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

Appendices

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.

$O(f(x))$ . . . . TODO: Big-O and Big-Omega notation?

(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.

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

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 $\mathbf{w}$.

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

$\texttt{node}(s)$ . . . . . The decision node corresponding to the state s.

$\texttt{node}(s)\texttt{.chldrn}$ The set of chance nodes that are children of $\texttt{node}(s)$.

$\texttt{node}(s)\texttt{.V}$ . . . The set of variables stored at decision node $\texttt{node}(s)$, typically used for estimating values.

$\texttt{node}(s,a)$ . . . The chance node corresponding to the state-action pair (s,a).

$\texttt{node}(s,a)\texttt{.chldrn}$ The set of decision nodes that are children of $\texttt{node}(s,a)$.

$\texttt{node}(s,a)\texttt{.Q}$ . . The set of variables stored at decision node $\texttt{node}(s,a)$, 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**)

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 [30], 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 [23] 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:

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

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

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 [8], 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. [8] 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):

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

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

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

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

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 [20, 39], 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 [39] 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 [20, 39], 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 [39] 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 $A^{m+1} = A^m$.

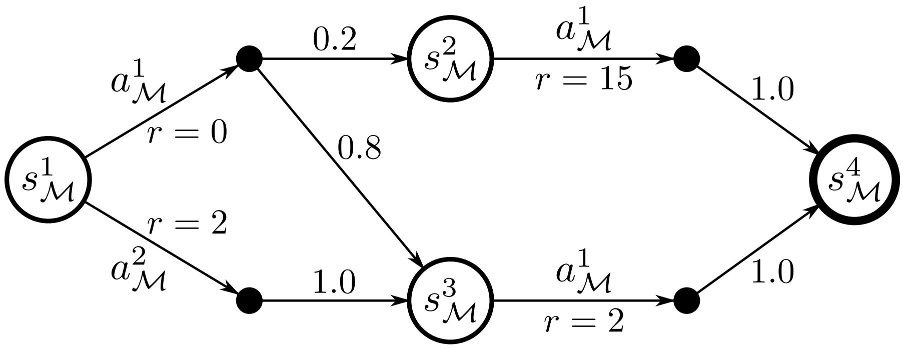


Figure 2.4: An example MDP $\mathcal{M}$, where $s_0 = s^1_{\mathcal{M}}$, the finite horizon is $H = 2$, and terminal states are marked with a thicker border. Edges are appropriately marked with actions and rewards, or marked with transition probabilities.

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).

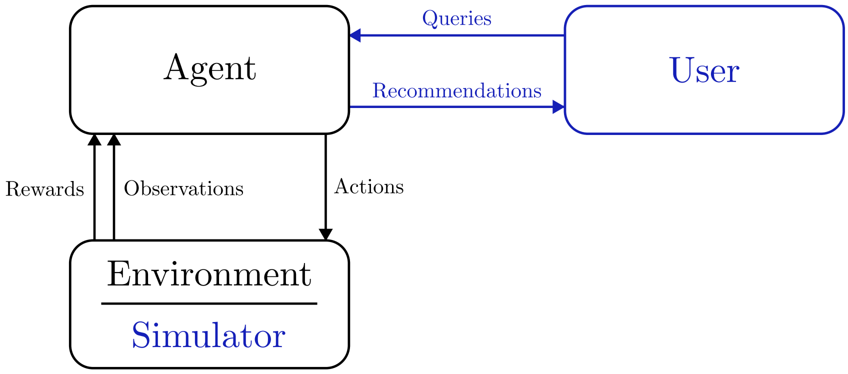


Figure 2.5: An overview of reinforcement learning. In black depicts the standard agent-environment interface for reinforcement learning [40], where an agent can perform actions in an environment and is given feedback in the form of observations and rewards. In blue the environment is replaced by a simulated 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 [40], 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:*

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

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 [27]:

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* [6, 27]:

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

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

The Bellman equations admit a *dynamic programming* approach which can be used to computer the optimal value functions, known as *Value Iteration* [6, 27]. 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 [15].

Let $\mathcal{H}$ denote the (Shannon) entropy function [34]:

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

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:*

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

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 [15]:*

$$\pi_{\text{sft}}^*(a|s;t) = \exp\left(\left(Q_{\text{sft}}^*(s,a;t) - V_{\text{sft}}^*(s;t)\right)/\alpha\right). \tag{2.46}$$

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 [15]:

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

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

Analogous to standard reinforcement learning, the soft Bellman equations can be used in *soft Bellman backups* for a *Soft Value Iteration* algorithm [15]:

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

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

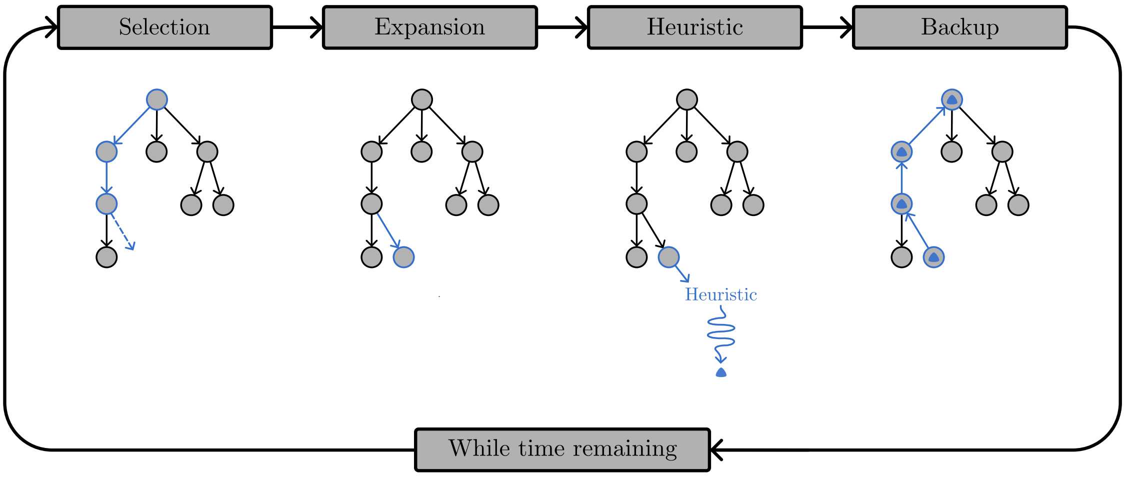


Figure 2.6: Overview of one trial of MCTS-1, depicting the four commonly presented stages [7]. First, in the selection phase a search policy is used to sample a path down the tree. Next a new node is added to the search tree. Then the new node is initialised using a heuristic, often using a *simulation* 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.

MCTS trials are commonly presented using four stages, selection, expansion, simulation/heuristic and backup phases [7], 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) [19] generalises heuristic tree search methods used for planning, such as Trial-Based Real Time Dynamic Programming [5] and LAO* [16]. 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++` [28] 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 [21, 22], which is a commonly used MCTS algorithm, and Section 2.4.4 presents the Maximum ENtropy Tree Search (MENTS) algorithm [46], 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 [19], 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 [35]. 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:

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);

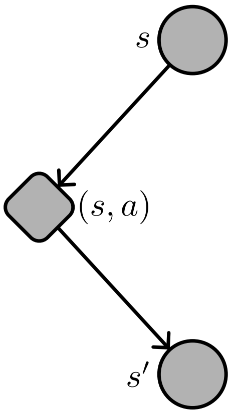


Figure 2.7: Tree diagrams notation, circles will be used to denote *decision nodes* that are associated with states, diamonds will be used to denote *chance nodes* that are associated with state-action pairs and arrows are used to denote parent/child relationships in the tree.

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 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.

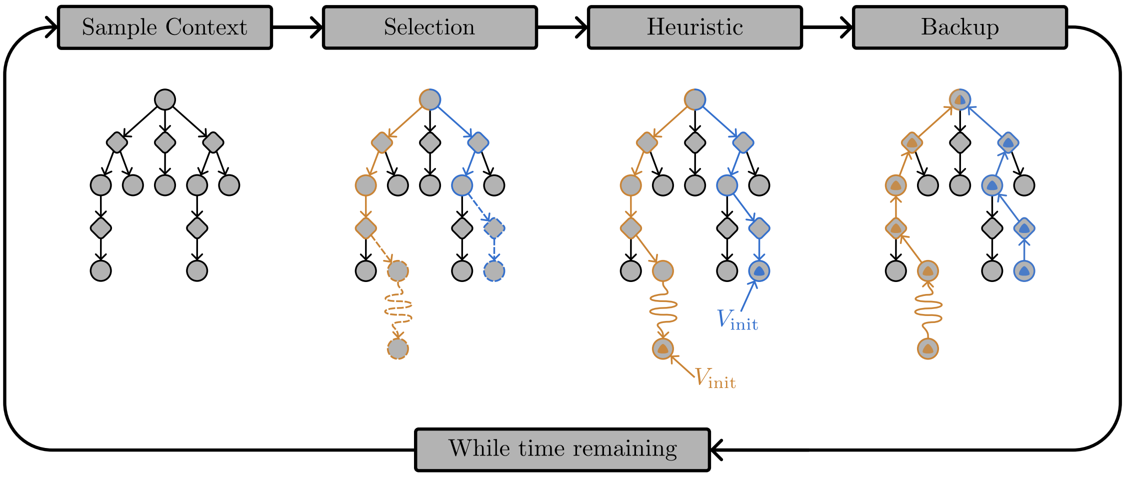


Figure 2.8: Overview of one trial of `THTS++`, where orange shows an example when `mcts_mode` is True, and blue shows an example when `mcts_mode` is False. From left to right: first a context is sampled, which stores any necessary per-trial state (not depicted) and the search tree at the beginning of 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).

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

Additionally, let `node`(s)`.chldrn` be a mapping from actions to the chance nodes that are children of `node`(s). Likewise, `node`(s,a)`.chldrn` is a mapping from outcomes (successor states) to the decision nodes that are children of `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 *[7]. 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 [19].*

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, `node(`s`).V` and `node(`s,a`).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) [21, 22] 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:

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

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

```
def run_trial(search_tree: $\mathcal{T}$,
              search_policy: $\pi_{\text{search}}$,
              heuristic_fn: $\hat{V}_{\text{init}}$,
              heuristic_action_fn: $\hat{Q}_{\text{init}}$,
              mcts_mode,
              planning_horizon: $H_{\texttt{THTS++}}$):
    # context sampling
    ctx = sample_context()
    # simulation phase
    $\tau_{:h}$ = sample_trajectory($\mathcal{T}$, $\pi_{\text{search}}$, $\hat{Q}_{\text{init}}$, mcts_mode, $H_{\texttt{THTS++}}$, ctx)
    $\mathcal{T} \leftarrow \mathcal{T} \cup \tau_{:h}$
    # heuristic phase
    node($s_h$).V $\leftarrow \hat{V}_{\text{init}}(s_h$, 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, $\tau_{:h}$, $\hat{V}_{\text{init}}(s_h)$, ctx)
        node($s_i$).backup_v(node($s_i$).chldrn, $\tau_{:h}$, $\hat{V}_{\text{init}}(s_h)$, ctx)

def sample_trajectory(search_tree: $\mathcal{T}$,
                      search_policy: $\pi_{\text{search}}$,
                      heuristic_action_fn: $\hat{Q}_{\text{init}}$,
                      mcts_mode,
                      planning_horizon: $H_{\texttt{THTS++}}$,
                      ctx):
    i = 0
    while ((not mcts_mode or $s_{\texttt{i}} \in \mathcal{T}$)
           and (i < $H_{\texttt{THTS++}}$)
           and (not terminal($s_{\texttt{i}}$)):
        $a_{\texttt{i}} \sim \pi_{\text{search}}(\cdot|s_{\texttt{i}}$, ctx)
        $r_{\texttt{i}} \leftarrow R(s_{\texttt{i}}, a_{\texttt{i}})$
        $s_{\texttt{i+1}} \leftarrow$ sample_outcome($s_{\texttt{i}}$, $a_{\texttt{i}}$, ctx)
        i += 1
    return $(s_0, a_0, r_0, s_1, ..., s_{\texttt{i-1}}, a_{\texttt{i-1}}, r_{\texttt{i-1}}, s_{\texttt{i}})$

def sample_outcome($s$, $a$, ctx):
    $s' \sim p(\cdot|s, a)$
    return $s'$
```

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

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 [19] 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 $\hat{V}_{\text{init}} = \tilde{V}$ [37, 38, 26] , where $\hat{V}$ aims to approximate the optimal value function V^*; the second approach consists of using a *rollout policy* [7, 14, 18, 9]. 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 [7, 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) [46], 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 [46], 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. [17]. 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 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

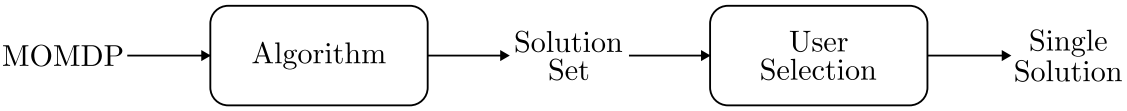


Figure 2.9: The decision-support scenario for multi-objective reinforcement learning [17].

