On Monte Carlo Tree Search With Multiple Objectives

Michael Painter

Pembroke College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2024

Acknowledgements

TODO: acknowledgements here

Abstract

TODO: abstract here

Contents

List of Figures

List of Tables

List of Code Listings

List of Abbreviations

BTS	Boltzmann Tree Search.
CHVI	Convex Hull Value Iteration.
CHVS	Convex Hull Value Set.
CMAB	Contextual Multi-Armed Bandit (problem).
CZ	Contextual Zooming.
DENTS	Decaying ENtropy Tree Search
EMAB	Exploring Multi-Armed Bandit (problem).
MAB	Multi-Armed Bandit (problem).
MCTS	Monte Carlo Tree Search.
MCTS-1	A specific and common presentation of Monte Carlo Tree Search, presented in Section 2.4.
MDP	Markov Decision Process.
MENTS	Maximum ENtropy Tree Search.
MOMDP	Multi-Objective Markov Decision Process.
THTS	Trial-based Heuristic Tree Search.
`THTS++`	An extension of THTS used in this thesis.
UCB	Upper Confidence Bound (algorithm).
UCT	Upper Confidence Bound applied to Trees.

List of Notation

General Notation

$\mathbb{1}$ . . . . . The indicator function, where $\mathbb{1}(A) = 1$ when A is true, and $\mathbb{1}(A) = 0$ when A is false.

$\hat{A}$ . . . . . The hat notation is used to denote an estimate, i.e. $\hat{A}$ is an estimate for some optimal value A^*

$\bar{A}$ . . . . . The bar notation is used to denote sample averages, i.e. $\bar{A}$ is a sample average of a number of samples $A_1, ..., A_k$, or $\bar{A} = \frac{1}{n}\sum_{i=1}^{k} A_i$.

$\tilde{A}$ . . . . . The tilde notation is used to denote a function approximator, such as a neural network.

$\mathcal{A}$ . . . . . Calligraphic font is used to denote variables that are sets or tuples (ordered sets).

$\mathbf{A}$ . . . . . Bolt font is used to denote vectors, and A_i is used to denote the ith component of the vector $\mathbf{A}$.

$\mathtt{A}$ . . . . . Typewriter text is used to refer to variables relating to an algorithm (or the analysis of an algorithm).

$\boldsymbol{\mathcal{A}}$ . . . . . The notations above will also be combined at times, so $\boldsymbol{\mathcal{A}}$ is used to denote a set that contains vectors.

$\mathbb{E}$ . . . . . The expectation operator. $\mathbb{E}[f]$ denotes the expectation of a distribution f, and $\mathbb{E}_{x\sim f}[g(x)]$ denotes the expected value of $g(x)$ under the distribution f.

$x \sim f$ . . . . . If $f : X \to \mathbb{R}$ specifies a probability distribution in some way (e.g. a probability density function or cumulative density function), then $x \sim f$ denotes that the variable x is sampled the distribution f.

(Multi-Objective) Markov Decision Processes (Defined in Sections 2.2 and 2.5)

$\mathcal{A}$ . . . . . A (finite) set of actions.

a_t . . . . . The action at the tth timestep of a trajectory.

D . . . . . : The dimension of rewards in a MOMDP.

H . . . . . The finite-horizon time bound of an MDP.

$\mathcal{M}$ . . . . . A Markov Decision Process, which is a tuple $\mathcal{M} = (\mathcal{S}, s_0, \mathcal{A}, R, p, H)$.

$\boldsymbol{\mathcal{M}}$ . . . . . A Multi-Objective Markov Decision Process, which is a tuple $\boldsymbol{\mathcal{M}} = (\mathcal{S}, s_0, \mathcal{A}, \mathbf{R}, p, H)$.

p . . . . . The next-state transition distribution of an MDP. $p(s'|s, a)$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$.

π . . . . . : A policy, mapping a state $s \in \mathcal{S}$ to a probability distribution over actions $\mathcal{A}$.

R . . . . . The reward function of an MDP: $R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

$\mathbf{R}$ . . . . . The D dimensional reward function of a MOMDP: $\mathbf{R}(s, a)$: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}^D$.

r_t . . . . . The reward at the tth timestep of a trajectory.

$\mathcal{S}$ . . . . . A (finite) set of states.

$\text{Succ}(s, a)$ . . . . The set of successor states of a state-action pair (s, a), with respect to an MDP: $\text{Succ}(s, a) = \{s' \in \mathcal{S} | p(s'|s, a) > 0\}$.

s_0 . . . . . $s_0 \in \mathcal{S}$ is the initial starting state of an MDP.

s_t . . . . . The state at the tth timestep of a trajectory.

$\texttt{terminal}(s)$ . . Denotes if a state s is terminal or not. That is, if the transition distribution and reward function of the MDP is such that once reached, the state will never be left and no more reward will be received.

τ . . . . . A trajectory, or sequence, of states, actions and rewards that are sampled according to a policy π and an MDP $\mathcal{M}$: $\tau = (s_0, a_0, r_0, s_1, ..., s_{H-1}, a_{H-1}, r_{H-1}, s_H)$.

τ . . . . . A multi-objective trajectory, of states, actions and rewards that are sampled according to a policy π and an MDP $\mathcal{M}$: $\tau = (s_0, a_0, \mathbf{r}_0, s_1, ..., s_{H-1}, a_{H-1}, \mathbf{r}_{H-1}, s_H)$.

$\tau_{i:j}$ . . . . . A truncated trajectory, starting at timestep i, and ending at timestep j: $\tau_{i:j} = (s_i, a_i, r_i, s_{i+1}, ..., s_{j-1}, a_{j-1}, r_{j-1}, s_j)$.

$\tau_{i:j}$ . . . . . A multi-objective truncated trajectory, starting at timestep i and ending at timestep j: $\tau_{i:j} = (s_i, a_i, \mathbf{r}_i, s_{i+1}, ..., s_{j-1}, a_{j-1}, \mathbf{r}_{j-1}, s_j)$.

Reinforcement Learning (Section 2.3)

$J(\pi)$ . . . . . The objective function for (standard) reinforcement learning: $J(\pi) = V^{\pi}(s_0; 0)$.

π^* . . . . . The optimal standard policy, that maximises the objective function $J(\pi)$.

Q^* . . . . . The optimal Q-value function. $Q^*(s, a; t)$ denotes the maximal expected value that can be achieved by any policy, starting from state $s_t = s$, with action $a_t = a$.

Q^{π} . . . . . The Q-value function of a policy π. $Q^{\pi}(s, a; t)$ denotes the expected cumulative reward that policy π will obtain, starting from state $s_t = s$, starting by taking action $a_t = a$.

V^* . . . . . The optimal value function. $V^*(s; t)$ denotes the maximal expected value that can be achieved by any policy, starting from state $s_t = s$.

V^{π} . . . . . The value of a policy π. $V^{\pi}(s; t)$ denotes the expected cumulative reward that policy π will obtain, starting from state $s_t = s$.

Maximum Entropy Reinforcement Learning (Section 2.3.1)

α . . . . . The temperature parameter, or, the coefficient of the entropy term in the maximum entropy (soft) objective.

$\mathcal{H}$ . . . . . The Shannon entropy, of a probability distribution or policy.

$J_{\text{sft}}(\pi)$ . . . . . The objective function for maximum entropy (soft) reinforcement learning: $J_{\text{sft}}(\pi) = V^{\pi}_{\text{sft}}(s_0; 0)$.

π^*_{sft} . . . . . The optimal soft policy, that maximises the soft objective function $J_{\text{sft}}(\pi)$.

Q^*_{sft} . . . . . The optimal soft Q-value function. $Q^*(s, a; t)$ denotes the maximal expected soft value that can be achieved by any policy, from state $s_t = s$ and taking action $a_t = a$.

Q^{π}_{sft} . . . . . The soft Q-value function of a policy π. $Q^{\pi}_{\text{sft}}(s, a; t)$ is the expected value of the policy from $s_t = s$, starting with action $a_t = a$, with an addition of the entropy of the policy weighted by the temperature α.

V^*_{sft} . . . . . The optimal soft value function. $V^*(s; t)$ denotes the maximal expected soft value that can be achieved by any policy, from state $s_t = s$.

V^{π}_{sft} . . . . . The soft value of a policy π. $V^{\pi}_{\text{sft}}(s;t)$ is the expected value of the policy from $s_t = s$, with an addition of the entropy of the policy weighted by the temperature α.

Multi-Objective Reinforcement Learning (Section 2.5)

$CCS(\Pi)$ . . . . A convex coverage set of policies, which contains at least one policy that maximises the linear utility $u_{\text{lin}}(\cdot;\mathbf{w})$ for each weight vector $\mathbf{w}$.

$CCS_{\text{min}}(\Pi)$ . . . The minimal convex coverage set of policies, which has no redundant policies and for each policy in $CCS_{\text{min}}(\Pi)$ there is some weight that it uniquely (with respect to $CCS_{\text{min}}(\Pi)$) obtains the optimal linear utility.

$CH(\Pi)$ . . . . . The undominated set of policies in Π (the set of all possible policies), with respect to the linear utility function u_{lin}. $CH(\Pi) = U(\Pi; u_{\text{lin}})$.

$CS(\Pi;u)$ . . . . A coverage set of policies, which contains at least one policy that maximises the utility $u(\cdot;\mathbf{w})$ for each weight vector $\mathbf{w}$.

`cvx_prune` . . . An operation that takes an arbitrary set of vectors, and returns the subset that lie at the vertices of the geometric (partial) convex hull.

Δ^D . . . . . The D dimensional simplex, or, the set of possible weightings over the D objectives of a MOMDP. $\Delta^D = \{\mathbf{w} \in \mathbb{R}^D | w_i > 0, \sum_i w_i = 1\}$.

Π . . . . . The set of all possible policies in a MOMDP.

$\mathbf{Q}^{\pi}$ . . . . . The multi-objective Q-value function of a policy π. $Q^{\pi}(s,a;t)$ denotes the expected cumulative vector reward that policy π will obtain, starting from state $s_t = s$, starting by taking action $a_t = a$.

$U(\Pi;u)$ . . . . . The undominated set of policies in Π (the set of all possible policies), with respect to the utility function u. Each policy in $U(\Pi;u)$ achieves a maximal utility for some weighting over the objectives.

u . . . . . A utility function, mapping multi-objective values to scalar values, that depends on a weighting over objectives. The multi-objective value $\mathbf{V}^{\pi}(s,a;t)$ is mapped to $u(\mathbf{V}^{\pi}(s,a;t);\mathbf{w})$, where $\mathbf{w}$ is a weighting over the objectives.

u_{lin} . . . . . The linear utility function: $u_{\text{lin}}(\mathbf{v}; \mathbf{w}) = \mathbf{w}^\top \mathbf{v}$.

$\mathbf{V}^\pi$ . . . . . The multi-objective value of a policy π. $V^\pi(s;t)$ denotes the expected cumulative reward that policy π will obtain, starting from state $s_t = s$.

$\boldsymbol{\mathcal{V}}\mathbf{als}(\Pi')$ . . . . The set of multi-objective values obtained by a set of policies Π'.

$\mathbf{w}$ . . . . . A weighting over objectives that quantifies preferences over the multiple objectives, to be used in a utility function.

Trial Based Heuristic Tree Search (Section 2.4)

`backup_v` . . . . Updates the values at a decision node in the backup phase of a trial `THTS++`, using the decision node's children, the trajectory sampled for the trial and the heuristic value function.

`backup_q` . . . . Updates the values at a chance node in the backup phase of a trial `THTS++`, using the chance node's children, the trajectory sampled for the trial and the heuristic value function.

$H_{\texttt{THTS++}}$ . . . . . The planning horizon used in `THTS++`, with $H_{\texttt{THTS++}} \leq H$.

`mcts_mode` . . . Specifies if `THTS++` will sample a trajectory such that only one decision node is added to the search tree per trial. If not running in `mcts_mode` the `THTS++` will sample a trajectory until the planning horizon $H_{\texttt{THTS++}}$.

$N(s)$ . . . . . The number of visits at the decision node corresponding to state s.

$N(s,a)$ . . . . . The number of visits at the chance node corresponding to state-action pair (s,a).

`node`(s) . . . . . The decision node corresponding to the state s.

`node`(s)`.chldrn` The set of chance nodes that are children of `node`(s).

`node`(s)`.V` . . . The set of variables stored at decision node `node`(s), typically used for estimating values.

`node`(s,a) . . . The chance node corresponding to the state-action pair (s,a).

`node`(s,a)`.chldrn` The set of decision nodes that are children of `node`(s,a).

`node`(s,a)`.Q` . . The set of variables stored at decision node `node`(s,a), typically used for estimating Q-values.

π_{search} . . . . . The search policy used in `THTS++` to sample a trajectory in the selection phase.

$\hat{Q}_{\text{init}}$ . . . . . The heuristic action function used in `THTS++`, used to provide a Q-value estimate for any state-action pairs that aren't in the search tree.

`sample_context` A function used in `THTS++` that creates a context, or key-value story, and samples any initial values to be stored in the context.

`sample_outcome` A function used in `THTS++` to sample outcomes (successor states) from the environment (MDP).

$\mathcal{T}$ . . . . . The `THTS++` search tree. $\mathcal{T} \subseteq \mathcal{S} \cup \mathcal{S} \times \mathcal{A}$.

$\hat{V}_{\text{init}}$ . . . . . The heuristic value function used in `THTS++`, used to initialise the value of a new decision node.

Maximum ENtropy Tree Search (Section 2.4.4)

TODO . . . . . TODO: Clean up notation used for maximum entropy tree search

α_{MENTS} . . . . . The temperature parameter used in MENTS, which is the coefficient of the entropy terms in the maximum entropy objective.

ϵ . . . . . An exploration parameter used in MENTS.

λ_s . . . . . An exploration parameter used in MENTS.

$\tilde{\pi}$ . . . . . A policy (neural) network, used to initialise soft Q-value estimates in MENTS, if available.

π_{MENTS} . . . . . The search policy used in MENTS.

$\hat{Q}_{\text{MENTS}}(s, a)$ . . A soft Q-value estimate at the chance node (s, a) in MENTS.

$\tilde{V}$ . . . . . A function approximation to V^*.

$\hat{V}_{\text{MENTS}}(s)$ . . . A soft value estimate at the decision node s in MENTS.

TODO: add MENTS, UCT, BTS, DENTS, and all of ch4 stuff

1 Introduction

Contents

TODO: chapter structure (i.e. in the introduction section I give some background in the field(s), cover the main contributions of this thesis, etc, etc).

1.1 Overview

TODO: list

- Give some context around MCTS (and talk about exploration and exploitation), and why we might use it
  - Larger scale than tabular methods
  - Can do probability and theory stuff (and some explainability, by looking at stats in the tree the agent used)
  - Can use tree search with neural networks to get some of the above (and use for neural network training as in alpha zero)

- Argument from DENTS paper for exploration > exploitation (in context of planning in a simulator)
- Give high level overview of Multi-Objective RL, and why it can be useful
- Give an idea of how my work fits into MCTS and MORL as a whole
- Discuss research questions/issues with current literature (i.e. introduce some of the ideas from contributions section below)

1.2 Contributions

TODO: Inline acronyms used, or make sure that they're defined before hand

Throughout this thesis, we will consider the following questions related to Monte Carlo Tree Search and Multi-Objective Reinforcement Learning:

Q1 - Exploration: When planning in a simulator with limited time, how can MCTS algorithms best explore to make good decisions?

Q1.1 - Entropy: Entropy is often used as an exploration objective in RL, but can it be used soundly in MCTS?

Q1.2 - Multi-Objective Exploration: How can Multi-Objective MCTS methods explore to find optimal actions for different objectives?

Q2 - Scalability: How can the scalability of (multi-objective) MCTS methods be improved?

Q2.1 - Complexity: MCTS algorithms typically run in $O(nAH)$, but are there algorithms that can improve upon this?

Q2.2 - Multi-Objective Scalability: With respect to the size of environments, how scalable are Multi-Objective MCTS methods?

Q2.3 - Curse of Dimensionality: With respect to the number of objectives, to what extent do Multi-Objective MCTS methods suffer from the curse of dimensionality?

Q3 - Evaluation: How can we best evaluate a search tree produced by a Monte Carlo Tree Search algorithm?

Q3.1 - Tree Policies: Does it suffice to extract a policy from a single search tree for evaluation? TODO: going to have to run some extra experiments for that, but I probably should do that for completeness anyway

Q3.2 - Multi-Objective Evaluation: Can we apply methods from the MORL literature to theoretically and empirically evaluate Multi-Objective MCTS?

TODO: some words about how below is the contributions we're making in this thesis and expand these bullets a bit more

- Max Entropy can be misaligned with reward maximisation (**Q1.1 - Entropy**)
- Boltzmann Search Policies - BTS and DENTS (**Q1.1 - Entropy**, and with extra results **Q3.1 - Tree Policies**)
- Use the alias method to make faster algorithms (**Q2.1 - Complexity**)
- Simple regret (**Q1 - Exploration**)
- Use of contexts in THTS to make consistent decisions in each trial (**Q1.2 - Multi-Objective Exploration**)
- Contextual regret introduced in CHMCTS (**Q2.2 - Multi-Objective Scalability**, **Q3.2 - Multi-Objective Evaluation**)
- Contextual Zooming and CHMCTS (designed for **Q1.2 - Multi-Objective Exploration**, runtimes cover **Q2.3 - Curse of Dimensionality**, results **Q3.2 - Multi-Objective Evaluation**)
- Simplex maps (**Q1.2 - Multi-Objective Exploration**, **Q2.2 - Multi-Objective Scalability**, **Q2.3 - Curse of Dimensionality**)
- Contextual Simple Regret (**Q3.2 - Multi-Objective Evaluation**)

TODO: Would like to do the comparing different types of eval, even if not listing it as a research question (compare giving it X seconds per decision and evaluating that policy (SLOW), and comparing policy extracted from the tree)

TODO: can make an argument that the best bound achieved by theory is given by letting temperature go to max. Which is consistent with the exploring bandits results

1.3 Structure of Thesis

TODO: a paragraph with a couple lines to a paragraph about each chapter. This is the high level overview/intro to the thesis paragraph. I.e. this section is "this is the story of my thesis in a page or two"

1.4 Publications

TODO: update final publication when submit

The work covered in this thesis also appears in the following publications:

- Painter, M; Lacerda, B; and Hawes, N. "Convex Hull Monte-Carlo Tree-Search." In *Proceedings of the international conference on automated planning and scheduling. Vol. 30. 2020*, ICAPS, 2020.
- Painter, M; Baioumy, M; Hawes, N; and Lacerda, B. "Monte Carlo Tree Search With Boltzmann Exploration." In *Advances in Neural Information Processing Systems, 36, 2023*, NeurIPS, 2023.
- Painter, M; Hawes, N; and Lacerda, B. "Simplex Maps for Multi-Objective Monte Carlo Tree Search." In *TODO*, **Under Review at conf_name**.

2
Background

Contents

This chapter introduces the fundamental concepts that will be used throughout this thesis, in particular the `THTS++` schema is introduced in Section 2.4.2, in which monte carlo tree search algorithms will be defined. Section 2.1 begins with multi-armed bandit problems and decision theory, which provides a fundamental building block used in monte carlo tree search methods (Section 2.4). Sections 2.2 and 2.3 provide a framework for sequential decision making under uncertainty.

Section 2.4 introduces monte carlo tree search methods methods that can be used for solving problems in sequential decision making under uncertainty. Section 2.5 extends the frameworks to include multiple objectives. And Section 2.6 discusses sampling from categorical distributions efficiently.

2.1 Multi-Armed Bandits

This section introduces the K-armed bandit problem [23], which is a foundational problem considered in decision theory. In the multi-armed bandit (MAB) problem, an agent is tasked with deciding to pull one of K arms, and the decision typically needs to be made multiple times. For example, the MAB problem could be used in the context of clinical trials, where each "arm" corresponds to a treatment option. As such, the MAB problem is presented in rounds, where an arm is selected each round. In each round an random outcome is observed in form of a reward.

The distribution of rewards is unknown ahead of time, and so an agent needs to explore to gather information about what rewards can be obtained by pulling each arm, and if an agent does not explore then it may miss out on discovering the best strategy. Conversely, the agent should want to exploit, using the information that it has obtained, so that it gather high rewards. In the clinical trials example, exploitation is desirable so that patients receive the best treatment. The problem of balancing these two aspects is known as the *exploration-exploitation trade off*.

The remainder of this section will formalise the MAB problem, and give details of the widely used Upper Confidence Bound (UCB) algorithm [2]. Sections 2.1.1 and 2.1.2 considers two variations on the MAB problem that will be relevant in this thesis.

Figure 2.1 shows how the operation of a K-armed bandit problem proceeds. Formally, let $f(1), ..., f(K)$ be the probability distributions for the rewards of each of the K arms, with expected values of $\mu(1), ..., \mu(K)$ respectively. On the mth round, if the arm $x^m \in \{1, ..., K\}$ is pulled, then a reward $y^m \sim f(x^m)$ is received. To specify a *strategy* σ, on each round m, a probability distribution σ^m is over $\{1, ..., K\}$ is given, which is sampled to decide which arm to pull, i.e. $x^m \sim \sigma^m$.

Parameters: K probability distributions for the rewards of each arm $f(1), ..., f(K)$.

- For each round $m = 1, 2, ...$:
  - the agent selects an arm $x^m \sim \sigma^m$ to pull;
  - the agent receives a reward $y^m \sim f(x^m)$ from the environment;
  - if the environment sends a stop signal, then the game ends, otherwise the next round starts.

Figure 2.1: The procedure of a multi-armed bandit problem, following a strategy σ.

To assess algorithmic strategies for MAB problems, a quantity known as *regret* is commonly used, which compares the cumulative reward obtained, compared to the maximal expected reward that could be obtained with full knowledge of $\{f(i)\}$.

Definition 2.1.1. *The* (cumulative) regret *of strategy σ after n rounds in the MAB process is:*

$$\texttt{cum\_regr}_{\text{MAB}}(n, \sigma) = n\mu^* - \sum_{m=1}^{n} y^m, \tag{2.1}$$

where $\mu^* = max_i \mu(i)$.

To theoretically analyse algorithms for MAB problems, the quantity of expected regret, $\mathbb{E}[\texttt{cum\_regr}_{\text{MAB}}(n, \sigma)]$ is considered. Lai and Robbins [18] show using information theory that there is a lower bound on the expected regret that an agent can achieve of $\Omega(\log n)$. And Auer [2] introduces the Upper Confidence Bound (UCB) algorithm, which achieves a matching upper bound of $O(\log n)$ on the expected regret.

To define the UCB strategy for pulling the arms, a few quantities need to be defined first. Let $N^m(x)$ be the number of times that arm x has been pulled after m rounds. And let $\bar{y}^m(x)$ be the average reward that has been received as a result of pulling arm x after m rounds. Mathematically:

$$N^m(x) = \sum_{i=1}^{m} \mathbb{1}[x^i = x], \tag{2.2}$$

$$\bar{y}^m(x) = \frac{1}{N^m(x)} \sum_{i=1}^{m} y^i \mathbb{1}[x^i = x]. \tag{2.3}$$

The strategy followed by the UCB algorithm on the mth round is given by:

$$x^m_{\text{UCB}} = \arg\max_{i \in \{1,..,K\}} \bar{y}^{m-1}(i) + b_{\text{UCB}} \sqrt{\frac{\log(m)}{N^{m-1}(i)}}, \tag{2.4}$$

$$\sigma^m_{\text{UCB}}(x) = \mathbb{1}[x = x^m_{\text{UCB}}], \tag{2.5}$$

where b_{UCB} is a bias parameter used to control how much the strategy explores. Additionally, each arm is pulled once in the first K rounds by the UCB strategy, to avoid division by zero in Equation 2.4. Alternatively, the division by zero can be considered to give a value of ∞, to give the same effect.

2.1.1 Exploring Bandits

In the pure exploration problem for K-armed bandits [7], the format of each round is changed slightly. The agent still gets to pull an arm each round (according to the *exploration strategy* σ), but after it recieves a reward on each round it is given the opportunity to output a *recommendation strategy* ψ. On the mth round, the recommendation strategy takes the form of ψ^m, a probability distribution over the K arms. In exploring multi-armed bandit (EMAB) problems, the emphasis is now on the algorithm being able to provide the best recommendations possible, rather than trying to exploit pulling the best arm each round. In essence, this separates the needs to explore and exploit, the agent needs to explore with its arm pulls, and output an exploiting recommendation at the end of each round. Figure 2.2 gives an overview of the EMAB problem.

The *(expected) instantaneous regret* for pulling an arm $x \in \{1, ..., K\}$ is given by:

$$\texttt{inst\_regr}(x) = \mu^* - \mu(x), \tag{2.6}$$

where again $\mu^* = max_i \mu(i)$.

Under the exploring regime of EMABs, the performance of an algorithm can be analysed by considering the quantity of *simple regret* of the recommendation policy. The simple regret is the expected value of the instantaneous regret which would come from following the recommendation policy.

Parameters: K probability distributions for the rewards of each arm $f(1), ..., f(K)$.

- For each round $m = 1, 2, ...$:
  - the agent selects an arm $x^m \sim \sigma^m$ to pull;
  - the agent receives a reward $y^m \sim f(x^m)$ from the environment;
  - the agent outputs a recommendation distribution ψ^m, over the arms $\{1, ..., K\}$;
  - if the environment sends a stop signal, then the game ends, otherwise the next round starts.

Figure 2.2: The procedure of an exploring multi-armed bandit problem, following an exploration strategy σ, and recommendation strategy ψ.

Definition 2.1.2. *The* simple regret *of following the exploration strategy σ and recommending a distribution ψ^m on the mth round is:*

$$\texttt{sim\_regr}_{\text{EMAB}}(m, \sigma, \psi^m) = \mathbb{E}_{i \sim \psi^m}[\mu^* - \mu(i)]. \tag{2.7}$$

Bubeck et al. [7] analyse a range of exploration and recommendation strategies in their work. Two of the recommendation strategies considered are the *empirical best arm* (EBA) and *most played arm* (MPA):

$$x^m_{\text{EBA}} = \arg\max_{i \in \{1,..,K\}} \bar{y}^m(i), \tag{2.8}$$

$$\psi^m_{\text{EBA}}(x) = \mathbb{1}[x = x^m_{\text{EBA}}], \tag{2.9}$$

$$x^m_{\text{MPA}} = \arg\max_{i \in \{1,..,K\}} N^m(i), \tag{2.10}$$

$$\psi^m_{\text{MPA}}(x) = \mathbb{1}[x = x^m_{\text{MPA}}]. \tag{2.11}$$

In addition to the UCB strategy, a *uniform exploration strategy* is considered:

$$x^m_{\text{uniform}} = (m \mod K) + 1, \tag{2.12}$$

$$\sigma^m_{\text{uniform}}(x) = \mathbb{1}[x = x^m_{\text{uniform}}]. \tag{2.13}$$

It is shown that exploring with UCB leads to a polynomial simple regret bound for both recommendation strategies: $\texttt{sim\_regr}_{\text{EMAB}}(m, \sigma_{\text{UCB}}, \psi^m_{\text{EBA}}) = O(m^{-k_{\text{UCB,EBA}}})$

Parameters: the set of arms $\mathcal{X}$, the set of contexts $\mathcal{W}$ and a mapping from $\mathcal{W} \times \mathcal{X}$ to probability distributions: $f(w, x)$.

- For each round $m = 1, 2, ...$:
  - the agent recieves a context $w^m \in \mathcal{W}$ from the environment;
  - the agent selects an arm $x^m \sim \sigma^m(w^m)$ to pull;
  - the agent receives a reward $y^m \sim f(w^m, x^m)$ from the environment;
  - if the environment sends a stop signal, then the game ends, otherwise the next round starts.

Figure 2.3: The procedure of a contextual multi-armed bandit problem, following a strategy σ.

and $\texttt{sim\_regr}_{\text{EMAB}}(m, \sigma_{\text{UCB}}, \psi^m_{\text{MPA}}) = O(m^{-k_{\text{UCB,MPA}}})$ for some appropriate constants $k_{\text{UCB,EBA}}, k_{\text{UCB,MPA}} > 0$. It is also shown that exploring with the uniform strategy (and recommendating the empirical best arm) leads to an exponential simple regret bound, $\texttt{sim\_regr}_{\text{EMAB}}(m, \sigma_{\text{UCB}}, \psi^m_{\text{EBA}}) = O(e^{-k_{\text{uniform}} m})$ for an appropriate constant $k_{\text{uniform}} > 0$.

2.1.2 Contextual Bandits

In the contextual multi-armed bandit (CMAB) problem [15, 30], before an arm is chosen, the agent is provided with a context w. In the CMAB problem the set of arms, now $\mathcal{X}$, and the set of contexts, $\mathcal{W}$ are compact sets (i.e. the sets are closed and bounded). The distribution of rewards now also depends on the context w and arm pulled x, and is written $f(w, x)$, with mean $\mu(w, x)$. The distribution of arms pulled on the mth round can now depend on the context: $\sigma^m(w)$. Figure 2.3 gives an overview of the CMAB problem.

Given this setup some further assumptions are required for the problem to be more tractable. Slivkins [30] make a fairly general assumption that some *metric* d over the *similarity space* $\mathcal{W} \times \mathcal{X}$ is known and is such that the following Lipschitz condition holds:

$$|\mu(x, y) - \mu(x', y')| \leq d((x, y), (x', y')). \tag{2.14}$$

For d to be a metric, the following conditions must hold for some arbitrary $z = (x, y), z' = (x', y'), z'' = (x'', y'')$ with $z \neq z' \neq z''$:

$$d(z, z) = 0, \tag{2.15}$$

$$d(z, z') > 0, \tag{2.16}$$

$$d(z, z') = d(z', z), \tag{2.17}$$

$$d(z, z'') \leq d(z, z') + d(z', z''). \tag{2.18}$$

Similar to MABs and EMABs, the notion of regret is used to analyse CMABs. Specifically, *contextual regret* is defined similarly to cumulative regret [15, 30], while taking into account the contexts drawn.

Definition 2.1.3. *The* (cumulative) contextual regret *of the strategy σ in the process given in Figure 2.3 is defined as:*

$$\texttt{ctx\_regr}_{\text{CMAB}}(\pi, n) = \sum_{m=1}^{n} \mu^*(w^m) - y^m, \tag{2.19}$$

where $\mu^*(w) = \sup_i \mu(w, i)$.

Slivkins [30] also introduces the Contextual Zooming (CZ) algorithm. CZ runs over a fixed number of rounds T, and achieves the contextual regret of $O(T^{\frac{1+c}{2+c}} \log(T))$, where c is the *(Lebesgue)covering dimension* of the similarity space $\mathcal{W} \times \mathcal{X}$. For the purpose of this thesis the covering dimension (a notion about topological spaces), will always be equal to the typical notion of dimension. That is, if $\mathcal{W} \times \mathcal{X}$ has covering dimension c, then $\mathcal{W} \times \mathcal{X}$ can be mapped to a subspace of $\mathbb{R}^c$.

Throughout the CZ algorithm, a set of balls in the similarity space is maintained, called *active balls*. Let A^m denote the set of active balls at the start of the mth round. A ball with center (w, x) and radius r is given by $\{(w', x') \in \mathcal{W} \times \mathcal{X} \mid d((w, x), (w', x')) < r\}$.

The CZ algorithm operates using two rules each round m: the *selection rule* is used to select a relevant ball B^m from the set of active balls A^m; then, an *activation rule* is optionally applied that adds a new ball with smaller radius.

Now the selection rule is described in more detail. Firstly, let $N^m(B)$ be the number of times that a ball $B \in A^m$ has been selected in the first m rounds, and $\bar{y}^m(B)$ the average reward received after m rounds when selecting ball B.

$$N^m(B) = \sum_{i=1}^{m} \mathbb{1}[B = B^m], \tag{2.20}$$

$$\bar{y}^m(B) = \frac{1}{N^m(B)} \sum_{i=1}^{m} y^m \mathbb{1}[B = B^m]. \tag{2.21}$$

Let $r(B)$ denote the radius of ball B and let $\texttt{dom}^m(B)$ be the domain of ball B on the mth round, which is the subset of B that excludes all balls of smaller radius, or:

$$\texttt{dom}^m(B) = B - \left(\bigcup_{B' \in A^m : r(B') < r(B)} B' \right). \tag{2.22}$$

The *confidence radius* $\texttt{conf}^m(B)$ of ball B on the mth round is:

$$\texttt{conf}^m(B) = 4\sqrt{\frac{\log T}{1 + N^{m-1}(B)}}. \tag{2.23}$$

Let $\texttt{relevant}^m(w, A^m)$ denote the set of balls that are relevant for context w on the mth round, which is the set of balls that contain an arm x such that (w, x) is in the domain of the ball:

$$\texttt{relevant}^m(w, A^m) = \{B \in A^m | \exists x \in \mathcal{X}.(w, x) \in \texttt{dom}^m(B)\}. \tag{2.24}$$

The ball B^m selected on the mth round with context w^m can now be given as:

$$I^m(B) = \bar{y}^{m-1}(B) + r(B) + \texttt{conf}^m(B), \tag{2.25}$$

$$B^m = \max_{B \in \texttt{relevant}^m(w^m, A^m)} r(B) + \min_{B' \in A^m} \left(I^m(B') + d(B, B') \right), \tag{2.26}$$

where $d(B, B')$ is the distance between the centers of balls B, B' according to the metric d.

Once a ball has been selected, the arm sampled by CZ, x^m_{CZ}, is sampled randomly from the domain of ball B^m (from the set $\{x \in \mathcal{X} | (w^m, x) \in \texttt{dom}^m(B^m)\}$), which concludes the selection rule.

The activation rule is then used if the confidence radius from Equation (2.23) has shrunk to smaller than the radius of ball selected this round, i.e. if $\texttt{conf}^{m+1}(B^m) \leq r(B^m) < \texttt{conf}^m(B^m)$. In this case, a new ball B' is added to the set of active balls, with radius $r(B') = \frac{1}{2} r(B^m)$ and center (w^m, x^m). If a new ball is added, then $A^{m+1} = A^m \cup \{B'\}$ and otherwise $A^{m+1} = A^m$.

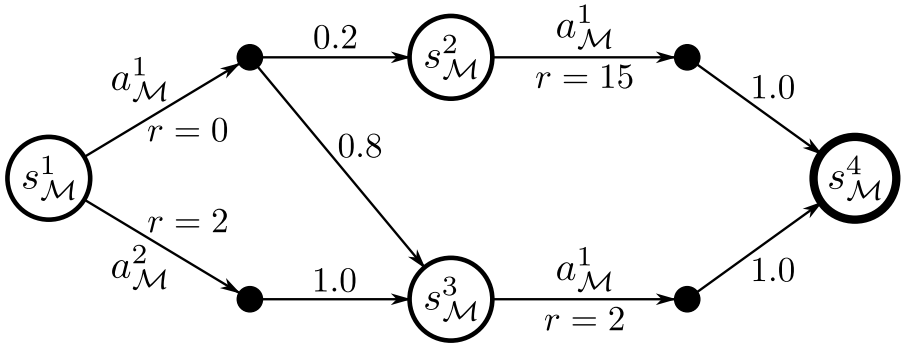


Figure 2.4: An example MDP $\mathcal{M}$, where $s_0 = s^1_{\mathcal{M}}$, the finite horizon is $H = 2$, and terminal states are marked with a thicker border. Edges are appropriately marked with actions and rewards, or marked with transition probabilities.

2.2 Markov Decision Processes

In this section *Markov decision processes* (MDPs) are introduced, along with related definitions of *policies* and *trajectories*. MDPs give a mathematical framework for problems concerning sequential decision making under uncertainty, and in this thesis will be the framework used to model the environment that agents act in. An MDP contains, among other things, a set of states and actions. States are sampled according to a transition distribution which depends on the current state and current action being taken (the Markov assumption). Any time an action is taken from a state the agent receives an instantaneous reward, according to a reward function that depends on the state and action taken.

This thesis is concerned with discrete, finite, fully-observable and finite-horizon Markov decision processes, meaning that the state and action spaces are discrete and finite, and any *trajectories* (sequences of states, actions and rewards) are of a finite length.

Definition 2.2.1. *A* Markov decision process *(MDP) is a tuple* $\mathcal{M} = (\mathcal{S}, s_0, \mathcal{A}, R, p, H)$*, where* $\mathcal{S}$ *is a set of states,* $s_0 \in \mathcal{S}$ *is an initial state,* $\mathcal{A}$ *is a set of actions,* $R(s, a)$ *is a reward function* $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$*,* $p(s'|s, a)$ *is a next state transition distribution* $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *and* $H \in \mathbb{N}$ *is a finite-horizon time bound.*

An example MDP is shown in Figure 2.4. Notationally, it is convenient to define the set of successor states, that is the set of states that could be reached after

taking an action from the current state of the MDP:

Definition 2.2.2. *The set of* successor states $\mathrm{Succ}(s, a)$ *of a state-action pair* (s, a)*, with respect to an MDP, is defined as:*

$$\mathrm{Succ}(s, a) := \{s' \in \mathcal{S} | p(s'|s, a) > 0\}. \tag{2.27}$$

Additionally, throughout this thesis, MDPs will be defined with *terminal states*, which are states that once reached will never be left, and no more reward can be obtained.

Definition 2.2.3. *A state* $s \in \mathcal{S}$ *is a* terminal *state (or a* trap *state), if the transition distribution always returns the same state (i.e.* $p(s|s, a) = 1$ *for all actions* a*), and if no more reward is obtained from that state (*$R(s, a) = 0$ *for all* a*). The function* `terminal`(s) *returns a boolean in* $\{\mathrm{True}, \mathrm{False}\}$ *which will be used to denote if a state* s *is terminal or not.*

To define a strategy that an agent will follow in an MDP, and agent defines a *policy*. A policy maps each state in the state space to a probability distribution over the action space. To "follow" a policy, actions are sampled from the distribution. Often it is desirable to define deterministic policies, which always produce the same action when given the same state, and can be represented as *one-hot* distributions.

Definition 2.2.4. *A* (stochastic) policy $\pi : \mathcal{S} \to (\mathcal{A} \to [0, 1])$ *is a mapping from states to a probability distributions over actions and* $\pi(a|s)$ *is the probability of sampling action* a *at state* s*. The policy* π *must satisfy the conditions: for all* $s \in \mathcal{S}$ *we have* $\sum_{a \in \mathcal{A}} \pi(a|s) = 1$ *and for all actions* $a \in \mathcal{A}$ *that* $\pi(a|s) \geq 0$ *.*

Additionally, a deterministic policy *is defined as a one-hot policy, that is, the policy* π *is deterministic iff it can be written as* $\pi(a|s) = \mathbb{1}[a = a']$ *for some* $a' \in \mathcal{A}$*.*

Moreover, the following notations are used for policies:

- $a \sim \pi(\cdot|s)$ *denotes sampling an action* a *from the distribution* $\pi(\cdot|s)$*;*
- $\pi(s) = a'$ *is used as a shorthand to define the deterministic policy* $\pi(a|s) = \mathbb{1}[a = a']$*;*

- $\pi(s)$ *is used as a shorthand for the action* $a' \sim \pi(\cdot|s)$ *in the case of a deterministic policy.*

Given an MDP $\mathcal{M}$ and a policy π it is then possible to sample a sequence of states, actions and rewards, known as a *trajectory*. A trajectory *simulates* one possible sequence that could occur if an agent follows policy π in $\mathcal{M}$, and in Section 2.4 these simulations are used to incrementally build a search tree.

Definition 2.2.5. *A* trajectory τ*, is a sequence of state, action and rewards, that is induced by a policy* π *and MDP* $\mathcal{M}$ *pair. Let the trajectory be* $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ..., s_{H-1}, a_{H-1}, r_{H-1}, s_H)$*, where* $a_t \sim \pi(\cdot|s_t)$*,* $r_t = R(s_t, a_t)$ *and* $s_{t+1} \sim p(\cdot|s_t, a_t)$*.*

The following notations will also be used for trajectories:

- $\tau \sim \pi$ *denotes a trajectory that is sampled using the policy* π*, where the MDP* $\mathcal{M}$ *is implicit;*
- $\tau_{i:j}$ *denotes the* truncated trajectory $\tau_{i:j} := (s_i, a_i, r_i, s_{i+1}, ..., s_{j-1}, a_{j-1}, r_{j-1}, s_j)$*, between the timesteps* $0 \leq i < j \leq H$ *inclusive;*
- $\tau_{:j} := \tau_{0:j}$ *denotes a trajectory that is trunacted on only one end,*
- *given a trajectory* τ*, the following set notation is used,* $s \in \tau$*,* $(s, a) \in \tau$ *as a shorthand for* $s \in \{s_i | i = 0, ..., H\}$ *and* $(s, a) \in \{(s_i, a_i) | i = 0, ..., H-1\}$ *respectively,*

and finally, when states, actions and rewards a timestep indexed (i.e. s_t, a_t, r_t*), they will always correspond to a trajectory.*

2.3 Reinforcement Learning

This section serves as a brief introduction to fundamental concepts in reinforcement learning, and motivates an alternative lens on reinforcement learning that this thesis will consider. The field of reinforcement learning considers an agent that has to learn how to make decisions by interacting with its environment (Figure 2.5).

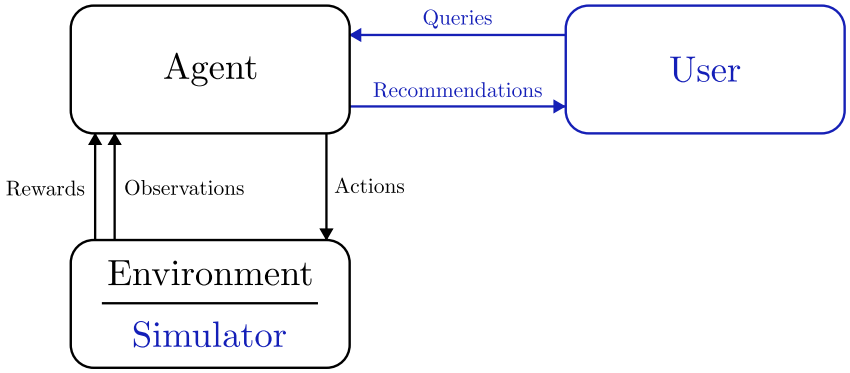


Figure 2.5: An overview of reinforcement learning. In black depicts the standard agent-environment interface for reinforcement learning [31], where an agent can perform actions in an environment and is given feedback in the form of observations and rewards. In blue the environment is replaced by a simulated environment which the agent can use for planning, and is queried by a user for recommendations on how to act in the real environment.

The agent can take actions in the environment, recieving in return *observations* and *rewards*, which can be used to update internal state and used to make further decisions, and the goal of the agent is to maximise the rewards that it recieves.

Classically the agent is considered to interact with its environment directly [31], and thus must make a trade-off between exploring new strategies and exploiting learned strategies. That is, reinforcement learning agents must consider the *exploration-exploitation trade-off* discussed in Section 2.1 too.

Also depicted in Figure 2.5 is a scenario where the agent is equiped with a simulator that it can use to plan and explore, and is either asked to recommend a strategy after a planning/learning phase, or is occassionally queried to recommend actions. This scenario more closely resembles how reinforcement learning is used in the modern era with greater amounts of compute power available, and interactions with the simulator occur at orders of magnitude quicker. Hence, in this scenario, the only significant real-world cost comes from following the recommendations output, to be used in the real-world environment. This changes the nature of the exploration-exploitation trade off, almost separating the two issues, where there is an emphasis on exploring during the planning phase, and the problem of providing good recommendations is concerned with pure exploitation. This

perspective on reinforcement learning motivates the research quesitons around exploration: **Q1 - Exploration**.

In this thesis, the environment will always take the form of an MDP (Defintion 2.2.1), and observations will always be *fully-observable*, meaning that the agent is provided with full access to the states of the MDP.

Following on from Section 2.2, the remainder of this section defines *value functions* and the objectives of reinforcement learning and covers *Value Iteration*, a tabular dynamic programming approach to reinforcement learning. Finally Section 2.3.1 covers *maximum-entropy reinforcement learning*.

The value of a policy π is the expected cumulative reward that will be obtained by following the policy:

Definition 2.3.1. *The* value *of a policy π from state s at time t is:*

$$V^{\pi}(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} r_t \middle| s_t = s\right]. \tag{2.28}$$

The Q-value *of a policy π, from state s, with action a, at time t is:*

$$Q^{\pi}(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}[V^{\pi}(s';t+1)]. \tag{2.29}$$

From the definition of the values functions the optimal value functions can be defined by taking the maximum value over all policies:

Definition 2.3.2. *The* Optimal (Q-)Value *of a state(-action pair) is defined as:*

$$V^{*}(s;t) = \max_{\pi} V^{\pi}(s;t) \tag{2.30}$$

$$Q^{*}(s,a;t) = \max_{\pi} Q^{\pi}(s,a;t). \tag{2.31}$$

Value functions can also be used to define an objective function:

Definition 2.3.3. *The* (standard) reinforcement learning objective function $J(\pi)$ *is defined as:*

$$J(\pi) = V^{\pi}(s_0;0). \tag{2.32}$$

The objective of (standard) reinforcement learning can then be stated as finding $\max_{\pi} J(\pi)$.

The *optimal policy* is the policy that maximises the objective function J, and can be shown to be deterministic [21]:

Definition 2.3.4. *The* optimal (standard) policy π^* *is the policy maximising the standard objective function:*

$$\pi^* = \arg\max_{\pi} J(\pi) \tag{2.33}$$

or equivalently, the optimal standard policy can be in terms of the optimal Q-value function:

$$\pi^*(s) = \arg\max_{a} Q^*(s, a). \tag{2.34}$$

It can be shown that the optimal (Q-)value functions satisfy the *Bellman equations* [5, 21]:

$$V^*(s; t) = \max_{a \in \mathcal{A}} Q^*(s, a; t), \tag{2.35}$$

$$Q^*(s, a; t) = R(s, a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}[V^*(s'; t + 1)]. \tag{2.36}$$

The Bellman equations admit a *dynamic programming* approach which can be used to computer the optimal value functions, known as *Value Iteration* [5, 21]. In Value Iteration, a table of value estimates $\hat{V}(s; t)$ are kept for each s, t. Given any initial estimate of the value function $\hat{V}^0$, the *Bellman backup* operations are:

$$\hat{V}^{k+1}(s; t) = \max_{a \in \mathcal{A}} \hat{Q}^{k+1}(s, a; t), \tag{2.37}$$

$$\hat{Q}^{k+1}(s, a; t) = \mathbb{E}_{s' \sim p(\cdot|s,a)}[R(s, a) + \hat{V}^k(s'; t + 1)]. \tag{2.38}$$

In each iteration of Value Iteration, these values are computed for all $s \in \mathcal{S}$, $a \in \mathcal{A}$ and $t \in \{0, 1, ..., H\}$. The value estimates computed from the Bellman backups will converge to the optimal values in a finite number of iterations: $V^k \to V^*$.

2.3.1 Maximum Entropy Reinforcement Learning

In *maximum-entropy reinforcement learning*, the objective function is altered to include the addition of an entropy term. The addition of an entropy term is motivated by wanting to learn stochastic behaviours, that better explore large state spaces, and learn more robust behaviours under uncertainty by potentially learning multiple solutions rather than a single deterministic solution [10].

Let $\mathcal{H}$ denote the (Shannon) entropy function [26]:

$$\mathcal{H}(\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi(\cdot|s)}[-\log\pi(a|s)] = \sum_{a\in\mathcal{A}} \pi(a|s)\log\pi(a|s). \tag{2.39}$$

In the maximum entropy objective, the relative weighting of entropy terms is included using a coefficient α, called the *temperature*. In the maximum entropy objective, analogues of the value functions can be defined, which are typically referred to as *soft (Q-)values*, and similarly the maximum entropy objective is often referred to as the *soft objective*.

Definition 2.3.5. *The* soft value *of a policy π from state s at time t is:*

$$V^{\pi}_{\text{sft}}(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} r_t + \alpha\mathcal{H}(\pi(\cdot|s_i))\middle| s_t = s\right]. \tag{2.40}$$

The soft Q-value *of a policy π, from state s, with action a, at time t is:*

$$Q^{\pi}_{\text{sft}}(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}[V^{\pi}_{\text{sft}}(s';t+1)]. \tag{2.41}$$

The optimal soft (Q-)values can be defined by taking the maximum over policies, similarly to the standard objective:

Definition 2.3.6. *The* optimal soft (Q-)Value *of a state(-action pair) is defined as:*

$$V^{*}_{\text{sft}}(s;t) = \max_{\pi} V^{\pi}_{\text{sft}}(s;t), \tag{2.42}$$

$$Q^{*}_{\text{sft}}(s,a;t) = \max_{\pi} Q^{\pi}_{\text{sft}}(s,a;t). \tag{2.43}$$

In maximum entropy reinforcement learning, the objective is to find a policy with maximal soft value.

Definition 2.3.7. *The* maximum entropy (or soft) reinforcement learning objective function $J_{\text{sft}}(\pi)$ *is defined as:*

$$J_{\text{sft}}(\pi) = V^{\pi}_{\text{sft}}(s_0; 0). \tag{2.44}$$

The objective of maximum entropy (or soft) reinforcement learning can then be stated as finding $\max_\pi J_{\text{sft}}(\pi)$.

The optimal soft policy is defined as the policy that maximises the soft objective function J_{sft}, and it can be computed from the optimal soft (Q-)values.

Definition 2.3.8. *The* optimal soft policy π^*_{sft} *is the policy maximising the soft objective function:*

$$\pi^*_{\text{sft}} = \arg\max_{\pi} J_{\text{sft}}(\pi). \tag{2.45}$$

Given V^*_{sft} *and* Q^*_{sft} *the optimal soft police is known to take the form [10]:*

$$\pi^*_{\text{sft}}(a|s; t) = \exp\left(\left(Q^*_{\text{sft}}(s, a; t) - V^*_{\text{sft}}(s; t)\right)/\alpha\right). \tag{2.46}$$

Equations similar to the Bellman equations, aptly named the *soft Bellman equations*, can be defined for maximum entropy reinforcement learning. These equations differ to equations (2.35) and (2.36) by the replacement of the max operation with a *softmax* or *log-sum-exp* operation, and explain why the maximum entropy analogues are referred to as the *soft* versions of their standard reinforcement learning counterparts. The *soft Bellman equations* are [10]:

$$V^*_{\text{sft}}(s; t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(Q^*_{\text{sft}}(s, a; t)/\alpha\right), \tag{2.47}$$

$$Q^*_{\text{sft}}(s, a; t) = R(s, a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}[V^*_{\text{sft}}(s'; t + 1)]. \tag{2.48}$$

Analogous to standard reinforcement learning, the soft Bellman equations can be used in *soft Bellman backups* for a *Soft Value Iteration* algorithm [10]:

$$\hat{V}^{k+1}_{\text{sft}}(s; t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha}\hat{Q}^{k+1}_{\text{sft}}(s, a; t)\right), \tag{2.49}$$

$$\hat{Q}^{k+1}_{\text{sft}}(s, a; t) = R(s, a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}[\hat{V}^{k}_{\text{sft}}(s'; t + 1)]. \tag{2.50}$$

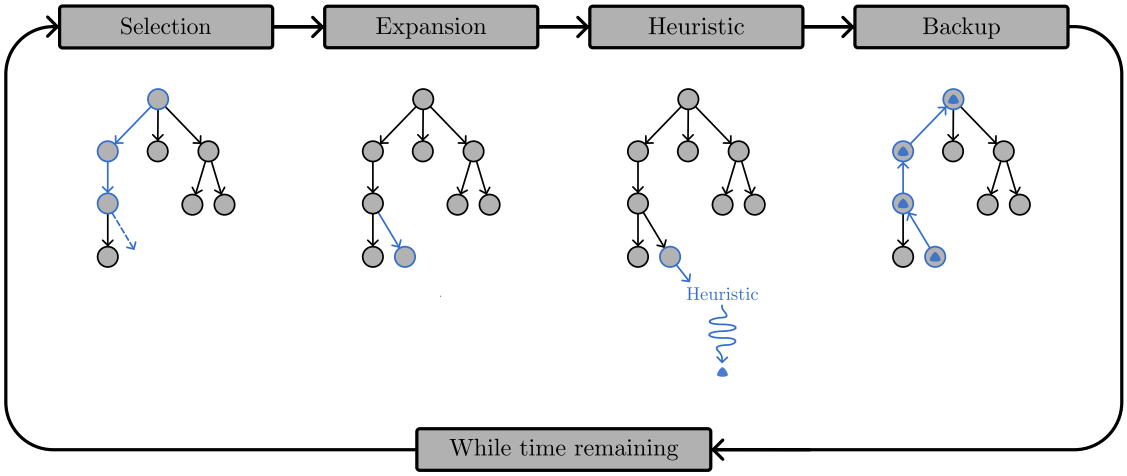


Figure 2.6: Overview of one trial of MCTS-1, depicting the four commonly presented stages [6]. First, in the selection phase a search policy is used to sample a path down the tree. Next a new node is added to the search tree. Then the new node is initialised using a heuristic, often using a *simulation* using a *rollout policy* as depicted. Finally, values (triangles) are updated through *backups* along the path sampled in the selection phase.

2.4 Trial-Based Heuristic Tree Search and Monte Carlo Tree Search

When considering MDPs with large state-action spaces, using tabular dynamic programming methods becomes infeasibly slow. One approach for handling large state-action spaces is to use *heuristic methods*, that consider a subset of the state-action space and utilise heuristic estimates for the value of states. Often these heuristic methods build a *search tree* from the initial state of the MDP.

Monte Carlo Tree Search (MCTS) refers to heuristic algorithms that build a search tree using *Monte Carlo Trials*, where nodes are added to the tree based off randomly sampled trajectories. Two advantages of using MCTS methods in the modern day are: that they allow for statistical analysis, which can provide *probabilistic guarantees* for the performance of the algorithm; and they offer interpretability, as the search tree can be inspected post-hoc to identify why the algorithm made certain decisions. TODO: consider adding interperobility refs when litrev'ed

MCTS trials are commonly presented using four stages, selection, expansion, simulation/heuristic and backup phases [6], which are depicted in Figure 2.6. To

avoid confusion with the above definition of MCTS, this thesis will refer to this specific form of MCTS as MCTS-1. Each of the four phases of MCTS-1 are described in more detail below:

1. *Selection phase* - a *search policy* is used to traverse the search tree from the root node until it reaches a state not contained in the search tree;

2. *Expansion phase* - a new child node is added to the search tree;

3. *Simulation/Heuristic phase* - the new child node's value is initialised using a *heuristic*, often taking the form of a Monte Carlo return, obtained using a *simulation* with a *rollout policy*;

4. *Backup phase* - the value from the heuristic or simulation is used to backup values through the path traversed in the selection phase.

Trial-Based Heuristic Tree Search (THTS) [14] generalises heuristic tree search methods used for planning, such as Trial-Based Real Time Dynamic Programming [4] and LAO* [11]. However, THTS differs slightly from how MCTS-1 is presented above. In a THTS trial, after the expansion/heurisitc phases are run, the trial "switches back to the selection phase and alternate between those phases", until the planning horizon is reached. This is similar to the trial depicted in blue in Figure 2.8.

`THTS++` [22] is introduced in Section 2.4.2, which is an open-source, parallelised extension of the THTS schema (including being able to implement MCTS-1 algorithms), and was built to facilitate the work in this thesis. Section 2.4.3 covers the Upper Confidence Bound applied to Trees (UCT) algorithm [16, 17], which is a commonly used MCTS algorithm, and Section 2.4.4 presents the Maximum ENtropy Tree Search (MENTS) algorithm [35], which is related to the algorithms introduced in Chapter 4 of this thesis.

2.4.1 Notation

To simplify notation in the presentation and analysis of THTS++ algorithms, this thesis assumes that states and state-action pairs have a one-to-one correspondance with nodes in the search tree. This assumption is purely to simplify notation for a clean presentation, and any results discussed in this thesis generalise to when this assumption does not hold.

Specifically, this allows the notation for value functions to avoid explicitly writing the timestep parameter, so that $V^\pi(s)$ can be written instead of $V^\pi(s;t)$.

2.4.2 Trial Based Heuristic Tree Search

This subsection introduces THTS++ , which extends the Trial-Based Heuristic Tree Search (THTS) schema [14], and aims to provide a modular library which can implement any MCTS algorithm. The changes made to the original schema allow algorithms following the four-phase MCTS-1 trials described previously to be encorporated. Additionally, the notion of a per-trial *context* is added, which will be used in Chapters 5 and 6 for multi-objective MCTS algorithms. Additionally, beyond the scope of this thesis, contexts allow THTS++ to implement algorithms for *partially-observable* environments, such as Partially Observable Monte Carlo Planning [27]. The remainder of this section will be presented in the context of planning for fully-observable MDPs, but THTS++ generalises to partially-observable environments if *observation-action histories* are used in place of states.

In THTS++ trees consist of *decision nodes* and *chance nodes*. Decision nodes sample actions that can be taken by the agent, and chance nodes sample *outcomes* (MDP states) that depend on the action taken and the transition function of the MDP. As such, each decision node has an associated *state* and each chance node has an associated *state-action pair*. Figure 2.7 shows how decision and chance nodes will be depicted as circles and diamonds in diagrams.

A search tree $\mathcal{T}$ is built using Monte Carlo *trials* and an overview of one trial in THTS++ is given in Figure 2.8. Each THTS++ trial is also split into four phases:

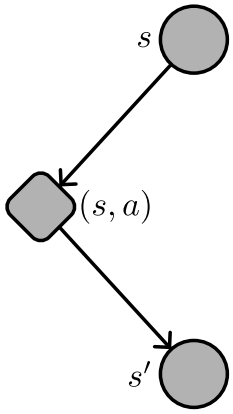


Figure 2.7: Tree diagrams notation, circles will be used to denote *decision nodes* that are associated with states, diamonds will be used to denote *chance nodes* that are associated with state-action pairs and arrows are used to denote parent/child relationships in the tree.

1. *Context sampling* - a *context* is sampled for the trial;
2. *Selection phase* - a trajectory is sampled using a *search policy*, any newly visited states (and state-action pairs) are added to $\mathcal{T}$ as decision nodes (and chance nodes);
3. *Heuristic phase* - any new leaf nodes added in the selection phase are initialised using a *heuristic function*;
4. *Backup phase* - value estimates in the search tree are updated, along the path sampled in the selection phase.

Note that the selection and expansion phases of MCTS-1 are encapsulated by the selection phase of `THTS++`.

Definition 2.4.1. *A* search tree $\mathcal{T}$ *is a subset of the state and state-action spaces, that is* $\mathcal{T} \subseteq \mathcal{S} \cup \mathcal{S} \times \mathcal{A}$*, where for each* $s \in \mathcal{T}$*, there exists some truncated trajectory* $\tau_{:h}$ *such that* $s_h = s$*, each* $s' \in \tau_{:h}$ *is also in the tree,* $s' \in \mathcal{T}$*, and each* $s', a' \in \tau$*:* h *is also in the tree,* $(s', a') \in \mathcal{T}$*.*

A *decision node* refers to any state that is in the search tree: $s \in \mathcal{T}$. A *chance node* refers to any state-action pair that is in the search tree: $(s, a) \in \mathcal{T}$. And a *node* is used to refer to any decision or chance node in the tree. Sometimes the

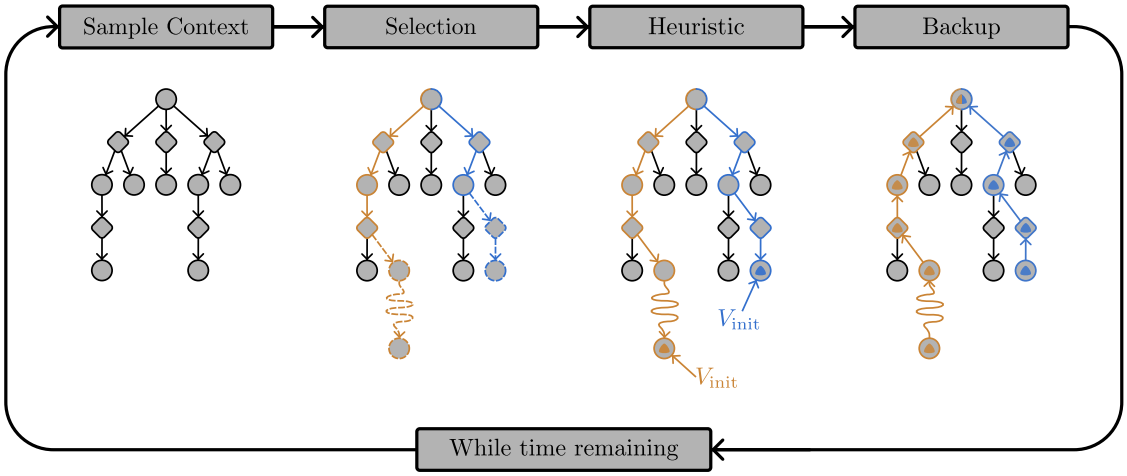


Figure 2.8: Overview of one trial of `THTS++`, where orange shows an example when `mcts_mode` is True, and blue shows an example when `mcts_mode` is False. From left to right: first a context is sampled, which stores any necessary per-trial state (not depicted) and the search tree at the beginning of the trial is shown; second shows the selection phase, where a trajectory is sampled and where dashed lines indicate any new nodes added; third shows that new leaf nodes are initialised using the $\hat{V}_{\text{init}}$ heuristic function; and finally on the right, shows the backup phase, where the arrows directions are changed to show that information is being propogated back up the tree to update the values in the nodes (triangles).

notation $\texttt{node}(s)$ and $\texttt{node}(s, a)$ will be used to make it clear that a node is being discussed, rather than a state or state-action pair.

$N(s)$ and $N(s, a)$ denote the number of times $\texttt{node}(s)$ and $\texttt{node}(s, a)$ have been visited, or equivalently, the number of times s and (s, a) appear in trajectories sampled in `THTS++` trials.

Each decision and chance node will generally store value estimates that are algorithm dependent. $\texttt{node}(s)\texttt{.V}$ is used to denote the set of values stored at node $\texttt{node}(s)$, and $\texttt{node}(s, a)\texttt{.Q}$ for the set of values stored at node $\texttt{node}(s, a)$.

Additionally, let $\texttt{node}(s)\texttt{.chldrn}$ be a mapping from actions to the chance nodes that are children of $\texttt{node}(s)$. Likewise, $\texttt{node}(s, a)\texttt{.chldrn}$ is a mapping from outcomes (successor states) to the decision nodes that are children of $\texttt{node}(s, a)$.

`THTS++` introduces the idea of `mcts_mode` into the THTS schema. When `mcts_mode` is True, a single decision node is added to the search tree on each trial, similarly to MCTS-1. When it is False, the trajectory is sampled until a terminal state or the planning horizon is reached. These two modes of operation

are depicted in blue and orange respectively in Figure 2.8.

Definition 2.4.2. *When running in* `mcts_mode`*, the trajectory sampled in the selection phase is truncated, and will end if a state not in the search tree $\mathcal{T}$ is reached, similarly to* MCTS-1 *[6]. When not running in* `mcts_mode`*, the trajectory is sampled until a terminal state is reached, or the planning horizon is reached, similarly to THTS [14].*

There are two main benefits to running an algorithm in `mcts_mode`. The first is that it uses significantly less memory, which can be a concern if using a large planning horizon H and the tree search is going to run for a long time. The second is that if `mcts_mode` is used with an informative heuristic function, then it allows the algorithm to avoid wasting resouces (time and memory) on parts of the search tree that are not promising.

In contrast, when no informative heuristic is available, a random simulation is often used in MCTS-1 algorithms. In such cases, where memory allows, running with `mcts_mode` set to False can be beneficial, because states that would have been visited in the simulation phase of MCTS-1 are added to the tree and avoids throwing away potentially useful information.

`THTS++` also introduces the notion of a *context* that is sampled for each trial. The context is passed to every subsequent function call in a trial of `THTS++`, and can be used to store temporary state. This context will go unused for the remainder of the chapter, but will be useful in Chapters 5 and 6 when *contextual tree search* is discussed.

Definition 2.4.3. *A* context *is an arbitrary key-value store that is used to store any relevant data that varies from trial to trial.*

To specify an algorithm in the `THTS++` schema, the following need to be specified:

Value Estimates: What value estimates will be stored at each decision and chance node. That is, $\texttt{node}(s)\texttt{.V}$ and $\texttt{node}(s, a)\texttt{.Q}$ need to be defined;

Search policy: A policy π_{search} used to sample a trajectory for the trial, which can use values in the current search tree $\mathcal{T}$, and values from the heuristic action function (below);

Outcome Sampler: A function `sample_outcome` that samples outcomes according to the environment, given a current state and action. In this thesis, this will always sample a state from the transition distribution of the MDP: $s' \sim p(\cdot|s, a)$;

Heuristic value function: A function $\hat{V}_{\text{init}}$ used as a heuristic to initialise values for new decision nodes added to the tree;

Heuristic action function: A function $\hat{Q}_{\text{init}}$ used as a heuristic for Q-values when a state-action pair is not in the current search tree;

Backup functions: Two functions `backup_v` and `backup_q` which updates the values in decision and chance nodes respectively. These functions can use values from their children, from the sampled trajectory and from the heuristic value function;

Context sampler: A function `sample_context` which creates a context key-value store, and samples any initial values to be stored in the context;

MCTS mode: A boolean (which will also be denoted `mcts_mode`) specifying if `THTS++` should operate in `mcts_mode`;

Planning horizon: A (problem dependent) planning horizon for the tree search $H_{\texttt{THTS++}}$, which is no longer than the horizon of the MDP, $H_{\texttt{THTS++}} \leq H$.

Figure 2.8 depicts a trial in `THTS++`, and psuedocode for running a trial is given in Listing 2.1. At the beginning of runnning a `THTS++` algorithm, the search tree is initialised to $\mathcal{T} \leftarrow \{s_0\}$. When the components detailed above are provided, the operation of a trial in `THTS++` is as follows:

- Firstly, a context is sampled using the `sample_context` function, which is available to be used by any other function in the trial.

- A trajectory is sampled $\tau_{:h} \sim \pi_{\text{search}}$ according to the search policy (which may use $\hat{Q}_{\text{init}}$ as necessary) and `sample_outcome`.
  - If running in `mcts_mode`, then $\tau_{:h}$ is such that $s_t \in \mathcal{T}$ for $t = 0, ..., h-1$ and $s_h \notin \mathcal{T}$, or $h = H_{\texttt{THTS++}}$, or s_h is terminal.
  - If not running in `mcts_mode`, then $h = H_{\texttt{THTS++}}$, or s_h is terminal.
- The search tree is updated to include any new nodes from the sampled trajectory, $\mathcal{T} \leftarrow \mathcal{T} \cup \tau_{:h}$.
- The heuristic value function is used to initialise the value of the new leaf node $\texttt{node}(s_h)\texttt{.V} \leftarrow \hat{V}_{\text{init}}(s_h)$.
- Finally, for the backup phase, the `backup_q` and `backup_v` functions are used to update the values of $\texttt{node}(s_t, a_t)\texttt{.Q}$ and $\texttt{node}(s_t)\texttt{.V}$ for $t = h-1, ..., 0$.

2.4.3 Upper Confidence Bounds Applied to Trees (UCT)

Upper Confidence Bound applied to Trees (UCT) [16, 17] is a commonly used tree search algorithm, which is based on the Upper Confidence Bound (UCB) algorithm covered in Section 2.1. UCT is a good example of a common paradigm in tree search algorithms for sequential decision making, where each node in the tree is tasked with making a single decision in the sequence, and so adapts methods from the MAB literature. More specifically, UCT can be viewed as running UCB on a non-stationary MAB at every decision node, where it is non-stationary because the rewards obtained depend on the decisions that children make. The remainder of this subsection will outline how UCT can be defined using the the `THTS++` schema given in Section 2.4.2.

In UCT `mcts_mode` is set to True. And at each decision and node of UCT a sample average $\bar{V}_{\text{UCT}}$ and $\bar{Q}_{\text{UCT}}$ is used for a value estimate. The search policy that UCT then follows is given by:

$$\pi_{\text{UCT}}(s) = \arg\max_{a \in \mathcal{A}} \bar{Q}_{\text{UCT}}(s, a) + b_{\text{UCT}} \sqrt{\frac{\log(N(s))}{N(s, a)}}, \tag{2.51}$$

```
def run_trial(search_tree: 𝒯,
              search_policy: π_search,
              heuristic_fn: V̂_init,
              heuristic_action_fn: Q̂_init,
              mcts_mode,
              planning_horizon: H_THTS++):
    # context sampling
    ctx = sample_context()
    # simulation phase
    τ_:h = sample_trajectory(𝒯, π_search, Q̂_init, mcts_mode, H_THTS++, ctx)
    𝒯 ← 𝒯 ∪ τ_:h
    # heuristic phase
    node(s_h).V ← V̂_init(s_h, ctx)
    # backup phase
    for i in {h−1, h−2, ..., 1, 0}:
        node(s_i, a_i).backup_q(node(s_i, a_i).chldrn, τ_:h, V̂_init(s_h), ctx)
        node(s_i).backup_v(node(s_i).chldrn, τ_:h, V̂_init(s_h), ctx)

def sample_trajectory(search_tree: 𝒯,
                      search_policy: π_search,
                      heuristic_action_fn: Q̂_init,
                      mcts_mode,
                      planning_horizon: H_THTS++,
                      ctx):
    i = 0
    while ((not mcts_mode or s_i ∈ 𝒯)
           and (i < H_THTS++)
           and (not terminal(s_i))):
        a_i ~ π_search(·|s_i, ctx)
        r_i ← R(s_i, a_i)
        s_i+1 ← sample_outcome(s_i, a_i, ctx)
        i += 1
    return (s_0, a_0, r_0, s_1, ..., s_i-1, a_i-1, r_i-1, s_i)

def sample_outcome(s, a, ctx):
    s' ~ p(·|s, a)
    return s'
```

Listing 2.1: Psuedocode for running a trial in THTS++.

where b_{UCT} is a *bias* parameter that controls the amount of exploration UCT will perform. In Equation (2.51), when $N(s,a) = 0$ there is a division by zero, which is taken as ∞, and ties are broken uniformly randomly. Note that like UCB (Section 2.1), this results in every action being taken once to obtain an initial value estimate, and as such, setting $\hat{Q}_{\text{init}}$ is unnecessary for UCT.

TODO: Added this quickly when writing ch4 Because the best setting of the bias parameter will depend on the scale of the reward function, Keller and Helmert

[14] suggest setting the bias parameter b_{UCT} adaptively to $\bar{V}_{\text{UCT}}(s)$, which is defined in Equation (2.55) below. In this thesis UCT can be considered to have a boolean parameter `adaptive_bias`, where an additional factor of $\bar{V}_{\text{UCT}}(s)$ is added to the confidence interval term as follows:

$$\pi_{\text{UCT}}(s) = \arg\max_{a \in \mathcal{A}} \bar{Q}_{\text{UCT}}(s,a) + b_{\text{UCT}} \bar{V}_{\text{UCT}}(s) \sqrt{\frac{\log(N(s))}{N(s,a)}}, \tag{2.52}$$

There are two common approaches to implementing $\hat{V}_{\text{init}}$ in UCT: the first consisting of using a function approximation $\tilde{V}$ and setting $\hat{V}_{\text{init}} = \tilde{V}$ [28, 29, 20] TODO: find another paper or two that is just an application of alphazero?, where $\hat{V}$ aims to approximate the optimal value function V^*; the second approach consists of using a *rollout policy* [6, 9, 13, 8]. When a rollout policy π_{rollout} is used, a Monte Carlo estimate $\hat{V}^{\pi_{\text{rollout}}}$ is used to estimate the value function $V^{\pi_{\text{rollout}}}$ and is used for $\hat{V}_{\text{init}}$.