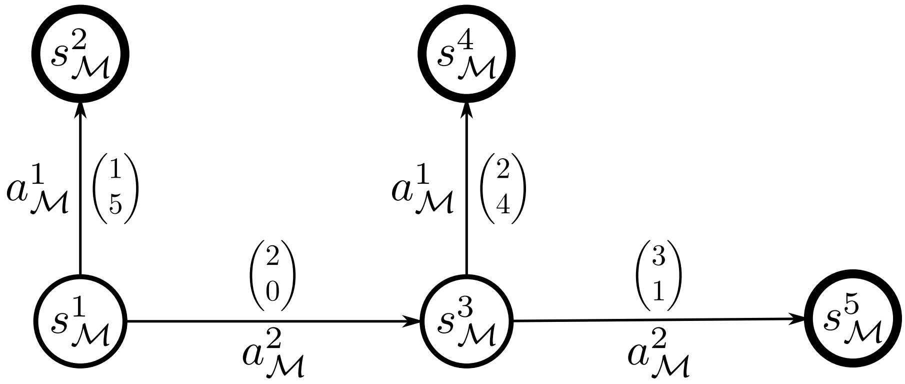


Figure 2.10: An example MOMDP $\boldsymbol{\mathcal{M}}$, where $s_0 = s^1_{\mathcal{M}}$, the finite horizon is $H = 2$ and terminal states are marked with thicker borders. This particular MOMDP is deterministic, so all transition probabilities are one.

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:*

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

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* $CCS_{\min}(\Pi) \subseteq CH(\Pi)$ *is minimal if the following holds*

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

Now the geometry of convex coverage sets is considered, to see how something like $\boldsymbol{\mathcal{V}}\mathbf{als}(CCS_{\min}(\Pi))$ 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, $\boldsymbol{\mathcal{V}}\mathbf{als}(CH(\Pi))$ lie on a *geometric (partial) convex hull* (see (b) in Figure 2.11). Moreover, any points in $\boldsymbol{\mathcal{V}}\mathbf{als}(CH(\Pi))$ that lie on the

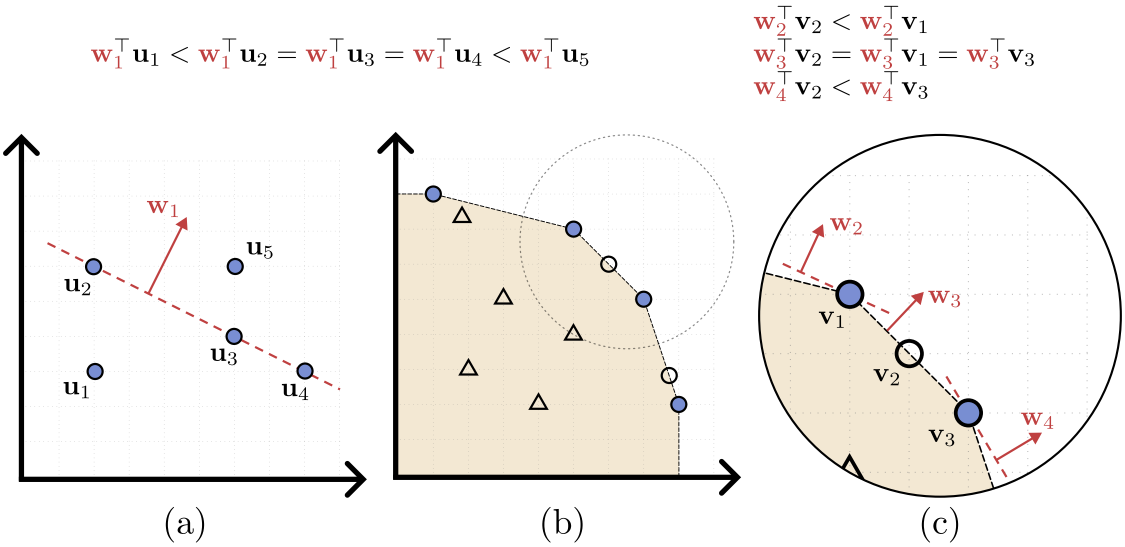

$\mathbf{w}_1^\top \mathbf{u}_1 < \mathbf{w}_1^\top \mathbf{u}_2 = \mathbf{w}_1^\top \mathbf{u}_3 = \mathbf{w}_1^\top \mathbf{u}_4 < \mathbf{w}_1^\top \mathbf{u}_5$
$\mathbf{w}_2^\top \mathbf{v}_2 < \mathbf{w}_2^\top \mathbf{v}_1$
$\mathbf{w}_3^\top \mathbf{v}_2 = \mathbf{w}_3^\top \mathbf{v}_1 = \mathbf{w}_3^\top \mathbf{v}_3$
$\mathbf{w}_4^\top \mathbf{v}_2 < \mathbf{w}_4^\top \mathbf{v}_3$
$\mathbf{w}_1$
$\mathbf{u}_5$
$\mathbf{u}_2$
$\mathbf{u}_3$
$\mathbf{u}_1$
$\mathbf{u}_4$
$\mathbf{w}_2$
$\mathbf{w}_3$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$
$\mathbf{w}_4$
(a)
(b)
(c)

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, $\boldsymbol{\mathcal{V}}\mathbf{als}(CCS_{\min}(\Pi'))$, 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 $\boldsymbol{\mathcal{V}}\mathbf{als}(\Pi')$

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 $\boldsymbol{\mathcal{V}}\mathbf{als}(CCS_{\min}(\Pi))$ 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 $\texttt{cvx\_prune}(\boldsymbol{\mathcal{V}}\mathbf{als}(\Pi')) = \boldsymbol{\mathcal{V}}\mathbf{als}(CCS_{\min}(\Pi'))$. `cvx_prune` can be implemented using *linear programming* [31], 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

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 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) [4] is a tabular dynamic programming algorithm similar to Value Iteration (Section 2.3) that will compute optimal convex

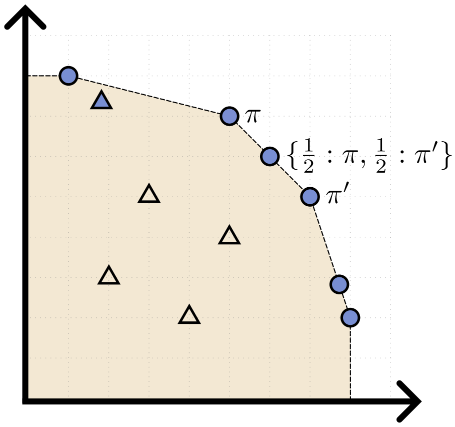


Figure 2.12: An example Pareto front, where the circles correspond to a convex coverage set, and the filled blue points denote the Pareto front. Two points are marked with their corresponding policies π and π', and a mixture policy is shown, denoted $\{\frac{1}{2}:\pi, \frac{1}{2}:\pi'\}$ that follows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

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:

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

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

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: $\bigcup_{a\in\mathcal{A}} \hat{\boldsymbol{\mathcal{Q}}}^{k+1}_{\text{CHVI}}(s,a;t)$.

$$\frac{1}{2} \times \left\{ \begin{pmatrix} 2 \\ 8 \end{pmatrix}, \begin{pmatrix} 4 \\ 6 \end{pmatrix} \right\} + \frac{1}{2} \times \left\{ \begin{pmatrix} 4 \\ 4 \end{pmatrix}, \begin{pmatrix} 8 \\ 2 \end{pmatrix} \right\} = \left\{ \begin{pmatrix} 3 \\ 6 \end{pmatrix}, \begin{pmatrix} 4 \\ 5 \end{pmatrix}, \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 6 \\ 4 \end{pmatrix} \right\}$$

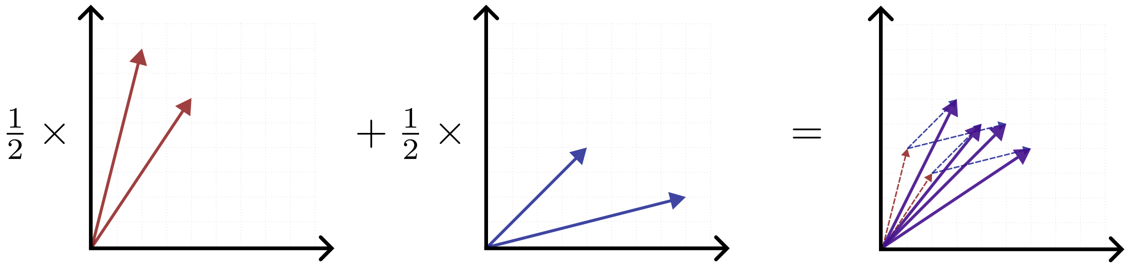


Figure 2.13: An example of arithmetic over vector sets. Two vector sets are shown in red and blue, which are multiplied by $\frac{1}{2}$ using Equation (2.71), and then added using Equation (2.72). On the right the resulting vector set is shown in purple, which are the sum of appropriately scaled red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

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 [4]:

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

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

TODO: make scalarised value notation consistent with how defined in ch5

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 $\mathbf{v}$ 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).

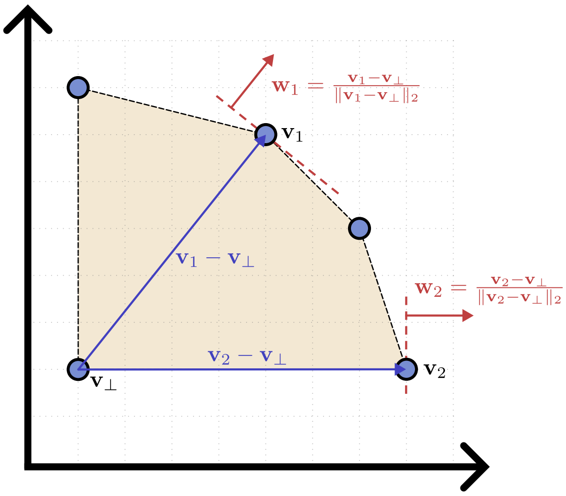

$\mathbf{w}_1 = \frac{\mathbf{v}_1 - \mathbf{v}_\perp}{\|\mathbf{v}_1 - \mathbf{v}_\perp\|_2}$
$\mathbf{v}_1$
$\mathbf{v}_1 - \mathbf{v}_\perp$
$\mathbf{w}_2 = \frac{\mathbf{v}_2 - \mathbf{v}_\perp}{\|\mathbf{v}_2 - \mathbf{v}_\perp\|_2}$
$\mathbf{v}_2 - \mathbf{v}_\perp$
$\mathbf{v}_2$
$\mathbf{v}_\perp$

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 $\mathbf{v}_i$.

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.

However, the *alias method* [43, 42] 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

```
def sample_catagorical($f$):
    threshold $\sim \text{Uniform}(0,1)$
    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 $\sim \text{Uniform}(\{1, ..., m\})$
    $(c_0, c_1,$ thrsh$)$ = alias_table[index]
    $u \sim \text{Uniform}(0,1)$
    if ($u <=$ thrsh):
        return $c_0$
    return $c_1$
```

Listing 2.3: Psuedocode for sampling from an alias table.

categories and `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 [42] 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 [43, 42]. 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 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.