Let $\tau_{:h} \sim \pi_{\text{UCT}}$, be the trajectory sampled in the selection phase of UCT, meaning that a decision node for s_h is added to the search tree, and needs to be initialised with $\hat{V}_{\text{init}}$. When using a rollout policy, the truncated trajectory is completed using the rollout policy, $\tau_{h:H} \sim \pi_{\text{rollout}}$, to provide the Monte Carlo estimate at s_h:

$$\hat{V}^{\pi_{\text{rollout}}}(s_h) = \sum_{i=h}^{H-1} r_i. \tag{2.53}$$

If no informative policy is available to be used for π_{rollout}, then uniformly random policy is often used [6, 1].

In UCT the backups `backup_v` and `backup_q` update the (sample average) value estimates. Letting heuristic value for the leaf node be $\tilde{r} = \hat{V}_{\text{init}}(s_h)$, they are computed as follows:

$$\bar{Q}_{\text{UCT}}(s_t,a_t) \leftarrow \frac{1}{N(s_t,a_t)} \left((N(s_t,a_t)-1)\bar{Q}_{\text{UCT}}(s_t,a_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i \right) \tag{2.54}$$

$$\bar{V}_{\text{UCT}}(s_t) \leftarrow \frac{1}{N(s_t)} \left((N(s_t)-1)\bar{V}_{\text{UCT}}(s_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i \right) \tag{2.55}$$

2.4.4 Maximum Entropy Tree Search

Maximum ENtropy Tree Search (MENTS) [35], in contrast to UCT, focuses on the maximum-entropy objective, and uses soft Bellman backups to update its value estimates. In its original presentation `mcts_mode` is set to True, and it uses the soft value estimates $\hat{V}_{\text{MENTS}}$ and $\hat{Q}_{\text{MENTS}}$. The MENTS search policy is given by:

$$\pi_{\text{MENTS}}(a|s) = (1 - \lambda(s, \epsilon_{\text{MENTS}}))\rho_{\text{MENTS}}{}^{k}(a|s) + \frac{\lambda(s, \epsilon_{\text{MENTS}})}{|\mathcal{A}|} \tag{2.56}$$

$$\rho_{\text{MENTS}}(a|s) = \exp\left(\frac{1}{\alpha_{\text{MENTS}}}\left(\hat{Q}_{\text{MENTS}}(s,a) - \hat{V}_{\text{MENTS}}(s)\right)\right), \tag{2.57}$$

$$\lambda(s,x) = \min\left(1, \frac{x}{\log(e + N(s))}\right). \tag{2.58}$$

where $\epsilon \in (0, \infty)$ is an exploration parameter, and α_{MENTS} is the temperature paramter used in MENTS for the maximum entropy objective (the coefficient of the entropy term in Equation (2.40)).

In MENTS the backups `backup_v` and `backup_q` update the soft value estimates and are updated using soft Bellman backups (Equations (2.49) and (2.50)) as follows:

$$\hat{Q}_{\text{MENTS}}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \text{Succ}(s,a)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}_{\text{MENTS}}(s')\right), \tag{2.59}$$

$$\hat{V}_{\text{MENTS}}(s_t) \leftarrow \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha} \hat{Q}_{\text{MENTS}}(s_t, a)\right). \tag{2.60}$$

In [35], the heuristic value function left as an arbitrary evaluation function, but is set using a function approximation $\hat{V}_{\text{init}} = \tilde{V}$ in experiments. The heuristic action function is set to zero, $\hat{Q}_{\text{init}}(s,a) = 0$, but they also suggest that if *policy network* $\tilde{\pi}$ is available, then the heuristic action function can alternatively be set to $\hat{Q}_{\text{init}}(s,a) = \log \tilde{\pi}(s|a)$.

2.5 Multi-Objective Reinforcement Learning

This thesis follows a utility based approach to multi-objective reinforcement learning similar to the review of Hayes et al. [12]. This work will specifically consider *linear utility* functions and the *decision support scenario* which is depicted in Figure 2.9. In the decision support scenario, an algorithm computes a *solution set* consisting

Figure 2.9: The decision-support scenario for multi-objective reinforcement learning [12].

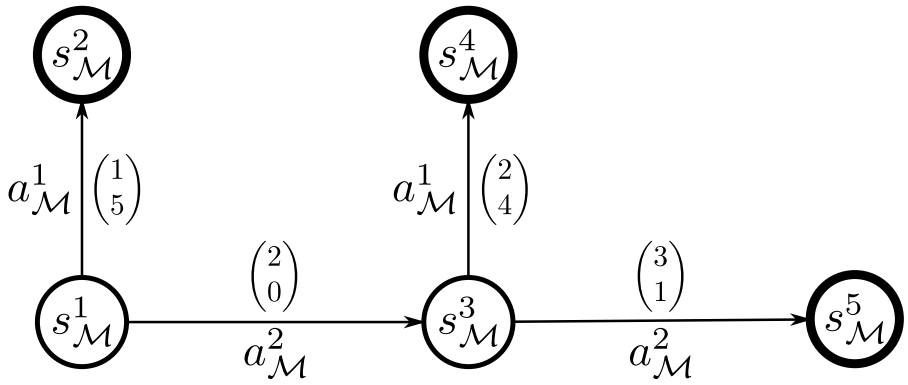


Figure 2.10: An example MOMDP $\boldsymbol{\mathcal{M}}$, where $s_0 = s^1_{\mathcal{M}}$, the finite horizon is $H = 2$ and terminal states are marked with thicker borders. This particular MOMDP is deterministic, so all transition probabilities are one.

of multiple *possibly optimal* solutions, from which a user then picks their most preferred option to be used. This scenario is useful when a user's preferences over the multiple objectives is unknown or difficult to specify in advance.

This section defines the multi-objective counterparts to various definitions given in Section 2.2 and 2.3. Outside of this section, the prefix "multi-objective" may be dropped where it should be clear from context, however bold typeface will consistently be used to denote any vector variables or functions. In Section 2.5.1, a multi-objective extension of Value Iteration is given.

To specify problems with multiple objectives, the reward function of an MDP changed to give a vector of rewards, rather than a scalar reward:

Definition 2.5.1. *A* Multi-Objective Markov Decision Process *(MOMDP) is a tuple* $\boldsymbol{\mathcal{M}} = (\mathcal{S}, s_0, \mathcal{A}, \mathbf{R}, p, H)$*, where* $\mathcal{S}$ *is a set of states,* $s_0 \in \mathcal{S}$ *is an initial state,* $\mathcal{A}$ *is a set of actions,* $\mathbf{R}(s, a)$ *is a vector reward function* $\mathcal{S} \times \mathcal{A} \to \mathbb{R}^D$*, where* D *is the dimension of the rewards and the MOMDP,* $p(s'|s, a)$ *is a next state transition distribution* $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *and* $H \in \mathbb{N}$ *is a finite-horizon time bound.*

Now multi-objective trajectories are defined:

Definition 2.5.2. *A* multi-objective trajectory τ*, is a sequence of state, action and vector rewards, that is induced by a policy* π *and MOMDP* $\boldsymbol{\mathcal{M}}$ *pair. Let the trajectory be* $\tau = (s_0, a_0, \mathbf{r}_0, s_1, a_1, \mathbf{r}_1, ..., s_{H-1}, a_{H-1}, \mathbf{r}_{H-1}, s_H)$*, where* $a_t \sim \pi(\cdot|s_t)$*,* $\mathbf{r}_t = \mathbf{R}(s_t, a_t)$ *and* $s_{t+1} \sim p(\cdot|s_t, a_t)$*.*

The notations used for single-objective trajectories (Definition 2.2.5) will also be used for multi-objective trajectories too. Such as, $\tau \sim \pi$ *for sampling trajectories using policies, and* $\tau_{i:j}$ *for truncated trajectories.*

Similarly, multi-objective variants of the (Q-)value of a policy are defined:

Definition 2.5.3. *The* multi-objective value *of a policy* π *from state* s *at time* t *is:*

$$\mathbf{V}^{\pi}(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} \mathbf{r}_t \middle| s_t = s\right]. \tag{2.61}$$

The multi-objective Q-value *of a policy* π*, from state* s*, with action* a*, at time* t *is:*

$$\mathbf{Q}^{\pi}(s,a;t) = \mathbf{R}(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}[\mathbf{V}^{\pi}(s';t+1)]. \tag{2.62}$$

In the corresponding point of the single-objective reinforcement learning section (Section 2.3), the the optimal (Q-)value functions and the objective of single-objective reinforcement learning were defined. However, in a multi-objective setting there is no longer a *total ordering* over values, and so there maybe be multiple vectors that could be "optimal". For example, consider two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ which are such that $u_1 > v_1$ and $u_2 < v_2$. To resolve this issue, a *utility function* or *scalarisation function* is used to map multi-objective values to scalar values.

Definition 2.5.4. *The* (D-dimensional) Simplex *consists of the set of* D*-dimensional vectors, whose entries are non-negative and sum to one. More formally, the* D *dimensional simplex is*

$$\Delta^D = \{\mathbf{w} \in \mathbb{R}^D | w_i > 0, \sum_i w_i = 1\}. \tag{2.63}$$

The elements of the D*-dimensional Simplex will be referred to as* weight vectors *(or just* weights*) in this thesis, as they will be used to specify preferences over the* D *dimensions of the reward function.*

Definition 2.5.5. *A* utility function *(or* scalarisation function*)* $u : \mathbb{R}^D \times \Delta^D \to \mathbb{R}$ *is used to map from a multi-objective value* $\mathbf{v} \in \mathbb{R}^D$ *and a weighting over the objectives* $\mathbf{w} \in \Delta^D$ *to a scalar value. That is, according to the utility function* $u(\cdot; \mathbf{w})$ *the multi-objective value* $\mathbf{v}$ *is mapped to the scalar value* $u(\mathbf{v}; \mathbf{w})$*.*

Of particular interest in this thesis is the *linear utility function* where the scalar value takes the form of a dot-product:

Definition 2.5.6. *The* linear utility function u_{lin} *is the utility function defined by:*

$$u_{\text{lin}}(\mathbf{v}; \mathbf{w}) = \mathbf{w}^\top \mathbf{v}. \tag{2.64}$$

Equipped with a utility function and a weight vector any set of multi-objective values can be mapped to scalars and ordered. A policy is *possibly optimal*, or *undominated*, with respect to utility function u if it achieves a maximal utility for some weight $w \in \Delta^D$. In contrast, a policy is *dominated* if for every weight there is another policy that achieves a better utility.

Let Π be the set of all possible policies for a given MOMDP. When constructing a solution set, a subset of Π, it makes sense that it should not contain any dominated policies. This leads to the first notion of a solution set, called an *undominated set*, consisting of all undominated policies:

Definition 2.5.7. *The* undominated set *of policies* $U(\Pi; u) \subseteq \Pi$*, with respect to a utility function* u*, is the set of policies for which there is a weight vector* $\mathbf{w} \in \Delta^D$ *where the utility (scalarised value) is maximised:*

$$U(\Pi; u) = \left\{ \pi \in \Pi \;\middle|\; \exists \mathbf{w} \in \Delta^D . \forall \pi' \in \Pi : u(\mathbf{V}^\pi(s_0; 0); \mathbf{w}) \geq u(\mathbf{V}^{\pi'}(s_0; 0); \mathbf{w}) \right\}. \tag{2.65}$$

In particular, the convex hull *of policies* $CH(\Pi)$ *is the undominated set with respect to the linear utility function* u_{lin}*. That is* $CH(\Pi) = U(\Pi; u_{\text{lin}})$*.*

Question: How do you feel about removing the timestep in the value functions in this section too? So it can be $\mathbf{V}^\pi(s)$ rather than $\mathbf{V}^\pi(s; t)$? I think it

would make some definitions like undominated sets a bit cleaner. Alternatively, maybe I should write it $u_{\mathbf{w}}$.

When computing solutions sets, it is often useful to first compute the multi-objective values that could be obtained, and then later use the data structures used by the algorithm to read out the selected policy. This thesis will refer to the set of values obtained by a set of policies as a *value set*:

Definition 2.5.8. *The* (multi-objective) value set *with respect to a set of policies* $\Pi' \subseteq \Pi$ *is defined as* $\boldsymbol{\mathcal{V}}\mathbf{als}(\Pi') = \{\mathbf{V}^{\pi}(s_0; 0) | \pi \in \Pi'\}$.

Undominated sets often have an infinite cardinality, and as such are infeasible to compute. However, in undominated sets there are often many reduntant policies that obtain the same scalarised values. Instead of computing an undominated set, it is more feasible to compute a *coverage sets* which contain at least one policy that maximises the scalarised value given any weight vector $\mathbf{w}$:

Definition 2.5.9. *A set* $CS(\Pi; u) \subseteq U(\Pi)$*, is a* coverage set *with respect to a utility function* u*, if for every weight vector* $\mathbf{w} \in \Delta^D$*, there is a policy* $\pi \in CS(\Pi; u)$ *that maximises the value of* $u(\cdot; \mathbf{w})$*. That is, for* $CS(\Pi; u)$ *to be a coverage set, the following statement must be true:*

$$\forall \mathbf{w} \in \Delta^D . \exists \pi \in CS(\Pi; u) . \forall \pi' \in \Pi : u(\mathbf{V}^{\pi}(s_0; 0); \mathbf{w}) \geq u(\mathbf{V}^{\pi'}(s_0; 0); \mathbf{w}). \quad (2.66)$$

Again, in particular, any set $CCS(\Pi)$ *is a* convex coverage set *if it is a coverage set with respect to the linear utility function* u_{lin}.

Note that it is still possible for there to be multiple coverage sets. Algorithms that compute convex coverage sets typically compute a unique minimal convex coverage set that contains no redundant policies, that is, each policy has some weight for which it uniquely gives an optimal solution in the set.

Definition 2.5.10. *A set* $CCS_{\text{min}}(\Pi) \subseteq CH(\Pi)$ *is minimal if the following holds*

$$\forall \pi \in CCS_{\text{min}}(\Pi) . \exists \mathbf{w} \in \Delta^D . \forall \pi' \in CCS_{\text{min}}(\Pi) - \{\pi\} . u_{\text{lin}}(\mathbf{V}^{\pi}(s_0; 0); \mathbf{w}) > u_{\text{lin}}(\mathbf{V}^{\pi'}(s_0; 0); \mathbf{w}). \quad (2.67)$$

Now the geometry of convex coverage sets is considered, to see how something like $\boldsymbol{\mathcal{V}}\textbf{als}(CCS_{\text{min}}(\Pi))$ can be computed. Firstly, any multi-objective values that obtain the same linear utility will lie on a hyperplane, whose normal is the weight vector used in the linear utility function (see (a) in Figure 2.11). As a result, the values of the convex hull, $\boldsymbol{\mathcal{V}}\textbf{als}(CH(\Pi))$ lie on a *geometric (partial) convex hull* (see (b) in Figure 2.11). Moreover, any points in $\boldsymbol{\mathcal{V}}\textbf{als}(CH(\Pi))$ that lie on the edge of the geometric convex hull are redundant, as the values corresponding to the neighbouring vertices of geometric convex hull will always achieve the same or greater utility (see (c) in Figure 2.11).

Because of the ambiguity between the convex hull of policies, and the geometric convex hulls of values, this thesis will refer to any value set that forms a geometric convex hull as a *convex hull value set* (CHVS) and will refer to the set $\boldsymbol{\mathcal{V}}\textbf{als}(CCS_{\text{min}}(\Pi))$ as the *optimal convex hull value set*. Question: should I make this paragraph a definition to emphasise it a bit more?

Given a value set, the `cvx_prune` operation removes any vectors that are dominated or redundant. That is, for every vector that remains after the pruning operation, there is some weight for which it *uniquely* gives the maximal utility for. Given set of vectors $\boldsymbol{\mathcal{V}}$ it is defined as:

$$\texttt{cvx\_prune}(\boldsymbol{\mathcal{V}}) = \{\mathbf{v} \in \boldsymbol{\mathcal{V}} | \exists \mathbf{w} \in \Delta^D . \forall \mathbf{v}' \in \boldsymbol{\mathcal{V}} - \{\mathbf{v}\} . \mathbf{w}^\top \mathbf{v} > \mathbf{w}^\top \mathbf{v}'\}. \tag{2.68}$$

Note that the `cvx_prune` operator computes the values of a minimal convex coverage set, that is, for some set of policies Π' the `cvx_prune` operation satisfies $\texttt{cvx\_prune}(\boldsymbol{\mathcal{V}}\textbf{als}(\Pi')) = \boldsymbol{\mathcal{V}}\textbf{als}(CCS_{\text{min}}(\Pi'))$. `cvx_prune` can be implemented using *linear programming* [24], and an example of its operation on a set of vectors can be seen in (b) of Figure 2.11, where `cvx_prune` would prune all points but the blue circles.

TODO: Not sure where to put this PF stuff, cant see where to put it in to make it flow well. Also proof read

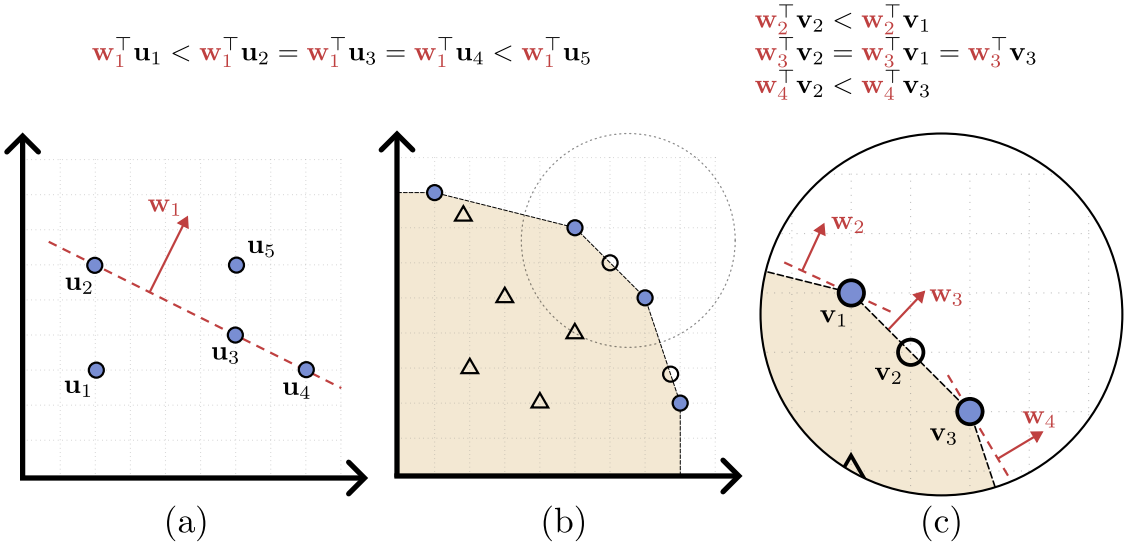


Figure 2.11: The geometry of convex coverage sets, shown with $D = 2$. In each images the points depicted correspond to a value set $\boldsymbol{\mathcal{V}}\mathbf{als}(\Pi')$, for some set of policies Π'. (a) demonstrates that values ($\mathbf{u}_2$,$\mathbf{u}_3$,$\mathbf{u}_4$) obtaining the same utility lie on a hyperplane. Moving in the direction of the weight vector increases the utility ($\mathbf{u}_5$) and likewise moving in the opposite direction decreases utility ($\mathbf{u}_1$); (b) circles correspond to values that are on the convex hull, so form the set $\boldsymbol{\mathcal{V}}\mathbf{als}(CH(\Pi'))$, the blue circles form the minimal convex coverage set, $\boldsymbol{\mathcal{V}}\mathbf{als}(CCS_{\text{min}}(\Pi'))$, and triangles are not part of the convex hull. All of the unfilled shapes would be pruned by `cvx_prune`(Equation (2.68)); (c) shows a magnification of (b) with additional labels, it shows that the value $\mathbf{v}_2$ is redundant because one of $\mathbf{v}_1$ or $\mathbf{v}_3$ will achieve an equal or greater utility. Additionally, is shows $\mathbf{w}_2$ and $\mathbf{w}_4$ which are weights that the values $\mathbf{v}_1$ and $\mathbf{v}_3$ uniquely maximise the utility within $\boldsymbol{\mathcal{V}}\mathbf{als}(\Pi')$

Another commonly used solution set is the *Pareto front*, which is defined using a *strict partial ordering* over vectors, known as *Pareto domination*, as opposed to using utility. An example is shown in Figure 2.12.

Definition 2.5.11. *The* Pareto front $PF(\Pi) \subseteq \Pi$ *is defined as:*

$$PF(\Pi) = \{\pi \in \Pi | \nexists \pi' \in \Pi . \mathbf{V}^{\pi'}(s_0;0) \succ_P \mathbf{V}^{\pi}(s_0;0)\}, \tag{2.69}$$

where the Pareto domination *relation is defined as:*

$$\mathbf{v} \succ_P \mathbf{v}' \iff (\forall i . v_i \geq v'_i) \wedge (\exists j . v_j > v'_j). \tag{2.70}$$

The Pareto front can be related to the previously defined solution sets if *mixture policies* are allowed, where a mixture policy between π and π' will follow each

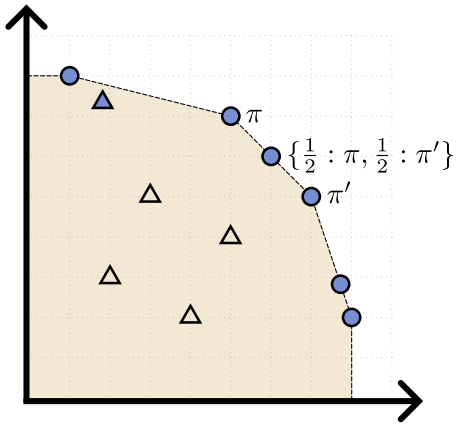


Figure 2.12: An example Pareto front, where the circles correspond to a convex coverage set, and the filled blue points denote the Pareto front. Two points are marked with their corresponding policies π and π', and a mixture policy is shown, denoted $\{\frac{1}{2}:\pi,\frac{1}{2}:\pi'\}$ that follows each of π and π' with probability $\frac{1}{2}$.

policy with some probability. Specifically, if Π is closed with respect to mixture policies, then the Pareto front is a convex coverage set.

2.5.1 Convex Hull Value Iteration

TODO: Should we also be defining undominated set/coverage sets for a given $(s;t)$? As in just add them as a param? As kind of using them in this sect as if we did

Convex Hull Value Iteration (CHVI) [3] is a tabular dynamic programming algorithm similar to Value Iteration (Section 2.3) that will compute optimal convex hull value sets. In CHVI the value estimates of Value Iteration are replaced by CHVSs, which are estimates of the optimal CHVS. These estimates will be denoted $\hat{\boldsymbol{\mathcal{V}}}_{\text{CHVI}}(s;t)$ and $\hat{\boldsymbol{\mathcal{Q}}}_{\text{CHVI}}(s,a;t)$.

Firstly, to define a multi-objective version of Value Iteration, an arithmetic over vector sets needs to be defined. An example of the following arithmetic is given in Figure 2.13 Given the vector sets $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$, define multiplication by a scalar s, addition with a vector $\mathbf{x}$ and addition between sets as follows:

$$\mathbf{x} + s\boldsymbol{\mathcal{V}} = \{\mathbf{x} + s\mathbf{v} | \mathbf{v} \in \boldsymbol{\mathcal{V}}\} \tag{2.71}$$

$$\boldsymbol{\mathcal{U}} + \boldsymbol{\mathcal{V}} = \{\mathbf{u} + \mathbf{v} | \mathbf{u} \in \boldsymbol{\mathcal{U}}, \mathbf{v} \in \boldsymbol{\mathcal{V}}\}. \tag{2.72}$$

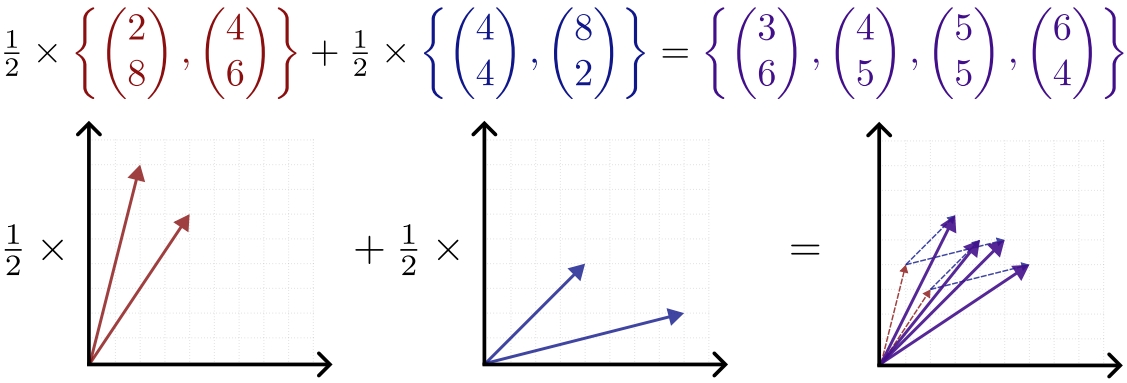


Figure 2.13: An example of arithmetic over vector sets. Two vector sets are shown in red and blue, which are multiplied by $\frac{1}{2}$ using Equation (2.71), and then added using Equation (2.72). On the right the resulting vector set is shown in purple, which are the sum of appropriately scaled red and blue vectors.

Now to define the multi-objective Bellman backups used in CHVI, let $\hat{\boldsymbol{\mathcal{V}}}^0_{\text{CHVI}}(s;t) = \{\mathbf{0}\}$, where $\mathbf{0} = (0, ..., 0) \in \mathbb{R}^D$. The CHVI backups are then:

$$\hat{\boldsymbol{\mathcal{V}}}^{k+1}_{\text{CHVI}}(s;t) = \texttt{cvx\_prune}\left(\bigcup_{a\in\mathcal{A}} \hat{\boldsymbol{\mathcal{Q}}}^{k+1}_{\text{CHVI}}(s,a;t)\right), \tag{2.73}$$

$$\hat{\boldsymbol{\mathcal{Q}}}^{k+1}_{\text{CHVI}}(s,a;t) = \mathbb{E}_{s'\sim p(\cdot|s,a)}[\mathbf{R}(s,a) + \hat{\boldsymbol{\mathcal{V}}}^{k}_{\text{CHVI}}(s';t+1)]. \tag{2.74}$$

This again parallels the Bellman backups use in single-objective Value Iteration, where the max operation is replaced by the `cvx_prune` operation over the set of achievable values from the current state s: $\bigcup_{a\in\mathcal{A}} \hat{\boldsymbol{\mathcal{Q}}}^{k+1}_{\text{CHVI}}(s,a;t)$.

Once the algorithm has terminated, and a weight $\mathbf{w}$ is provided, a policy can be extracted by computing scalar Q-values from the CHVSs [3]:

$$Q_{\mathbf{w}}(s,a;t) = \max_{\mathbf{q}\in\hat{\boldsymbol{\mathcal{Q}}}_{\text{CHVI}}(s,a;t)} \mathbf{w}^\top\mathbf{q}, \tag{2.75}$$

$$\pi_{\text{CHVI}}(s;t,\mathbf{w}) = \arg\max_{a\in\mathcal{A}} Q_{\mathbf{w}}(s,a;t). \tag{2.76}$$

If the user would rather select a particular value from the CHVS, say $\mathbf{v} \in \hat{\boldsymbol{\mathcal{V}}}_{\text{CHVI}}(s_0;0)$, rather than provide a weight, one can be computed. Let $\mathbf{v}_\perp$ be a *reference point*, defined at each index as:

$$(v_\perp)_i = \min\{v'_i | \mathbf{v}' \in \hat{\boldsymbol{\mathcal{V}}}_{\text{CHVI}}(s_0;0)\}. \tag{2.77}$$

The vector that runs from $\mathbf{v}_\perp$ to $\mathbf{v}$ provides an appropriate weighting over objectives for which $\mathbf{v}$ will produce the largest utility out of $\hat{\boldsymbol{\mathcal{V}}}(s_0;0)$, as is demonstrated in Figure 2.14. Hence, to extract a policy that achieves the value $\mathbf{v}$

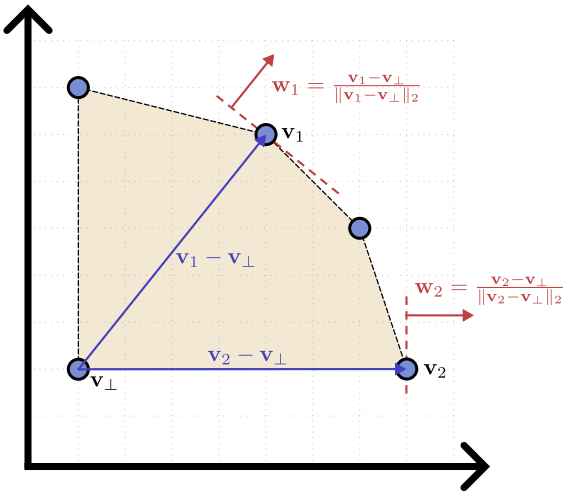


Figure 2.14: A visualisation, reusing the convex hull from Figure 2.11, demonstrating how to compute a weight vector that can be used to extract a specific value from a vector set. The hyperplane running through $\mathbf{v}_i$ with normal $\mathbf{w}_i = \frac{\mathbf{v}_i - \mathbf{v}_\perp}{\|\mathbf{v}_i - \mathbf{v}_\perp\|_2}$ is tangential to the geometric convex hull. Hence $\mathbf{v}_i$ is the optimal value for the weight $\mathbf{w}_i$ and following the CHVI policy in Equation (2.76) will give the desired value $\mathbf{v}_i$.

the weight $\mathbf{w}$ should be set to:

$$\mathbf{w} = \frac{\mathbf{v} - \mathbf{v}_\perp}{\|\mathbf{v} - \mathbf{v}_\perp\|_2}, \tag{2.78}$$

and then follow $\pi_{\text{CHVI}}(s; t, \mathbf{w})$ as defined in Equation (2.76).

2.6 Sampling From Catagorical Distributions

Question: Maybe heres a good time to ask about some notation stuff. So I used m in MABs section, and I reuse it here. Is that ok? I was trying to keep as much notation unique as possible, but noticed I've reused a couple variables. I was hoping its ok because I'm using them "locally".

Some of the work in this thesis will involve sampling from catagorical distributions, and sometimes it will be helpful to do so efficiently, as in Chapter 4. Let $f : \{C_1, ..., C_m\} \to [0, 1]$ be the probability mass function of a catagorical distribution with m categories. Suppose that we want to sample $c \sim f$. A naive method to sample from f will take $O(m)$ time, where a continuous uniform random value is sampled, $u \sim \text{Uniform}(0, 1)$, and f is linearly scanned until a probability mass of u has been passed. See Listing 2.2 for psuedocode of the naive method.

```
def sample_catagorical(f):
    threshold ~ Uniform(0, 1)
    i = 0
    accumulated_mass = 0
    while (accumulated_mass < threshold):
        i += 1
        accumulated_mass += f(i)
    return i
```

Listing 2.2: Psuedocode for naively sampling from a catagorical distribution.

```
def sample_from_alias_table(alias_table):
    index ~ Uniform({1, ..., m})
    (c_0, c_1, thrsh) = alias_table[index]
    u ~ Uniform(0, 1)
    if (u <= thrsh):
        return c_0
    return c_1
```

Listing 2.3: Psuedocode for sampling from an alias table.

However, the *alias method* [33, 32] can instead be used to sample from a catagorical distribution. The alias method uses $O(m)$ preprocessing time to construct an *alias table*, and after can sample from f, using the table, in $O(1)$ time. An alias table consists of m entries, accounting for an mth of the probability mass of f. Each entry takes the form $(c_0, c_1, \texttt{thrsh})$, where $c_0, c_1 \in \{C_1, ..., C_m\}$ are categories and $\texttt{thrsh} \in [0, 1]$ gives the ratio of the $\frac{1}{m}$ probability mass to assign to c_0 and c_1. Sampling from the alias table can be performed by sampling two random numbers, one to index into the table and one to compare against `thrsh`, from which $c \sim f$ can be returned by looking it up in the table. Listing 2.3 gives psuedocode for sampling from an alias table, and Figure 2.15 visualises the construction and sampling of an alias table.

Vose [32] proves that an alias table can always be constructed from an arbitrary catagorical distribution, such that the probability of sampling any catagory from the alias table is identical to the probability of sampling it from the original probability mass function.

Psuedocode for constructing an alias table is given in Listing 2.4, following [33, 32]. An alias table can be constructed by keeping two sets of categories, one set for "big" categories that currently have more than $\frac{1}{m}$ mass to be added to the

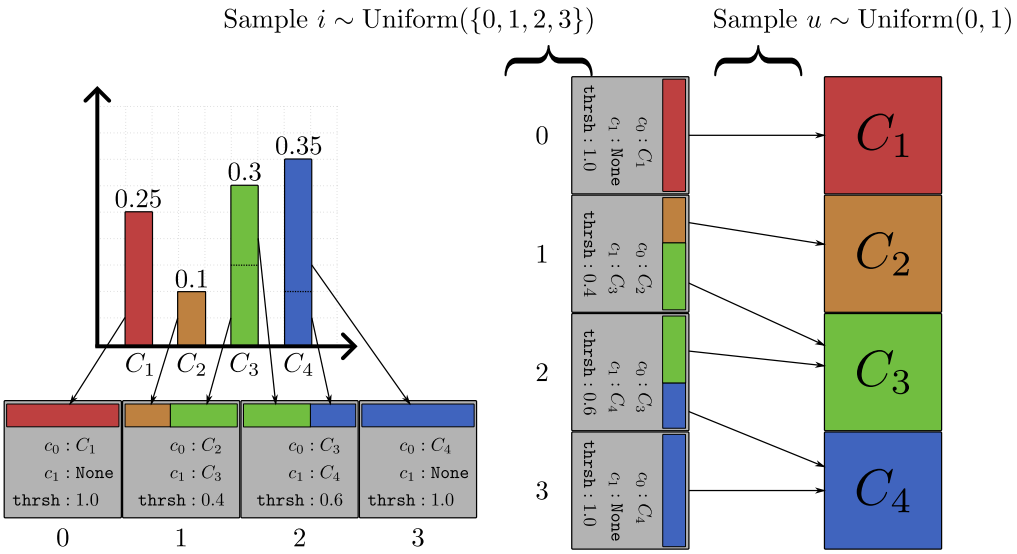


Figure 2.15: Example of building and sampling from an alias table, on a distribution with four categories. Left: shows how the probability mass is split into the four entries of the table. Right: shows how two random variables can be used to sample from the table in $O(1)$ time, where i is used to index into the table, and u is used to compare with `thrsh` and decide which of the two arrows from the entry to follow.

table, and one for "small" catagories that have at most $\frac{1}{m}$ mass left to be added to the table. To create a new entry in the table, take one category from the small set, and take a category from the big set to complete the remainder of the $\frac{1}{m}$ mass that the small set didn't fill. After creating a new entry, move the big category to the little set if it no longer has more than $\frac{1}{m}$ mass to be added.

```
def build_alias_table(f):
    # lists of categories and mass remaining
    small = []
    big = []

    # fill big and small lists
    for c in f:
        if f(c) > 1/m:
            big.append((c, f(c)))
        else:
            small.append((c, f(c)))

    # construct alias table
    alias_table = []
    while len(small) > 0:
        # create entry
        c_0, c0_mass_remaining = small.pop()
        if mass_remaining == 1/m:
            alias_table.append((c_0, None, 1.0))
            continue
        c_1, c1_mass_remaining = big.pop()

        # add entry to table (accounting for 1/mth of total mass)
        thrsh = m * c0_mass_remaining
        alias_table.append((c_0, c_1, thrsh))

        # put c1 back in lists appropriately
        c1_mass_remaining -= c0_mass_remaining
        if c1_mass_remaining > 1/m:
            big.append((c_1, c1_mass_remaining))
        else:
            small.append((c_1, c1_mass_remaining))

    return alias_table
```

Listing 2.4: Psuedocode for constructing an alias table.

3 Literature Review

Contents

TODO: currently this is a copy and paste of what I originally wrote for background chapter 2. Deleted parts which are irrelevant for litreview here (and vice versa for the background section).

TODO: I'm also going to use this as a space to paste papers I should write about as they come up while writing later chapters

3.1 Multi-Armed Bandits

TODO: Maybe dont need to cover this in litrev, but should talk about exploring bandits, UCT and contextual bandits either in background or in litrev

TODO: linUCB for contextual bandits think this is the linucb paper: https://arxiv.org/pdf/1003

Designing multi-objective multi-armed bandits algorithms: a study - Madalina M. Drugan and Ann Nowe

TODO: MAB book

Talk about Exp3?

3.2 Reinforcement Learning

TODO: Intro should say that look at Sutton and Barto and something else for deep RL, for a more complete overview. Here we will just discuss papers that consider entropy in their work, as thats the most relevant part for this thesis.

TODO: list

- Talk about entropy and some of that work (probably a subsection)

TODO: In the entropy bit talk add this, removed from ch2: Note that there are other forms of entropy, such as relative and Tsallis entropy, which can be used in place of Shannon entropy (TODO cite). For the work considered in this thesis, the other forms of entropy can be used by replacing the definition of $\mathcal{H}$ by the relevant definition.

TODO: talk about some deep learning methods here

TODO: Talk about entropy in (deep) RL methods too. Say that it is primarily introduced primarily for an exploration benifit. A3C "prevent converging to deterministic suboptimal behaviour". While there is some additional. Intro in the soft Q learning provides some alternative motivations, "In some cases, we might actually prefer to learn stochastic behaviors. In this paper, we explore two potential reasons for this: exploration in the presence of multimodal objectives, and compositionality attained via pretraining. Otherbenefits include robustness in the face of uncertain dynamics (Ziebart, 2010), imitation learning (Ziebart et al., 2008), and improved convergence and computational properties (Gu et al., 2016a). Multi-modality also has application in real robot tasks, as demonstrated in (Daniel et al., 2012). However, in order to learn such policies, we must define an objective that promotes stochasticity.". ANd also talks about adversarital pertobations. So

basically: good for meta learning/pretraining/foundation models, good for robustness against uncertain dynamics. BASICALLY, HERE we want to talk about the other reasons why we might want to use maximum entropy.

3.3 Trial-Based Heuristic Tree Search and Monte-Carlo Tree Search

TODO: talk about UCT, and multiarmed bandits. Specifically that the theoretical lower bound is log T and UCT achieves log T lower bound. Define cumulative regret here.

3.3.1 Trial Based Heuristic Tree Search

TODO: THTS paper, talk about the differences that the paper has to our presentation of `THTS++`

TODO: cut from ch2: Finally we will briefly point out the differences between `THTS++`and the original THTS schema in subsection

TODO: talk about how these methods are still relevant with deep learning because of algorithms that use both, such as alpha zero

3.3.2 Monte-Carlo Tree Search

TODO: list

- Talk about the things that are ambiguous from literature (e.g. people will just say UCT, which originally presented doesn't run in `mcts_mode`, but often assumed it does)
- Should talk about multi-armed bandits here?

https://inria.hal.science/inria-00164003/document

https://pdf.sciencedirectassets.com/271585/1-s2.0-S0004370211X0005X/1-s2.0-S000437021100052X/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEOP

TODO: removed from ch2, this can be litrev: add polynomial UCT here? and or prioritised UCT from alpha go here?

TODO: removed from ch2: add stuff about regret here, give the O(logn) bound for UCT, and also talk about the papers that

TODO: removed from ch4: H-MCTS algorithm <TODO: cite>which combines UCT and Sequential Halving.

3.3.3 Maximum Entropy Tree Search

TODO: MENTS, RENTS and TENTS

3.4 Multi-Objective Reinforcement Learning

TODO: list

- Should talk about multi-objective and/or contextual multi-armed bandits here?
- Bunch of the work covered in recent MORL survey [12]
- Mention some deep MORL stuff, say that this work (given AlphaZero) is adjacent work

TODO: removed from ch2: comment here or in literature review about there being more types of scalariation function that arent necessarily weighted by a weight, and ESR vs SER stuff

TODO: talk about more of the MORL survey, including some of the other motivating scenarios

TODO: Talk about the better way of doing CHVI? https://www.jmlr.org/papers/volume13/lizotte12a and Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. Also by Lizotte. But also say that its approximate for $D > 2$ and so we stick with the slower one? But the CHVS operations can be computed more efficiently in $D = 2$ using this stuff. Did it for the python implementation.

TODO: also need to talk about the MO sequential decision survey. Define Pareto Front. Say that the Pareto Front "Definition 3 If the utility function u

is any monotonically increasing function, then the Pareto Front (PF) is the undominated set [Roijers et al., 2013]:"

TODO: need to talk about this because some prior/related work uses PF rather than CH

TODO: cover some inner loop and outer loop things? Roijers computing CCS work? See what MORL survey says about it

TODO: Maybe talk about the *witness algorithm* for Partially Observable MDPs as its very similar to CHVI stuff

Ann Nowe papers:

https://www.jmlr.org/papers/volume15/vanmoffaert14a/vanmoffaert14a.pdf

https://arxiv.org/pdf/2402.07182

https://www.ifaamas.org/Proceedings/aamas2024/pdfs/p1611.pdf

TODO: Go through Roijers and Ann Nowe and Mykel Kochenderfer

TODO: Deep MORL things in MO-gymnasium

TODO: Some of the outer and inner loop things - prism, computing convex coverage sets

3.5 Multi-Objective Monte Carlo Tree Search

TODO: I think this whole section can just go in litrev

TODO: list

- Define the old methods (using the CH object methods, so clear that not doing direct arithmetic)
- Mention that old method could be written using the arithmetic of CHMCTS (but they don't)
- TODO: write about & make sure its implemented - its because just updating for 1 is more efficient in deterministic, and say that the additions can be implemented as updating for 1 value when determinstic
- Different flavours copy UCT action selection, but with different variants

- Link back to contributions and front load our results showing that all of the old methods don't explore correctly

TODO: There has been some prior work in multi-objective MCTS which we will outline here

TODO: Write out implementations of prior works using THTS

TODO: define pareto front

TODO: perez algorithms

TODO: https://ieeexplore.ieee.org/document/8107102

TODO: https://www.roboticsproceedings.org/rss15/p72.pdf TODO: https://arxiv.org/abs/2111.0182

TODO: https://proceedings.mlr.press/v25/wang12b/wang12b.pdf

TODO: https://ifmas.csc.liv.ac.uk/Proceedings/aamas2021/pdfs/p1530.pdf TODO: https://link.springer.com/article/10.1007/s10458-022-09596-0

TODO: Some stuff we wrote that didnt use in ch2. Might want to talk about it with eval:

Moreover, in the case of MCTS algorithms, by having the user select a preferred policy, it implicitly forces the user to chose a preference over the objectives, as the policy corresponds to a weight vector that it is optimal for. As MCTS algorithms are often used in an online fashion, where planning is interleaved with execution, this implicitly selected weight can be used for any online execution needed, effectively reducing the multi-objective problem into a single-objective problem.

4

Monte Carlo Tree Search With Boltzmann Exploration

Contents

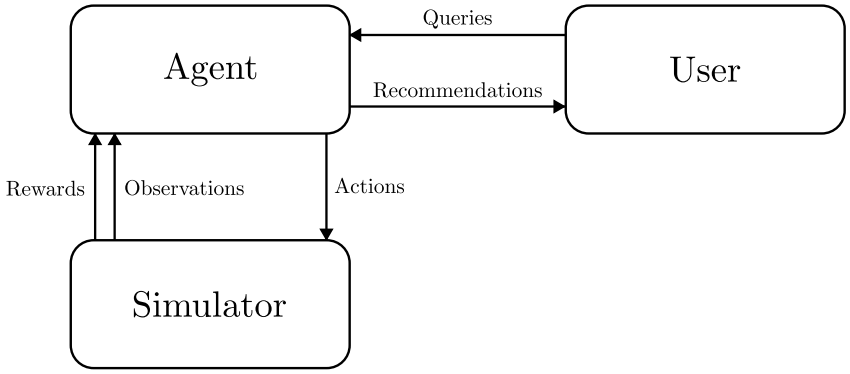


Figure 4.1: Exploration setting in reinforcement learning (edited from Figure 2.5), where the agent is only assessed on the recommendations that it provides, and is not penalised for considering poor actions in the simulator. So this setting places greater emphasis on exploration over exploitation.

This chapter considers MCTS algorithms for planning in single-objective environments, where the algorithm may consider a secondary entropy objective for exploration. In the maximum entropy setting the optimal soft policy takes the form of a Boltzmann distribution (Section 2.3.1), and the chapter will predominantly discuss MCTS algorithms whose search policies take the form of Boltzmann distributions. Question **Q1.1 - Entropy**, about how entropy can be used soundly in MCTS planning algorithms is answered here, and the foundations are laid for answering **Q1.2 - Multi-Objective Exploration** in Chapter 6.

The discussion will focus on the exploration setting for reinforcement learning (see Section 2.3 and Figure 4.1), where agents are assessed purely on the recommendations it makes, rather than what actions it explored in simulation.

Entropy is widely used in the reinforcement learning literature, commonly introduced to promote exploration and discourage convergence to suboptimal deterministic policies [25, 10, 19, 34]. MENTS (Section 2.4.4), RENTS and TENTS (Section 3.3.3) are all MCTS algoritms that optimise for maximum entropy objectives. While RENTS and TENTS will be considered as baselines in empirical experiments,

MENTS will be used to facilitate and provide discussion around the use of entropy and maximum entropy objective.

Section 4.1 discusses limitations of existing MCTS algorithms in the exploration setting, motivating the work covered in the remainder of the chapter. Additionally, the planning framework is formally defined so that the algorithms can be theoretically analysed using *simple regret*.

In Section 4.2 the *Boltzmann Tree Search* (BTS) and *Decaying ENtropy Tree Search* (DENTS) algorithms are defined. Section 4.2.3 additionally discusses some useful properties of using a stochastic search policy in MCTS, such as naturally including prior knowledge through mixed policies, and how the Alias method (Section 2.6) can be used to improve on computational complexity to answer **Q2.1 - Complexity**.

Section 4.3 considers some theoretical MDPs, which are used to empirically demonstrate the limitations of the existing MCTS algorithms, and are additionally used to provide discussion around **Q1.1 - Entropy**.

Results on grid world environments and the game of Go are given in section 4.4.

Finally, in Section 4.5 the main theoretical analysis and proofs are given. Convergence properties of MENTS, BTS and DENTS are proven to answer **Q1.1 - Entropy**.

TODO: After finished writing, do another pass through neurips paper and check no results/writing missing from thesis that really should be included. (Thinking about some of the frozen lake plots in the appendix when writing this.)

4.1 Introduction and Motivation

.

In this section a grid world shortest path problem will be used to discuss the behaviour of UCT and MENTS in the exploration setting, and highlight some limitations of these algorithms. In this grid world the agent may move deterministically in any cardinal direction, North/East/South/West, provided it stays on the grid. The agent starts at the origin $(0, 0)$ and the goal is to reach

the other side of the grid, at $(G-1, G-1)$, where G is the grid size. The cost of a path is equal to its length, and an optimal policy will always select actions that move the agent `UP` or `RIGHT`.

This MDPs can be defined formally:

$$\mathcal{S} = \{(x, y) \in \mathbb{N}^2 | x, y \in [0, G]\} \tag{4.1}$$

$$s_0 = (0, 0) \tag{4.2}$$

$$\mathcal{A} = \{(1, 0), (0, -1), (-1, 0), (0, 1)\} = \{\texttt{UP}, \texttt{LEFT}, \texttt{DOWN}, \texttt{RIGHT}\} \tag{4.3}$$

$$\text{clip}((x, y)) = (\max(\min(x, G-1), 0), \max(\min(y, G-1), 0)) \tag{4.4}$$

$$p(s'|s, a) = \begin{cases} \mathbb{1}[s' = s] & \text{if } s = (G-1, G-1) \\ \mathbb{1}[s' = \text{clip}(s + a)] & \text{otherwise} \end{cases} \tag{4.5}$$

$$H = 6G \tag{4.6}$$

$$R(s, a) = -1 \tag{4.7}$$

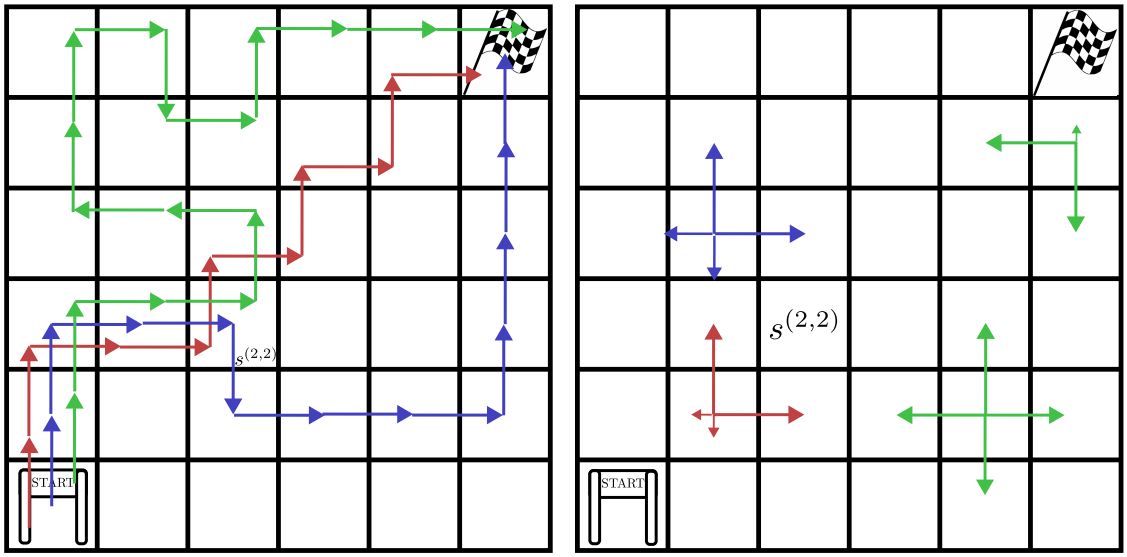
In Section 4.4 similar grid world problems will be considered with greater complexities, such as sparse rewards and stochastic transition distributions.

4.1.1 UCT

The UCT algorithm (Section 2.4.3) is designed in the traditional reinforcement learning setting described in Section 2.3, where the UCT agent aims to minimise the cumulative regret. Thus UCT makes a trade off between exploration and exploitation during it trials, and as such will frequently choose the same action to exploit, which can result in it getting stuck in local optima when rewards are sparse or not informative.

A sparse reward is where the reward signal is infrequent, that is, most actions give a reward of zero. In the shortest path example, the reward is dense but relatively uninformative, as the immediate cost of taking any action is the same.

In Figure 4.2a the shortest path problem is depicted, with an optimal path shown in red, and two paths that UCT may have explored in blue and green. When UCT is selecting an action to take from state $s^{(2,2)}$, it will consider the value estimates $\bar{Q}_{\text{UCT}}(s^{(2,2)}, \texttt{DOWN}) \approx -8$ and $\bar{Q}_{\text{UCT}}(s^{(2,2)}, \texttt{UP}) \approx -12$ (note that at



(a) Some possible paths from start to finish.

(b) Optimal policy with the maximum entropy objective for varying temperatures.

Figure 4.2: An example grid world shortest path problem. (a) shows three possible paths, that could be explored by UCT. The red path shows one possible optimal path from the start to finish (cost of 10). The blue path shows a near optimal path (cost of 12), and the green path shows a suboptimal path (cost of 16). Supposing UCT has searched the blue and green paths, when the UCT search policy selects actions at state $s^{(2,2)}$, it will often pick action `DOWN` over action `UP`, as the Q-value estimates it has are $\bar{Q}_{\text{UCT}}(s^{(2,2)}, \texttt{DOWN}) = -8$ and $\bar{Q}_{\text{UCT}}(s^{(2,2)}, \texttt{UP}) = -12$. (b) shows the probability of sampling actions from the optimal soft policy for varying temperature α. Red corresponds to a small value of α, and is close to the optimal standard policy. Blue corresponds to a mid-range value of α where the optimal policy will still try to reach the goal, but still act randomly to obtain entropy reward. Green corresponds to a large value of α, where obtaining entropy reward outweighs ever reaching the goal, and the optimal soft policy is nearly uniform. For large values of entropy the optimal soft policy actively avoids reaching the goal that will end the trial, and will have a slight preference to stay near the middle of the grid, as there are fewer available actions at the edges and corners (less entropy available). TODO: clean up alignment of arrows in figs and make state label clearer in fig a

state $s^{(2,2)}$ the agent has already taken 4 steps along the path). As the Q-value estimate for taking the suboptimal action `DOWN` is higher, UCT will most frequently select this action from state $s^{(2,2)}$.

UCT will select every action infinitely often (that is, $N(s,a) \to \infty$ for all reachable s, a) and in theory will eventually converge to the optimal policy. However, in practise this could take a long time. In Section 4.3 a theoretical MDP is considered where UCT needs much more than $\exp(|\mathcal{S}|)$ trials for the search policy

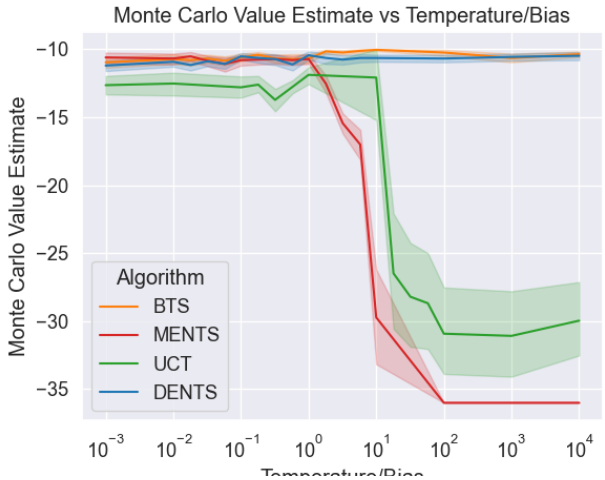


Figure 4.3: Results on the example grid world problem for a variety of UCT bias and temperature parameters. For comparison, the Boltzmann Tree Search (BTS) and Decaying ENtropy Tree Search (DENTS) algorithms defined in Section 4.2 are also shown. Each algorithm was run for 5000 trials, and results were averaged over 25 repeated experiments. UCT is able to find the optimal solution in some of the repeats with a bias of 10, but for larger biases 5000 trials was not enough to converge. MENTS successfully finds the shortest path of length 10 for small values of α_{MENTS}, but for values in the inteval $[1, 100)$ the entropy objective encourages a scenic route to the goal, and for values 100+ acting randomly to gain entropy reward far outweighs reaching the goal. TODO: Make sure can actually see x-axis label (Temperature/Bias)

to converge to the optimal policy.

In Figure 4.3 the performance of UCT after 5000 trials is given for a range of values of the UCT bias parameter b_{UCT}. This bias parameter controls the amount of exploration that UCT performs, and as the bias parameter is increased UCT will explore more. It can be seen that for low values of the bias parameter UCT is stuck in a suboptimal solution. When the bias parameter is increased UCT sufficiently explores to find the optimal policy. Finally, as UCT uses sample averages, for a very large bias parameter, 5000 trials is not sufficient enough for the average return value estimates to converge. Note that the (Q-)values of UCT can also be written as:

$$\bar{V}_{\text{UCT}}(s) = \sum_{a \in \mathcal{A}} \frac{N(s,a)}{N(s)} \bar{Q}_{\text{UCT}}(s,a), \tag{4.8}$$

$$\bar{Q}_{\text{UCT}}(s,a) = R(s,a) \sum_{s' \in \text{Succ}(s,a)} \frac{N(s')}{N(s,a)} \bar{V}_{\text{UCT}}(s,a), \tag{4.9}$$

and so the empirical distribution $\frac{N(s,a)}{N(s)}$ needs to a converge to a one hot distribuion before the value estimates can converge to the optimal values.

TODO: This later issue can be resolved by using MaxUCT instead. Add? Also add in results Section 4.4?

4.1.2 MENTS

One can argue that when using the maximum entropy objective, it is possible to recover the standard objective by setting the temperature parameter to zero, or an infitesimally small value. However, in practise, setting α to a tiny value will nullify the exploration advantages of using entropy. Considering the other extreme, when a very large temperature is used, entropy becomes the dominant objective in the maximum entropy objective, and the optimal soft policy will be (almost) uniform.

The optimal soft policy for a large temperature may be very different to the optimal policy for the standard objective. In cases such as this, it can be said that the maximum entropy objective is *misaligned* with the standard objective. More precisely, the maximum entropy objective is misaligned when the policy $\pi^*_{\text{sft,eval}}(s) = \arg\max_{a'} \pi^*_{\text{sft}}(a'|s)$ that would be followed at test time differs from the optimal standard policy $\pi^*(s)$.

For a more concrete example of this phenominon, consider the shortest path example and what the optimal soft policy look like at a state next to the goal? The agent can either reach the goal in the next step, or, alternatively move away from the goal so that it can collect more entropy reward. This is depicted in Figure 4.2b in purple.

Hence, when using the maximum entropy objective the temperature parameter often needs to be carefully tuned to the MDP. Using too small of a value will nullify the exploration benefits, and using too large of a value will result in the maximum entropy objective being misaligned with the standard objective. In Section 4.3 a theoretical MDP is considered where the temperature parameter needs to be made prohibitively small to avoid the maximum entropy objective from being misaligned.

In Figure 4.3 the performance of MENTS after 5000 trials is given for a range of values of the temperature parameter α_{MENTS}. For temperature values up to a value of 1, a close to optimal path is found in the shortest path problem. In

the range of temperatures between 1 and 100, it becomes more beneficial for MENTS to act randomly to obtain entropy but will still tend to move towards the goal, and beyond a value of 100 it becomes much more valueable to act randomly and optimise for entropy.

Building off this intuition, a good rule of thumb when using the maximum entropy objective is to set the temperature parameter large enough to gain an exploration benefit, but small enough to avoid misalignment issues. Additionally, the threshold for which the maximum entropy objective will become misaligned is dependent on the MDP itself, considering that a uniform policy over $|\mathcal{A}|$ actions has an entropy of $\log(|\mathcal{A})$. Generally this means that the temperature will need to be carefully tuned for any new MDP that it is used with.

Furhermore, in Section 4.3, a theoretical MDP will be constructed where MENTS will either suffer from the issue of the entropy objective misalignment, or the temperature must be set to a small value that does not effectively utilise the entropy exploration.

While only the MENTS algorithm has been discussed in this section, the issue arises from mixing entropy into the scalar objective. Similar issues still arise no matter the form of entropy considered.

4.1.3 Simple Regret and Consistency

TODO: Move to chapter 2 in RL section. Also add cumulative regret in RL section.

Now *simple regret* is defined, which will be used motivate and analyse the algorithms developed. The simple regret of a policy is the difference between the value of the policy and optimal value of the policy (Equation (4.10)). By definition, the optimal policy achieves a simple regret of zero, and an MCTS algorithm is considered *consistent* if it's expected simple regret tends to zero. In plain english, an algorithm is consistent if left to run forever it would eventually output an optimal policy. An implication of consistency is that if an algorithm can be run for longer, then it is expected to improve on its solution.

Parameters: An MDP $\mathcal{M}$.

- For each round $m = 1, 2, ...$:
  1. the agent produces a search policy π^m to follow;
  2. the environment samples a trajectory $\tau \sim \pi^m$ (including rewards $r_t = R(s_t, a_t)$ for each s_t, a_t pair in τ);
  3. the agent produces a recommendation policy ψ^m;
  4. if the environment sends a stop signal, then the game ends, otherwise the next round starts.

Figure 4.4: The procedure of an exploring planning problem for MDPs, where ψ^m is the recommendation policy the agent produces after m trajectories are sampled using the exploration policy π^m.

While *cumulative regret* has been used to motivate and analyse algorithms such as UCT, it is not the most appropriate measure for the exploration setting. In the exploration setting, the agent is only assessed on the recommendations it makes, and is not penalised for considering poor actions in the simulator. See Figure 4.4 for the formal setup of the exploring planning problem for MDPs. As such, the focus of this chapter will be on the (expected) simple regret of the algorithms developed.

Definition 4.1.1. *The* simple regret *of a policy ψ at state $s \in \mathcal{S}$ is the difference between the value of the policy and the optimal value at that state:*

$$\texttt{sim\_regr}(s, \psi) = V^*(s) - V^{\psi}(s). \tag{4.10}$$

Note that the simple regret is a random variable, as it depends on the recommendation policy ψ, which itself depends on the random trajectories that are sampled. Hence the expected simple regret will be the main value of interest in the theoretical analysis of Section 4.5.

Now that simple regret has been defined, the concept of consistency can be defined formally:

Definition 4.1.2. *An agent, that produces recommendation policies $\psi^1, \psi^2, ...$ is said to be* consistent *if $\mathbb{E}[\texttt{sim\_regr}(s, \psi^m)] \to 0$ as $m \to \infty$.*

Returning to the discussion around UCT and MENTS, now that simple regret and consistency have been defined, it can be shown the UCT is always consistent TODO: cite?, whereas MENTS is only consistent for a sufficiently small temperature parameter that depends on the MDP (Section 4.3 and Theorem TODO: ref theorem). Although UCT is consistent, it can take a long time to converge to the optimal policy (as will be seen in 4.3), and so it is of interest to develop algorithms that can explore more that are also consistent.

4.2 Boltzmann Search

TODO: Double check NeurIPS appendix B coverered properly here

TODO: Double check NeurIPS appendix C coverered properly here (some disucssion around when to use different algorithms and multithreading in MCTS)

This section defines new algorithms, Boltzmann Tree Search (BTS) and Decaying ENtropy Tree Search (DENTS). Following the discussion from Section 4.1, the design aims of these algorithms are as follows:

- to be as simple to define and implement as UCT and MENTS (i.e. each decision node operates on a multi-armed bandit problem);
- be able to explore effectively when rewards are sparse or uninformative;
- be able to exploit when necessary (dense informative rewards, large environments or stochastic environments);
- can be shown to be consistent for parameters that do not depend on the MDP.

In Section 4.2.1 BTS is defined, which adapts MENTS to run on the standard objective to arrive at a consistent algorithm. Then Section 4.2.2 defines DENTS which re-introduces entropy as an exploration bonus while maintaining consistency.

4.2.1 Boltzmann Tree Search

The *Boltzmann Tree Search* (BTS) algorithm uses value estimates $\hat{V}_{\text{BTS}}$ and $\hat{Q}_{\text{BTS}}$ that are computed using *Bellman backups*. Additionally, BTS uses a variable temperature specified by the positive function $\alpha_{\text{BTS}}(x) > 0$, and at a decision node s the temperature used will be $\alpha_{\text{BTS}}(N(s))$. BTS promotes exploration through the use of a stochastic Boltzmann search policy.

The search policy of BTS is defined as:

$$\pi_{\text{BTS}}(a|s) = (1 - \lambda(s, \epsilon_{\text{BTS}}))\rho_{\text{BTS}}(a|s) + \frac{\lambda(s, \epsilon_{\text{BTS}})}{|\mathcal{A}|}, \tag{4.11}$$

$$\rho_{\text{BTS}}(a|s) \propto \exp\left(\frac{1}{\alpha_{\text{BTS}}(N(s))}\left(\hat{Q}_{\text{BTS}}(s,a)\right)\right). \tag{4.12}$$

$$\lambda(s, \epsilon) = \min\left(1, \frac{\epsilon}{\log(e + N(s))}\right) \tag{4.13}$$

where $\epsilon_{\text{BTS}} \in [0, \infty)$ is an exploration parameter.

Given a trajectory $\tau = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h) \sim \pi_{\text{BTS}}$ the value estimates are updated for $t = h - 1, ..., 0$:

$$\hat{Q}_{\text{BTS}}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \text{Succ}(s_t, a_t)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}_{\text{BTS}}(s')\right), \tag{4.14}$$

$$\hat{V}_{\text{BTS}}(s_t) \leftarrow \max_{a \in \mathcal{A}} \hat{Q}_{\text{BTS}}(s_t, a). \tag{4.15}$$

In line with the `THTS++` schema, the values of $\hat{V}_{\text{BTS}}(s)$ are initialised with the function $\hat{V}_{\text{init}}$ and any missing values of $\hat{Q}_{\text{BTS}}(s, a)$ are completed using the function $\hat{Q}_{\text{init}}$. By default these functions can be set to a constant value, or could encorporate prior information, for example with the use of neural networks.

When BTS needs to recommend a policy, it can use it's Q-value estimates:

$$\psi_{\text{BTS}}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_{\text{BTS}}(s, a). \tag{4.16}$$

Alternatively, the node visit counts can be used in the recommendation policy

$$\texttt{mv}_{\text{BTS}}(s) = \arg\max_{a \in \mathcal{A}} N(s, a). \tag{4.17}$$

Similarly to UCT the best setting of the temperature function will be dependent on the reward scaling. Rather than adding a scaling factor to this parameter, BTS

can operate in an `adaptive_policy` mode, where the Q-values are normalised to the unit interval before they are used in the policy:

$$\rho_{\text{BTS}}(a|s) \propto \exp\left(\frac{1}{\alpha_{\text{BTS}}(N(s))}\left(\hat{Q}^{\text{norm}}_{\text{BTS}}(s,a)\right)\right). \tag{4.18}$$

$$\hat{Q}^{\text{norm}}_{\text{BTS}}(s,a) = \frac{\hat{Q}_{\text{BTS}}(s,a) - \min_{a\in\mathcal{A}} \hat{Q}_{\text{BTS}}(s,a)}{\max_{a\in\mathcal{A}} \hat{Q}_{\text{BTS}}(s,a) - \min_{a\in\mathcal{A}} \hat{Q}_{\text{BTS}}(s,a)}. \tag{4.19}$$

The BTS search policy can still be used with average returns. In *Boltzmann Tree Search with Average Returns* (AR-BTS) the Bellman value estimates are replaced with average returns $\hat{V}_{\text{AR-BTS}}$ and $\hat{Q}_{\text{AR-BTS}}$. Given a trajectory $\tau = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h)$ the average returns are updated for $t = h-1, ..., 0$: and the leaf node value estimate $\tilde{r} =$, the value estimates are updated for $t = h-1, ..., 0$:

$$\hat{Q}_{\text{AR-BTS}}(s_t, a_t) \leftarrow \frac{N(s_t,a_t)-1}{N(s_t,a_t)}\left(\hat{Q}_{\text{AR-BTS}}(s_t,a_t) + \frac{\hat{V}_{\text{init}}(s_h) + \sum_{i=t}^{h-1} r_i}{N(s_t,a_t)-1}\right) \tag{4.20}$$

$$\hat{V}_{\text{AR-BTS}}(s_t) \leftarrow \frac{N(s_t)-1}{N(s_t)}\left(\hat{V}_{\text{AR-BTS}}(s_t) + \frac{\hat{V}_{\text{init}}(s_h) + \sum_{i=t}^{h-1} r_i}{N(s_t)-1}\right). \tag{4.21}$$

The corresponding definitions of $\pi_{\text{AR-BTS}}$, $\psi_{\text{AR-BTS}}$, $\texttt{mv}_{\text{BTS}}$ are similar to the definitions for BTS, but for completeness are given in Appendix B.

In Section 4.5 it is shown that BTS recommendation policy will converge to the optimal policy.

Theorem 4.2.1. *For any MDP $\mathcal{M}$, the expected simple regret of ψ_{BTS} tends to zero:* $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\text{BTS}})] \to 0$.

Additionally, if the temperature function is allowed to tend to zero, then the search policy will also converge to the optimal policy, and the most visited recommendation policy will also converge to the optimal policy as a result.

Theorem 4.2.2. *For any MDP $\mathcal{M}$, if $\alpha_{\text{BTS}}(x) \to 0$, then $\pi_{\text{BTS}} \xrightarrow{p} \pi^*$ and* $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\text{BTS}})] \to 0$. *TODO: Double check converge in prob is correct for search policy*

Conversely, if the temperature always positive, then the BTS recommendation policy will converge to the optimal policy exponentially fast.

Theorem 4.2.3. *For any MDP $\mathcal{M}$, if there exists $\alpha_{\text{BTS}}(x) \geq L > 0$, then there exists constants $C, k > 0$ such that $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\text{BTS}})] \leq C \exp(-kn)$ after running n trials of BTS.*

For AR-BTS, the temperature function needs to be restricted to functions that tend to zero. This is necessary so that the search policy converges to the optimal policy, so that the average return value estimates can converge to the optimal (Q-)values.

Theorem 4.2.4. *For any MDP $\mathcal{M}$, if $\alpha_{\text{AR-BTS}}(x) \to 0$, then $\pi_{\text{AR-BTS}} \xrightarrow{p} \pi^*$, $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\text{AR-BTS}})] \to 0$ and $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\text{AR-BTS}})] \to 0$. TODO: Double check converge in prob is correct for search policy*

TODO: Make sure theorem numbers in theory section match (for all 4 of the above)

4.2.2 Decaying ENtropy Tree Search

Decaying ENtropy Tree Search (DENTS) extends the BTS algorithm by adding *entropy estimates*. In DENTS nodes maintain value estimates $\hat{V}_{\text{D}}$ and $\hat{Q}_{\text{D}}$, but also a secondary value estimate, $\bar{\mathcal{H}}_{V,\text{D}}$ and $\bar{\mathcal{H}}_{Q,\text{D}}$, which are monte carlo estimates the entropy of the search policy rooted from the relevant node. By keeping a separate estimate for entropy, DENTS is able to use entropy in it's search policy, and discard it for recommendations.

The entropy values are used as an exploration bonus in the search policy, and are weighted by a non-negative function $\beta_{\text{D}}(x) \geq 0$. Generally this will be set to a function such that $\beta_{\text{D}}(x) \to 0$ as $x \to \infty$, so that the weighting on entropy reduces over time and deeper parts of the tree can be explored. The DENTS search policy π_{D} is defined follows:

$$\pi_{\text{D}}(a|s) = (1 - \lambda(s, \epsilon_{\text{D}}))\rho_{\text{D}}(a|s) + \frac{\lambda(s, \epsilon_{\text{D}})}{|\mathcal{A}|}, \tag{4.22}$$

$$\rho_{\text{D}}(a|s) \propto \exp\left(\frac{1}{\alpha_{\text{D}}(N(s))}\left(\hat{Q}_{\text{D}}(s,a) + \beta(N(s))\bar{\mathcal{H}}_{Q,\text{D}}(s,a)\right)\right), \tag{4.23}$$

$$\lambda(s, \epsilon) = \min\left(1, \frac{\epsilon}{\log(e + N(s))}\right). \tag{4.24}$$

Given a trajectory $\tau = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h) \sim \pi_{\mathrm{D}}$, the entropy values are updated as follows for $t = h-1, ..., 0$:

$$\bar{\mathcal{H}}_{Q,\mathrm{D}}(s_t, a_t) \leftarrow \sum_{s' \in \mathrm{Succ}(s_t, a_t)} \frac{N(s')}{N(s_t, a_t)} \bar{\mathcal{H}}_{V,\mathrm{D}}(s'), \tag{4.25}$$

$$\bar{\mathcal{H}}_{V,\mathrm{D}}(s_t) \leftarrow \mathcal{H}(\pi_{\mathrm{D}}(\cdot|s_t)) + \sum_{a \in \mathcal{A}} \pi_{\mathrm{D}}(a|s_t) \bar{\mathcal{H}}_{Q,\mathrm{D}}(s_t, a), \tag{4.26}$$

where $\mathcal{H}$ is the Shannon entropy function. Initially, each of the entropy estimates are set to zero: $\bar{\mathcal{H}}_{V,\mathrm{D}}(s) \leftarrow 0$ and $\bar{\mathcal{H}}_{Q,\mathrm{D}}(s, a) \leftarrow 0$.