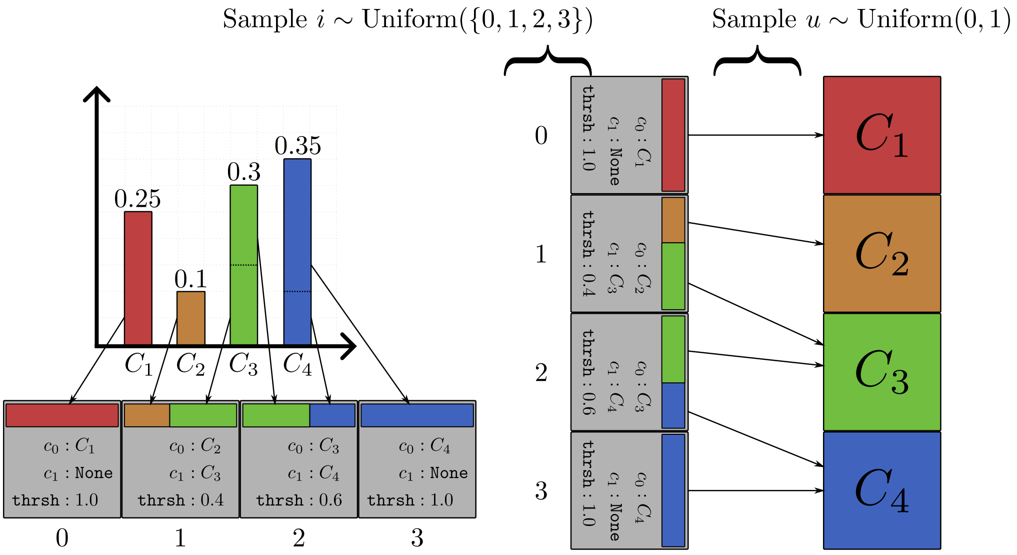


Figure 2.15: Example of building and sampling from an alias table, on a distribution with four categories. Left: shows how the probability mass is split into the four entries of the table. Right: shows how two random variables can be used to sample from the table in $O(1)$ time, where i is used to index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

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

    # fill big and small lists
    for $c$ in $f$:
        if $f(c) > \frac{1}{m}$:
            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 == $\frac{1}{m}$:
            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 > $\frac{1}{m}$:
            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. BD: H-MCTS is used as an UCT style MCTS algorithm that is also designed using simple regret for comparison. (this was in intro to section 4.4, DENTS results section)

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 [17]
- 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)

- 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

This chapter considers MCTS algorithms for planning in single-objective environments, where the algorithm may consider a secondary entropy objective for

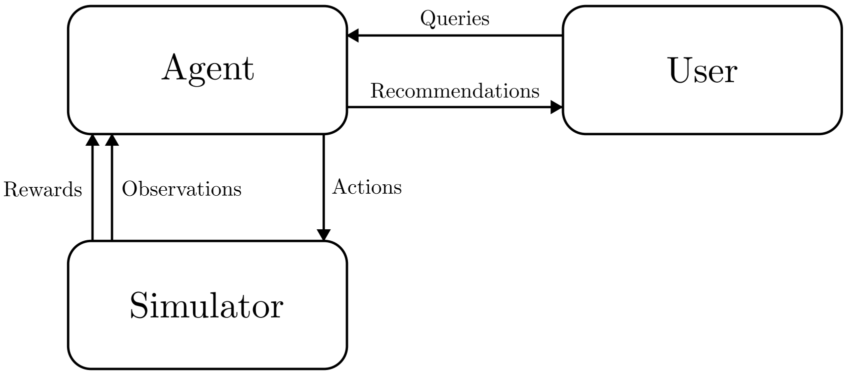


Figure 4.1: Exploration setting in reinforcement learning (edited from Figure 2.5), where the agent is only assessed on the recommendations that it provides, and is not penalised for considering poor actions in the simulator. So this setting places greater emphasis on exploration over exploitation.

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 [33, 15, 25, 44]. 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**.

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:

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

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

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

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

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

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

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

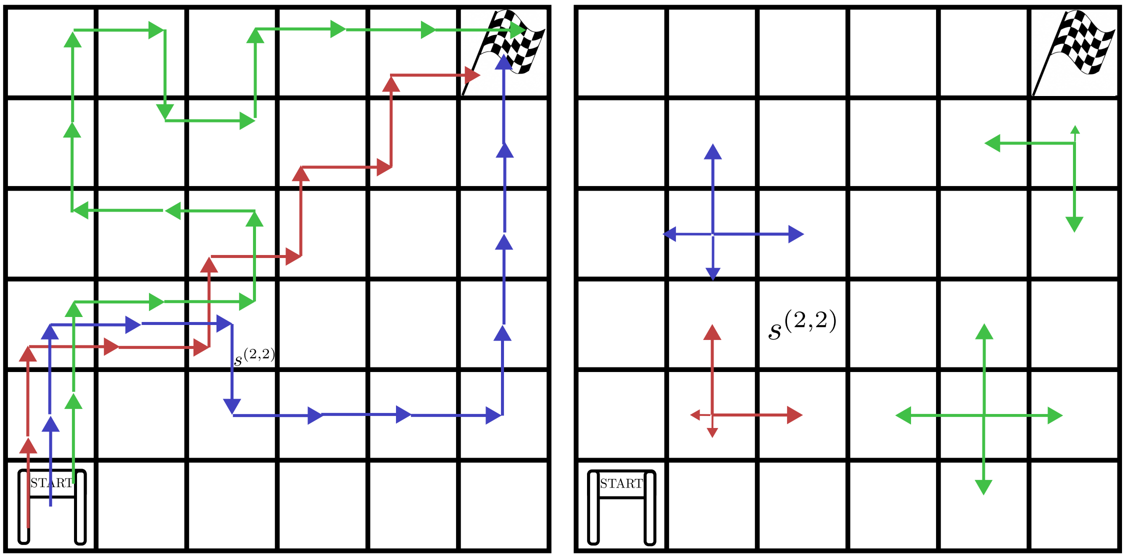
In Section 4.4 similar grid world problems will be considered with greater complexities, such as sparse rewards and stochastic transition distributions.

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 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 $s^{(2,2)}$.



(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

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 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

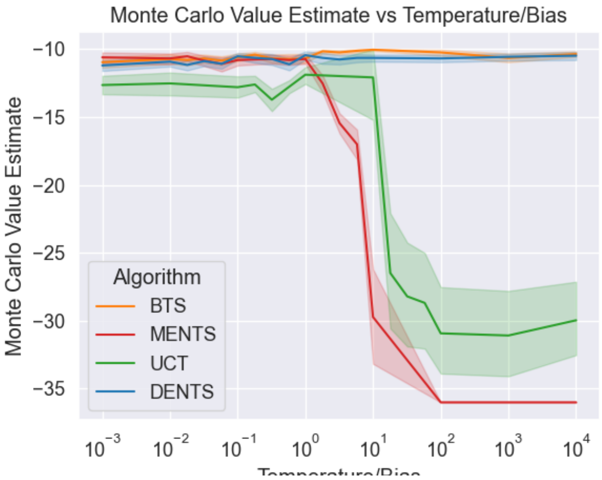


Figure 4.3: Results on the example grid world problem for a variety of UCT bias and temperature parameters. For comparison, the Boltzmann Tree Search (BTS) and Decaying ENtropy Tree Search (DENTS) algorithms defined in Section 4.2 are also shown. Each algorithm was run for 5000 trials, and results were 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)

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.

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 and add cumulative regret. Call subsection regret and consistency.

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.

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

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.

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 (TODO: ref section/theorem, where show MENTS garunteed to converge given a small enough temperature. Maybe quote theorem here?). 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

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 A.

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* $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\text{BTS}})] \to 0$.

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* $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\text{AR-BTS}})] \to 0$*.*

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 $\hat{V}_{\text{DNTS}}$ and $\hat{Q}_{\text{DNTS}}$, but also a secondary value estimate, $\bar{\mathcal{H}}_{V,\text{DNTS}}$ and $\bar{\mathcal{H}}_{Q,\text{DNTS}}$, 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 $\beta_{\text{DNTS}}(x) \geq 0$. Generally this will be set to a function such that $\beta_{\text{DNTS}}(x) \to 0$ 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_{DNTS} is defined follows: TODO: highlight the changes from BTS in red?

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

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

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

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

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

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

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

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

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

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

The recommendation policies for DENTS are also defined identically to BTS:

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

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

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 A for completeness.

Corresponding theoretical results hold for DENTS and AR-DENTS to those stated for BTS. For results that require the temperature function α_{DNTS} to tend to zero, it will now also be required for the entropy weighting function β_{DNTS} 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_{DNTS} tends to zero:* $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\text{DNTS}})] \to 0$.

Theorem 4.2.6. *For any MDP $\mathcal{M}$, if $\alpha_{\text{DNTS}}(x) \to 0$ and $\frac{\beta_{\text{DNTS}}(x)}{\alpha_{\text{DNTS}}(x)} \to 0$, then $\pi_{\text{DNTS}} \xrightarrow{p} \pi^*$ and* $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\text{DNTS}})] \to 0$.

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

Theorem 4.2.8. *For any MDP $\mathcal{M}$, if $\alpha_{\text{AR-DNTS}}(x) \to 0$ and $\frac{\beta_{\text{AR-DNTS}}(x)}{\alpha_{\text{AR-DNTS}}(x)} \to 0$, then $\pi_{\text{AR-DNTS}} \xrightarrow{p} \pi^*$, $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi_{\text{AR-DNTS}})] \to 0$ and $\mathbb{E}[\texttt{sim\_regr}(s_0, \texttt{mv}_{\text{AR-DNTS}})] \to 0$.*

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*.

4.2.3.1 The Alias Method in MCTS

TODO: This subsubsection feels unnecessarily dense right now

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:

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

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

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:

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

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

where S computes the corrected average of $\hat{V}(s')$ for $s' \in \mathrm{Succ}(s_t, a_t) - \{s_{t+1}\}$, $\hat{V}^{\mathrm{old}}(s_{t+1})$ is the previous value of $\hat{V}(s_{t+1})$, and $N^{\mathrm{old}}(s_{t+1}) = N(s_{t+1}) - 1$, $N^{\mathrm{old}}(s_t, a_t) = N(s_t, a_t) - 1$ 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 (4.26) and (4.25), the entropy backups of DENTS:

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

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

Equation (4.35) is of the same form as Equation (4.31), so can also be computed in $O(1)$ time similarly. Equation (4.36) 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 $\pi^{\text{old}}_{\text{DNTS}}$ be the search policy from before the node was visited this trial, when Equation (4.36) can be computed by TODO: Fix overflow. Just use subscript D rather than DNTS?:

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

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

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 from Equations (4.20) and (A.4). These backups are already in a form that takes $O(1)$ to compute. Because the entropy backups can also be computed in amortised $O(1)$ time, the backups for AR-DENTS have an amortised complexity of $O(1)$. Hence, AR-DENTS can be run in `mcts_mode` with the Alias method with an overall complexity of $O(n(H+A))$ to run n trials. As will be seen later in Table 4.1, this can lead to an order of magnitude more trials being run in the same timespan.

4.2.3.2 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

This section builds on the discussion from Section 4.1 to consider extreme cases for which UCT and MENTS will struggle to solve. Coquelin and Munos [10] introduce the *D-chain problem*, which is a theoretical MDP for which UCT stuggles with. They also prove that UCT will suffer a *hyperexponential* regret with respect to the number of states.

The D-chain problem has a sparse unit reward at the end of the chain, and use of the maximum entropy objective is able to solve the problem easily. However, the D-chain problem can be extended to include a *entropy trap*, to create an MDP for which both UCT and MENTS cannot solve in any feasible time.

4.3.1 The D-Chain Problem

In the *D-chain problem*, depicted in Figure 4.5, an agent may take one of two actions at any point throughout the chain. If the action $a^L_{\mathcal{M}}$ is taken from state $s^i_{\mathcal{M}}$, then it collects an immediate reward of $R(s^i_{\mathcal{M}}, a^L_{\mathcal{M}}) = \frac{D-i}{D}$. If the action $a^R_{\mathcal{M}}$ is instead taken all the way to the end of the chain, then the agent will receive a

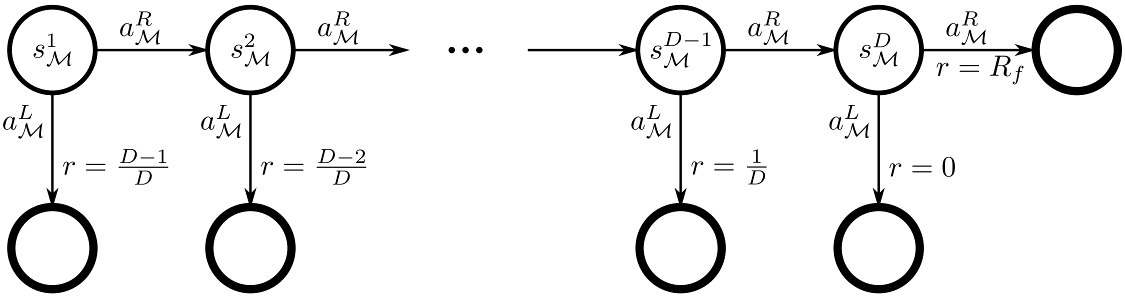


Figure 4.5: An illustration of the *(modified) D-chain problem*, where $s^1_{\mathcal{M}}$ is the starting state and transitions are deterministic. The reward for taking action $a^R_{\mathcal{M}}$ from state $s^D_{\mathcal{M}}$ at the end of the chain is R_f, where $R_f = 1$ in the original D-chain, and $R_f = \frac{1}{2}$ in the modified D-chain.

reward of $R(s^D_{\mathcal{M}}, a^R_{\mathcal{M}}) = 1$. By inspection, the optimal policy is $\pi * (s) = a^R_{\mathcal{M}}$, to follow the chain to the end and collect the maximum reward of one.

To intuitively understand why UCT will struggle on this MDP, the reward function can be decomposed into a dense and sparse component:

$$R_{\text{dense}}(s^i_{\mathcal{M}}, a^L_{\mathcal{M}}) = \frac{D-i}{D}, \tag{4.40}$$

$$R_{\text{dense}}(s^i_{\mathcal{M}}, a^R_{\mathcal{M}}) = 0, \tag{4.41}$$

$$R_{\text{sparse}}(s^i_{\mathcal{M}}, a) = \mathbb{1}[i = D] \cdot \mathbb{1}[a = a^R_{\mathcal{M}}], \tag{4.42}$$

$$R(s,a) = R_{\text{dense}}(s,a) + R_{\text{sparse}}(s,a). \tag{4.43}$$

The two components of the rewards can be seen to encourage the agent to perform two different behaviours. In English, the reward R_{dense} is telling the agent to "leave the chain as soon as possible and collect a reward as soon as possible", whereas the sparse reward R_{sparse} is telling the agent to "stay on the chain until the end". As UCT tries to interleve exploration and exploitation, UCT will take the immediate reward of $\frac{D-1}{D}$ on most trials, rather than exploring to find the reward of one.

Compounded on this, once a trial of UCT finally reaches the unit reward for the first time, the value estimates of $\bar{V}_{\text{UCT}}(s^i_{\mathcal{M}})$ still need many more trials to reach the unit reward for the values to become close to accurate. Coquelin and Munos [10] make these arguments more formally to prove that UCT will suffer a regret of $\Omega((\exp \circ \exp \circ ... \circ \exp)(1))$, where $\circ$ denotes functional composition, and there are D many composed exponential functions.

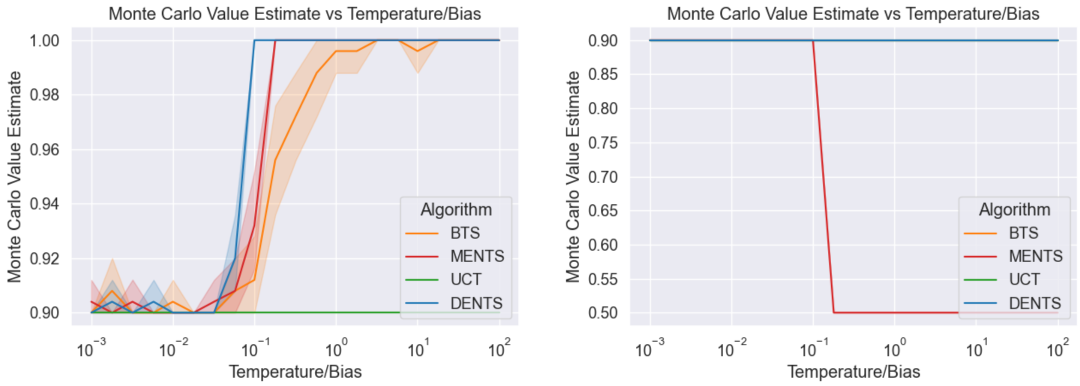


Figure 4.6: Results on the D-chain problem (left), and the modified D-chain problem (right), for a variety of parameters and with $D = 10$. Each algorithm was run for 5000 trials, and results are averaged over 25 repeated experiments. For BTS the temperature function is set to a constant value, and in DENTS the temperature function and entropy weighting function are set to the same constant value. TODO: Make yaxis go from 0 to 1, and add some markers again to make things clearer with overlapping lines.

While from the lens of practical (single-objective) reinforcement the issue may seem like one of poor reward engineering, rewards with conflicting objectives similar to this will naturally arise in the context of multi-objective reinforcement learning. An example of this will be seen in the *Deep Sea Treasure* problem in Chapter 5. TODO: Here I'm referencing that in deep sea treasure, the scalarised reward can be written as $R = w \cdot R_{\text{time}} + (1-w)R_{\text{treasure}}$, TODO: and its very similar in form to my above decomposition.

Introducing entropy as an exploration bonus in this problem helps to solve the problem easily. Once the agent leaves the chain, it can no longer collect an bonus entropy reward, encouraging it to explore down the chain. As can be seen in Figure 4.6, the D-chain problem is an example of an MDP where the standard and maximum-entropy objectives are well aligned.

To highlight that in maximum entropy methods that the temperature parameter needs to be tuned precisely per MDP, consider the *modified D-chain problem*, where the sparse reward at the end of the chain is replaced with $R(s^D_{\mathcal{M}}, a^R_{\mathcal{M}}) = \frac{1}{2}$. In this case the optimal policy takes the immediate reward of $\frac{D-1}{D}$ from the initial state, and the maximum entropy objective is misaligned with the standard objective. TODO: Comment about results

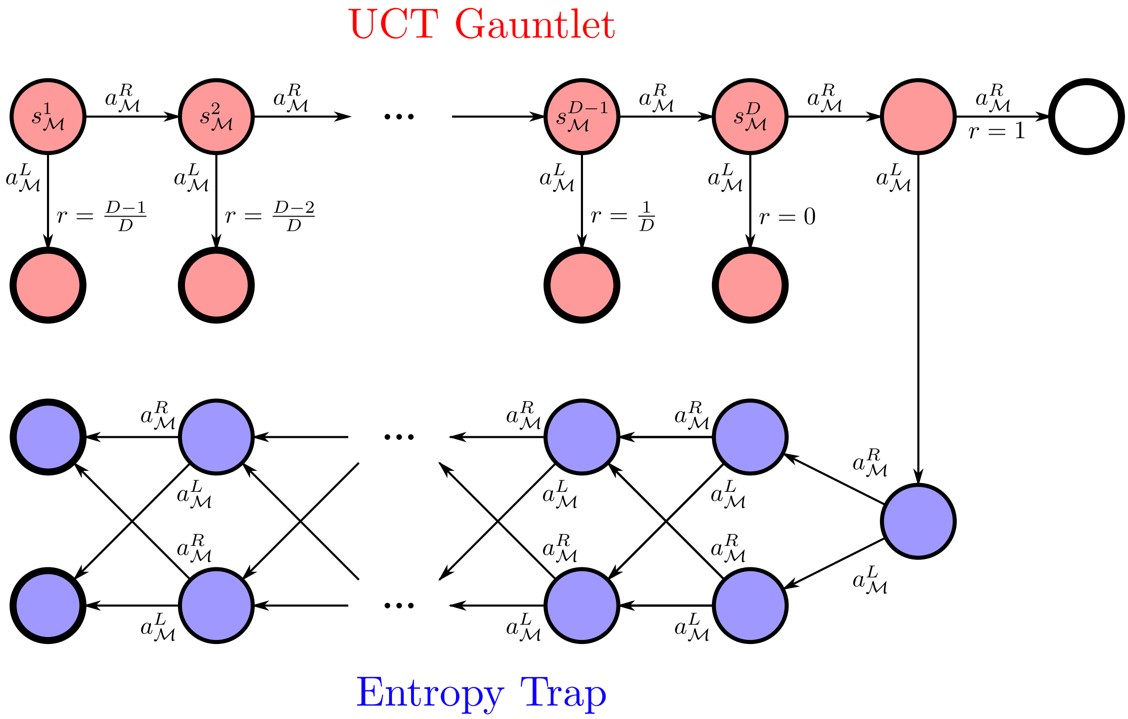


Figure 4.7: An illustration of the *Entropy Trap problem*. In red, the original D-chain problem (Figure 4.5) is labelled as the UCT Gauntlet, and the additional states added for the Entropy Trap are in blue. If a reward is not specified for a transition it is zero.

4.3.2 The Entropy Trap

In the *D-chain with Entropy Trap problem*, or *Entropy Trap problem* for short, an additional chain of states is added to end of the original D-chain to throw off maximum entropy agents. Now at the end of the D-chain, or the *UCT Gauntlet*, the agent has to make a final choice. It can either collect the final sparse reward of one, or it can fall into the *Entropy Trap*, a chain of length E, where zero reward is given, but the agent can collect an entropy bonus of up to $E\log(2)$. The Entropy Trap problem is depicted in Figure 4.7.

When using the maximum entropy objective, with temperature α, in the Entropy Trap problem, an agent can gain an entropy bonus of $\alpha E\log(2)$ from falling in the Entropy Trap. And hence, if $\alpha > \frac{1}{E\log(2)}$, the maximum entropy objective must be misaligned with the standard objective, as the entropy agent can gain an entropy bonus of greater than one from the Entropy Trap. So a maximum entropy agent can be restricted to using an arbitrarily small temperature, by increasing the value of E.

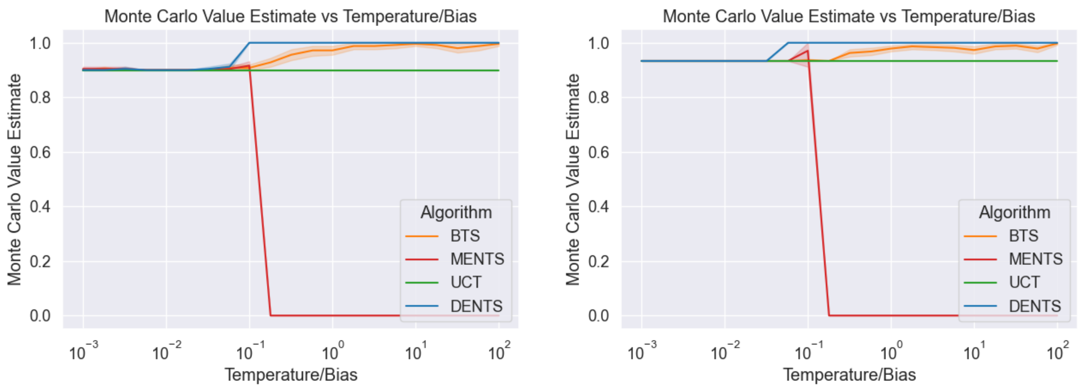


Figure 4.8: Results on the Entropy Trap problem, with $D = E = 10$ (left) and $D = E = 15$ (right). Each algorithm was run with a variety of temperature/bias parameters, and results are averaged over 25 repeated experiments. In the $D = E = 10$ case, each algorithm was run for 5000 trials, and in the $D = E = 15$ case each algorithm was run for 100000 trials. For BTS the temperature function is set to a constant value, and in DENTS the temperature function and entropy weighting function are set to the same constant value.