Given the same trajectory,the Bellman value estimates $\hat{V}_{\mathrm{D}}$ and $\hat{Q}_{\mathrm{D}}$ are updated identically to BTS for $t = h-1, ..., 0$:

$$\hat{Q}_{\mathrm{D}}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \mathrm{Succ}(s_t, a_t)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}_{\mathrm{D}}(s) \right), \tag{4.27}$$

$$\hat{V}_{\mathrm{D}}(s_t) \leftarrow \max_{a \in \mathcal{A}} \hat{Q}_{\mathrm{D}}(s_t, a), \tag{4.28}$$

The recommendation policies for DENTS are defined identically to BTS:

$$\psi_{\mathrm{D}}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_{\mathrm{D}}(s, a), \tag{4.29}$$

$$\texttt{mv}_{\mathrm{D}}(s) = \arg\max_{a \in \mathcal{A}} N(s, a), \tag{4.30}$$

DENTS can also be run in the `adaptive_policy` mode similarly to BTS, and the Bellman value estimates can be with average returns to arrive at the *Decaying ENtropy Tree Search with Average Returns* (AR-DENTS) algorithm. To avoid repetition, full details are given in Appendix B for completeness.

Corresponding theoretical results hold for DENTS and AR-DENTS to those stated for BTS. For results that require the temperature function α_{D} to tend to zero, it will now also be required for the entropy weighting function β_{D} to tend to zero at a faster rate than the temperature function. This is necessary so that the convergent search policy is not skewed by any entropy estimate. Proofs of the theorems below are given in Section 4.5.

Theorem 4.2.5. *For any MDP $\mathcal{M}$, the expected simple regret of ψ_{D} tends to zero:* $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\mathrm{BTS}})] \to 0$.

Theorem 4.2.6. *For any MDP $\mathcal{M}$, if $\alpha_{\mathrm{D}}(x) \to 0$ and $\frac{\beta_{\mathrm{D}}(x)}{\alpha_{\mathrm{D}}(x)} \to 0$, then $\pi_{\mathrm{D}} \xrightarrow{p} \pi^*$ and $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\mathrm{D}})] \to 0$. TODO: Double check converge in prob is correct for search policy*

Theorem 4.2.7. *For any MDP $\mathcal{M}$, if there exists $\alpha_{\mathrm{D}}(x) \geq L > 0$ and $U \geq \beta_{\mathrm{D}}(x)$, then there exists constants $C, k > 0$ such that $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\mathrm{BTS}})] \leq C \exp(-kn)$ after running n trials of DENTS.*

Theorem 4.2.8. *For any MDP $\mathcal{M}$, if $\alpha_{\text{AR-D}}(x) \to 0$ and $\frac{\beta_{\text{AR-D}}(x)}{\alpha_{\text{AR-D}}(x)} \to 0$, then $\pi_{\text{AR-D}} \xrightarrow{p} \pi^*$, $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\text{AR-D}})] \to 0$ and $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\text{AR-D}})] \to 0$. TODO: Double check converge in prob is correct for search policy*

TODO: Make sure theorem numbers in theory section match (for all 4 of the above)

4.2.3 Advantages of Stochastic Search Policies

Using a stochastic search policy in MCTS (or `THTS++`) provides more benefits than just encouraging exploration through randomly sampled actions. In particular, the Alias Method (Section 2.6) can be used to trade off using the most up to date policy for computational speed, yeilding an answer to **Q2.1 - Complexity**. Moreover, when prior knowledge (in the form of a policy prior) is available, it can naturally be integrated into the search using a *mixture policy*.

The Alias Method in MCTS

TODO: This section feels unnecessarily dense right now

TODO: Dont like that subsubsection and paragraph headings seem the same size. Fix later though

The Alias Method (Section 2.6) can be used sample from a categorical distribution in constant time, with a linear cost for preprocessing. However this assumes that the categorical distribution is fixed, which is not the case for search policies in MCTS. However, the up-to-dateness of the search policy can be traded off for sampling speed.

Any MCTS algorithm that samples actions from a stochastic (categorical) distribution can make use of the alias method. To do so, when a new decision node is made, construct an initial alias table, with $O(A)$ cost, where $A = |\mathcal{A}|$. Then, update the alias table every A visits to the decision node. As sampling from the alias table has a cost of $O(1)$, and there is an $O(A)$ cost to update the table every A visits, the amortised cost of sampling actions is reduced to $O(1)$ using this method.

To fully answer **Q2.1 - Complexity**, the cost of backups needs to be considered too. The remainder of this subsection will consider how efficiently DENTS and AR-DENTS can be implemented.

The cost of running n trials of MCTS, on an MDP with horizon H and A actions is typically $O(nAH)$. Which is because on each of the n trials, up to H many decision nodes may be visited, and each decision node needs to consider A many values in sampling actions.

When not running in `mcts_mode` the worst case complexity while using the Alias method will still be $O(nAH)$, when each trial creates many ($O(H)$) new decision nodes, each incurring an $O(A)$ cost to build the alias table. Although the asymptotic cost is not improved by using the Alias method without `mcts_mode`, in practise it may still run an order of magnitude faster.

When running `mcts_mode` however, only one new decision node is created per trial, which leads to a complexity of $O(n(BH + A))$, where B is the worst cost for computing backups in the trial, as backups still need to be computed for every node visited per trial.

Bellman Backups. For some $\tau = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h)$, consider Bellman backups used of the form:

$$\hat{Q}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \text{Succ}(s_t, a_t)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}(s) \right), \tag{4.31}$$

$$\hat{V}(s_t) \leftarrow \max_{a \in \mathcal{A}} \hat{Q}(s_t, a). \tag{4.32}$$

Although the cost to compute the right hand side of each of these is $O(A)$, they can be more efficiently computed by observing that only one child value is updated

per trial. As such, Equation (4.31) can be computed in $O(1)$ time as:

$$\hat{Q}(s_t, a_t) \leftarrow \begin{cases} R(s_t, a_t) + \hat{V}(s_{t+1}) & \text{if } N(s_t, a_t) = 1, \\ R(s_t, a_t) + S(s_t, a_t) + \frac{N(s_{t+1})}{N(s_t, a_t)} \hat{V}(s_{t+1}) & \text{otherwise,} \end{cases} \tag{4.33}$$

$$S(s_t, a_t) = \frac{N^{\text{old}}(s_t, a_t)}{N(s_t, a_t)} \left(Q(s_t, a_t) - R(s_t, a_t) - \frac{N^{\text{old}}(s_{t+1})}{N^{\text{old}}(s_t, a_t)} \hat{V}^{\text{old}}(s_{t+1}) \right), \tag{4.34}$$

where S computed the corrected average of $\hat{V}^(s')$ for $s' \in \text{Succ}(s_t, a_t) - \{s_{t+1}\}$, $\hat{V}^{\text{old}}(s_{t+1})$ is the previous value of $\hat{V}(s_{t+1})$, and $N^{\text{old}}(s_{t+1}) = N(s_{t+1}) - 1$, $N^{\text{old}}(s_t, a_t) = N(s_t, a_t) - 1$ are the previous values of $N(s_{t+1})$, $N(s_t, a_t)$.

Equation (4.32) can be most efficiently implemented in the general case by using a *max heap*. The max heap keeps track of the values of $\hat{Q}(s_t, a)$, on each backup the value of $\hat{Q}(s_t, a_t)$ needs to be updated in the heap, taking $O(\log(A))$ time, and then the maximum can be read out in $O(1)$ time.

Entropy Backups. Recall Equations (4.25) and (4.26), the entropy backups of DENTS:

$$\bar{\mathcal{H}}_{Q,\text{D}}(s_t, a_t) \leftarrow \sum_{s' \in \text{Succ}(s_t, a_t)} \frac{N(s')}{N(s_t, a_t)} \bar{\mathcal{H}}_{V,\text{D}}(s'). \tag{4.35}$$

$$\bar{\mathcal{H}}_{V,\text{D}}(s_t) \leftarrow \mathcal{H}(\pi_{\text{D}}(\cdot|s_t)) + \sum_{a \in \mathcal{A}} \pi_{\text{D}}(a|s_t) \bar{\mathcal{H}}_{Q,\text{D}}(s_t, a), \tag{4.36}$$

Equation (**??**) is of the same form as Equation (4.31), so can also be computed in $O(1)$ time similarly. Equation (**??**) can also be implemented in amortised $O(1)$ time, as in most backups the search policy does not change. Hence, every A visits, and $O(A)$ backup is required, when the search policy is updated, and otherwise an $O(1)$ backup is sufficient. Let $\pi_{\text{D}}^{\text{old}}$ be the search policy from before the node was visited this trial, when Equation (**??**) can be computed by:

$$\bar{\mathcal{H}}_{V,\text{D}}(s_t) \leftarrow \begin{cases} \mathcal{H}(\pi_{\text{D}}(\cdot|s_t)) + \sum_{a \in \mathcal{A}} \pi_{\text{D}}(a|s_t) \bar{\mathcal{H}}_{Q,\text{D}}(s_t, a) & \text{if } \pi_{\text{D}} \neq \pi_{\text{D}}^{\text{old}}, \\ \bar{\mathcal{H}}_{V,\text{D}}(s_t) + \pi_{\text{D}}(a_t|s_t) \left(\bar{\mathcal{H}}_{Q,\text{D}}(s_t, a_t) - \bar{\mathcal{H}}_{Q,\text{D}}^{\text{old}}(s_t, a_t) \right) & \text{if } \pi_{\text{D}} = \pi_{\text{D}}^{\text{old}}, \end{cases} \tag{4.37}$$

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

DENTS complexity. From the above arguments, the most time consuming backup operation costs $O(\log(A))$, and hence DENTS can be run in `mcts_mode` with the Alias method with an overall complexity of $O(n(H \log(A) + A))$ to run n trials.

AR-DENTS complexity. The average return values in AR-DENTS are computed similarly to AR-BTS, in Equations (4.20) and (B.4). These backups are already in a form that takes constant time to compute. Because the entropy backups can also be computed in amortised $O(1)$ time, the backups for AR-DENTS have a constant complexity. Hence, AR-DENTS can be run in `mcts_mode` with the Alias method with an overall complexity of $O(n(H + A))$.

Encorporating Prior Knowledge Through Mixture Policies

Let π be some `THTS++` search policy, and supposed that we have access to another policy $\tilde{\pi}$ that encapsulates some prior knowledge, such as a neural network. Then a new search policy π_{mix} can be naturally be defined using a mixture of π and $\tilde{\pi}$.

The mixture policy is defined by:

$$\pi_{\text{mix}}(a|s) = (1 - \lambda(s, \epsilon_{\text{mix}}))\pi(a|s) + \lambda(s, \epsilon_{\text{mix}})\tilde{\pi}(a|s), \tag{4.38}$$

$$\lambda(s, \epsilon) = \min\left(1, \frac{\epsilon}{\log(e + N(s))}\right). \tag{4.39}$$

where $\epsilon_{\text{mix}} \in (0, \infty)$ controls how heavily to weight the prior knowledge by in the mixture policy. The weighting of the prior policy $\tilde{\pi}$ is decayed with the number of visits, and eventually to zero, so that the behaviour of the algorithm is unchanged in the limit.

4.3 Toy Environments

TODO: Sometimes feel like I'm bashing on MENTS. Want to make it clearer that I'm not bashing on MENTS, but using MENTS (as the max entropy MCTS method) to demonstrate issues with the max entropy objective itself.

This subsection uses theoretical MDPs to further highlight and discuss the benefits and pitfalls of UCT and MENTS, TODO: and compares the performance of BTS and DENTS on these MDPs.

The *D-chain problem* introduced by TODO: cite (Figure 4.5), is a deterministic MDP for which TODO: UCT "struggles" (find better words, maybe here say that

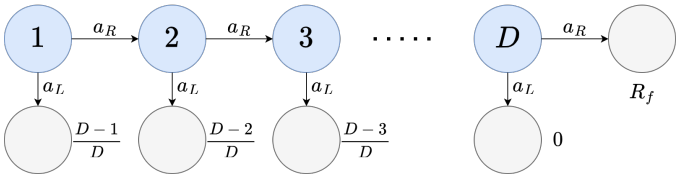


Figure 4.5: An illustration of the *(modified) D-chain problem*, TODO: where 1 is the starting state, and values next to sink states represent the reward for arriving in that state. At the end of the chain a reward of R_f TODO: is recieved, where in TODO: cite the reward at the end of the chain is set to $R_f = 1$. TODO: Temp fig

wont feasibly solve). From some state d if the action a_L is taken the trial ends and a reward of $(D - d)/D$ is received, and from the final state in the chain, D, if the action a_R is taken then the maximum reward of $R_f = 1$ is received. Hence, the optimal standard policy will always take action a_R at every state achieve a return of 1. TODO: Make sure that notation is consistent with the new fig

TODO: cite show that UCT requires $\Omega(\exp(...\exp(1)...))$ many trials (D composed exponential functions) to recommend the optimal actions a_R. BD: Informally, this behaviour stems from the value estimate of state 2 remaining below D-1/D for a long time, and UCT repeatedly taking action aL on its trials once its confidence intervals become relatively concentrated. UCT will still always explore the aR action and eventually converge to the optimal value estimates (and recommendation policy), but in practise it would take longer than a human lifetime.

Where in Section TODO: ref UCT can be seen to perform well in the presence of an informative dense reward, and struggle when only given an uninformative sparse reward, this MDP sets the rewards to take advantage of this BD: behaviour maximally. The MDPs rewards can be viewed from the perspective of the dense rewards along the chain that essentially suggest "don't explore this way", which are hiding the sparse reward of one at the end of the chain.

In stark contrast, when MENTS is run on the D-chain problem, it quickly explores and finds the sparse reward of one at the end of the chain. This is largely because of the maximum entropy objective, where there BD: is no entropy reward to be gained by traversing to a sink state, thus encouraging MENTS to follow the chain. However, because the chain is explored largely due to the maximum

entropy objective, consider what happens in the *modified D-chain problem*, where the reward at the end of the chain is set to $R_f = 1/2$. TODO: copy out the soft Q value computations with temp of one.

These two cases $R_f \in 1/2, 1$ demonstrate that MENTS, and more generally whenever using the maximum entropy objective, the optimal temperature parameter to be used is dependent on the MDP, and can vary massively even with small changes in the MDP, as for example in this the value of a single reward was changed. TODO: Reference the result that for any temperature there is an MDP where MENTS will be inconsistent

TODO: Generally write up a bit cleaner once have plots in

BTS and DENTS improve on this theoretically as convergence can be guarunteed by parameter settings which are independent of the MDP. In practise, for example consider using a constant value for α_{D} and β_{D}, the search policy can behave similarly to MENTS, BD: which could be an issue in more complex MDPs TODO: some of this discussion more appropriate for entropy trap bit maybe?

TODO: plots to add: the two plots from neurips, performance of MENTS/BTS/DENTS for varying settings of temperature parameters. Same plots for entropy trap D-chain. Also demonstrate that if temperatures decayed properly then DENTS will visit the optimal sink state in entropy D-chain lots, while if not decayed, then will recommend the correct thing, but not visit. Last one is of practical importance, where the sparse reward of one at the end isn't just a sink state but the rest of the MDP where you actually want to do something important.

BD: (Commented out old writing below this). It is often argued in the maximum entropy objective that the standard objective can be recovered by setting the temperature suffficiently small, however this looses the benefits of using entropy for exploration. TODO: Ref the result about ments converging for a sufficiently small temperature here.

By making this observation that maximum entropy algorithms can be mislead by providing an opportunity to accrue the 'entropy reward', the D-chain problem can be further adapted, such that neither UCT or MENTS will perform well for

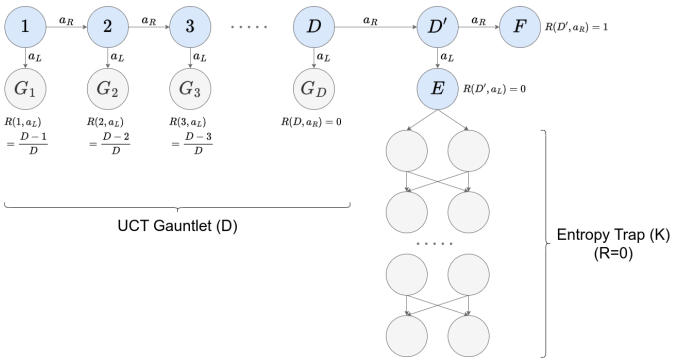


Figure 4.6: An illustration of the *(modified) D-chain problem with entropy trap*, TODO: write caption describing MDP after made it. TODO: Temp fig

any parameter settings. To construct the *D-chain with entropy trap* BD: problem? MDP?, the same chain of length D, or the *UCT gauntlet*, is used from the D-chain problem, and as such UCT will also struggle on this problem. However, at the end of the chain this time, one more choice is left, to either take the immediate reward of one, or to enter the *entropy trap*, which is a sequence of states with zero rewards that allows a policy to act randomly to gain an entropy reward.

TODO: These claims made more concrete in (ref theorem)

TODO: Some writing about what the plots show empirically once run this.

TODO: Wrap up this section saying that although these are highly theoretical and constructed examples, it can help even in practise to think about these things and consider the performance on these envs to understand why these algorithms behave the way they do on real problems. For example, it may be the case that there are some subtle entropy traps, consider if instead that Rf was 10, and the entropy trap had a reward of 9 (including the entropy bit), such that the performance looked good but TODO: Basically a bit of chatting to map back to these things in practise and why useful, maybe also add something like this at the top. Actually, add this at the top, and then point out the specifics at the end. Good sandwitch storytelling.

TODO: Below is parameter sensitivity section from neurips appendix D. Extract whats useful, but better results should demonstrate this better

We run each algorithm with a variety of α temperatures, and the ϵ exploration parameter on the 10-chain environments (Figure **??**). Additionally, we ran UCT

with a variety of bias parameters. Figures **??**, **??**, **??**, **??**, **??** and **??** give results for the 10-chain environment, with algorithms UCT, MENTS, RENTS, TENTS, BTS and DENTS respectively. Figures **??**, **??**, **??**, **??**, **??** and **??** 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 **??** and **??** for plots.

As discussed in Section **??**, 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 **??**), but will still recommend a policy that gives the reward of $R_f = 0.5$ on the modified 10-chain (Figure **??**). 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 **??**). 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 **??** and **??**. 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 **??** and **??**.

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 **??**). And, in the modified 10-chain, BTS always achieves a reward of 0.9 regardless of how the parameters are set (Figure **??**). DENTS performance on the 10-chain (Figure **??**) and modified 10-chain (Figure **??**) 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 **??** 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 **??**).

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 **??**). 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 **??**). 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 **??**, 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 **??** 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).

TODO: A MILLION FIGURES WERE HERE

4.4 Empirical Results

TODO: Talk about how exploitation is useful in two cases in the exploration setting. When have stochastic env, need to exploit to get better value estimates. When env is very large, need exploitation to explore deeper parts of env. Also exploiting helpful when reward is dense and/or informative

TODO: SHOULD WE DO VALUE NORMALISATION IN THE AUX EXPR TOO?

TODO: Wasn't too careful about changing from active voice to passive voice here in this entire section

TODO: Get some better results using hyperparam optimise //// want a dense env where the (not entropy) temp decay fn gets optimised to soemthing that decays a lot /// want a sparse env where the temp decay fn gets optimised to something flat (or basically flat)

This section provides empirical results on gridworld environments and on the game of Go. It begins by describing the evaluation setup, and environments, followed by the empirical results and discussion. The main algorithms discussed in this

chapter, BTS and DENTS will be evaluated, using UCT TODO: ref, MENTS TODO: ref, RENTS TODO: ref, TENTS TODO: ref and H-MCTS TODO: ref litrev. BD: H-MCTS is used as an UCT style MCTS algorithm that is also designed using simple regret for comparison.

4.4.1 Environments

(Deterministic) Frozen Lake

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. TODO: this is c and p from neurips TODO: also run with the more reasonable reward, was only doing this reward to make plots look nicer, but should just put in a bit of time to make the plots not look terrible with the more sensible reward

BD: This Frozen Lake environment is used to evaluate the algorithms in a sparse reward environment

TODO: TWO: would like to have experiments which vary the proportion of the sparse reward. So have FL(lambda), where lambda specifies the ratio between the dense and sparse rewards. Then investigate what happens as vary lambda. So this is the grid world experiments run again,

TODO: mention some of the above talked about simpler versions of this without holes.

TODO: Why did we keep it deterministic? Maybe do some smaller ones with transition noise?

TODO: Below is the frozen lake details from appendix, merge into main prose

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.

```
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 4.7: Maps used for experiments using the Frozen Lake environment in Sections **??**, **??** and **??**. `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.

In Figure 4.7a we give an 8x8 Frozen Lake environment that is used in Section **??** to demonstrate how the different algorithms perform with a variety of temperatures. Figure 4.7b gives the 8x12 Frozen Lake Environment that is used for hyperparameter selection in Section **??**. And in Figure 4.7c we give the 8x12 Frozen Lake Environment that is used to test the algorithms in Section **??**. 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.

Sailing Problem

The *Sailing Problem* is a grid world environment with one goal state, at the opposite corner to the starting location of the agent. TODO: This problem was introduced in <TODO, think ref is this commented out one> and has often been used to evaluate MCTS algorithms <TODO add uct cites etc>. 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.

BD: The sailing problem is used to evaluate the algorithms in a dense reward enviroinment

TODO: Below is the saling problem details from appendix, merge into main prose

```
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 4.8: 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.

Aditionally, we give the map used in the 6x6 Sailing Problem, and the wind transition probabilities in Figure 4.8. 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 **??** the wind direction was set to North (or 0) in the initial state, and for testing in Section **??**, the initial wind direction was set to South-East (or 3).

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 TODO: cite.

TODO: below is description of go in neurips paper. Think most of this is appropriate here, but some should be moved to results/eval proceedure

Area scoring is used to score the games, with a *komi* (a score handicap for black) of 7.5. We used an openly available value network $\tilde{V}$ and policy network $\tilde{\pi}$ from KataGo TODO: cite. Our baseline was the PUCT algorithm TODO: cite, as described in Alpha Go Zero TODO: citeusing prioritised UCB TODO: citeto 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 **??**. TODO: need to copy this over, appendix c.2.

4.4.2 Evaluation Proceedure

Consider an algorithm with search tree $\mathcal{T}$, which provides a partial recommendation policy ψ_{alg} TODO: ref back to defns of recommendation policies. The recommendation policy is made complete by using a uniformly random policy for states and actions that are outside of the tree $\mathcal{T}$ as follows:

$$\psi(a|s) = \begin{cases} 1 & \text{if } s \in \mathcal{T} \text{ and } a = \psi_{\text{alg}}(s), \\ 0 & \text{if } s \in \mathcal{T} \text{ and } a \neq \psi_{\text{alg}}(s), \\ \frac{1}{|\mathcal{A}|} & \text{otherwise.} \end{cases} \tag{4.40}$$

Using the completed policy ψ, a monte carlo value estimate of V^{ψ} is computed by sampling a number of trajectories from ψ and averaging the returns. TODO: Kind of want a better argument for this line of eval here. Although this proceedure is evaluating the algorithms in an *offline planning* setting, it still indicates how the algorithms perform in an *online* setting when planning in simulation is interleaved with letting the agent act in the real environment.

TODO: Experiments run without mcts mode, apart from go, which was run in mcts mode

Gridworld Hyperparameter Selection

To select hyperparameters for the algorithms in the gridworld environments, the BayesOpt package TODO: cite was used.

TODO: State the ranges of params used, including categorical parameters, such as the type of decay function used, state that no prior information used, and heurisitc/initialisation values set to zero

TODO: Commented out below this is the original hyperparameter selection details from appendix D of neurips

Go Hyperparameter Selection

To tune hyperparemters for algorithms used in the Go experiments, a methodical sequence of round robin tournaments were run on a 9×9 board, where in each tournament one parameter was tuned at a time. Full details of these tournaments is given in Appendix TODO: ref.

TODO: Summarise the selected parameters used for the experiments here? Or in the results section?

TODO: State that prior knowledge used in the form of the KataGo networks

BD: One thing of note in the Go section is that the AR versions are considered, and specifically, AR versions of MENTS, RENTS and TENTS are used. Below is copy and past of neurips appendix B.3 which gave the details. Doesn't feel too apropriate for this section right now, but cant think of more appropriate place so ehhh

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. TODO: ppara c and p from neurips, clean?

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. TODO: ppara c and p from neurips, clean?

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 (**??**), (**??**) and (**??**) for backups, but replace the Shannon entropy function $\mathcal{H}$, with a relative entropy function $\mathcal{H}_{\text{relative}}$ in Equation (**??**). 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 TODO: cite and or reffor full details on computing

relative entropy and the search policy used in RENTS. TODO: ppara c and p from neurips, clean?

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 (**??**), (**??**) and (**??**) for backups. This time, we replace the Shannon entropy function $\mathcal{H}$, with a Tsallis entropy function $\mathcal{H}_{\text{Tsallis}}$ in Equation (**??**). Again, we use the same search policy used in TENTS, with the substitution for soft values. See TODO: cite and or reffor how Tsallis entropy is computed and the corresponding search policy for TENTS. TODO: ppara c and p from neurips, clean?

BD: although details of the tournaments is left to appendix <TODO>, of note is that each DP and AR set of algorithms were tuned using the same sequence of tournaments, and then each AR version was played off against the DP version to compare. In each case the AR versions won out and hence were used in the final round robin. <TODO clean up some words about why this might to better: my original hypothesis is that, like using the most visited recommendation poilcy, that the average returns has a smoothing effect on the recommendation poilcy, and lead the algorithms to being less sensitive to noise from the neural nets.>

TODO: Here was the writing where we talked about the above from original neurips Additionally, we found that adapting the algorithms to use average returns (recall Equation (**??**)) outperformed using Bellman backups for Go (Appendix C.0.1). 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 **??**.

4.4.3 Results and Discussion

TODO: Should also eval AR-BTS etc on these envs? Might make the arguments/results nicer and more complete

BD: This subsection first uses the Frozen lake environment to give a demonstration of MDP dependent parameter sensitivity in the Frozen lake, followed by results using optimised parameters, with discussion, on the gridworld envs and Go

TODO: <beg> is original writing on encorporating prior knowledge from neurips - moved from encorporating prior knowledge to results now, as a bit is relevent here This section describes how to use value and policy networks in BTS. Adapting MENTS and DENTS are similar (Appendix **??**). 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. TODO: cite. 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 (**??**)) with $\pi_{\text{BTS,mix}}$:

$$\pi_{\text{BTS,mix}}(a|s) = \lambda_{\tilde{\pi}}\tilde{\pi}(a|s) + (1-\lambda_{\tilde{\pi}})\pi_{\text{BTS}}(a|s) \tag{4.41}$$

$$= \lambda_{\tilde{\pi}}\tilde{\pi}(a|s) + (1-\lambda_{\tilde{\pi}})(1-\lambda_s)\rho_{\text{BTS}}(a|s) + \frac{(1-\lambda_{\tilde{\pi}})\lambda_s}{|\mathcal{A}|}, \tag{4.42}$$

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. TODO: <end> is original writing on encorporating prior knowledge from neurips

Parameter sensitivity in Frozen Lake

TODO: this subsubsection currently just a c and p from neurips

In this section we provide a more detailed discussion on sensitivity to the temperature parameter in MENTS TODO: cite, RENTS and TENTS TODO: cite. Additionally, we provide results using the 8x8 Frozen Lake environment (Figure 4.7a) using a variety of temperatures to demonstrate how each algorithm performs with different temperatures in that domain.

We also ran MENTS, RENTS, TENTS, BTS and DENTS with a variety of temperatures on the 8x8 Frozen Lake environment given in Figure 4.7a. 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 4.9 and 4.11 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 4.10 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 4.9, 4.11 and 4.10 for reference and as a comparison for MENTS, RENTS and TENTS.

DENTS with a constant β

TODO: this subsubsection currently just a c and p from neurips

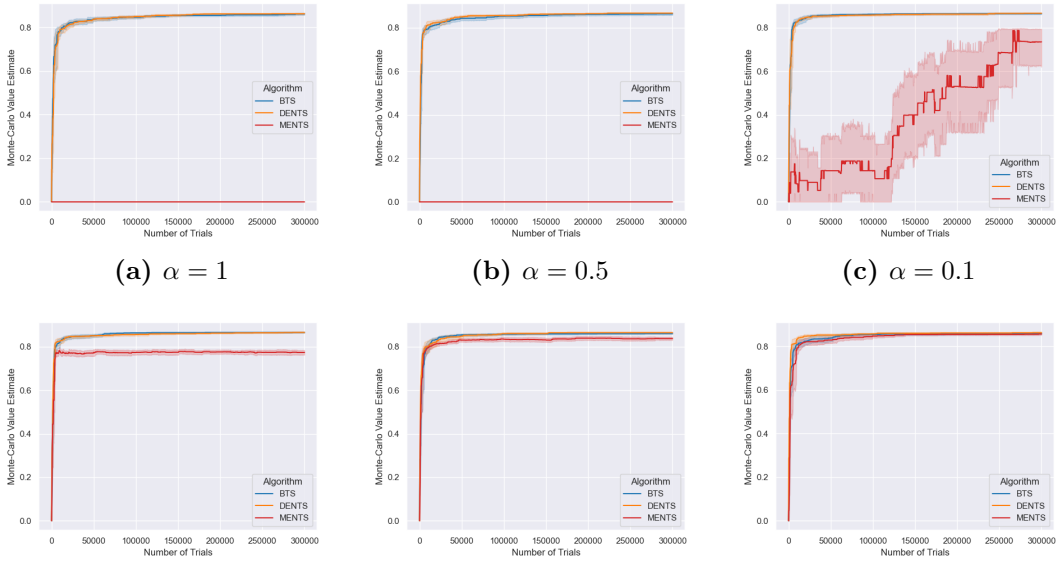
TODO: Place this better? Or maybe just remove from thesis?

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 **??**). Results are given in Figure 4.12. 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.

Gridworld Results

TODO: this subsubsection currently just a c and p from neurips

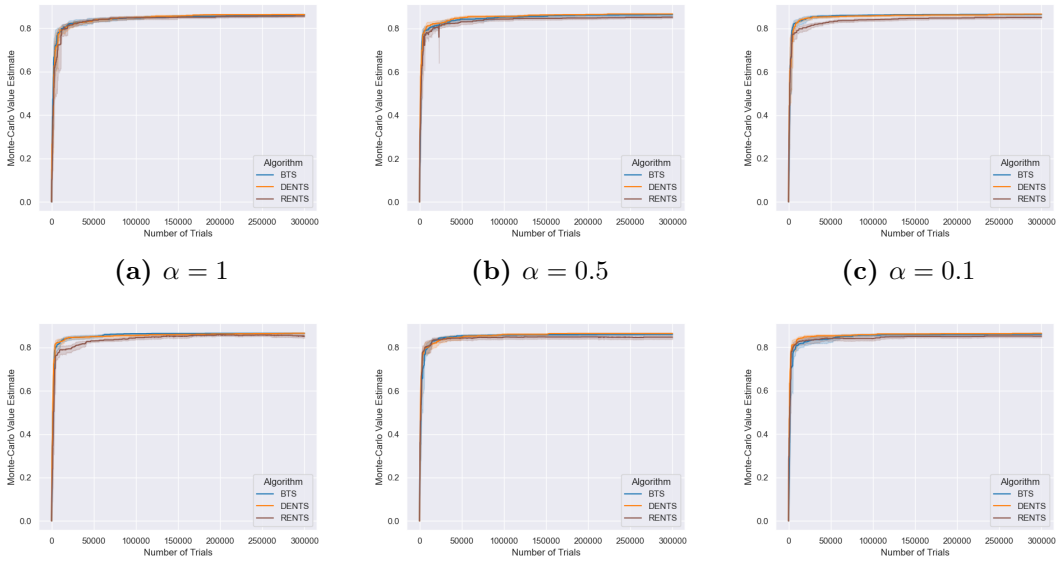
TODO: Ref the above We used an 8x12 Frozen Lake environment and a 6x6 Sailing Problem for evaluation



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

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

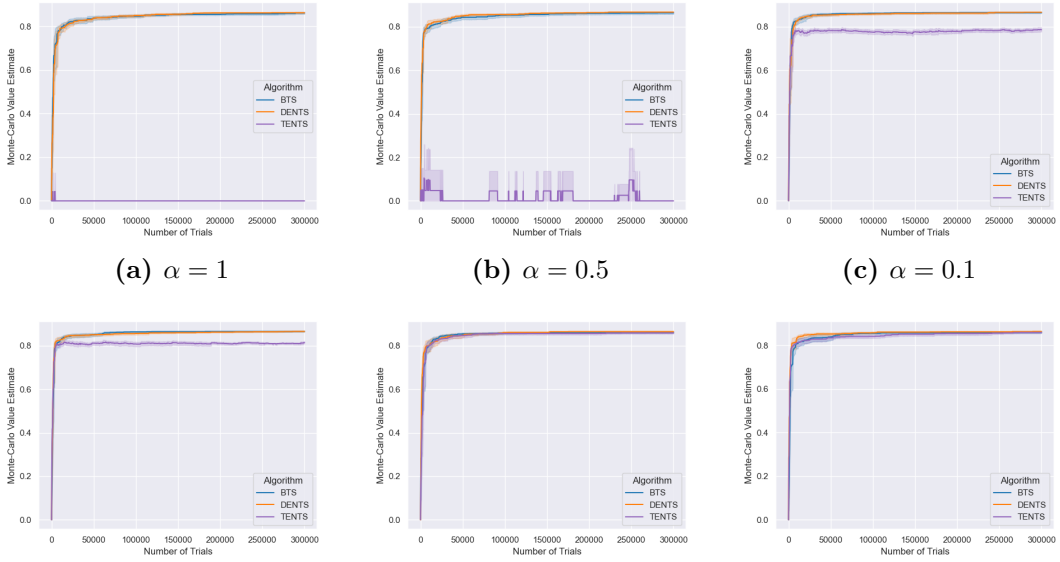
Figure 4.9: MENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference. TODO: Update fig?



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

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

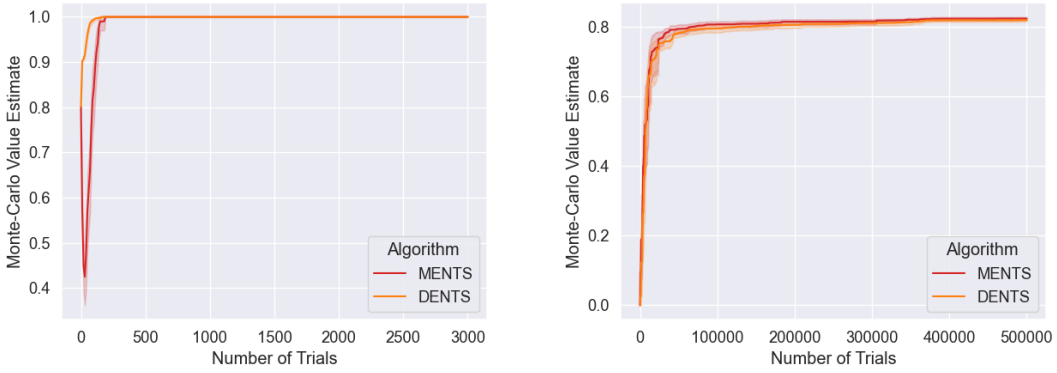
Figure 4.10: RENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference. TODO: Update fig?