By setting $D = E$ it can be shown that expectation MENTS will need exponentially many trials with respect to the size of the MDP state space to find the optimal policy:

Theorem 4.3.1. *There exists an MDP such that MENTS (for any value of α_{MENTS}) is either not consistent, or requires an exponential number of trials in the size of the state space. That is, either $\mathbb{E}\texttt{sim\_regr}(s_0, \psi_{\text{MENTS}}) \not\to 0$ or there exists constants $c, k > 0$ such that $\mathbb{E}\texttt{sim\_regr}(s_0, \psi_{\text{MENTS}}) \geq c(1 - \frac{n}{k^{|S|}})$. The latter case implies that $\mathbb{E}\texttt{sim\_regr}(s_0, \psi_{\text{MENTS}}) > 0$ and $\psi_{\text{MENTS}} \neq \pi^*$ for $n < k^{|S|}$.*

Results on the Entropy Trap problem are given for $D = E = 10$ and $D = E = 15$ in Figure 4.8. When using $D = E = 15$ it is feasible that BTS can make progress given enough trials, but ultimately if D is made large enough BTS will also struggle on this environment, as it can only explore randomly. As the likeliehood of reaching the end of the chain through random exploration becomes exponentially less likely, only DENTS is able to solve the Entropy Trap in a reasonable amount of time as $D = E$ is made larger.

4.4 Empirical Results

This section provides empirical results on gridworld environments and on the game of Go. It begins by describing the environments, the evaluation setup, and is followed by the empirical results and discussion. The main algorithms discussed in this chapter, BTS and DENTS will be evaluated, using UCT, MENTS, RENTS, TENTS and H-MCTS as baselines .

4.4.1 Environments

4.4.1.1 (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 sparse reward of 0.99^t is received.

Figure 4.9 gives the specific maps used for Frozen Lake in these experiments. In the 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.

An 8x8 map (Figure 4.9a) is used in Section 4.4.3.1 to demonstrate how the different Boltzmann search and entropy based algorithms perform with a variety of temperature parameters. The 8x12 map (Figure 4.9b) is used for an environment that is sufficiently large map to test and compare algorithms. 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.

4.4.1.2 Sailing Problem

The *Sailing Problem* is another grid world environment with one goal state, at the opposite corner to the starting location of the agent. This problem has often been used to evaluate MCTS algorithms [29, 21, 41, 12].

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

(a) 8x8 Frozen Lake.

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

(b) 8x12 Frozen Lake.

Figure 4.9: Maps used for the Frozen Lake environment. `S` is the starting location for the agent, `F` represents floor that the agent can move too, `H` are holes that end the agents trial and `G` is the goal location.

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

(a) 6x6 Sailing Problem map.

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

(b) Wind transition probabilities.

Figure 4.10: The map and wind transition probabilities used for the 6x6 Sailing Problem. The same notation is used for the map as Figure 4.9 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.

The agent has 8 different actions, where it can travel in any cardinal or semi-cardinal direction. In each state of the MDP, the wind is blowing in a given direction and will stochastically change after every transition. The agent cannot sail directly into the wind. 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. The cost of each action depends on the *tack*, the angle between the direction of the agent's travel and the wind.

Figure 4.10 gives the map and wind transition probabilities for the Sailing problem, using the same map notation as for Frozen Lake.

4.4.1.3 Go

For a more challenging domain, a round-robin tournament was run on the game of Go, which has widely motivated the development of MCTS methods [13, 36, 37]. *Area scoring* is used to score the games, with a *komi* (a score handicap for black) of 7.5.

The open-source library KataGo [45] was used to provide a value network $\tilde{V}$ and prior policy network $\tilde{\pi}$ for the algorithms.

PUCT algorithm [3] was used as a baseline algorithm, as described in Alpha Go Zero [37], using prioritised UCB [32] to utilise the policy neural network. Each algorithm was limited to 5 seconds of compute time per move, allowed to use 32 search threads per move, and had access to 80 Intel Xeon E5-2698V4 CPUs clocked at 2.2GHz, and a single Nvidia V100 GPU on a shared compute cluster.

The methods described in this chapter can be adapted for two-player games by using a discount factor in the MDP. The value of a policy π with a discount factor γ is defined as:

$$V^{\pi}(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1}\gamma^{i-t}r_i \middle| s_t = s\right]. \tag{4.44}$$

TODO: should define this in ch2? It's only really relevant here, and kind of in Frozen lake If the two players are using the policies π_0 and π_1, and π is the policy that interleves π_0 at even timesteps with π_1 at odd timesteps, then using a discount factor of $\gamma = -1$ is sufficient to handle two-player games.

4.4.2 Evaluation Proceedure

Consider an algorithm with search tree $\mathcal{T}$, which provides a partial recommendation policy ψ_{alg}. 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:

$$\psi(a|s) = \begin{cases} 1 & \text{if } s \in \mathcal{T} \text{ and } a = \psi_{\text{alg}}(s), \\ 0 & \text{if } s \in \mathcal{T} \text{ and } a \neq \psi_{\text{alg}}(s), \\ \frac{1}{|\mathcal{A}|} & \text{otherwise.} \end{cases} \tag{4.45}$$

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. 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.

All experiments in this chapter were run with `mcts_mode` turned off and did not use the Alias method as described in Section 4.2.3, apart from the experiments on Go, which used `mcts_mode`and the Alias method.

4.4.2.1 Gridworld Hyperparameter Selection

To select hyperparameters for the algorithms in the gridworld environments, the BayesOpt package [24] was used. Full details on the hyperparameter tuning and the selected parameters are given in Appendix B.1.

These experiments were run with `mcts_mode` turned off and did not use the Alias method (Section 4.2.3).

4.4.2.2 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 and the selected hyperparamters are given in Appendix B.3.

Of note, both Bellman value estimates and average returns were considered for the Go round robin. In the hyperparamter tuning the two versions of each algorithm were tuned seperately and then played against each other. In each case the average returns algorithms decisively beat the Bellman versions. This could have been because of average returns helping to average out noisy neural network outputs, and the ability of the AR algorithms to run more trials (as described in Section 4.2.3) in the given search time.

These experiments were run with `mcts_mode` turned on and made use of the Alias method (Section 4.2.3). Additionally, the value network $\tilde{V}$ was used for the value initialisation function $\hat{V}_{\text{init}}$, and prior policy network $\tilde{\pi}$ was mixed into the search policy as described in Section 4.2.3.

4.4.3 Results and Discussion

This subsection first demonstrates the parameter sensitivity of maximum entropy methods on the Frozen Lake environment. Then results on Frozen Lake and the Sailing problem are given with optimised parameters, and finally results from the Go round robin.

4.4.3.1 Parameter sensitivity in Frozen Lake

Further discussion on sensitivity to the temperature parameter in MENTS [46], RENTS and TENTS [11] is now given. Results on the 8x8 Frozen Lake environment (Figure 4.9a) are given using a variety of temperatures to demonstrate how each algorithm performs with different temperatures in that domain.

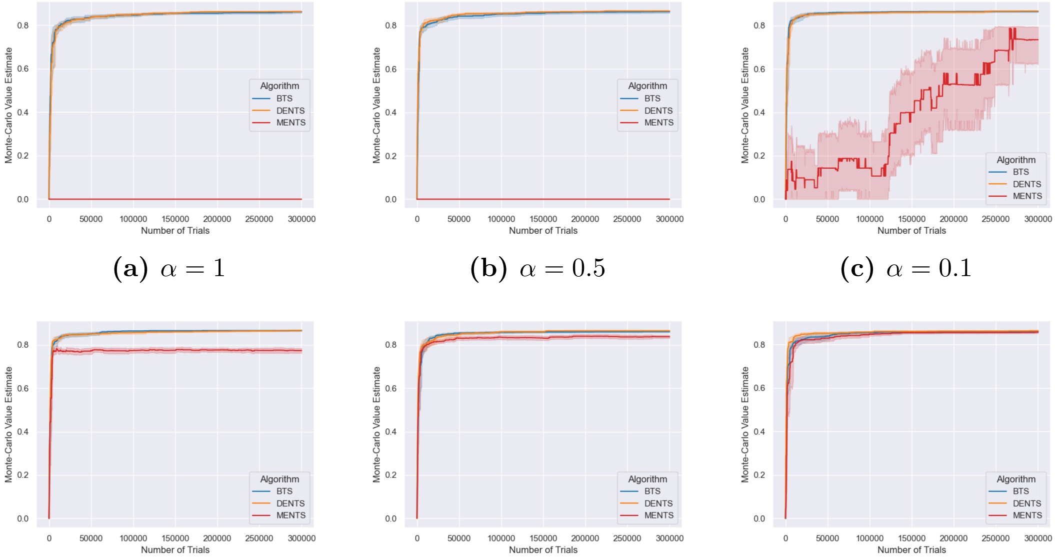
In Figures 4.11 and 4.13 MENTS and TENTS take the scenic route to the goal state for medium temperatures, where the reward for reaching the goal is still significant, but they can obtain more entropy reward by wondering around the gridworld for a while first. For higher temperatures they completely ignore the goal state, opting to rather maximise policy entropy. Interestingly, RENTS in Figure 4.12 fared better than MENTS and TENTS and never really ignored the goal state for all of the temperatures considered.

In contrast, both BTS and DENTS were relativly agnostic to the temperature parameter in this environment (with $\epsilon = 1$) and were always able to find the goal state. Curves for BTS and DENTS are included in all of Figures 4.11, 4.13 and 4.12 for reference and as a comparison for MENTS, RENTS and TENTS.

4.4.3.2 Gridworld Results

The 8x12 Frozen Lake environment (Figure 4.9b) and a 6x6 Sailing Problem (Figure 4.10) were used to evaluate the algorithms introduced in this chapter against the baseline algorithms.

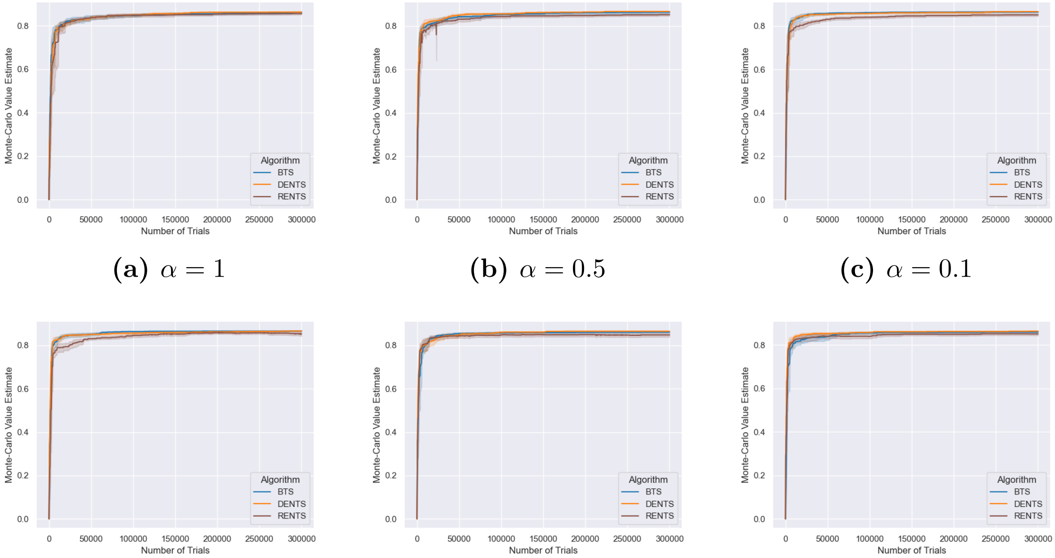
Each algorithm is run 25 times on each environment and evaluated every 250 trials using 250 trajectories. A horizon of 100 was used for Frozen Lake and 50 for the Sailing Problem. Curves of these evaluations are given in Figure 4.14.



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

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

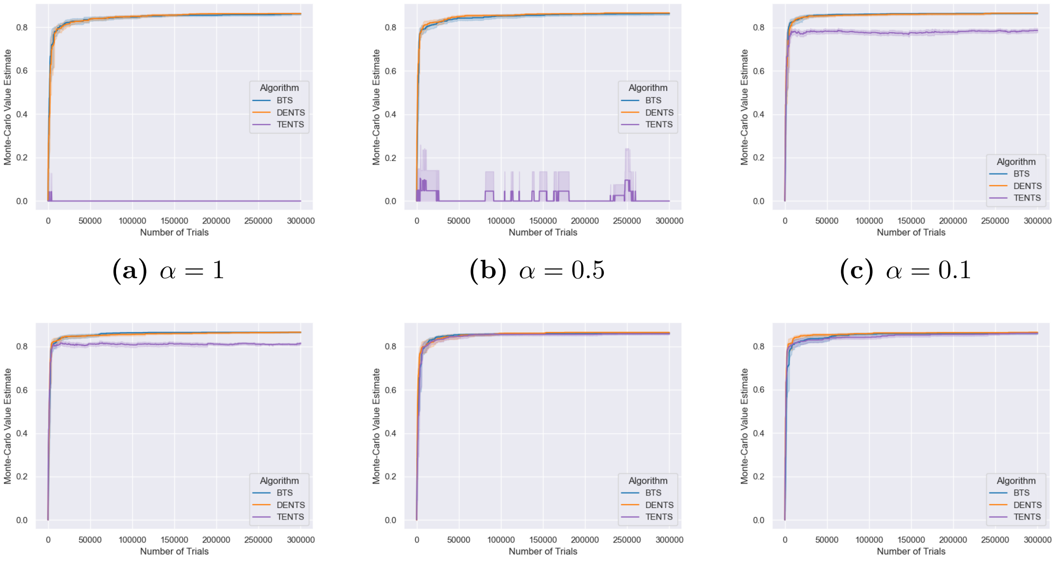
Figure 4.11: MENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference. TODO: Update fig?



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

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

Figure 4.12: 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)** $\alpha = 0.1$

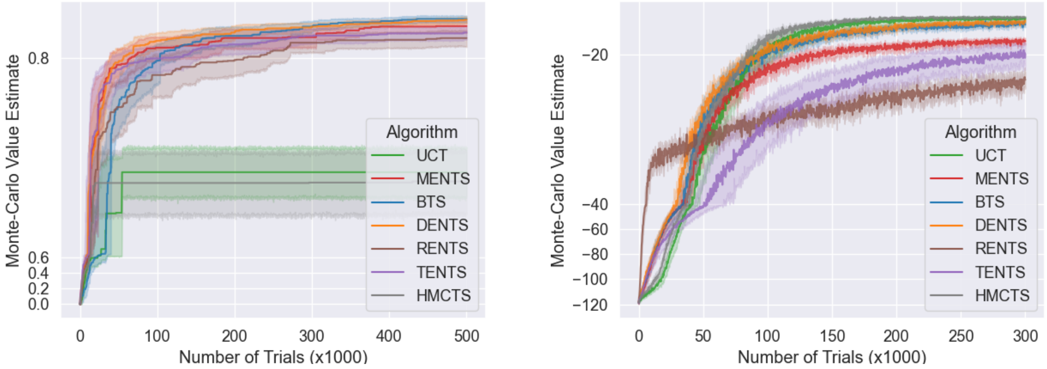
(d) $\alpha = 0.05$ **(e)** $\alpha = 0.01$ **(f)** $\alpha = 0.005$

Figure 4.13: TENTS with a variety of temperatures on an 8x8 Frozen Lake environment. BTS and DENTS are included for reference. TODO: Update fig?

In Frozen Lake (Figure 4.14a), 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 4.14b) 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.



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

Figure 4.14: Results for gridworld environments. TODO: update fig?

4.4.3.3 Go Results

Results of the round-robin are summarised in Table 4.1, where each algorithm played 50 games as black and 50 as white. BTS was able to run the most trials per move and beat all of the other algorithms other than DENTS which it drew. The Alias method 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 [14] algorithm used: the value of a move is typically unaffected by other moves on the board.

To validate the strength of the THTS++ PUCT agent, it was played directly against KataGo [45], with a budget of 1600 trials per move. Interestingly, the THTS++ 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 the neural networks from

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.

KataGo were not fine-tuned, so the Boltzmann search algorithms directly used the networks that were trained for use in PUCT (with average returns).

4.5 Theoretical Results

This Section provides the theoretical analysis and proofs discussed in this chapter. This section is structured as follows:

1. First, in Section 4.5.1, additional notation is defined so that the MCTS process can be reasoned about with respect to the number of trials run;
2. Second, in Section 4.5.2, BD: some preliminary results, lemmas and definitions are given that are used in later results;
3. Third, in Section 4.5.3, the consistency results for (AR-)BTS and (AR-)DENTS are proven (Theorems 4.2.1, 4.2.2, 4.2.4, 4.2.5, 4.2.6, 4.2.8);
4. Fourth, in Section 4.5.4, Theorem 4.3.1 is proven, showing that the maximum entropy objective is either misaligned or BD: not useful in the Entropy Trap problem from Section 4.3;

Some of the particularly verbose arguments are left to Appendix C so that they do not distract from the string of thought in this section. In particular, Theorems 4.2.3 and 4.2.7, which show exponential simple regret bounds on BTS and DENTS are left to Appendix C.2. This section will refer to particular parts of the appendix where they provide any relevant extra details.

4.5.1 Notation

Firstly some additional notation is needed to be able to reason about the MCTS processes over time. Specifically, notation is needed to reason about random variables that vary with the number of MCTS trials run. Similarly to the main text, the assumption that for any two states $s, s' \in \mathcal{S}$ that $s = s'$ if and only if the trajectories leading to them are identical 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: Either say because this assumption is made, can take an arbitrary $s_t \in \mathcal{S}$ TODO: where t denotes the timestep where it can appear, because trajectories unique. OR, try to clean up that we sometimes use an arbitrary st from S. Its probably quite confusion.

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}.

Most variables be denoted with a superscript n to denote that it is the value used on the nth trial of MCTS. For example:

- If the algorithm uses a search policy π and outputs a recommendation policy ψ, then the search policy followed on the nth trial is π^n, and the recommendation policy after n trials is denoted ψ^n.

- When it is necessary to reason about multiple trajectories, the superscript notation will also be used. The trajectory sampled in the nth trial is $\tau^n = (s_0^n, a_0^n, r_0^n, ..., s_{h-1}^n, a_{h-1}^n, r_{h-1}^n, s_h^n)$. If it is sampled from a search policy π, then $\tau^n \sim \pi^n$.

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

It will also be useful to have some additional notation for counting the number of times that nodes have been visited. Let the mth trajectory be $\tau^m = (s_0^m, a_0^m, r_0^m, ..., s_{h-1}^m, a_{h-1}^m, r_{h-1}^m, s_h^m)$ for $1 \leq m \leq n$. The number of times state s_t

was visited in the first n trials is $N^n(s_t)$, and the number of times action a_t was selected from state s_t in the first n trials is $N^n(s_t, a_t)$. Also, let $T^n(s_t)$ (and $T^n(s_t, a_t)$) be the set of trajectory indices that s_t was visited on (and action a_t selected) in the first n trials:

$$T^n(s_t) = \{i | 1 \leq i \leq n, s_t^i = s_t\} \tag{4.46}$$

$$T^n(s_t, a_t) = \{i | 1 \leq i \leq n, s_t^i = s_t, a_t^i = a_t\}. \tag{4.47}$$

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

$$N^n(s_t) = \sum_{i=1}^{n} \mathbb{1}[s_t^i = s_t] = |T(s_t, m)|, \tag{4.48}$$

$$N^n(s_t, a_t) = \sum_{i=1}^{n} \mathbb{1}[s_t^i = s_t, a_t^i = a_t] = |T(s_t, a_t, m)|, \tag{4.49}$$

$$N^n(s_t, a_t) = \sum_{i \in T^n(s_t)} \mathbb{1}[a_t^i = a_t], \tag{4.50}$$

$$N^n(s_{t+1}) = \sum_{i \in T^n(s_t, a_t)} \mathbb{1}[s_{t+1}^i = s_{t+1}]. \tag{4.51}$$

The exception to the superscript notation with respect to the number of trials will be value estimates. They will instead be denoted with respect to the number of times the value has been backed up. To clearly differentiate this counting with respect to backups, instead of with respect to trials, a superscript with brackets will be used. For example, the backups for DENTS value estimates would be written as follows. Let $\tau^n \sim \pi^n_{\text{DNTS}}$ and $\tau^n = (s_0, a_0, r_0, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h)$, the backups for $t = h-1, ..., 0$ are:

$$\hat{Q}_{\text{DNTS}}^{(N^n(s_t,a_t))}(s_t, a_t) = R(s_t, a_t) + \sum_{s' \in \text{Succ}(s_t, a_t)} \left(\frac{N^n(s')}{N^n(s_t, a_t)} \hat{V}_{\text{DNTS}}^{(N^n(s'))}(s') \right), \tag{4.52}$$

$$\hat{V}_{\text{DNTS}}^{(N^n(s_t))}(s_t) = \max_{a \in \mathcal{A}} \hat{Q}_{\text{DNTS}}^{(N^n(s_t,a))}(s_t, a), \tag{4.53}$$

$$\bar{\mathcal{H}}_{Q,\text{DNTS}}^{(N^n(s_t,a_t))}(s_t, a_t) = \sum_{s' \in \text{Succ}(s_t, a_t)} \frac{N^n(s')}{N^n(s_t, a_t)} \bar{\mathcal{H}}_{V,\text{DNTS}}^{(N^n(s'))}(s'), \tag{4.54}$$

$$\bar{\mathcal{H}}_{V,\text{DNTS}}^{(N^n(s_t))}(s_t) = \mathcal{H}(\pi^n_{\text{DNTS}}(\cdot | s_t)) + \sum_{a \in \mathcal{A}} \pi^n_{\text{DNTS}}(a_t | s_t) \bar{\mathcal{H}}_{Q,\text{DNTS}}^{(N^n(s_t,a_t))}(s_t, a_t). \tag{4.55}$$

TODO: Double check we've actually defined the value at leaf nodes as Vinit somewhere in the thesis at least. And make a comment about it here?

For reference, all of the algorithms considered in this Section are written out in full with this additional notation in Appendix C.1.

4.5.2 Preliminaries

Firstly, it is worth noting that mathematically (AR-)BTS is a special case of (AR-)DENTS, with $\beta_{\text{DNTS}}(x) = 0$ and $\beta_{\text{AR-DNTS}}(x) = 0$. Results will be proven about (AR-)DENTS, and the corresponding results for BTS follow from setting the entropy weighting function to zero.

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

$$\Pr\left(|X_n - X| > \varepsilon\right) \to 0 \textit{ as } n \to \infty. \tag{4.56}$$

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

$$\Pr\left(\lim_{n\to\infty} X_n = X\right) = 1. \tag{4.57}$$

The definition of simple regret (Definition 2.1.2) used in this thesis is different to what has been considered by the tree search literature so far TODO: cite: mcts_simple_regret,brue1, which only considers the *immediate regret* of taking a single action step. However, the definition used in this thesis 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.

However, the *immediate simple regret* will be a simpler quantity to work with for proofs:

Definition 4.5.3. *The* immediate simple regret *is defined as as:*

$$\texttt{imm\_regr}_I(s_t, \psi^n) = V^*(s) - \mathbb{E}_{a_t\sim\psi^n(s_t)}[Q^*(s_t, a_t)]. \tag{4.58}$$

The differences between these alternative definitions of simple regret can be reconciled by showing that any bound and convergence results that hold for one must also hold for the other.