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

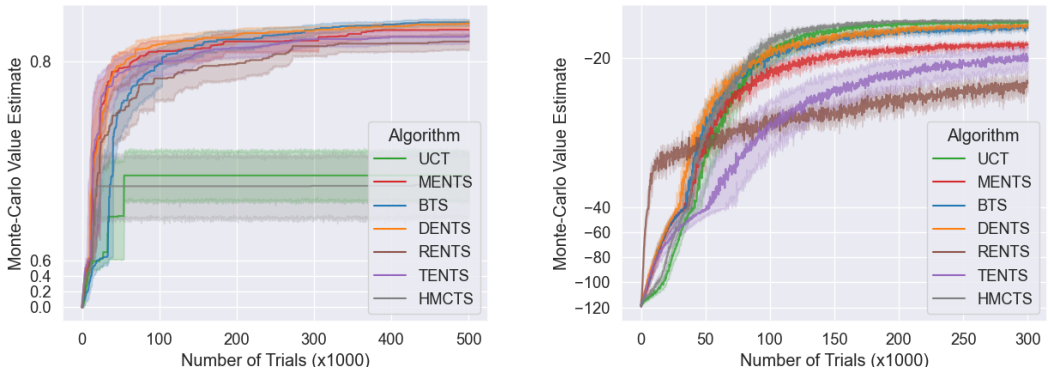
(d) $\alpha = 0.05$ **(e)** $\alpha = 0.01$ **(f)** $\alpha = 0.005$

Figure 4.11: TENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference. TODO: Update fig?



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

Figure 4.12: 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. TODO: Update fig?



(a) 8x12 Frozen Lake. **(b)** 6x6 Sailing Problem.

Figure 4.13: Results for gridworld environments. Further results are given in Appendix ??. TODO: update fig?

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 4.13a), 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 4.13b) 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.

Go Results

TODO: this subsubsection currently just a c and p from neurips

In each match, each algorithm played 50 games as black and 50 as white. TODO: this was moved from another parahraph

Results of the round-robin are summarised in Table 4.1, and we discuss how parameters were selected in Appendix C.0.1. 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 **??** 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 TODO: citealgorithm 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 TODO: cite, 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.

4.5 Theoretical Results

TODO: First thing to do in this section is write up the results for what we're showing (in the chapter). Then write up the results we're showing in the appendix. Then move the proofs needed to build the results showing in this chapter. Then dump all of the remainder in the appendix.

TODO: Double check appendix B coverered properly here

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 4.1: 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.

	DENTS	AR-DENTS	MENTS
TODO: ψ		$\alpha_{\text{AR-D}}(m) \to 0$, $\frac{\beta_{\text{AR-D}}(m)}{\alpha_{\text{AR-D}}(m)} \to 0$	$\alpha_{\text{MENTS}} < \frac{\Delta_{\mathcal{M}}}{3H \log \lvert\mathcal{A}\rvert}$ TODO: H consister
TODO: `mv`	$\frac{\beta_{\text{D}}(m)}{\alpha_{\text{D}}(m)} \to 0$	$\alpha_{\text{AR-D}}(m) \to 0$, $\frac{\beta_{\text{AR-D}}(m)}{\alpha_{\text{AR-D}}(m)} \to 0$	$\alpha_{\text{MENTS}} < \frac{\Delta_{\mathcal{M}}}{3H \log \lvert\mathcal{A}\rvert}$ TODO: H consister

Table 4.2: TODO: Summary of conditions for algorithms to be consistent.

TODO: Here we should front load the "main results" of the theory section, and write them up in a story, for example showing that DENTS can always be made to converge, although you may have to tune results for good performance in practise. THEN give an outline of how the proofs are laid out, and if the proofs are in the appendix or the rest of this chapter.

TODO: Below until end of list is some notes pre writing up theory and bit copied from the neurips paper directly

Should we try to add a result about most visited? Where will be concentration bounds around the number of visits being proportional to the values. So think can bootstrap those proofs.

- Generalise the AR proof to say that for decaying search functions still work (as long as they are bounded) - Have a result that using most visited is theoretically sound

The Neurips paper proof section was structured as follows:

1. First, in Section **??** 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 **??** we introduce preliminary results, that will be useful building blocks for proofs in later Theorems;

3. Third, in Section 4.5.4 we show some general results about soft values that will also be useful later;

4. Fourth, in Section 4.5.5 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 4.5.6 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 4.5.7 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 4.5.8 and 4.5.9 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 4.5.10 we consider results that are relavant for the algorithms using average returns from Section **??**.

TODO: Some comment about using a policy of the form XXX will be refered to a Boltzmann MCTS Process.

4.5.1 MCTS As A Stochastic Process

Now the MENTS, BTS and DENTS are written as a stochastic process, where values are indexed (with a superscript) by the number of times that the corresponding node has been visited. For completeness and reference, the stochastic processes for TODO: UCT?, MENTS, BTS and DENTS are given below.

TODO: change m to n, and use mth when need to reason about prev trials (whole subsection)

When reasoning about a *MCTS stochastic process* the following notations will be helpful.

- The search policy used on the mth trial is π^m, and if the process were stopped after m trials, the recommendation policy that the algorithm would output is denoted ψ^m. Where if the process is run for n trials, then m ranges from 1 to n.

- The m trajectory sampled is $\tau^m = (s_0^m, a_0^m, ..., s_{h-1}^m, a_{h-1}^m, s_h^m)$ TODO: add reward, also make it h subscript m instead of h?, and is sampled using the search policy $\tau^m \sim \pi^m$ (that is $a_i^m \sim \pi^m(\cdot|s_i^m)$ and $s_{i+1}^m \sim p(\cdot|s_i^m, a_i^m)$). TODO: comment that kth trial is run until h where s subscript h is termal in some thtspp sense. Comment about how the superscripts on s a and r will be used when necessary, but not always

- The search tree after m trials is denoted $\mathcal{T}^m$, the initial search tree is $\mathcal{T}^0 = \{s_0\}$, and $\mathcal{T}^k = \mathcal{T}^{k-1} \cup \tau^k$.

When making arguments that apply to multiple algorithms the general policies π and ψ will be used, and when making arguments about specific algorithms the subscripts will be used, such as π_{BTS}. Note that superscript is used to index with respect to the trial, and superscript with parenthasis is used to denote the number of visits.

In the proofs of following sections, it will be useful to write the number of times state s was visited in the first m trials as $N(s, m)$, and the number of times action a was selected from state s in the first m trials as $N(s, a, m)$. Additionally, it will be useful to write these quantities in terms of indicator random variables.

Let $T(s_t, m)$ (and $T(s_t, a_t, m)$) be the set of trajectory indices that s_t was visited on (and action a_t selected) in the first m trials, that is: TODO: this

can avoid using s subscript t

$$T(s_t, m) = \{i | i \leq m, s_t^i = s_t\} \tag{4.43}$$

$$T(s_t, a_t, m) = \{i | i \leq m, s_t^i = s_t, a_t^i = a_t\}. \tag{4.44}$$

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

$$N(s_t, m) = \sum_{i=1}^{m} \mathbb{1}[s_t^i = s_t] = |T(s_t, m)|, \tag{4.45}$$

$$N(s_t, a_t, m) = \sum_{i=1}^{m} \mathbb{1}[s_t^i = s_t, a_t^i = a_t] = |T(s_t, a_t, m)|, \tag{4.46}$$

$$N(s_t, a_t, m) = \sum_{i \in T(s_t, m)} \mathbb{1}[a_t^i = a_t], \tag{4.47}$$

$$N(s_{t+1}, m) = \sum_{i \in T(s_t, a_t, m)} \mathbb{1}[s_{t+1}^i = s_{t+1}]. \tag{4.48}$$

Additionally, the assumption that for any two states $s, s' \in \mathcal{S}$ TODO: double check use calligraphic S for state space? that $s = s'$ if and only if the trajectories leading to them are identical is made. 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). TODO: move this up

TODO: do we do anything at all with the UCT process?

The MENTS Process

Below is a complete summary of the MENTS process, for reference. On the mth trial, the MENTS process follows the search policy:

TODO: make ments section in ch2 use consistent definitions of alphaments, lambdaments and so on

$$\pi_{\text{MENTS}}^m(a|s) = (1 - \lambda^m(s, \epsilon_{\text{MENTS}}))\rho_{\text{MENTS}}^m(a|s) + \frac{\lambda^m(s, \epsilon_{\text{MENTS}})}{|\mathcal{A}|}, \tag{4.49}$$

$$\rho_{\text{MENTS}}^m(a|s) = \exp\left(\frac{1}{\alpha_{\text{MENTS}}}\left(\hat{Q}_{\text{MENTS}}^{(N(s,a,m-1))}(s, a) - \hat{V}_{\text{MENTS}}^{(N(s,m-1))}(s)\right)\right). \tag{4.50}$$

$$\lambda^m(s, x) = \min\left(1, \frac{x}{\log(e + N(s, m-1))}\right), \tag{4.51}$$

where $\epsilon_{\text{MENTS}} \in (0, \infty)$ is an exploration parameters, and α_{MENTS} is the temperature parameter.

The mth trajectory is sampled using the search policy: $\tau^m \sim {\pi_{\text{MENTS}}}^m$. Letting $\tau^m = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h)$, the MENTS value estimates are computed using backups for $t = h-1, ..., 0$: TODO: either add the superscript m into every s and a, or update words to say implicit

$$\hat{V}_{\text{MENTS}}^{(N(s_t,m))}(s_t) = \alpha_{\text{MENTS}} \log \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha_{\text{MENTS}}} \hat{Q}_{\text{MENTS}}^{(N(s_t,a,m))}(s_t, a)\right), \tag{4.52}$$

$$\hat{Q}_{\text{MENTS}}^{(N(s_t,a_t,m))}(s_t, a_t) = R(s_t, a_t) + \sum_{s' \in \text{Succ}(s,a)} \left(\frac{N(s', m)}{N(s_t, a_t, m)} \hat{V}_{\text{MENTS}}^{(N(s',m))}(s')\right). \tag{4.53}$$

TODO: also want something about initialisation and everything in line with thtspp

TODO: recommendation policies and some comment about it being the soft values used?

$$\psi_{\text{MENTS}}^m(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_{\text{MENTS}}^{(N(s,a,m))}(s, a), \tag{4.54}$$

$$\texttt{mv}_{\text{MENTS}}^m(s) = \arg\max_{a \in \mathcal{A}} N(s, a, m). \tag{4.55}$$

The DENTS Process

TODO: this whole sections math is pretty mess and a struggle to fit onto one line at a time. Maybe it suffices to just wordily describe the values of the sufixed values?

Follow policy: TODO: add more wordy things. Some things we said here before were: (1) defining beta $\beta : \mathbb{R} \to [0, \infty)$ be a bounded function; TODO: (2) defined epsilon as exploration policy; (3) in Neurips we defined the propto version of rho and also the full version saying "becasue we need to reason about the search policy we give the exact form," for the moment I've kept both. I think just copy the

full version where relevent in the proofs.

$$\pi_{\mathrm{D}}^{m}(a|s) = (1 - \lambda^{m}(s, \epsilon_{\mathrm{D}}))\rho_{\mathrm{D}}^{m}(a|s) + \frac{\lambda^{m}(s, \epsilon_{\mathrm{D}})}{|\mathcal{A}|}, \tag{4.56}$$

$$\rho_{\mathrm{D}}^{m}(a|s) \propto \exp\left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))}\left(\hat{Q}_{\mathrm{D}}^{(N(s,a,m-1))}(s,a) + \beta_{\mathrm{D}}(N(s,m))\bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a,m-1))}(s,a)\right)\right). \tag{4.57}$$

$$\rho_{\mathrm{D}}^{m}(a|s) = \frac{\exp\left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))}\left(\hat{Q}_{\mathrm{D}}^{(N(s,a,m-1))}(s,a) + \beta_{\mathrm{D}}(N(s,m))\bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a,m-1))}(s,a)\right)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))}\left(\hat{Q}_{\mathrm{D}}^{(N(s,a',m-1))}(s,a') + \beta_{\mathrm{D}}(N(s,m))\bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a',m-1))}(s,a')\right)\right)}. \tag{4.58}$$

$$\lambda^{m}(s,x) = \min\left(1, \frac{x}{\log(e + N(s,m-1))}\right), \tag{4.59}$$

The mth trajectory is sampled using the search policy: $\tau^{m} \sim {\pi_{\text{MENTS}}}^{m}$. Letting $\tau^{m} = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h)$, the DENTS value and entropy estimates are computed using backups for $t = h-1, ..., 0$: TODO: either add the superscript m into every s and a, or update words to say implicit

$$\hat{Q}_{\mathrm{D}}^{(N(s_t,a_t,m))}(s_t,a_t) = R(s_t,a_t) + \sum_{s' \in \mathrm{Succ}(s_t,a_t)}\left(\frac{N(s',m)}{N(s_t,a_t,m)}\hat{V}_{\mathrm{D}}^{(N(s',m))}(s')\right), \tag{4.60}$$

$$\hat{V}_{\mathrm{D}}^{(N(s_t,m))}(s_t) = \max_{a \in \mathcal{A}} \hat{Q}_{\mathrm{D}}^{(N(s_t,a,m))}(s_t,a), \tag{4.61}$$

$$\bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s_t,a_t,m))}(s_t,a_t) = \sum_{s' \in \mathrm{Succ}(s_t,a_t)} \frac{N(s',m)}{N(s_t,a_t,m)}\bar{\mathcal{H}}_{V,\mathrm{D}}^{(N(s',m))}(s'), \tag{4.62}$$

$$\bar{\mathcal{H}}_{V,\mathrm{D}}^{(N(s_t))}(s_t) = \mathcal{H}(\pi_{\mathrm{D}}^{m}(\cdot|s_t)) + \sum_{a \in \mathcal{A}} \pi_{\mathrm{D}}^{m}(a_t|s_t)\bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s_t,a_t,m))}(s_t,a_t). \tag{4.63}$$

TODO: also want something about initialisation and everything in line with tht-spp

TODO: recommendation policies word stuff

$$\psi_{\mathrm{D}}^{m}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_{\mathrm{D}}^{(N(s,a,m))}(s,a), \tag{4.64}$$

$$\mathtt{mv}_{\mathrm{D}}^{m}(s) = \arg\max_{a \in \mathcal{A}} N(s,a,m). \tag{4.65}$$

TODO: From the neurips, also wrote the following (everything left in this subsubsection is copied and cleaned up, but probably needs a proof read) Let

$\hat{V}_{\rho,\mathrm{D}}^{(N(s))}(s)$ be defined as the value:

$$\hat{V}_{\rho,\mathrm{D}}^{(N(s))}(s) = \alpha_{\mathrm{D}}(N(s,m)) \log \left[\sum_{a' \in \mathcal{A}} \exp \left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))} \left(\hat{Q}_{\mathrm{D}}^{(N(s,a'))}(s,a') + \beta_{\mathrm{D}}(N(s,m)) \bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a'))}(s,a') \right) \right) \right], \tag{4.66}$$

and notice that the value of $\exp(\hat{V}_{\rho,\mathrm{D}}^{(N(s,m))}(s)/\alpha_{\mathrm{D}}(N(s,m)))$ is equal to the denominator in Equation (4.54), and so by rearranging we can write ρ_{D}^{m} as

$$\rho_{\mathrm{D}}^{m}(a|s) = \exp \left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))} \left(\hat{Q}_{\mathrm{D}}^{(N(s,a,m))}(s,a) + \beta_{\mathrm{D}}(N(s,m)) \bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a,m))}(s,a) - \hat{V}_{\rho,\mathrm{D}}^{(N(s,m))}(s) \right) \right), \tag{4.67}$$

and subsequently, the full DENTS policy can be written as:

$$\pi_{\mathrm{D}}^{m}(a|s) = (1 - \lambda(s,m)) \exp \left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))} \left(\hat{Q}_{\mathrm{D}}^{(N(s,a,m))}(s,a) + \beta_{\mathrm{D}}(N(s,m)) \bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a))}(s,a) - \hat{V}_{\rho,\mathrm{D}}^{(N(s} \right. \right. \tag{4.68}$$

The AR-DENTS Process

The (AR-)BTS process.

The (AR-)BTS process is a special case of the (AR-)DENTS process when $\beta_{\mathrm{D}}(x) = 0$. As such, results about BTS will be corollaries from proofs about the (AR-)DENTS process.

4.5.2 Preliminaries

Definition 4.5.1. *A sequence of random variables* X_i converges in probability *to a random variable* X*, written as* $X_n \xrightarrow{p} X$*, if for all* $\varepsilon > 0$ *the following holds:*

$$\Pr\left(|X_n - X| > \varepsilon\right) \to 0 \text{ as } n \to \infty. \tag{4.69}$$

Definition 4.5.2. *A sequence of random variables* X_i converges almost surely *to a random variable* X*, written as* $X_n \xrightarrow{as} X$*, if the following holds:*

$$\Pr\left(\lim_{n\to\infty} X_n = X\right) = 1. \tag{4.70}$$

TODO: had this (commented out below) in the Neurips paper, but probably want to actually just define convergence in probability here, and say "letting n tend to infinity" gives the convergence in probability result.

4.5.3 Preliminaries - Exp

TODO: This section contains things related to exponential bounds, which might move to appendix This subsection contains lemmas that will be useful in multiple times and are used to avoid repeating the same argument multiple times.

TODO: change m to n, and use mth when need to reason about prev trials (whole subsection)

Firstly it will be that the union of exponentially unlikely events is also exponentially unlikely, and that the intersection of exponentially likely events is exponentially likely. Although this is a special case of the union bound TODO: ref? TODO: it is stated here so that it can be referenced, because this fact is used regularly TODO: Some comment about how this is just a special case of the union bound (and ref that?) and also

Lemma 4.5.1. *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:*

$$\Pr\left(\bigcup_{i=1}^{\ell} \neg A_i\right) \leq C \exp(-k), \tag{4.71}$$

$$\Pr\left(\bigcap_{i=1}^{\ell} A_i\right) = 1 - \Pr\left(\bigcup_{i=1}^{\ell} \neg A_i\right) \geq 1 - C \exp(-k), \tag{4.72}$$

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

Proof outline. Lemma 4.5.1 is a consequence of the union bound, using $\exp(k_i) \leq \exp(k)$ and simplifying. Inequality (4.68) is just a negation of Inequality (4.67). □

The following two lemma's show that the MENTS and DENTS processes always have a minimum probability of picking any action. TODO: not in good headspace writing up these two lemmas. Do a clean up, double checking maths and that have eliminated using 'we' and so on

Lemma 4.5.2. *For any MENTS process there exists some $\pi^{\min} > 0$, for any state $s_t \in \mathcal{S}$, for all $a_t \in \mathcal{A}$ and any number of trials $m \in \mathbb{N}$, such that $\pi^m_{\text{MENTS}}(a_t|s_t) \geq \pi^{\min}$.*

Proof outline. TODO: Fix N having to be N(s,m) not N(s) etc Define the $Q^{\min}$ function as follows:

$$Q^{\min}(s_t, a_t) = \min_{s_{t+1}, a_{t+1}, \ldots, s_H, a_H} \sum_{i=t}^{H} \min(0, Q^{\text{init}}_{\text{sft}}(s_i), R(s_i, a_i)). \tag{4.73}$$

TODO: clean up with respect to init function

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

$$V^{\max}(s_t) = \alpha_{\text{MENTS}} \log \sum_{a \in \mathcal{A}} \exp(Q^{\max}(s_t, a)/\alpha_{\text{MENTS}}), \tag{4.74}$$

$$Q^{\max}(s_t, a_t) = R(s_t, a_t) + \max_{s_{t+1} \in \text{Succ}(s_t, a_t)} V^{\max}(s_{t+1}). \tag{4.75}$$

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

$$\pi^{\min} = \inf_{\lambda \in [0,1]} \min_{(s,a) \in \mathcal{S} \times \mathcal{A}} (1 - \lambda) \exp\left(\left(Q^{\min}(s, a) - V^{\max}(s)\right)/\alpha\right) + \frac{\lambda}{|\mathcal{A}|}. \tag{4.76}$$

Because the value of an exponential is positive, as is $1/|\mathcal{A}|$, it follows that $\pi^{\min} > 0$. Recall the MENTS policy (Equation (4.45)). By the monotonicity of the exponential function, it follows that for any $s_t \in \mathcal{S}, a_t \in \mathcal{A}, n \in \mathbb{N}$:

$$\pi^m_{\text{MENTS}}(a_t|s_t) = (1 - \lambda(s_t, m)) \exp\left(\left(\frac{1}{\alpha_{\text{MENTS}}}\left(\hat{Q}^{(N(s_t,a_t,m))}_{\text{MENTS}}(s_t, a_t) - \hat{V}^{(N(s_t,m))}_{\text{MENTS}}(s_t)\right)\right)\right) + \frac{\lambda(s_t, m)}{|\mathcal{A}|} \tag{4.77}$$

$$\geq (1 - \lambda(s_t, m)) \exp\left(\left(Q^{\min}(s_t, a_t) - V^{\max}(s_t)\right)/\alpha\right) + \frac{\lambda(s_t, m)}{|\mathcal{A}|} \tag{4.78}$$

$$\geq \pi^{\min}. \tag{4.79}$$

□

TODO: this whole lemma just needs cleaning up really

TODO: this lemma requires $\beta_{\text{D}}(x) < U$ for all x, or β_{D} is bounded. AND it requires $\alpha_{\text{D}}(x) < U'$ so α_{D} to be bounded. TODO: THERE IS CURRENTLY ERROR IN THIS. EXP BOUND REQUIRES A MINIMUM TEMP. Might need to reason around logsumexp's around here. But, as alpha tends to infinity, the probability distribution will tend to a uniform distribution, and min prob is fine,

if alpha tends to zero, then softmax tends to max, and then min prob can be zero, so dont have a min prob. Might need to use a different value of alpha for the computation of V, so might need to have both a lower and upper bound on alpha.

Lemma 4.5.3. *For any DENTS process there exists some $\pi^{\min} > 0$, for any state $s_t \in \mathcal{S}$, for all $a_t \in \mathcal{A}$ and any number of trials $m \in \mathbb{N}$, such that $\pi_{\mathrm{D}}^m(a_t|s_t) \geq \pi^{\min}$.*

Proof outline. This proof outline follows similar reasoning to Lemma 4.5.2. The same definition for $Q^{\min}$ can be used:

$$Q^{\min}(s_t, a_t) = \min_{s_{t+1}, a_{t+1}, \ldots, s_H, a_H} \sum_{i=t}^{H} \min(0, Q_{\mathrm{sft}}^{\mathrm{init}}(s_i), R(s_i, a_i)). \tag{4.80}$$

TODO: clean up with respect to init function

However, the definition of $V^{\max}$ from Equation (4.70) needs to altered to:

$$V^{\max}(s_t) = \max_{a \in \mathcal{A}} Q^{\max}(s_t, a) \tag{4.81}$$

$$Q^{\max}(s_t, a_t) = R(s_t, a_t) + \max_{s' \in \mathrm{Succ}(s_t, a_t)} V^{\max}(s') \tag{4.82}$$

$$V_\rho^{\max}(s_t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\left(Q^{\max}(s_t, a) + \beta_{\max} H \log |\mathcal{A}| \right) / \alpha \right), \tag{4.83}$$

TODO: clean up the use of alpha in this one. Should be alpha max (and also define alpha max), and need to work out if should be using a sup or inf or whatever. Maybe just put a sup outside it. TODO: H was the horizon, make that consistent with what we've defined before (above and in below paragraph) where $\beta_{\max} = \sup_{x \in \mathbb{R}} \beta(x)$. Similarly to the MENTS process case, it can be shown by induction that $Q^{\min}(s_t, a_t) \leq \hat{Q}_{\mathrm{D}}^{(N(s_t, a_t, m))}(s_t, a_t)$ and $V_\rho^{\max}(s_t) \geq V_\rho^{N(s_t, m)}(s_t)$, where the latter implicitly uses that $0 \leq \mathcal{H}_Q^{N(s_t, a_t)}(s_t, a_t) \leq (H - t) \log |\mathcal{A}| \leq H \log |\mathcal{A}|$ TODO: using well-known properties of entropy (left was what was originally written. Consider rewording this entire paragraph tho).

Define $\pi^{\min}$ similarly to Equation (4.72), with the updated definition of $V_\rho^{\max}$:

$$\pi^{\min} = \inf_{\lambda \in [0,1]} \min_{(s,a) \in \mathcal{S} \times \mathcal{A}} (1 - \lambda) \exp\left(\left(Q^{\min}(s, a) - V_\rho^{\max}(s) \right) / \alpha_{\max} \right) + \frac{\lambda}{|\mathcal{A}|}. \tag{4.84}$$

where $\alpha_{\max} = \inf_x \alpha_{\mathrm{D}}(x)$ TODO: double check this is right. Its not. Although exp(1/x) is decreasing in x, which is why we reasoned that it should be max, it

ignores that V rho max depends on alpha too. Also V rho max should be using the . Recalling the DENTS policy (Equation (4.64)), and using similar reasoning to before, as well as $\beta_{\mathrm{D}}(N(s,m))\bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a))}(s,a) \geq 0$, the result follows:

$$\pi_{\mathrm{D}}^{m}(a_t|s_t) = (1-\lambda(s,m))\exp\left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))}\left(\hat{Q}_{\mathrm{D}}^{(N(s,a,m))}(s,a) + \beta_{\mathrm{D}}(N(s,m))\bar{\mathcal{H}}_{Q,\mathrm{D}}^{(N(s,a))}(s,a) - \hat{V}_{\rho,\mathrm{D}}^{(N(s} \right.\right. \tag{4.85}$$

$$\geq (1-\lambda(s,m))\exp\left(\frac{1}{\alpha_{\mathrm{D}}(N(s,m))}\left(\hat{Q}_{\mathrm{D}}^{(N(s,a))}(s,a) - \hat{V}_{\rho,\mathrm{D}}^{(N(s))}(s)\right)\right) + \frac{\lambda(s,m)}{|\mathcal{A}|} \tag{4.86}$$

$$\geq (1-\lambda(s,m))\exp\left(\left(Q^{\min}(s_t,a_t) - V_{\rho}^{\max}(s_t)\right)/\alpha_{\max}\right) + \frac{\lambda(s,m)}{|\mathcal{A}|} \tag{4.87}$$

$$\geq \pi^{\min}. \tag{4.88}$$

□

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

Theorem 4.5.4. *Let $\{X_i\}_{i=1}^{k}$ be indicator random variables (i.e. $X_i \in \{0,1\}$), and $S_k = \sum_{i=1}^{k} X_i$. Then Hoeffding's inequality for indicator random variables states for any $\varepsilon > 0$ that:*

$$\Pr(|S_k - \mathbb{E}S_k| > \varepsilon) \leq 2\exp\left(-\frac{2\varepsilon^2}{k}\right). \tag{4.89}$$

Proof. This is a specific case of Hoeffding's inequality. See TODO: add cite for proof. □

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

Lemma 4.5.5. *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. $\Pr(a_t^i = a_t | s_t^i = s_t) \geq \eta$. And suppose for some $C', k' > 0$ that some event E admits a bound:*

$$\Pr(E) \leq C'\exp(-k'N(s_t,a_t,m)). \tag{4.90}$$

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

$$\Pr(E) \leq C \exp(-kN(s_t, m)). \tag{4.91}$$

Proof. TODO: Define m again somewhere? Recall from Equation (4.43) that $N(s_t, a_t, m) = \sum_{i \in T(s_t,m)} \mathbb{1}[a_t^i = a_t] = \sum_{i \in T(s_t,m)} \mathbb{1}[a_t^i = a_t | s_t^i = s_t]$ TODO: this should be the other sum, from i equal to one to N(st,m). By taking expectations TODO: with respect to at and using the assumed $\Pr(a_t^i = a_t | s_t^i = s_t) \geq \eta$ it follows that $\mathbb{E}N(s_t, a_t, m) \geq \eta N(s_t, m)$ (and more specifically as a consequence $\mathbb{E}N(s_t, a_t, m) - \eta N(s_t, m)/2 \geq \eta N(s_t, m)/2$). The probability of $N(s_t, a_t, m)$ being below a multiplicative ratio of $N(s_t, m)$ is bounded as follows:

$$\begin{aligned}
&\Pr\left(N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) && (4.92)\\
\leq &\Pr\left(N(s_t, a_t, m) < \mathbb{E}N(s_t, a_t, m) - \frac{1}{2}\eta N(s_t, m)\right) && (4.93)\\
= &\Pr\left(\mathbb{E}N(s_t, a_t, m) - N(s_t, a_t, m) > \frac{1}{2}\eta N(s_t, m)\right) && (4.94)\\
\leq &\Pr\left(|\mathbb{E}N(s_t, a_t, m) - N(s_t, a_t, m)| > \frac{1}{2}\eta N(s_t, m)\right) && (4.95)\\
\leq &2\exp\left(-\frac{1}{2}\eta^2 N(s_t, m)\right). && (4.96)
\end{aligned}$$

The first inequality follows from $\mathbb{E}N(s_t, a_t, m) - \eta N(s_t, m)/2 \geq \eta N(s_t, m)/2$, the second line is a rearrangement, the third line comes from the inequality in (4.90) implying the inequality in (4.91), and the final line uses Theorem 4.5.4, a Hoeffding bound for the sum of indicator random variables.

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

$N(s_t, m)$ using the law of total probability as follows:

$$\begin{aligned}\Pr(E) =& \Pr\left(E \middle| N(s_t, a_t, m) \geq \frac{1}{2}\eta N(s_t, m)\right) \Pr\left(N(s_t, a_t, m) \geq \frac{1}{2}\eta N(s_t, m)\right) \\ &+ \Pr\left(E \middle| N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) \Pr\left(N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) \quad (4.97)\\ \leq& \Pr\left(E \middle| N(s_t, a_t, m) \geq \frac{1}{2}\eta N(s_t, m)\right) \cdot 1 \\ &+ 1 \cdot \Pr\left(N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) \quad (4.98)\\ \leq& C' \exp\left(-\frac{1}{2}k'\eta N(s_t, m)\right) + 2\exp\left(-\frac{1}{2}\eta^2 N(s_t, m)\right) \quad (4.99)\\ \leq& C \exp(-kN(s_t, m)), \quad (4.100)\end{aligned}$$

where (4.95) uses the assumed Inequality (4.86) with the condition $N(s_t, a_t) \geq \eta N(s_t)/2$, and also uses the bound from (4.92). The final inequality (4.96) follows from Lemma 4.5.1, with $C = C' + 2$ and $k = \min(-k'\eta/2, -\eta^2/2)$. □

TODO: consider just defining a logsumexp function, and define the temp to have at zero it equal to max. Also finish writing up the preliminaries (commented out below)

Similar to Lemma 4.5.5, it will also be convenient to be able to translate bounds that depend on $N(s')$, for some $s' \in \text{Succ}(s, a)$, into bounds that depend on $N(s, a)$:

Lemma 4.5.6. *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:*

$$\Pr(E) \leq C' \exp(-k'N(s'_t)). \quad (4.101)$$

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

$$\Pr(E) \leq C \exp(-kN(s_t, a_t)). \quad (4.102)$$

Proof outline. Proof is similar to Lemma 4.5.5. 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 (4.44)) as the sum of indicator random variables will give the result. □

4.5.4 Maximum Entropy Reinforcement Learning

TODO: change m to n, and use mth when need to reason about prev trials (whole subsection)

TODO: DEFINITELY APPENDIX MATERIAL

This subsection shows some basic results that relate the maximum entropy and standard objectives, which will be used to show results about MENTS. This subsection temporarily reintroduces the time-step parameter into value functions to simplify other notation. Two results about soft Q-values are given: first, that the optimal standard Q-value is less than the optimal soft Q-value, and secondly, 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.

TODO: recall the max entropy objective, and ref that below

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

$$Q^\pi_{\text{sft}}(s,a;t) = Q^\pi(s,a;t) + \alpha \mathbb{E}_\pi \left[\sum_{i=t+1}^{H} \mathcal{H}\left(\pi(\cdot|s_i)\right) \middle| s_t = s, a_t = a \right], \tag{4.103}$$

$$= Q^\pi(s,a;t) + \alpha \mathcal{H}_{t+1}(\pi | s_t = s, a_t = a), \tag{4.104}$$

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 4.5.7. $Q^*(s,a;t) \leq Q^*_{\text{sft}}(s,a;t)$.

Proof. Taking a maximum over policies in Equation (4.100), and considering that π^*, the optimal standard policy, is one of the possible policies considered in the maximisation, gives the result:

$$Q^*_{\text{sft}}(s,a;t) = \max_\pi \left(Q^\pi(s,a;t) + \alpha \mathcal{H}_{t+1}(\pi | s_t = s, a_t = a) \right) \tag{4.105}$$

$$\geq Q^{\pi^*}(s,a;t) + \alpha \mathcal{H}_{t+1}(\pi^* | s_t = s, a_t = a) \tag{4.106}$$

$$\geq Q^*(s,a;t). \tag{4.107}$$

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

TODO: make below a defn? ALSO THIS DEFN IS NOT APPENDIX MATERIAL

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:

$$\Delta_{s,t} = \min_{(a,a')\in\delta(s,t)} |Q^*(s,a;t) - Q^*(s,a';t)|, \tag{4.108}$$

$$\Delta_{\mathcal{M}} = \min_{s,t} \Delta_{s,t}. \tag{4.109}$$

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

$$Q^*(s,a;t) + \Delta_{\mathcal{M}} \leq Q^*(s,a';t). \tag{4.110}$$

Using this problem dependent constant, $\alpha < \Delta_{\mathcal{M}}/H \log|\mathcal{A}|$ is a sufficient condition for the optimal standard and optimal soft policies to coincide, a consequence of the following Lemma.

Lemma 4.5.8. *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:

$$Q^*_{\text{sft}}(s,a;t) = \max_\pi \left(Q^\pi(s,a;t) + \alpha\mathcal{H}_{t+1}(\pi)\right) \tag{4.111}$$

$$\leq \max_\pi Q^\pi(s,a;t) + \max_\pi \alpha\mathcal{H}_{t+1}(\pi) \tag{4.112}$$

$$= Q^*(s,a;t) + \alpha(H-t)\log|\mathcal{A}| \tag{4.113}$$

$$\leq Q^*(s,a;t) + \alpha H \log|\mathcal{A}|. \tag{4.114}$$

Then, using $\alpha < \Delta_{\mathcal{M}}/H \log |\mathcal{A}|$, $Q^*(s,a;t) + \Delta_{\mathcal{M}} \leq Q^*(s,a';t)$ and $Q^*(s,a';t) \leq Q_{\text{sft}}(s,a;t)$ from Lemma 4.5.7 gives the desired inequality:

$$Q^*_{\text{sft}}(s,a;t) \leq Q^*(s,a;t) + \alpha H \log |\mathcal{A}| \tag{4.115}$$
$$< Q^*(s,a;t) + \Delta_{\mathcal{M}} \tag{4.116}$$
$$\leq Q^*(s,a';t) \tag{4.117}$$
$$\leq Q^*_{\text{sft}}(s,a';t). \tag{4.118}$$

($\Rightarrow$) To show that $Q^*_{\text{sft}}(s,a;t) < Q^*_{\text{sft}}(s,a';t) \Rightarrow Q^*(s,a;t) < Q^*(s,a';t)$ it is easier to 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:

$$Q^*(s,a;t) \geq Q^*(s,a';t) \tag{4.119}$$
$$\Rightarrow Q^*(s,a;t) > Q^*(s,a';t) \tag{4.120}$$
$$\Rightarrow Q^*_{\text{sft}}(s,a;t) > Q^*_{\text{sft}}(s,a';t) \tag{4.121}$$
$$\Rightarrow Q^*_{\text{sft}}(s,a;t) \geq Q^*_{\text{sft}}(s,a';t), \tag{4.122}$$

where the first implication uses that $(a,a') \in \delta(s)$, the second reuses the ($\Leftarrow$) proof. TODO: Try to respect reader more, dont need to explain the last line... (e.g. cut after this), and the final implication holds generally. □

4.5.5 Simple regret

TODO: DEFINITELY APPENDIX MATERIAL

This section revisits *simple regret* in more detail. Recall the definition of simple regret TODO: double check this is correct notation (brain ded today):

$$\texttt{sim\_regr}(s_t, \psi^n) = V^*(s_t) - V^{\psi^n}(s_t), \tag{4.123}$$

where ψ^n is the policy recommended after n rounds or trials. This definition is different to what has been considered by the tree search literature so far TODO: cite: mcts_simple_regret,brue1. However, this definition is a more natural extension

to the simple regret considered for multi-armed bandit problems TODO: cite: simple_regret_short,simple_regret_long, as it stems from a more general MDP planning problem that does not have to be solved using a tree search. For example, consider a non-tree search algorithm that outputs a recommendation policy rather than one action at a time. Besides, the difference can be reconciled by showing that any bound that holds for one definition also holds for the other.

Define the *simple regret of an action* (or *immediate simple regret*) as:

$$\texttt{imm\_regr}_I(s_t, \psi^n) = V^*(s) - \mathbb{E}_{a_t \sim \psi^n(s_t)}[Q^*(s_t, a_t)]. \tag{4.124}$$

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

From this definition of immediate simple regret, TODO: any big-O bounds on simple regret hold if and only if the same big-O bound holds for the immediate simple regret.

Lemma 4.5.9. $\mathbb{E}\texttt{sim\_regr}(s_t, \psi^n) = O(f(n))$ *for all* $s_t \in \mathcal{S}$ *iff* $\mathbb{E}\texttt{imm\_regr}(s_t, \psi^n) = O(f(n))$ *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:

$$\begin{aligned}
\texttt{sim_regr}(s_t, \psi^n) =& V^*(s_t) - V^{\psi^n}(s_t) && (4.125)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s)}[Q^{\psi^n}(s_t, a_t)] && (4.126)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s)}[R(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^{\psi^n}(s_{t+1})]] && (4.127)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s)}\Big[Q^*(s, a) - \mathbb{E}_{s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^*(s_{t+1})] \\
&+ \mathbb{E}_{s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^{\psi^n}(s_{t+1})]\Big] && (4.128)\\
=& V^*(s_t) - \mathbb{E}_{a_t \sim \psi^n(s_t)}[Q^*(s_t, a_t)] \\
&+ \mathbb{E}_{a_t \sim \psi_n(s_t), s_{t+1} \sim \Pr(\cdot|s_t, a_t)}[V^*(s_{t+1}) - V^{\psi^n}(s_{t+1})] && (4.129)\\
=& \texttt{imm_regr}(s, \psi^n) + \mathbb{E}_{a_t \sim \psi^n(s_t), s' \sim \Pr(\cdot|s_t, a_t)}[\texttt{sim_regr}(s_{t+1}, \psi^n)], && (4.130)
\end{aligned}$$

where line (4.124) uses $R(s,a) = Q^*(s,a) - \mathbb{E}_{s' \sim \Pr(\cdot|s,a)}[V^*(s')]$, a rearrangement of the Bellman optimality equation for $Q^*(s,a)$.

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

$$\mathbb{E}\texttt{sim\_regr}(s_t, \psi^n) = \mathbb{E}\left[O(f(n)) + \mathbb{E}_{a_t \sim \psi^n(s_t), s_{t+1} \sim \Pr(\cdot|s_t,a_t)}[O(f(n))]\right] = O(f(n)), \tag{4.131}$$

where the outer expectation is with respect ψ^n (as the recommendation policy is the output of a random process). □

TODO: Commented out corollary below. As we're now making N(s,a,n) explicit, exp(-kN(s,a,n)) is an f(n), so the above lemma is sufficient to handle the below.

TODO: Change any references to cor:imm_to_full_simple_regret to lem:imm_simple_regret instead.

4.5.6 General Q-value convergence result

TODO: This section contains things related to exponential bounds, which might move to appendix

Recall the (soft) Q-value backups used by Boltzmann MCTS processes (Equations (4.49) and (4.56)). Considering that these backups are of identical form, a reward TODO: plus an empirical averaging of child nodes / MC estimate of expected child value. TODO: clean up old writing: 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 4.5.10. *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*

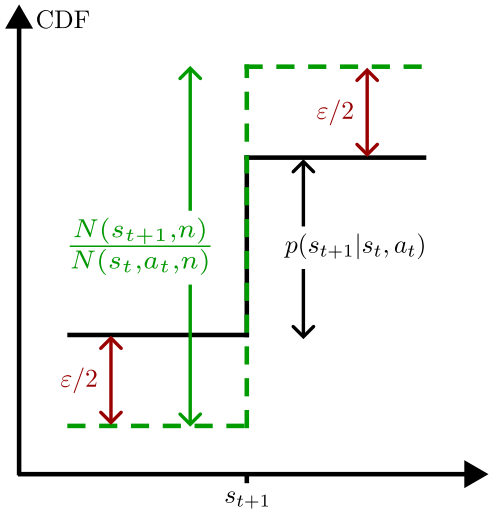


Figure 4.14: Bounding the empirical transition probabilities to the true transition probabilities. The true cdf is shown as a solid black line, the empirical cdf is shown as a dashed green line, and a worst case error of $\varepsilon/2$, using Theorem 4.5.10, is shown in red. The probability mass of $p(s_{t+1}|s_t, a_t)$ and empirical probability mass of $\frac{N(s_{t+1},n)}{N(s_t,a_t,n)}$ is also indicated to demonstrate how the constructed distribution gives Corollary 4.5.10.1.

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:*

$$\Pr\left(\sup_x |F_m(x) - F(x)| > \varepsilon\right) \leq 2\exp\left(-2m\varepsilon^2\right). \tag{4.132}$$

Proof. See TODO: fix cite □

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

Corollary 4.5.10.1. *Consider any Boltzmann MCTS process. For all* $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ *and for all* $\varepsilon > 0$ *we have:*

$$\Pr\left(\max_{s_{t+1}\in \text{Succ}(s_t,a_t)} \left|\frac{N(s_{t+1},n)}{N(s_t,a_t,n)} - p(s_{t+1}|s_t,a_t)\right| > \varepsilon\right) \leq 2\exp\left(-\frac{1}{2}\varepsilon^2 N(s_t,a_t)\right). \tag{4.133}$$

Proof outline. By considering some arbitrary ordering over the successor states in $\text{Succ}(s_t, a_t)$ and applying Theorem 4.5.10, replacing ε by $\varepsilon/2$, the result follows.

To see why the factor of $1/2$ is needed, consider Figure 4.14. Because the distribution is discrete, the cumulative distribution function is a (piecewise constant)

step function. As Theorem 4.5.10 bounds the maximum difference between the empirical and true cumulative distribution functions, the factor of 1/2 is needed to account for the error before and after each s_{t+1} in the worst case. □

Now that Corollary 4.5.10.1 can be used to bound the empirical transition distribution to the true transition distribution, a general purpose concentration inequality for Q-values can be proved for use in later proofs.

Lemma 4.5.11. *Consider any Boltzmann MCTS process, and some state action pair* $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. *Let* $\dot{V}^{N(s,n)}(s) : \mathcal{S} \to \mathbb{R}$, $\dot{V}^*(s) : \mathcal{S} \to \mathbb{R}$ *be some estimated and optimal value functions respectively and suppose that for all* $s_{t+1} \in \text{Succ}(s_t, a_t)$ *that there is some* $C_{s_{t+1}}, k_{s_{t+1}} > 0$ *such that for all* $\varepsilon_0 > 0$*:*

$$\Pr\left(\left|\dot{V}^{N(s_{t+1})}(s_{t+1}) - \dot{V}^*(s_{t+1})\right| > \varepsilon_0\right) \leq C_{s_{t+1}} \exp(-k_{s_{t+1}} \varepsilon_0^2 N(s')). \tag{4.134}$$

If the optimal and estimated Q-values are defined as follows:

$$\dot{Q}^*(s,a) = R(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\dot{V}^*(s')\right], \tag{4.135}$$

$$\dot{Q}^{N(s,a,n)}(s,a) = R(s,a) + \sum_{s' \in \text{Succ}(s,a)} \left[\frac{N(s',n)}{N(s,a,n)} \dot{V}^{N(s',n)}(s')\right]. \tag{4.136}$$

Then there exists some $C, k > 0$*, for all* $\varepsilon > 0$ *such that:*

$$\Pr\left(\left|\dot{Q}^{N(s,a,n)}(s,a) - \dot{Q}^*(s,a)\right| > \varepsilon\right) \leq C \exp(-k\varepsilon^2 N(s,a,n)). \tag{4.137}$$

Proof. By the assumed bounds, Lemma 4.5.1 and Lemma 4.5.6, there is some $C_1, k_1 > 0$, such that for any $\varepsilon_1 > 0$:

$$\Pr\left(\forall s_{t+1}. \left|\dot{V}^{N(s_{t+1})}(s_{t+1}) - \dot{V}^*(s_{t+1})\right| \leq \varepsilon_1\right) > 1 - C_1 \exp(-k_1 \varepsilon_1^2 N(s_t, a_t, n)). \tag{4.138}$$

And recall that for any $p(s_{t+1}|s_t, a_t) > \varepsilon_2 > 0$, using Corollary 4.5.10.1 that:

$$\Pr\left(\max_{s_{t+1} \in \text{Succ}(s_t, a_t)} \left|\frac{N(s_{t+1}, n)}{N(s_t, a_t, n)} - p(s_{t+1}|s_t, a_t)\right| \leq \varepsilon_2\right) > 1 - 2\exp\left(-\frac{1}{2}\varepsilon_2^2 N(s_t, a_t, n)\right). \tag{4.139}$$

If the events in Inequalities (4.134) and (4.135) hold, then the following inequalities must also hold:

$$\dot{V}^*(s_{t+1}) - \varepsilon_1 \leq \dot{V}^{N_n(s_{t+1},n)}(s_{t+1}) \leq \dot{V}^*(s_{t+1}) + \varepsilon_1 \tag{4.140}$$

$$p(s_{t+1}|s_t, a_t) - \varepsilon_2 \leq \frac{N(s_{t+1}, n)}{N(s_t, a_t, n)} \leq p(s_{t+1}|s_t, a_t) + \varepsilon_2. \tag{4.141}$$

The upper bounds on $\dot{V}^{N_n(s_{t+1},n)}(s_{t+1})$ and $N(s_{t+1}, n)/N(s_t, a_t, n)$ can be used to obtain an upper bound on $\dot{Q}^{N(s_t,a_t,n)}(s_t, a_t)$:

$$\dot{Q}^{N(s_t,a_t,n)}(s_t, a_t) \tag{4.142}$$

$$= R(s_t, a_t) + \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \frac{N(s_{t+1}, n)}{N(s_t, a_t, n)} \dot{V}^{N(s_{t+1},n)}(s_{t+1}) \tag{4.143}$$

$$\leq R(s_t, a_t) + \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} (p(s_{t+1}|s_t, a_t) + \varepsilon_2)(\dot{V}^*(s_{t+1}) + \varepsilon_1) \tag{4.144}$$

$$= R(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t,a_t)}[\dot{V}^*(s_{t+1})] + \varepsilon_1 + \varepsilon_2 \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1}) + \varepsilon_1\varepsilon_2 \tag{4.145}$$

$$\leq R(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t,a_t)}[\dot{V}^*(s_{t+1})] + \varepsilon_2 \left| \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1}) \right| + \varepsilon_1 + \varepsilon_1\varepsilon_2 \tag{4.146}$$

$$= \dot{Q}^*(s_t, a_t) + \varepsilon_2 \left| \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1}) \right| + \varepsilon_1 + \varepsilon_1\varepsilon_2. \tag{4.147}$$

Following the similar reasoning but using the lower bounds on $\dot{V}^{N_n(s_{t+1},n)}(s_{t+1})$ and $N(s_{t+1}, n)/N(s_t, a_t, n)$ gives: TODO: There is some special cases when V is negative, having to use the other bound, but the upper and lower bound have a bit of leway which handles all four cases of using plus minus epsilon one and two.

$$\dot{Q}^{N(s_t,a_t,n)}(s_t, a_t) \geq \dot{Q}^*(s_t, a_t) - \varepsilon_2 \left| \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1}) \right| - \varepsilon_1 - \varepsilon_1\varepsilon_2. \tag{4.148}$$

Now given an arbitrary $\varepsilon > 0$, recalling that $\varepsilon_1, \varepsilon_2 > 0$ were arbitrary, set $\varepsilon_1 = \varepsilon/3$ and $\varepsilon_2 = \min(\varepsilon/3, \varepsilon/3|\sum_{s_{t+1}} \dot{V}^*(s_{t+1})|)$ to give:

$$\dot{Q}^*(s_t, a_t) - \varepsilon \leq \dot{Q}^{N(s_t,a_t,n)}(s_t, a_t) \leq \dot{Q}^*(s_t, a_t) + \varepsilon. \tag{4.149}$$

Using Lemma 4.5.1 liberally, there is some $C_2, C_3, k_2, k_3 > 0$ such that:

$$\Pr\left(\left|\dot{Q}^{N(s_t,a_t,n)}(s_t,a_t) - \dot{Q}^*(s_t,a_t)\right| \leq \varepsilon\right) \tag{4.150}$$

$$> \left(1 - C_1 \exp(-k_1\varepsilon_1^2 N(s_t,a_t,n))\right)\left(1 - 2\exp\left(-\frac{1}{2}\varepsilon_2^2 N(s_t,a_t,n)\right)\right) \tag{4.151}$$

$$= \left(1 - C_2 \exp(-k_2\varepsilon^2 N(s_t,a_t,n)\right) \cdot \left(1 - C_3 \exp(-k_3\varepsilon^2 N(s_t,a_t,n)\right) \tag{4.152}$$

$$> 1 - C_2 \exp(-k_2\varepsilon^2 N(s_t,a_t,n)) - C_3 \exp(-k_3\varepsilon^2 N(s_t,a_t,n)). \tag{4.153}$$

Finally, by negating and setting $C = C_2 + C_3$ and $k = \min(k_2, k_3)$ in Lemma TODO: ref the result follows:

$$\Pr\left(\left|\dot{Q}^{N(s_t,a_t,n)}(s_t,a_t) - \dot{Q}^*(s_t,a_t)\right| > \varepsilon\right) \leq C \exp(-k\varepsilon^2 N(s_t,a_t,n)). \tag{4.154}$$

□

4.5.7 MENTS results

This subsection provides results related to MENTS in the setting of that standard reinforcement learning objective. Concentration inequalities are given around the optimal soft values to start with, although this result is similar to soem of the results provided in [35] TODO: this is still provided to keep this thesis' proofs self contained. (although its not as we use lots of things like hoeffdings, and also use UCT style proofs prsumably later on.) We could say "to keep the proofs compatible with the other results provided. Or we could just omit it and reference the MENTS paper..." Afterwards, the concentration inequalities are combined with results about maximum entropy reinforcement learning TODO: ref the section where we did that to provide 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 4.5.12. *Consider a MENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ TODO: update for thtspp notation. If for all $s_{t+1} \in \bigcup_{a\in\mathcal{A}} \text{Succ}(s_t, a)$ there is some $C_{s_{t+1}}, k_{s_{t+1}} > 0$ for any $\varepsilon_{s_{t+1}} > 0$:*

$$\Pr\left(\left|\hat{V}_{\text{MENTS}}^{(N(s_{t+1},n))}(s_{t+1}) - V_{\text{sft}}^*(s_{t+1})\right| > \varepsilon_{s_{t+1}}\right) \leq C_{s_{t+1}} \exp\left(-k_{s_{t+1}}\varepsilon_{s_{t+1}}^2 N(s_{t+1},n)\right), \tag{4.155}$$

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

$$\Pr\left(\left|\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) - V^*_{\text{sft}}(s_t)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 N(s_t,n)\right). \tag{4.156}$$

Proof. Given the assumptions and by Lemmas 4.5.11, 4.5.5 and 4.5.1, there is some $C, k > 0$ such that for any $\varepsilon > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}.\left|\hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t,a_t) - Q^*_{\text{sft}}(s_t,a_t)\right| \leq \varepsilon\right) > 1 - C\exp(-k\varepsilon^2 N(s_t,n)). \tag{4.157}$$

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

$$Q^*_{\text{sft}}(s_t,a_t) - \varepsilon \leq \hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t,a_t) \leq Q^*_{\text{sft}}(s_t,a_t) + \varepsilon. \tag{4.158}$$

Using the upper bound on $\hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t,a_t)$ in the soft backup equation for $\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t)$ (Equation (4.48) gives:

$$\begin{aligned}
\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) &= \alpha\log\sum_{a\in\mathcal{A}}\exp\left(\frac{\hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t,a_t)}{\alpha}\right) && (4.159)\\
&\leq \alpha\log\sum_{a\in\mathcal{A}}\exp\left(\frac{Q^*_{\text{sft}}(s_t,a_t)+\varepsilon}{\alpha}\right) && (4.160)\\
&= \left[\alpha\log\sum_{a\in\mathcal{A}}\exp\left(\frac{Q^*_{\text{sft}}(s_t,a_t)}{\alpha}\right)\right] + \varepsilon && (4.161)\\
&= V^*_{\text{sft}}(s_t) + \varepsilon, && (4.162)
\end{aligned}$$

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

$$\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) \geq V^*_{\text{sft}}(s_t) - \varepsilon, \tag{4.163}$$

and hence:

$$|\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) - V^*_{\text{sft}}(s_t)| \leq \varepsilon. \tag{4.164}$$

This therefore shows: TODO: some comment about how if A implies B then pr(B) more than pr(A)

$$\Pr\left(\left|\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) - V^*_{\text{sft}}(s_t)\right| \leq \varepsilon_1\right) > 1 - C\exp(-k\varepsilon^2 N(s_t,n)), \tag{4.165}$$

and negating probabilities gives the result. □

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

Theorem 4.5.13. *Consider a MENTS process, let $s_t \in \mathcal{S}$ then there is some $C, k > 0$ for any $\varepsilon > 0$:*

$$\Pr\left(\left|\hat{V}_{\text{MENTS}}^{(N(s_t,n))}(s_t) - V_{\text{sft}}^*(s_t)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 N(s_t, n)\right). \tag{4.166}$$

Moreover, at the root node s_0:

$$\Pr\left(\left|\hat{V}_{\text{MENTS}}^{(N(s_0,n))}(s_0) - V_{\text{sft}}^*(s_0)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 n\right). \tag{4.167}$$

And hence $\hat{V}_{\text{MENTS}}^{(N(s_t,n))}(s_t) \xrightarrow{p} V_{\text{sft}}^*(s_t)$.

Proof. Consider that the result holds for $t = H + 1$, because $\hat{V}_{\text{MENTS}}^{(N(s_{H+1},n))}(s_{H+1}) = V_{\text{sft}}^*(s_{H+1}) = 0$. Therefore the result holds for any $t = 0, ..., H+1$ by induction using Lemma 4.5.12. Noting that $N(s_0) = n$ gives (4.163). TODO: To see the convergence in probability, let $n \to \infty$. TODO: Double check the H stuf is consistent with the thtspp and MDP defns □

Provided MENTS's temperature parameter is set small enough, such that the optimal standard and soft values lead to identical recommendation policies TODO: ref the max entropy proof of this, then MENTS is consistent and the expected simple regret tends to zero exponentially. This is shown in the following lemma.

Lemma 4.5.14. *Consider a MENTS process with $\alpha_{\text{MENTS}} < \Delta_{\mathcal{M}}/3H \log|\mathcal{A}|$. TODO: Make sure replaced alpha with alphaments in ments theorems Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\texttt{sim\_regr}(s_t, \psi_{\text{MENTS}}^n) \leq C' \exp\left(-k' N(s_t, n)\right). \tag{4.168}$$

Proof. TODO: Why cant we just replace this with, the values converge in probability from the previous result, and then use that it means the recommendation policy converges in probability, and then use that to say simple regret go zero? The case where it doesnt work is if there is suboptimal soft value which equals the optimal soft

value (which has no entropy), in this case, becase we need the value ESTIMATES to converge, not the optimal values, we need the temperature to be a bit stricter. I think

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 4.5.8 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 4.5.13 and Lemmas 4.5.11 and 4.5.5 there exists $C_1, k_1 > 0$ such that for all $\varepsilon_1 > 0$: TODO: reword this paragraph, its a bit meh

$$\Pr\left(\forall a_t \in \mathcal{A} \left| \hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t) - Q^*_{\text{sft}}(s_t, a_t) \right| \leq \varepsilon_1\right) > 1 - C_1 \exp(-k_1 \varepsilon_1^2 N(s_t, n)). \tag{4.169}$$

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

$$Q^*_{\text{sft}}(s_t, a_t) - \Delta_{\mathcal{M}}/3 \leq \hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t) \leq Q^*_{\text{sft}}(s_t, a_t) + \Delta_{\mathcal{M}}/3. \tag{4.170}$$

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

$$\begin{aligned}
\hat{Q}^{(N(s_t,a,n))}_{\text{MENTS}}(s_t, a) \leq & Q^*_{\text{sft}}(s_t, a) + \Delta_{\mathcal{M}}/3 && (4.171)\\
\leq & Q^*(s_t, a) + \alpha H \log |\mathcal{A}| + \Delta_{\mathcal{M}}/3 && (4.172)\\
\leq & Q^*(s_t, a) + 2\Delta_{\mathcal{M}}/3 && (4.173)\\
\leq & Q^*(s_t, a^*) - \Delta_{\mathcal{M}}/3 && (4.174)\\
\leq & Q^*_{\text{sft}}(s_t, a^*) - \Delta_{\mathcal{M}}/3 && (4.175)\\
\leq & \hat{Q}^{(N(s_t,a^*,n))}_{\text{MENTS}}(s_t, a^*). && (4.176)
\end{aligned}$$

Where in the above, the first line holds from the upper bound on $\hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t)$; The second holds from maximising the standard return and entropy portions of the soft value separately (recall Inequality (4.108) in Lemma 4.5.8); The third holds from the assumption on α; The fourth holds from the definition of $\Delta_{\mathcal{M}}$ (also see Inequality (4.106)); The fifth holds from the optimal soft value being greater than

the optimal standard value (Lemma 4.5.7); And the final line holds by using the lower bound on $\hat{Q}_{\text{MENTS}}^{(N(s_t,a_t,n))}(s_t,a_t)$ given above with $a_t = a^*$.

Negating the probability that (4.172) holds gives:

$$\Pr\left(\exists a_t \in \mathcal{A} - \{a^*\}.\left(\hat{Q}_{\text{MENTS}}^{(N(s_t,a))}(s_t,a) > \hat{Q}_{\text{MENTS}}^{(N(s_t,a^*,n))}(s_t,a^*)\right)\right) \leq C_1 \exp(-k_2 N(s_t)). \tag{4.177}$$

The expected immediate regret can be bounded as follows:

$$\begin{aligned}
&\mathbb{E}\texttt{inst_regr}(s_t, \psi_{\text{MENTS}}^n) && (4.178)\\
&= \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\psi_{\text{MENTS}}^n(s_t) = a\right) && (4.179)\\
&= \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(a = \arg\max_{a'} \hat{Q}_{\text{MENTS}}^{(N(s_t,a',n))}(s_t,a')\right) && (4.180)\\
&\leq \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\hat{Q}_{\text{MENTS}}^{(N(s_t,a,n))}(s_t,a) > \hat{Q}_{\text{MENTS}}^{(N(s_t,a^*,n))}(s_t,a^*)\right) && (4.181)\\
&\leq \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) C_1 \exp(-k_2 N(s_t,n)), && (4.182)
\end{aligned}$$

where $k' = k_2$ and $C' = C_1 \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a))$. Finally, using TODO: ref lemma gives the result. □

In consequence to Lemma 4.5.14, provided the sufficient conditions are met, MENTS is consistent and will converge to a simple regret of zero.

Theorem 4.5.15. *Consider a MENTS process with $\alpha_{\text{MENTS}} < \Delta_{\mathcal{M}}/3H \log|\mathcal{A}|$. TODO: H consistent with thtspp Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ TODO: H consistent with thtspp then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\texttt{sim\_regr}(s_t, \psi_{\text{MENTS}}^n) \leq C' \exp\left(-k' N(s_t,n)\right). \tag{4.183}$$

And specifically, at the root node s_0:

$$\mathbb{E}\texttt{sim\_regr}(s_0, \psi_{\text{MENTS}}^n) \leq C' \exp\left(-k' n)\right). \tag{4.184}$$

Proof. This theorem holds as a consequence of Corollary **??** and Lemma 4.5.14, and noting that at the root node $N(s_0,n) = n$. □

Figure 4.15: TODO: Make fig and write caption for the MDP for the new proof.

TODO: this was a theorem in the main neurips paper. Need to decide how writing this up, so this may go TODO: We have now shown Theorem **??**:

Theorem 3.2. *For any MDP $\mathcal{M}$, after running n trials of the MENTS algorithm with $\alpha_{\text{MENTS}} \leq \Delta_{\mathcal{M}}/3H \log |\mathcal{A}|$, TODO: H consistent with thtspp there exists constants $C, k > 0$ such that: $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi^n_{\text{MENTS}})] \leq C \exp(-kn)$, 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 4.5.15. □

TODO: Want to change this result (below) to be, for any temperature alpha, there is some MDP that MENTS is not consistent. Then for theory story it can be that there is always a way to set params in DENTS that it will converge, but for MENTS the parameters depends on the MDP. The MDP to do this has one initial choice, with a reward of one, and then the other option is an entropy chain again, where the length is long enough such that MENTS will chose the entropy chain option.

Proposition 3.1. *TODO: Adapt this argument, which is just c and p from the neurips appendix There exists an MDP $\mathcal{M}$ and temperature α such that $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi^n_{\text{MENTS}})] \not\to 0$ as $n \to \infty$. That is, MENTS is not consistent.*

Proof. TODO: Adapt this argument, which is just c and p from the neurips appendix We give a proof by construction. Recall the modified 10-chain problem, with $R_f = 1/2$ in Figure **??**, and consider a MENTS process (i.e. running MENTS) with a temperature $\alpha_{\text{MENTS}} = 1$ for n trials. By considering the optimal soft Bellman

equations (??) and (??), one can verify that $Q^*_{\text{sft}}(1,2) = 0.9$ and $Q^*_{\text{sft}}(1,1) = \log\left(\exp(1/2) + \sum_{i=0}^{8}\exp(i/10)\right) \approx 2.74$.

Theorem 4.5.13, Lemma 4.5.11 and Lemma 4.5.5 implies that there is some $C, k > 0$ for any $\varepsilon > 0$:

$$\Pr\left(\left|\hat{Q}^{(N(1,1,n))}_{\text{MENTS}}(1,1) - Q^*_{\text{sft}}(1,1)\right| > \varepsilon\right) \leq C\exp(-k\varepsilon^2 N(1,n)) = C\exp(-k\varepsilon^2 n). \tag{4.185}$$

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

$$\begin{aligned}\Pr\left(\hat{Q}^{(N(1,1,n))}_{\text{MENTS}}(1,1) < 3/2\right) \leq& \Pr\left(\left|\hat{Q}^{(N(1,1,n))}_{\text{MENTS}}(1,1) - Q^*_{\text{sft}}(1,1)\right| > 1\right) &(4.186)\\ \leq& C\exp(-kn). &(4.187)\end{aligned}$$

And hence:

$$\Pr(\psi^n_{\text{D}}(1) = 1) > 1 - C\exp(-kn) \tag{4.188}$$

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 $\Pr(\psi^n_{\text{MENTS}}(1) = 1) > 1/2$, and hence:

$$\begin{aligned}\mathbb{E}\texttt{sim\_regr}(1, \psi^n_{\text{MENTS}}) >& \frac{1}{10}\cdot\Pr(\psi^n_{\text{MENTS}}(1) = 1) &(4.189)\\ >& \frac{1}{20}. &(4.190)\end{aligned}$$

Thus $\mathbb{E}\texttt{sim\_regr}(1, \psi^n_{\text{D}}) \not\to 0$. □

Finally for the MENTS proofs, an MDP can be constructed such that for any setting of α_{MENTS} MENTS will either not be consistent, or, will take exponentially long in the size of the state space of the MDP. This MDP is the TODO: whatever call the entropy trap env first seen in TODO: ref to toy envs section figure, where this phenominon was demonstrated empirically TODO: ref to graph in toy envs section. This figure is repeated in Figure 4.16 TODO: for ease of reading.

Theorem 4.5.16. *Consider a MENTS process with arbitrary temperature α_{MENTS}. There exists and MDP such that for any α_{MENTS} that MENTS is either not consistent,*

Figure 4.16: TODO: haven't edited this from the neurips paper, need to reproduce the fig and comment with the new notation 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.

or requires an exponential number of trials in the size of the state space. More precisely, either $\mathbb{E}\texttt{sim\_regr}(s_0, \psi^n_{\text{MENTS}}) \not\to 0$ *or* $\mathbb{E}\texttt{sim\_regr}(1, \psi^n_{\text{MENTS}}) \geq c(1 - \frac{n}{k^{|\mathcal{S}|}})$*, which implies that* $\mathbb{E}\texttt{sim\_regr}(s_0, \psi^n_{MENTS}) > 0$ *for* $n < k^{|\mathcal{S}|}$.

Proof outline. TODO: remember this one was written in a bit of a rush, so probably want to make it a little clearer. Also need to make it consistent with the states and actions notation

TODO: Also a bunch of N(s) instead of N(s,n) in the proof that need to be fixed

Proof is by construction. Consider the adapted-chain MDP TODO: update name for name change? defined in Figure 4.16, which is parameterised by D the length of the UCT gauntlet, and K, half the number of states in the entropy trap. To prove the claim, two cases need to be considered: when the temperature is sufficiently

high for MENTS to get 'caught' in the entropy trap when it is inconsistent; and, when the temperature is lower than this threshold.

Case 1: $\alpha_{\text{MENTS}} > \frac{1}{\log(2)K}$ (MENTS gets caught by the entropy trap).

If $\alpha_{\text{MENTS}} > \frac{1}{\log(2)K}$, 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 TODO: only for the entropy trap section):

$$V^*_{\text{sft}}(E) = 0 + \alpha_{\text{MENTS}} \cdot \mathcal{H}(\phi) \tag{4.191}$$

$$= \alpha_{\text{MENTS}} \cdot \log(2)K \tag{4.192}$$

$$> 1 \tag{4.193}$$

$$= V^*_{\text{sft}}(F). \tag{4.194}$$

Hence the optimal soft values (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 $\mathbb{E}\texttt{sim\_regr}(1, \psi^n_{\text{MENTS}}) \to 1 > 0$.

Case 2: $\alpha \leq \frac{1}{\log(2)K}$ TODO: asnother comment on this case, or remove comment from other case

In this case it is argued that with a low α_{MENTS} that MENTS will only have a low probability of ever 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:

$$\hat{V}^{(N(i,n))}_{\text{MENTS}}(i) = \alpha \log\left(\frac{1}{\alpha}\left(\frac{D-i}{D} + \hat{V}^{(N(i+1,n))}_{\text{MENTS}}(i+1)\right)\right) \tag{4.195}$$

$$\leq \max\left(\frac{D-i}{D}, \hat{V}^{(N(i+1,n))}_{\text{MENTS}}(i+1)\right) + \alpha\log(2) \tag{4.196}$$

$$\leq \max\left(\frac{D-i}{D}, \hat{V}^{(N(i+1,n))}_{\text{MENTS}}(i+1)\right) + \frac{1}{K} \tag{4.197}$$

where inequality (4.192) 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 $\hat{V}^{(N(D))}_{\text{MENTS}}(D) = 0$ (it will be

checked that these assumptions are valid later). Then, by induction, it holds that $\hat{V}^{(N(i))}_{\text{MENTS}}(i) \leq \frac{D-(i-1)}{D}$:

$$\hat{V}^{(N(i))}_{\text{MENTS}}(i) \leq \max\left(\frac{D-i}{D}, \hat{V}^{(N(i+1))}_{\text{MENTS}}(i+1)\right) + \frac{\log(2)}{\log(K)} \tag{4.198}$$

$$\leq \frac{D-i}{D} + \frac{1}{K} \tag{4.199}$$

$$\leq \frac{D-i}{D} + \frac{1}{D} \tag{4.200}$$

$$= \frac{D-(i-1)}{D}. \tag{4.201}$$

TODO: first line uses induction hypothesis, value at i plus one less than D-i/D.

Now let the event $Y(n)$ be that MENTS visits state D in its nth trial. Again assuming that $\hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0$, the probability of this event occuring is less than 2^{-D}. Because given this assumption it has just been shown that $\hat{V}^{(N(i+1,n))}_{\text{MENTS}}(i+1) \leq \frac{D-i}{D} = \hat{V}^{(N(G_i,n))}_{\text{MENTS}}(G_i)$, TODO: we are considering the choice between Gi and i plus 1 from state i here, which isn't too clear right now as there are only two actions and taking action a_R to continue down the chain has a lower soft value estimate, it must be that $\pi^n_{\text{MENTS}}(a_R|i) < \frac{1}{2}$, and as such, it must be that $\Pr\left(Y(n)\middle|\hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0\right) < \frac{1}{2^D}$.

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 $\hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0$: TODO: probably aught to say somewhere in the proofs section that all the initialisations are set to zero, and the theory is to analyse the algorithms by itself,

without any informative prior knowledge of the mdp

$$\begin{aligned}
\Pr(Z(n)) &= \Pr\left(Z(n) \cap \hat{V}_{\text{MENTS}}^{(N(D,n))}(D) = 0\right) && (4.202)\\
&= \Pr\left(\neg Y(n) \cap Z(n-1) \cap \hat{V}_{\text{MENTS}}^{(N(D,n))}(D) = 0\right) && (4.203)\\
&= \Pr\left(\neg Y(n) \middle| Z(n-1) \cap \hat{V}_{\text{MENTS}}^{(N(D,n))}(D) = 0\right) \Pr\left(Z(n-1) \cap \hat{V}_{\text{MENTS}}^{(N(D,n))}(D) = 0\right) && (4.204)\\
&\geq (1 - 2^{-D}) \Pr\left(Z(n-1) \cap \hat{V}_{\text{MENTS}}^{(N(D,n))}(D) = 0\right) && (4.205)\\
&= \ldots && (4.206)\\
&\geq (1 - 2^{-D})^n \Pr\left(Z(0) \cap \hat{V}_{\text{MENTS}}^{(N(D))}(D) = 0\right) && (4.207)\\
&= (1 - 2^{-D})^n && (4.208)\\
&\geq 1 - n2^{-D}, && (4.209)
\end{aligned}$$

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

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:

$$\begin{aligned}
\mathbb{E}\texttt{sim_regr}(1, \psi_{\text{MENTS}}^n) &\geq \left(1 - \frac{9}{10}\right) \Pr(Z(n)) && (4.210)\\
&\geq \frac{1}{10}\left(1 - n2^{-D}\right). && (4.211)
\end{aligned}$$

Finally, setting $K = D$, so that $|\mathcal{S}| = 4D + 3 < 5D$ for $D \in \mathbb{N}$ TODO: D more than 1 and so $D = \frac{|\mathcal{S}|-3}{4} > \frac{|S|}{5}$. Substituting this into the above inequality gives $\mathbb{E}\texttt{sim\_regr}(1, \psi_{\text{MENTS}}^n) \geq 1 - \frac{n}{\sqrt[5]{2}^{|S|}}$, which is greater than 0 for $n < \sqrt[5]{2}^{|S|}$. That is $c = \frac{1}{10}$ and $k = \sqrt[5]{2}$. □

TODO: Some note about how DENTS can handle this case by using entropy and then ignoring it for the recommendations. Moreover, by allowing the entropy temperature (beta) to be decayed, allows the entropy trap estimates to correctly converge to the correct value of zero, and could handle a case where there is something more complex than a single state with a reward of one when not taking the entropy trap.

TODO: Additionally, note that because the proofs for convergence for DENTS have constraints on parameters (or no constraints on params) that even on this case it will converge to the correct result.

4.5.8 DENTS results

TODO: This is mostly the exponential bound stuff, so can also probably be appendix stuff

This subsection provides theoretical results that give exponential convergence of DENTS value estimates and simple regret. TODO: Some note on no constraints necessary on the parameters of DENTS?