Lemma 4.5.1. *$\mathbb{E}\texttt{sim\_regr}(s_t,\psi^n) = O(f(n))$ for all $s_t \in \mathcal{S}$ iff $\mathbb{E}\texttt{imm\_regr}(s_t,\psi^n) = O(f(n))$ for all $s_t \in \mathcal{S}$. Consequently, $\mathbb{E}\texttt{sim\_regr}(s_t,\psi^n) \to 0$ iff $\mathbb{E}\texttt{imm\_regr}(s_t,\psi^n) \to 0$.*

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

$$
\begin{aligned}
\texttt{sim\_regr}(s_t,\psi^n) =& V^*(s_t) - V^{\psi^n}(s_t) && (4.59)\\
=& V^*(s_t) - \mathbb{E}_{a_t\sim\psi^n(s)}[Q^{\psi^n}(s_t,a_t)] && (4.60)\\
=& V^*(s_t) - \mathbb{E}_{a_t\sim\psi^n(s)}[R(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim\Pr(\cdot|s_t,a_t)}[V^{\psi^n}(s_{t+1})]] && (4.61)\\
=& V^*(s_t) - \mathbb{E}_{a_t\sim\psi^n(s)}\Big[Q^*(s,a) - \mathbb{E}_{s_{t+1}\sim\Pr(\cdot|s_t,a_t)}[V^*(s_{t+1})] \\
& \quad + \mathbb{E}_{s_{t+1}\sim\Pr(\cdot|s_t,a_t)}[V^{\psi^n}(s_{t+1})]\Big] && (4.62)\\
=& V^*(s_t) - \mathbb{E}_{a_t\sim\psi^n(s_t)}[Q^*(s_t,a_t)] \\
& \quad + \mathbb{E}_{a_t\sim\psi_n(s_t),s_{t+1}\sim\Pr(\cdot|s_t,a_t)}[V^*(s_{t+1}) - V^{\psi^n}(s_{t+1})] && (4.63)\\
=& \texttt{imm\_regr}(s,\psi^n) + \mathbb{E}_{a_t\sim\psi^n(s_t),s'\sim\Pr(\cdot|s_t,a_t)}[\texttt{sim\_regr}(s_{t+1},\psi^n)], && (4.64)
\end{aligned}
$$

where line (4.62) 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:

$$
\mathbb{E}\texttt{sim\_regr}(s_t,\psi^n) = \mathbb{E}\left[O(f(n)) + \mathbb{E}_{a_t\sim\psi^n(s_t),s_{t+1}\sim\Pr(\cdot|s_t,a_t)}[O(f(n))]\right] = O(f(n)), \tag{4.65}
$$

where the outer expectation is with respect ψ^n (as the recommendation policy is the output of a random process).

TODO: Setting f(n) equal to one of the expected simple regret gives the last bit. As f(n) is O(f(n)). □

4.5.3 Consistency Results

BD: This subsection gives consistency results for (AR-)DENTS and consequently (AR-)BTS. It begins by showing that the Boltzmann search policies will lead to every state and state-action pair being visited infinitely often. Then a lot of inductive step lemmas, and then finally its all tied up in the theorems using inductive arguments.

TODO: All nodes visited infinitely often

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 4.5.2. *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*

$$\pi^m(a) = (1 - \lambda(m))\rho^m(a) + \lambda(m)\frac{1}{|\mathcal{A}|}, \tag{4.66}$$

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:

$$\Pr\left(\bigcap_{m=\ell}^{\infty} a^m \neq b\right) > 0. \tag{4.67}$$

However, from the definition of equation (4.66) it must be that:

$$\Pr\left(a^m = b\right) \geq \frac{\lambda(m)}{|\mathcal{A}|} = \frac{\epsilon}{|\mathcal{A}|\log(e+m)}. \tag{4.68}$$

And so using this lower bound to work out the probability of never sampling action b again after the first $\ell - 1$ samples gives:

$$\begin{aligned}
\Pr\left(\bigcap_{m=\ell}^{\infty} a^m \neq b\right) &= \lim_{k\to\infty} \prod_{m=\ell}^{k} \Pr(a^m \neq b) && (4.69)\\
&\leq \lim_{k\to\infty} \prod_{m=\ell}^{k} \left(1 - \frac{\epsilon}{|\mathcal{A}|\log(e+k)}\right) && (4.70)\\
&\leq \lim_{k\to\infty} \left(1 - \frac{\epsilon}{|\mathcal{A}|\log(e+k)}\right)^{k-\ell} && (4.71)\\
&= 0, && (4.72)
\end{aligned}$$

which is in contradiction to inequality (4.67) that was assumed. Inequality (4.71) holds because the factors in the product are increasing with respect to m. To see why the final limit equality (4.72) holds, consider the simpler function $f(x) = (1 - 1/\log(x))^x$. Taking logarithms and applying L'Hopital's rule TODO: cite? generously gives:

$$\begin{aligned}
\lim_{x\to\infty} \log f(x) &= \lim_{x\to\infty} x \log\left(1 - \frac{1}{\log(x)}\right) && (4.73)\\
&= \lim_{x\to\infty} \frac{\log\left(1 - \frac{1}{\log(x)}\right)}{x^{-1}} && (4.74)\\
&= \lim_{x\to\infty} \frac{\frac{1}{x\log(x)(\log(x)-1)}}{-x^{-2}} && (4.75)\\
&= \lim_{x\to\infty} \frac{-x}{\log(x)(\log(x)-1)} && (4.76)\\
&= \lim_{x\to\infty} \frac{-x}{2\log(x)-1} && (4.77)\\
&= \lim_{x\to\infty} \frac{-x}{2} && (4.78)\\
&= -\infty. && (4.79)
\end{aligned}$$

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 4.5.2, every reachable state (and state-action pair) will be visited infinitely often in a Boltzmann MCTS Process.

Corollary 4.5.2.1. *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 $N(s,m) \overset{as}{\to} \infty$ and $N(s,a,m) \overset{as}{\to} \infty$ 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 4.5.2 when applied inductively at each node. □

TODO: Now already get core BTS and DENTS results.

TODO: DP backups converge in limit (Dynamic Programming cite)

TODO: And corollary is that DENTS with DP backups converges

Theorem 4.2.5 and 4.2.1.

TODO: If Q values converge then can make search policy converge. Generalise the below to be for AR-DENTS and DENTS in one

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 4.5.3. *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 $\bar{Q}_{\text{AR-DNTS}}(s,a) \overset{p}{\to} Q^*(s,a)$ then $\pi_{\text{AR-DNTS}} \overset{p}{\to} \pi^*$*

Proof outline. A full proof to show that these limits converge correctly needs to make use of the dominated convergence theorem TODO: cite. Also use this argument hashed out with gpt https://chatgpt.com/c/67b632f2-0554-8007-9d6c-d24b10a3cf99

The outline argument is that in $\pi_{\text{AR-DNTS}}$ that firstly $\lambda \to 0$ and so the limit of $\pi_{\text{AR-DNTS}}$ will be the same as the limit of $\rho_{\text{AR-DNTS}}$.

TODO: Words about the below maths, and below is very informal writing up

$$\frac{1}{\alpha(n)}\dot{Q}(s,a) + \frac{\beta(n)}{\alpha(n)}\bar{\mathcal{H}}_Q(s,a) \to \frac{1}{\alpha(n)}\dot{Q}(s,a) \tag{4.80}$$

$$\rho(a|s) \to \frac{1}{Z}\exp\left(\frac{\dot{Q}(s,a)}{\alpha(n)}\right) \tag{4.81}$$

$$\to \mathbb{1}[a = a^*] \tag{4.82}$$

TODO: Should probably show that softmax converges to max in limit □

TODO: if search policy converges, then most visited recommendation converge

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 4.5.4. *TODO: write this up properly If $\pi \xrightarrow{p} \pi_{\text{lim}}$ such that $\pi_{\text{lim}}(a*|s) > \pi(a|s)$ for all a, s, then the most visited recommendation policy will converge. Specifically, $\frac{N(s,a,n)}{N(s,n)} \xrightarrow{p} \pi_{\text{lim}}(a|s)$.*

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

$$\frac{N(s,a,n)}{N(s,n)} = \frac{1}{N(s,n)}\sum_{i=1}^{N}(s,n)\mathbb{1}[a^i = a] \tag{4.83}$$

$$\to \mathbb{E}\lim_{n\to\infty}\sum_{i=1}^{n}\frac{\mathbb{1}[a^i = a]}{n} \tag{4.84}$$

$$= \lim_{n\to\infty}\mathbb{E}\sum_{i=1}^{n}\frac{\mathbb{1}[a^i = a]}{n} \tag{4.85}$$

$$= \lim_{n\to\infty}\sum_{i=1}^{n}\frac{\pi^i(a|s)}{n} \tag{4.86}$$

$$= \pi^i(a|s) \tag{4.87}$$

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: if search policy converges and Q values converge, then V values converge.

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 4.5.5. *TODO: clean up this writing If $\pi_{\text{AR-DNTS}} \xrightarrow{p} \pi^*$ (i.e. TODO: conditions from above result), and $\bar{Q}_{\text{AR-DNTS}} \xrightarrow{p} Q^*$, then $\bar{V}_{\text{AR-DNTS}} \xrightarrow{p} V^*$.*

Proof outline. TODO: clean Using the result above TODO: ref, it must be that $\pi_{\text{AR-DNTS}}(a|s) \to \pi^*(a|s) = \mathbb{1}[a^*_{|s} = a]$. Then recalling equation TODO: ref that related the average returns values:

$$\bar{V}(s) = \sum_a \frac{N(s,a,n)}{N(s,n)} \bar{Q}(s,a) \tag{4.88}$$

$$\xrightarrow{p} \sum_a \mathbb{1}[a^*_{|s} = a] \bar{Q}(s,a) \tag{4.89}$$

$$= \bar{Q}(s, a^*_{|s}) \tag{4.90}$$

$$\xrightarrow{p} Q^*(s, a^*_{|s}) \tag{4.91}$$

$$= V^*(s). \tag{4.92}$$

□

TODO: Words about how doing the if V converges, the Q converges step.

Theorem 4.5.6. *Let $\{X_i\}_{i=1}^m$ be random variables drawn from a probability distribution with a cumulative distribution function of F. Let the empirical cumulative distribution function be $F_m(x) = \frac{1}{m} \sum_{i=1}^m \mathbb{1}[X_i < x]$. Then the Dvoretzky-Kiefer-Wolfowitz inequality is:*

$$\Pr\left(\sup_x |F_m(x) - F(x)| > \varepsilon\right) \leq 2\exp\left(-2m\varepsilon^2\right). \tag{4.93}$$

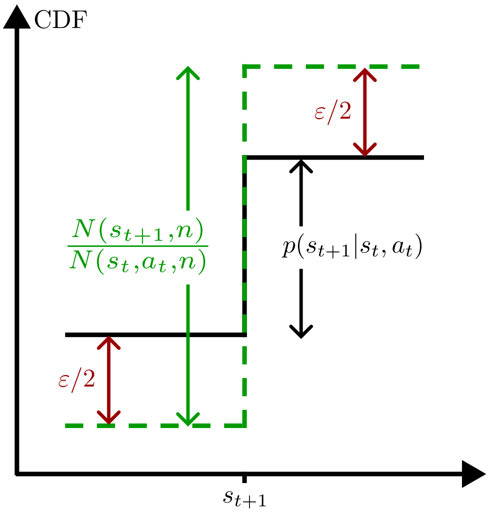


Figure 4.15: Bounding the empirical transition probabilities to the true transition probabilities. The true cdf is shown as a solid black line, the empirical cdf is shown as a dashed green line, and a worst case error of $\varepsilon/2$, using Theorem C.2.9, is shown in red. The probability mass of $p(s_{t+1}|s_t,a_t)$ 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 C.2.9.1.

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

$$\Pr\left(\max_{s_{t+1}\in\text{Succ}(s_t,a_t)}\left|\frac{N(s_{t+1},n)}{N(s_t,a_t,n)} - p(s_{t+1}|s_t,a_t)\right| > \varepsilon\right) \leq 2\exp\left(-\frac{1}{2}\varepsilon^2 N(s_t,a_t)\right). \tag{4.94}$$

Consequently $\frac{N^n(s_{t+1})}{N^n(s_t,a_t)} \xrightarrow{p} p(s_{t+1}|s_t,a_t)$.

Proof outline. By considering some arbitrary ordering over the successor states in $\text{Succ}(s_t,a_t)$ and applying Theorem C.2.9, replacing ε by $\varepsilon/2$, the result follows.

To see why the factor of 1/2 is needed, consider Figure C.1. Because the distribution is discrete, the cumulative distribution function is a (piecewise constant) step function. As Theorem C.2.9 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.

TODO: Final bit holds by taking the limit, specify that it is with respect to N(s,a) tending to inf, not n this time. (Need to also use that N(s,a) goes to inf as n goes to inf from above) □

TODO: Result of how if V rap V*, then Q rap Q*. Follows immediately from the corollary above.

TODO: Give the final theorems Theorems 4.2.2, 4.2.4, 4.2.6, 4.2.8

TODO: Use customthrms, repeat the BTS theorems, and say that they hold from setting beta to zero in the DENTS theorems above.

TODO: This was before the AR proofs originally. Should really just make sure that this is covered in the UCT section in Ch2, and then recall it. Either in the preliminaries, or before the first place its needed in this chapter here where roughly here where it's needed.