Lemma 4.5.17. *TODO: maybe should do directly from the Q values? Also same for the ments one? Consider a DENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$. TODO: make H consistent with thtspp If for all $s_{t+1} \in \bigcup_{a \in \mathcal{A}} \mathrm{Succ}(s_t, a)$ there is some $C_{s_{t+1}}, k_{s_{t+1}} > 0$ such that for any $\varepsilon_{s_{t+1}} > 0$:*

$$\Pr\left(\left|\hat{V}_{\mathrm{D}}^{(N(s_{t+1},n))}(s_{t+1}) - V^*(s_{t+1})\right| > \varepsilon_{s_{t+1}}\right) \leq C_{s_{t+1}} \exp\left(-k_{s_{t+1}} \varepsilon_{s_{t+1}}^2 N(s_{t+1}, n)\right), \tag{4.212}$$

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

$$\Pr\left(\left|\hat{V}_{\mathrm{D}}^{(N(s_t,n))}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 N(s_t, n)\right). \tag{4.213}$$

Proof. Given the assumptions and by Lemmas 4.5.11, 4.5.1 and 4.5.5, for some $C, k > 0$ and for any $\varepsilon_1 > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}_{\mathrm{D}}^{(N(s_t,a_t,n))}(s_t, a_t) - Q^*(s_t, a_t)\right| \leq \varepsilon_1\right) > 1 - C\exp(-k\varepsilon_1^2 N(s_t)). \tag{4.214}$$

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

$$Q^*(s_t, a_t) - \varepsilon_1 \leq \hat{Q}_{\mathrm{D}}^{(N(s_t,a_t,n))}(s_t, a_t) \leq Q^*(s_t, a_t) + \varepsilon_1. \tag{4.215}$$

Let $a^* = \max_{a \in \mathcal{A}} Q^*(s_t, a)$. Using $\varepsilon_1 \leq \Delta_{\mathcal{M}}/2$ in (4.211), then for any $a \in \mathcal{A} - \{a^*\}$:

$$\begin{aligned}
\hat{Q}_{\mathrm{D}}^{(N(s_t,a,n))}(s_t, a) \leq & Q^*(s_t, a) + \Delta_{\mathcal{M}}/2 && (4.216)\\
\leq & Q^*(s_t, a^*) - \Delta_{\mathcal{M}}/2 && (4.217)\\
\leq & \hat{Q}_{\mathrm{D}}^{(N(s_t,a^*,n))}(s_t, a^*), && (4.218)
\end{aligned}$$

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

$$\hat{V}_{\mathrm{D}}^{(N(s_t,n))}(s_t) = \max_a \hat{Q}_{\mathrm{D}}^{(N(s_t,a,n))}(s_t, a) = \hat{Q}_{\mathrm{D}}^{(N(s_t,a^*,n))}(s_t, a^*). \tag{4.219}$$

Then using (4.211) with $a_t = a^*$ (noting $V^*(s_t) = Q^*(s_t, a^*)$), using (4.215) and using $\varepsilon_1 \leq \varepsilon$ gives:

$$\begin{aligned}
V^*(s_t) - \varepsilon \leq & V^*(s_t) - \varepsilon_1 && (4.220)\\
\leq & \hat{V}_{\mathrm{D}}^{(N(s_t,n))}(s_t) && (4.221)\\
\leq & V^*(s_t) + \varepsilon_1 && (4.222)\\
\leq & V^*(s_t) + \varepsilon. && (4.223)
\end{aligned}$$

Hence:

$$\begin{aligned}
\Pr\left(\left|\hat{V}_{\mathrm{D}}^{(N(s_t,n))}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq & C \exp(-k\varepsilon_1^2 N(s_t, n)) && (4.224)\\
\leq & C \exp(-k\varepsilon^2 N(s_t, n)), && (4.225)
\end{aligned}$$

which is the result. □

Similarly to the MENTS section, Lemma 4.5.17 provides an inductive step, which is used in Theorem 4.5.18 to show concentration inequalities at all states that DENTS visits.

Theorem 4.5.18. *Consider a DENTS process, let $s_t \in \mathcal{S}$ then there is some $C, k > 0$ for any $\varepsilon > 0$:*

$$\Pr\left(\left|\hat{V}_{\mathrm{D}}^{(N(s_t,n))}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 N(s_t, n)\right). \tag{4.226}$$

Moreover, at the root node s_0:

$$\Pr\left(\left|\hat{V}_{\mathrm{D}}^{(N(s_0,n))}(s_0) - V^*(s_0)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 n\right). \tag{4.227}$$

Proof. The result holds for $t = H + 1$ TODO: consistent with thtspp. SHOULD JUST CTRL-F THIS and sort all of these out at once. because $\hat{V}_{\mathrm{D}}^{(N(s_{H+1}))}(s_{H+1}) = V^*(s_{H+1}) = 0$. TODO: might be an off by one error here dep on how defined H, check over this whole paragraph. Hence the result holds for all $t = 1, ..., H + 1$ by induction using Lemma 4.5.17. Noting that $N(s_0) = n$ gives (4.223). □

TODO: is this just repeated? Can we merge two of the proofs?

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 4.5.19. *Consider a DENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ TODO: defn with thtspp for H agen then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\texttt{inst\_regr}(s_t, \psi_{\mathrm{D}}^n) \leq C' \exp\left(-k' N(s_t, n)\right). \tag{4.228}$$

Proof. Let a^* be the locally optimal standard action, so $a^* = \arg\max_{a \in \mathcal{A}} Q^*(s_t, a)$. By Theorem 4.5.18 and Lemmas 4.5.11 and 4.5.5 there exists $C_1, k_1 > 0$ such that for all $\varepsilon_1 > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}_{\mathrm{D}}^{(N(s_t,a_t,n))}(s_t, a_t) - Q^*(s_t, a_t)\right| \leq \varepsilon_1\right) > 1 - C_1 \exp(-k_1 \varepsilon_1^2 N(s_t, n)). \tag{4.229}$$

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

$$Q^*(s_t, a) - \Delta_{\mathcal{M}}/2 \leq \hat{Q}_{\mathrm{D}}^{(N(s_t,a,n))}(s_t, a) \leq Q^*(s_t, a) + \Delta_{\mathcal{M}}/2. \tag{4.230}$$

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

$$\begin{aligned}
\hat{Q}_{\mathrm{D}}^{(N(s_t,a,n))}(s_t, a_t) \leq & Q^*(s_t, a_t) + \Delta_{\mathcal{M}}/2 && (4.231)\\
\leq & Q^*(s_t, a^*) - \Delta_{\mathcal{M}}/2 && (4.232)\\
\leq & \hat{Q}_{\mathrm{D}}^{(N(s_t,a^*,n))}(s_t, a^*). && (4.233)
\end{aligned}$$

Negating the probability of (4.229) then gives:

$$\Pr\left(\hat{Q}_{\mathrm{D}}^{(N(s_t,a,n))}(s_t,a) > \hat{Q}_{\mathrm{D}}^{(N(s_t,a^*,n))}(s_t,a^*)\right) \leq C_1 \exp(-k_2 N(s_t,n)) \tag{4.234}$$

Finally, the bound on the expected immediate regret follows:

$$\begin{aligned}
&\mathbb{E}\texttt{inst_regr}(s_t,\psi_{\mathrm{D}}^n) && (4.235)\\
&= \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\psi_{\mathrm{D}}^n(s_t) = a\right) && (4.236)\\
&= \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(a = \arg\max_{a'} \hat{Q}_{\mathrm{D}}^{(N(s_t,a,n))}(s_t,a)\right) && (4.237)\\
&\leq \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\hat{Q}_{\mathrm{D}}^{(N(s_t,a,n))}(s_t,a) > \hat{Q}_{\mathrm{D}}^{(N(s_t,a^*,n))}(s_t,a^*)\right) && (4.238)\\
&\leq \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \, C_1 \exp(-k_2 N(s_t,n)), && (4.239)
\end{aligned}$$

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 4.5.19, the bound on the immediate simple regret can be converted to a bound on the simple regret.

Theorem 4.5.20. *Consider a DENTS process. Let $s_t \in \mathcal{S}$, with $1 \leq t \leq H$ TODO: thtspp consistent agen then there is some $C', k' > 0$ such that:*

$$\mathbb{E}\texttt{sim\_regr}(s_t,\psi_{\mathrm{D}}^n) \leq C' \exp\left(-k' N(s_t,n)\right). \tag{4.240}$$

Moreover, at the root node s_0:

$$\mathbb{E}\texttt{sim\_regr}(s_0,\psi_{\mathrm{D}}^n) \leq C' \exp\left(-k' n\right). \tag{4.241}$$

Proof. Theorem holds as a consequence of Corollary **??** and Lemma 4.5.19. To arrive at (4.237) note that $N(s_0,n) = n$. □

Finally, Theorem **??** follows TODO: this was the neurips main paper theorem, changing how doing the main results bit from the results that we have just shown.

Finally, Theorem **??** hows as it is a subset of what has already been shown.

Theorem 4.2. *TODO: this was the neurips main paper theorem, only changed enough to get this compiling. Also need to change the results for correct conditions, dont think beta needs to be bounded actually For any MDP $\mathcal{M}$, after running n trials of the DENTS algorithm with a root node of s_0, if β_D is bounded above and $\beta_D(m) \geq 0$ for all $m \in \mathbb{N}$, then there exists constants $C, k > 0$ such that for all $\varepsilon > 0$ it holds that $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_D^n)] \leq C \exp(-kn)$, and also $\hat{V}_D^{(N(s_0,n))}(s_0) \xrightarrow{p} V^*(s_0)$ as $n \to \infty$.*

Proof. The simple regret bound follows immediately from Theorem 4.5.20. Using Theorem 4.5.18 it holds that $\Pr\left(\left|\hat{V}_D^{(N(s_0,n))}(s_0) - V^*(s_0)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 n\right) \to 0$ as $n \to \infty$, and hence $\hat{V}_D^{(N(s_0,n))}(s_0)$ converges in probability to $V^*(s_0)$. □

4.5.9 BTS results

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

TODO: probably want to delete this and be more clear in the intro to the theory section that BTS is just a special case of DENTS mathematically. Also want to say that BTS does have a purpose in being a standalone algorithm when entropy isn't useful and so doing the second backup is useless. Commented out below is the main BTS theorem from the neurips paper

4.5.10 Results for using average returns in Boltzmann MCTS processes

TODO: SOME of this should be integrated into the main arguments I think

TODO: N(s) to N(s,n) stuff

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

Figure 4.17: TODO: Haven't touched this, need to reproduce from neurips paperAn MDP that AR-BTS will not converge to recommending the optimal policy on, for a large enough value of D.

$$\bar{V}^{N(s_t)+1}(s_t) = \bar{V}^{N(s_t)}(s_t) + \frac{\bar{R}(s_t) - \bar{V}^{N(s_t)}(s_t)}{N(s_t)+1}, \tag{4.242}$$

$$\bar{Q}^{N(s_t,a_t)+1}(s_t,a_t) = \bar{Q}^{N(s_t,a_t)}(s_t,a_t) + \frac{\bar{R}(s_t,a_t) - \bar{Q}^{N(s_t,a_t)}(s_t,a_t)}{N(s_t,a_t)+1}, \tag{4.243}$$

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:

$$\bar{V}^{N(s_t)}(s_t) = \sum_{a\in\mathcal{A}} \frac{N(s_t,a)}{N(s_t)} \bar{Q}^{N(s_t,a_t)}(s_t,a_t), \tag{4.244}$$

$$\bar{Q}^{N(s_t,a_t)}(s_t,a_t) = R(s_t,a_t) + \sum_{a\in\mathcal{A}} \frac{N(s')}{N(s_t,a_t)} \bar{V}^{N(s')}(s'). \tag{4.245}$$

TODO: should we do the rearrangement/working out for this?

TODO: Sufficiency for ar-bts to have temp tend to zero, but would need to change the theorem to say lower bound on temp rather than just fixed temp. Firstly, it can be shown that using a non-decaying search temperature with AR-BTS is not guaranteed to recommend the optimal policy.

Proposition B.1. *TODO: Haven't touched this, just c and p from neurips paper*

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: $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi^n_{\text{AR-BTS}})] \not\to 0$ *as* $n \to \infty$.

Proof outline. TODO: Haven't touched this, just c and p from neurips paper

Consider the MDP given in Figure 4.17. 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:

$$\mathbb{E}\bar{V}^{N(k)}(k) = \frac{\exp\left(\bar{V}^{N(k+1)}(k+1)\right)}{1+\exp\left(\bar{V}^{N(k+1)}(k+1)\right)}\mathbb{E}\bar{V}^{N(k+1)}(k+1) + \frac{1}{1+\exp\left(\bar{V}^{N(k+1)}(k+1)\right)} \cdot 0 \tag{4.246}$$

$$\leq E \cdot \mathbb{E}\bar{V}^{N(k+1)}(k+1) \tag{4.247}$$

$$\leq E \cdot 2E^{D-k} \tag{4.248}$$

$$= 2E^{D-k+1}. \tag{4.249}$$

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. TODO: (was camera ready todo) Do the full proof, and show that AR-BTS converges to these value with non zero prob, hence the non zero simple regret.

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. TODO: (Was camera ready todo) Formally give the simple regret converges to something strictly greater than zero. Also can I even do that intersection to product equality? Doesn't that mean that they are independent. Are they independent? □

TODO: Say that AR-DENTS with beta not o(alpha) isn't consistent. Show that it essentially leads to a term of one times the entropy value, and so repeating the arguments that showed MENTS is inconsistent would show that AR-DENTS

is inconsistent in this case. So this and the above AR-BTS results show that the conditions are necessary too.

TODO: Sufficiency for most visited stuff could also be done? Probably extending from these arguments, or saying that it would contradict these results.

TODO: This proof is re-written and improved on below, so this should be deleted.

Lemma 4.5.21. *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:

$$\Pr\left(\bigcap_{i=\ell}^{m} a^i \neq b\right) = \prod_{i=\ell}^{m} \Pr(a^i \neq b) \tag{4.250}$$

$$\leq \prod_{i=\ell}^{m} \left(1 - \frac{\epsilon}{|\mathcal{A}| \log(e + i)}\right) \tag{4.251}$$

$$\leq \left(1 - \frac{\epsilon}{|\mathcal{A}| \log(e + m)}\right)^{m-\ell} \tag{4.252}$$

$$\to 0, \tag{4.253}$$

as $m \to \infty$. Which is a contradiction. □

TODO: This proof is re-written and improved on below, so this should be deleted.

Theorem B.2. *For any MDP $\mathcal{M}$, if $\alpha(m) \to 0$ as $m \to \infty$ then $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi^n_{\text{AR-BTS}})] \to 0$ 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 4.5.22, 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 (4.240), we can see that in the limit $\bar{V}^{N(s_t)}(s_t)$ tends to $\bar{Q}^{N(s_t,a*)}(s_t, a*)$. TODO: (old cr todo)formally prove that the distribution and value converges. Will need some epsilons and use the equation referenced for the value. the distribution bit needs to show that eventually the samples are dominated by the greedy boltzmann policy, and not the exploration part

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 $\alpha(m) \to 0$. TODO: (old cr todo)technically only the rho bit does, shrugs □

TODO: Cut this one, also assumption is a bit off, and I think we're proving stuff correctly below in the additional section later

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

Proof outline. Proof is similar to the proof for Theorem **??**. □

TODO: Probably delete this too 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 **??** and **??**.

4.5.11 Consistency Of Boltzmann MCTS Processes

TODO: Generally look into how to justify that we take limits in parts. Got the dominated convergence theorem from here: https://math.stackexchange.com/questions/15240/when-can-you-switch-the-order-of-limits/15296#15296. Probably just leave the places where we do it as handwavy proof outlines

TODO: Which on that note, make sure we're using proof outline in this section, not proof

TODO: Generally clean up the writing here.

TODO: Make this the main section showing consistency of Boltzmann MCTS processes and under what conditions. Have the missing proofs written up on ipad.

TODO: Also write up the general steps from the ipad notes (which I think need a bit of correction.)

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

TODO: This lemma restated means that Boltzmann search policies will select all actions infinitely often. TODO: Check the variables in this proof are consistent with the definitions of N(s,a,m) etc

Lemma 4.5.22. *Let ρ^m be an arbitrary policy, and the following policy: TODO: handle when epsilon is greater than one? I think its just let ell be large enough argument*

$$\pi^m(a) = (1 - \lambda(m))\rho^m(a) + \lambda(m)\frac{1}{|\mathcal{A}|}, \tag{4.254}$$

with $\lambda(m) = \min(1, \epsilon/\log(e+m))$, where $\epsilon \in (0, \infty)$. Let $a^m \sim \pi^m$ be the mth sampled action, and let $M(a,m)$ be the number of times action a was selected in the first m samples, i.e. $M(a,m) = \sum_{m=1}^{m} \mathbb{1}[a^m = a]$. Then for all $a \in \mathcal{A}$ we have $M(a,m) \overset{as}{\to} \infty$ as $m \to \infty$.

Proof. First consider that if $M(a,m) \not\to \infty$ then it is logically equivalent to say that there exists some ℓ such that from a^ℓ onwards that there is some action which is never sampled again. To prove the result, argue by contradiction, and suppose that there is some $\ell \in \mathbb{N}$ and $b \in \mathcal{A}$ such that:

$$\Pr\left(\bigcap_{m=\ell}^{\infty} a^m \neq b\right) > 0. \tag{4.255}$$

However, from the definition of equation (4.250) it must be that:

$$\Pr\left(a^m = b\right) \geq \frac{\lambda(m)}{|\mathcal{A}|} = \frac{\epsilon}{|\mathcal{A}|\log(e+m)}. \tag{4.256}$$

And so using this lower bound to work out the probability of never sampling action b again after the first $\ell - 1$ samples gives:

$$\Pr\left(\bigcap_{m=\ell}^{\infty} a^m \neq b\right) = \lim_{k\to\infty} \prod_{m=\ell}^{k} \Pr(a^m \neq b) \tag{4.257}$$

$$\leq \lim_{k\to\infty} \prod_{m=\ell}^{k} \left(1 - \frac{\epsilon}{|\mathcal{A}| \log(e+k)}\right) \tag{4.258}$$

$$\leq \lim_{k\to\infty} \left(1 - \frac{\epsilon}{|\mathcal{A}| \log(e+k)}\right)^{k-\ell} \tag{4.259}$$

$$= 0, \tag{4.260}$$

which is in contradiction to inequality (4.251) that was assumed. Inequality (4.255) holds because the factors in the product are increasing with respect to m. To see why the final limit equality (4.256) holds, consider the simpler function $f(x) = (1 - 1/\log(x))^x$. Taking logarithms and applying L'Hopital's rule TODO: cite? generously gives:

$$\lim_{x\to\infty} \log f(x) = \lim_{x\to\infty} x \log\left(1 - \frac{1}{\log(x)}\right) \tag{4.261}$$

$$= \lim_{x\to\infty} \frac{\log\left(1 - \frac{1}{\log(x)}\right)}{x^{-1}} \tag{4.262}$$

$$= \lim_{x\to\infty} \frac{\frac{1}{x\log(x)(\log(x)-1)}}{-x^{-2}} \tag{4.263}$$

$$= \lim_{x\to\infty} \frac{-x}{\log(x)(\log(x)-1)} \tag{4.264}$$

$$= \lim_{x\to\infty} \frac{-x}{2\log(x)-1} \tag{4.265}$$

$$= \lim_{x\to\infty} \frac{-x}{2} \tag{4.266}$$

$$= -\infty. \tag{4.267}$$

And thus $\lim_{x\to\infty} f(x) = \lim_{y\to-\infty} e^y = 0$.

TODO: cut this down maybe, its a little excessive showing all of the working out?

Hence, by contradiction, it must be the case that $\Pr(M(a,m) \not\to \infty) = 0$, or rather that $\Pr(M(a,m) \to \infty) = 1$ which is the desired result. □

TODO: Can we also say that if its (1 - 1/o(x)) to the x, then it will generally hold, so x to the power of 0.99 could equally be used. And also any polynomial of log(x) could be used

TODO: Words about the consequences of this lemma?

In direct consequence of Lemma 4.5.22, every reachable state (and state-action pair) will be visited infinitely often in a Boltzmann MCTS Process.

Corollary 4.5.22.1. *For any Boltzmann MCTS Process (i.e. a MCTS algorithm using a search policy of the form* $\pi^m(a) = (1-\lambda(m))\rho^m(a) + \lambda(m)\frac{1}{|\mathcal{A}|}$*), all reachable states from the root node are visited infinitely often. Specifically, for any reachable* $s \in \mathcal{S}$ *is must be that* $N(s,m) \xrightarrow{as} \infty$ *and* $N(s,a,m) \xrightarrow{as} \infty$ *as* $m \to \infty$*. TODO: handle reachability somewhere better? just make some assumption at the start of the theory section saying we assume all s are reacahable?*

Proof outline. This is a direct consequence of Lemma 4.5.22 when applied inductively at each node. □

TODO: DP backups converge in limit (Dynamic Programming cite)

TODO: And corollary is that DENTS with DP backups converges

TODO: Words about the below, which gives conditions for the search policy to tend to the optimal policy, and should add somewhere that can't have the search policy converge to the optimal policy and get exp simple regret convergence in theory

Lemma 4.5.23. *TODO: Write this up properly, and more generally, and use the indexed on num trials notation. Say that as long as the Q function converges to the optimal, beta is o(alpha) and alpha tends to zero, then search policy tends to optimal policy. If* $\bar{Q}_{\text{AR-D}}(s,a) \xrightarrow{p} Q^*(s,a)$ *then* $\pi_{\text{AR-D}} \xrightarrow{p} \pi^*$

Proof outline. A full proof to show that these limits converge correctly needs to make use of the dominated convergence theorem TODO: cite.

The outline argument is that in $\pi_{\text{AR-D}}$ that firstly $\lambda \to 0$ and so the limit of $\pi_{\text{AR-D}}$ will be the same as the limit of $\rho_{\text{AR-D}}$.

TODO: Words about the below maths, and below is very informal writing up

$$\frac{1}{\alpha(n)}\dot{Q}(s,a) + \frac{\beta(n)}{\alpha(n)}\bar{\mathcal{H}}_Q(s,a) \to \frac{1}{\alpha(n)}\dot{Q}(s,a) \tag{4.268}$$

$$\rho(a|s) \to \frac{1}{Z}\exp\left(\frac{\dot{Q}(s,a)}{\alpha(n)}\right) \tag{4.269}$$

$$\to \mathbb{1}[a = a^*] \tag{4.270}$$

TODO: Should probably show that softmax converges to max in limit □

TODO: Another proof about visits. For most visit recommendations, use this to justify that alpha can be arbitrary, but need beta equal to o(alpha), (which is alpha doesnt go to zero just means that beta goes to zero) in the DP case, and for the AR case need the same conditions with alpha to zero

Lemma 4.5.24. *TODO: write this up properly If* $\pi \xrightarrow{p} \pi_{\lim}$ *such that* $\pi_{\lim}(a* |s) > \pi(a|s)$ *for all* a, s*, then the most visited recommendation policy will converge. Specifically,* $\frac{N(s,a,n)}{N(s,n)} \xrightarrow{p} \pi_{\lim}(a|s)$.

Proof. TODO: write this up properly First use Kolmogorovs strong law TODO: cite the source downloaded, which valid because $\sum \frac{\mathbb{E}\mathbb{1}[a^i=a]^2}{k^2} \leq \sum \frac{1}{k^2} < \infty$. Using this gives

$$\frac{N(s,a,n)}{N(s,n)} = \frac{1}{N(s,n)}\sum_{i=1}^{N}(s,n)\mathbb{1}[a^i = a] \tag{4.271}$$

$$\to \mathbb{E}\lim_{n\to\infty}\sum_{i=1}^{n}\frac{\mathbb{1}[a^i = a]}{n} \tag{4.272}$$

$$= \lim_{n\to\infty}\mathbb{E}\sum_{i=1}^{n}\frac{\mathbb{1}[a^i = a]}{n} \tag{4.273}$$

$$= \lim_{n\to\infty}\sum_{i=1}^{n}\frac{\pi^i(a|s)}{n} \tag{4.274}$$

$$= \pi^i(a|s) \tag{4.275}$$

Where the last line uses if $x_n \to x$ then $\sum_{i=1}^{n}\frac{x_i}{n} \to x$. TODO: cite this somehow? Cesaro summation? □

TODO: Write up results that use the above lemma to have consequence that most visited recommendation policy is consistent.

TODO: AR-DENTS can use the generalised Q convergence result, via the equations above still.

TODO: AR-DENTS needs three steps for the induction, with the extra step of if Q values converge then policy converges as given above

Lemma 4.5.25. *TODO: clean up this writing If* $\pi_{\text{AR-D}} \xrightarrow{p} \pi^*$ *(i.e. TODO: conditions from above result), and* $\bar{Q}_{\text{AR-D}} \xrightarrow{p} Q^*$*, then* $\bar{V}_{\text{AR-D}} \xrightarrow{p} V^*$*.*

Proof outline. TODO: clean Using the result above TODO: ref, it must be that $\pi_{\text{AR-D}}(a|s) \to \pi^*(a|s) = \mathbb{1}[a^*_{|s} = a]$. Then recalling equation TODO: ref that related the average returns values:

$$\bar{V}(s) = \sum_a \frac{N(s,a,n)}{N(s,n)} \bar{Q}(s,a) \tag{4.276}$$

$$\xrightarrow{p} \sum_a \mathbb{1}[a^*_{|s} = a] \bar{Q}(s,a) \tag{4.277}$$

$$= \bar{Q}(s, a^*_{|s}) \tag{4.278}$$

$$\xrightarrow{p} Q^*(s, a^*_{|s}) \tag{4.279}$$

$$= V^*(s). \tag{4.280}$$

□

4.6 Bookend

TODO: Something to bookend the chapter. Conclusion?

TODO: At end of cleaning all this up, do a proof read of this chapter (and any appendices), and then go through Neurips paper one more time to see if any info in paper that want to cover isn't in there. (Can probably skip over the theory part of appendix.)

4.7 To Move To Appendix

TODO: A space to move things temorarily before moving it all to an appendix.

5 Convex Hull Monte Carlo Tree Search

Contents

5.1 Introduction

TODO: list

- high level overview of CHMCTS work
- discuss how CHMCTS answers the research questions from introduction chapter
- moving into multi-objective land now
- Comment about CHVI and prior MOMCTS work motivating this

5.2 Contextual Tree Search

TODO: list

- Discuss need for context when doing multi-objective tree Search
  - Use an example env where left gives (1,0) and right gives (0,1), optimal policy picks just left or just right, but hypervolume based methods wont
  - Use previous work on these examples and show they dont do well bad
- Discuss how UCT = running a non-stationary UCB at each node, so given above discussion, there is work in contextual MAB
- Introduce contextual regret here

5.3 Contextual Zooming for Trees

TODO: list

- Give contextual zooming for trees algorithm
- Discussion on the contextual MAB to non-stationary contextual MAB stuff (CZT is to CZ what UCT is to UCB) (and what theory carry over)

5.4 Convex Hull Monte Carlo Tree Search

TODO: list

- Give convex hull monte carlo tree search
- Contextual zooming with the convex hull backups

TODO: Repeat definition of cprune, and add to abbreviations there

5.5 Results

TODO: list

- Results from CHMCTS paper
- Get same plots from C++ code, but compare expected utility, rather than the confusing hypervolume ratio stuff

6

Simplex Maps for Multi-Objective Monte Carlo Tree Search

Contents

6.1 Introduction

TODO: list

- high level overview of simplex maps work
- discuss how simplex maps answer the research questions from introduction chapter
- staying in multi-objective land now
- Motivated by CHMCTS being slow

6.2 Simplex Maps

TODO: list

- Define simplex map interface
- Give details on how to efficiently implement the interface with tree structures
- (Good diagram is everything here I think)

6.3 Simplex Maps in Tree Search

TODO: list

- Come up with better title for section
- Use simplex maps interface to create algorithms from the dents work
- Give a high level idea of what δ parameter is (used in theory section)

6.4 Theoretical Results

TODO: list

- Convergence can build ontop of DENTS results
- Runtime bounds (better than $O(2^D)$ which is what using convex hulls has)
- Simplex map has a diameter δ (i.e. the furthest away a new context could be from a point in the map)
- Bounds can then come from that diameter (which is a parameter of the simplex map/algorithm) and DENTS results

6.5 Empirical Results

TODO: list

- Results from MO-Gymnasium
- Compare algorithms using expected utility

7 Conclusion

Contents

TODO: Something about we'll conclude by looking back at contributions and possible future work.

7.1 Summary of Contributions

TODO: go through each of the research questions and contributions, and write about how the work answers the research questions

7.2 Future Work

TODO: outline some avenues of potential future work

Appendices

A

List Of Appendices To Consider

- Multi Armed Bandits, maybe
- MAybe from of the things in background are more appropriate as appendices?

B

Boltzmann Search With Average Returns

TODO: Currently a pasting ground for currting average returns

B.0.1 AR-BTS

TODO: This is original writing for AR-BTS

This search policy can still be used with average returns. The following is a summary of the definitions for *Boltzmann Tree Search with Average Returns* (AR-BTS), which uses the value estimates of $\hat{V}_{\text{AR-BTS}}$ and $\hat{Q}_{\text{AR-BTS}}$ at each node, and temperature schedule $\alpha_{\text{AR-BTS}}(x) > 0$:

$$\pi_{\text{AR-BTS}}(a|s) = (1 - \lambda(s, \epsilon_{\text{AR-BTS}}))\rho_{\text{AR-BTS}}(a|s) + \frac{\lambda(s, \epsilon_{\text{AR-BTS}})}{|\mathcal{A}|}, \tag{B.1}$$

$$\rho_{\text{BTS}}(a|s) \propto \exp\left(\frac{1}{\alpha_{\text{AR-BTS}}(N(s))}\left(\hat{Q}_{\text{AR-BTS}}(s,a)\right)\right). \tag{B.2}$$

where $\epsilon_{\text{AR-BTS}} \in (0, \infty)$ is an exploration parameter and $\alpha_{\text{AR-BTS}}(x)$ is the search temperature schedule. Given a trajectory $\tau = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h)$ and the leaf node value estimate $\tilde{r} = \hat{V}_{\text{init}}(s_h)$, the value estimates are updated for $t = h - 1, ..., 0$:

$$\hat{Q}_{\text{AR-BTS}}(s_t, a_t) \leftarrow \frac{1}{N(s_t, a_t)}\left((N(s_t, a_t) - 1)\hat{Q}_{\text{AR-BTS}}(s_t, a_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i\right) \tag{B.3}$$

$$\hat{V}_{\text{AR-BTS}}(s_t) \leftarrow \frac{1}{N(s_t)}\left((N(s_t) - 1)\hat{V}_{\text{AR-BTS}}(s_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i\right). \tag{B.4}$$

Similarly to BTS, either the Q-value estimates can be used for a recommendation policy or the most visited child node can be used:

$$\psi_{\text{AR-BTS}}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_{\text{BTS}}(s, a), \tag{B.5}$$

$$\texttt{mv}_{\text{AR-BTS}}(s) = \arg\max_{a \in \mathcal{A}} N(s, a). \tag{B.6}$$

B.0.2 AR-DENTS

B.0.3 MENTS, RENTS and TENTS with Average Returns

TODO: Its DENTS, with alpha $\bar{\text{b}}$eta $\bar{\text{c}}$onst, and different entropy functions.

C

Go Parameter Tuning

TODO: Edit this down correctly for thesis and change ch title?

C.0.1 Additional Go details, results and discussion

Recall from Appendix **??** 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 TODO: fixcites For these 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.

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 C.1: Results for round robin to select the temperature parameter α for BTS. The value of 1.0 won all four of its matches so was selected.

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 C.2: Results for round robin to select the temperature parameter α_{init} for AR-BTS. The value of 0.1 won all four of its matches so was selected.

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.

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 **??**. 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 TODO: fix cites

Initially we set the values of $\epsilon, \epsilon_{\tilde{\lambda}}$ to $0.03, 1.0$. In Tables C.1 and C.2 we give results of tuning the search temperature for BTS and AR-BTS. For AR-BTS we tried different values of α_{init} in $\alpha(m) = \alpha_{\text{init}}/\sqrt{m}$.

We then tuned the weighting of the prior policy $\epsilon_{\tilde{\lambda}}$. In Tables C.3 and C.4 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 C.5 and C.6 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

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 C.3: 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 C.4: 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.

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 C.7, C.8,C.9, C.10,C.11 and C.12.

Finally, we considered entropy temperatures of the form $\beta(m) = \beta_{\text{init}} \frac{1+\exp(-5)}{1+\exp(m-5)}$ for DENTS and AR-DENTS, and tuned the value of β_{init}, in Tables C.13 and C.14 for DENTS and AR-DENTS respectively.

After tuning all of the algorithms, we compared each algorithm to their AR version in table C.15, and the AR versions universally outperformed their counterparts. We used all of the selected parameters in 19x19 Go to run our final

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 C.5: 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 C.6: Results for round robin to select the exploration parameter ϵ for AR-BTS. The value of 0.001 won the most matches so was selected.

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 C.7: 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 C.8: 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 C.9: 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 C.10: Results for round robin to select the temperature parameter α for ARRENTS. 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 C.11: Results for round robin to select the temperature parameter α for TENTS. The value of 30.0 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 C.12: Results for round robin to select the temperature parameter α for ARTENTS. 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
0.03	13-2	-	11-4	13-2	10-5
0.01	12-3	11-4	-	10-5	11-4
0.003	7-8	13-2	13-2	-	10-5
0.001	13-2	13-2	11-4	11-4	-

Table C.13: Results for round robin to select the initial entropy temperature β_{init} for DENTS. 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 C.14: Results for round robin to select the initial entropy temperature β_{init} for AR-DENTS. 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 C.15: Results for the matches of each algorithm against its AR version.

experiments given in Table 4.1.

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.

Bibliography

[1] Bruce Abramson. Expected-outcome: A general model of static evaluation. *IEEE transactions on pattern analysis and machine intelligence*, 12(2):182–193, 1990.

[2] P Auer. Finite-time analysis of the multiarmed bandit problem, 2002.

[3] Leon Barrett and Srini Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, pages 41–47, 2008.

[4] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138, 1995.

[5] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN 9780486428093.

[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 *Algorithmic Learning Theory: 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings 20*, pages 23–37. Springer, 2009.

[8] Tristan Cazenave, Flavien Balbo, Suzanne Pinson, et al. Monte-carlo bus regulation. In *12th international IEEE conference on intelligent transportation systems*, volume 340345, 2009.

[9] 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.

[10] 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.

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

[12] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

[13] Thomas Keller and Patrick Eyerich. Prost: Probabilistic planning based on uct. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 22, pages 119–127, 2012.

[14] Thomas Keller and Malte Helmert. Trial-based heuristic tree search for finite horizon mdps. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 23, pages 135–143, 2013.

[15] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690, 2008.

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

[17] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1:1–22, 2006.

[18] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[19] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

[20] Thomas M Moerland, Joost Broekens, Aske Plaat, and Catholijn M Jonker. A0c: Alpha zero in continuous action space. *arXiv preprint arXiv:1805.09613*, 2018.

[21] Mausam Natarajan and Andrey Kolobov. *Planning with Markov decision processes: An AI perspective*. Springer Nature, 2022.

[22] Michael Painter. THTS++. URL `https://github.com/MWPainter/thts-plus-plus`. Available at `https://github.com/MWPainter/thts-plus-plus`.

[23] Herbert Robbins. Some aspects of the sequential design of experiments. 1952.

[24] Diederik Marijn Roijers, Shimon Whiteson, and Frans A Oliehoek. Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, 52:399–443, 2015.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[26] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[27] David Silver and Joel Veness. Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010.

[28] 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.

[29] 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.

[30] Aleksandrs Slivkins. Contextual bandits with similarity information. In *Proceedings of the 24th annual Conference On Learning Theory*, pages 679–702. JMLR Workshop and Conference Proceedings, 2011.

[31] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

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

[33] Alastair J Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):253–256, 1977.

[34] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

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