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 $\bar{Q}^{N(s,a)}(s,a)$:

$$\bar{V}^{N(s_t)+1}(s_t) = \bar{V}^{N(s_t)}(s_t) + \frac{\bar{R}(s_t) - \bar{V}^{N(s_t)}(s_t)}{N(s_t)+1}, \tag{4.95}$$

$$\bar{Q}^{N(s_t,a_t)+1}(s_t,a_t) = \bar{Q}^{N(s_t,a_t)}(s_t,a_t) + \frac{\bar{R}(s_t,a_t) - \bar{Q}^{N(s_t,a_t)}(s_t,a_t)}{N(s_t,a_t)+1}, \tag{4.96}$$

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:

$$\bar{V}^{N(s_t)}(s_t) = \sum_{a\in\mathcal{A}} \frac{N(s_t,a)}{N(s_t)} \bar{Q}^{N(s_t,a_t)}(s_t,a_t), \tag{4.97}$$

$$\bar{Q}^{N(s_t,a_t)}(s_t,a_t) = R(s_t,a_t) + \sum_{a\in\mathcal{A}} \frac{N(s')}{N(s_t,a_t)} \bar{V}^{N(s')}(s'). \tag{4.98}$$

TODO: should we do the rearrangement/working out for this?

Figure 4.16: 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.

TODO: Necessity 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 4.16. 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:

$$\mathbb{E}\bar{V}^{N(k)}(k) = \frac{\exp\left(\bar{V}^{N(k+1)}(k+1)\right)}{1+\exp\left(\bar{V}^{N(k+1)}(k+1)\right)}\mathbb{E}\bar{V}^{N(k+1)}(k+1) + \frac{1}{1+\exp\left(\bar{V}^{N(k+1)}(k+1)\right)} \cdot 0 \tag{4.99}$$

$$\leq E \cdot \mathbb{E}\bar{V}^{N(k+1)}(k+1) \tag{4.100}$$

$$\leq E \cdot 2E^{D-k} \tag{4.101}$$

$$= 2E^{D-k+1}. \tag{4.102}$$

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.

Here is a bunch of todo's had for the new parts of proofs

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.

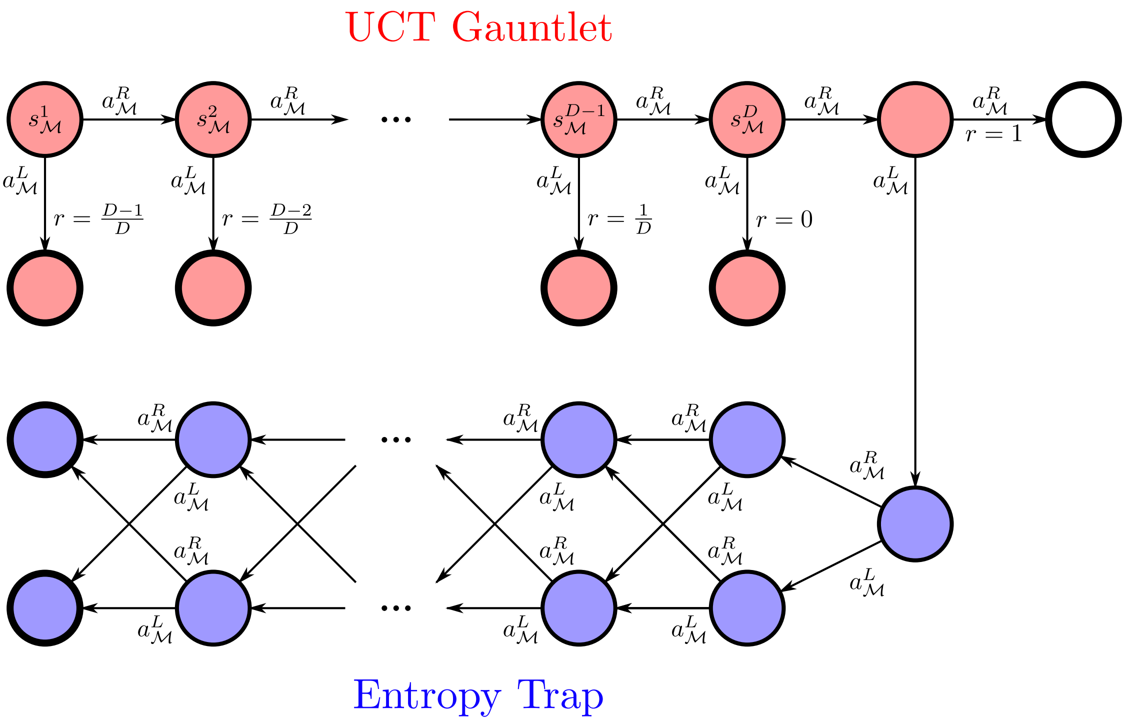


Figure 4.17: An illustration of the *Entropy Trap problem*. In red, the original D-chain problem (Figure 4.5) is labelled as the UCT Gauntlet, and the additional states added for the Entropy Trap are in blue. If a reward is not specified for a transition it is zero. TODO: this fig is copied, update caption?

4.5.4 Entropy Trap

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 4.17 TODO: for ease of reading.

Theorem 4.5.7. *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, 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 $n < k^{|\mathcal{S}|}$. TODO: make sure this has same label and same notation as in Sec 4.3*

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 4.17, 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):

$$\begin{aligned} V^*_{\text{sft}}(E) &= 0 + \alpha_{\text{MENTS}} \cdot \mathcal{H}(\phi) && (4.103)\\ &= \alpha_{\text{MENTS}} \cdot \log(2)K && (4.104)\\ &> 1 && (4.105)\\ &= V^*_{\text{sft}}(F). && (4.106) \end{aligned}$$

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:

$$\hat{V}^{(N(i,n))}_{\text{MENTS}}(i) = \alpha \log\left(\frac{1}{\alpha}\left(\frac{D-i}{D} + \hat{V}^{(N(i+1,n))}_{\text{MENTS}}(i+1)\right)\right) \tag{4.107}$$

$$\leq \max\left(\frac{D-i}{D}, \hat{V}^{(N(i+1,n))}_{\text{MENTS}}(i+1)\right) + \alpha\log(2) \tag{4.108}$$

$$\leq \max\left(\frac{D-i}{D}, \hat{V}^{(N(i+1,n))}_{\text{MENTS}}(i+1)\right) + \frac{1}{K} \tag{4.109}$$

where inequality (4.108) 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 $\hat{V}^{(N(i))}_{\text{MENTS}}(i) \leq \frac{D-(i-1)}{D}$:

$$\hat{V}^{(N(i))}_{\text{MENTS}}(i) \leq \max\left(\frac{D-i}{D}, \hat{V}^{(N(i+1))}_{\text{MENTS}}(i+1)\right) + \frac{\log(2)}{\log(K)} \tag{4.110}$$

$$\leq \frac{D-i}{D} + \frac{1}{K} \tag{4.111}$$

$$\leq \frac{D-i}{D} + \frac{1}{D} \tag{4.112}$$

$$= \frac{D-(i-1)}{D}. \tag{4.113}$$

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

$$\begin{aligned}
\Pr(Z(n)) &= \Pr\left(Z(n) \cap \hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0\right) && (4.114)\\
&= \Pr\left(\neg Y(n) \cap Z(n-1) \cap \hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0\right) && (4.115)\\
&= \Pr\left(\neg Y(n) \middle| Z(n-1) \cap \hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0\right) \Pr\left(Z(n-1) \cap \hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0\right) && (4.116)\\
&\geq (1-2^{-D}) \Pr\left(Z(n-1) \cap \hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0\right) && (4.117)\\
&= \dots && (4.118)\\
&\geq (1-2^{-D})^n \Pr\left(Z(0) \cap \hat{V}^{(N(D))}_{\text{MENTS}}(D) = 0\right) && (4.119)\\
&= (1-2^{-D})^n && (4.120)\\
&\geq 1 - n2^{-D}, && (4.121)
\end{aligned}$$

where the penultimate line used $\Pr\left(Z(0) \cap \left(\hat{V}^{(N(D))}_{\text{MENTS}}(D) = 0\right)\right) = 1$, as $Z(0)$ and $\hat{V}^{(N(D,n))}_{\text{MENTS}}(D) = 0$ 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 $V^{\psi^n_{\text{MENTS}}}(1) \leq \frac{9}{10}$, 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:

$$\begin{aligned}
\mathbb{E}\texttt{sim\_regr}(1, \psi^n_{\text{MENTS}}) &\geq \left(1 - \frac{9}{10}\right) \Pr(Z(n)) && (4.122)\\
&\geq \frac{1}{10}\left(1 - n2^{-D}\right). && (4.123)
\end{aligned}$$

Finally, setting $K = D$, so that $|\mathcal{S}| = 4D + 3 < 5D$ for $D \in \mathbb{N}$ TODO: D more than 1 and so $D = \frac{|\mathcal{S}|-3}{4} > \frac{|\mathcal{S}|}{5}$. Substituting this into the above inequality gives $\mathbb{E}\texttt{sim\_regr}(1, \psi^n_{\text{MENTS}}) \geq 1 - \frac{n}{\sqrt[5]{2}^{|\mathcal{S}|}}$, which is greater than 0 for $n < \sqrt[5]{2}^{|\mathcal{S}|}$. 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.

5

Convex Hull Monte Carlo Tree Search

Contents

BD: list, brain dumping it out.

This chapter. . .

- Sec 5.1, introduction. Starting with existing MOMCTS methods discussed in Ch3 TODO: ref. These algorithms come in two flavours. One where a point estimate is maintained, and one where a convex hull is maintained. This section demonstrates why point estimates are not sufficient, and
- Sec 5.2, convex hull monte carlo tree search
- Sec 5.3, contextual zooming for trees

- Sec 5.5, results
- Sec 5.6, theoretical results

TODO: link back to questions and how we're answering them here

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

First prior multi-objective MCTS (MOMCTS) algorithms are discussed.

BD: Point estimates insufficient can be copies from CHMCTS paper

BD: In prior methods that use a convex hull TODO: ref BD: , the environment is assumed to be deterministic, and backups can be seen to be a special case of Convex Hull backups from Chapter 2 TODO: ref. BD: Therefore, for the remainder of this chapter, most algorithms will use convex hull backups, similar to Convex Hull Value Iteration (todo ref), as in the deteminstic MOMDP case it reduces to the original algorithms, and extends them for stochastic MOMDPs. As such, this chapter, will predominantly focus on how perform action selection in MOMCTS (exploration?)

BD: this paper (below) talks about the need for making consistent action selections. Because the aim is to discover for all weights w an optimal policy. Given a w, actions should be taken consistently to try and maximise the objective.

BD: The hypervolume is the area under the convex hull. The optimal convex hull is the unique convex hull that maximised the hypervolume. The premise of some of the prior work is to maximise the hypervolume at every node. Or the premise is to select from the "optimal actions" that lie on the convex hull. However, reducing the problem back to discovering the optimal policies from the root node, this objective is a bit excessive. The optimal convex hull is not needed at non root nodes. TODO: wrote a bit about this in confirmation in a wordy way, copy that? BD: Consider a 2d problem, with two actions, which correspond to optimising each of the objectives. Once a step has been taken in the MOMDP, the agent has in some sence committed to a weighting over the objectives, and it does not make sense for the agent to then select actions to optimise the other objective, as if the agent wanted to optimise the other objective it would have made a different first step. This has also been observed by XXX et al, who do YYY to solve this issue. In this work, a different solution is considered, by turning the problem into a contextual one.

TODO: want details in contextual tree search, but want to link to it from here. The detail above really should go below in sec 5.1.1

BD: This example, and an example from XXX et al, are given in subsection TODO ref, to concretely demonstrate these issues.

TODO: Demonstrate the need for consistent action selection according to objectives. This paper talks about the same issue: https://jmlr.org/papers/volume15/vanmoffaert14a/var so should read, cite and discuss. They essentially use some policy tracking

5.1.1 Contextual Tree Search

TODO: Talk about how values can be extracted from the convex hull objects given a context, and how this can be used to solve this consistent action problem. Refer back to section 2, and say that for decision support, when a policy is chosen, it implies a weight vector, which can be used to extract the policies.

TODO: use this to highlight that the intuition, of maximising the hypervolume or trying to find an optimal convex hull at every node is unnecessary. And leads to suboptimal behaviour and exploration.

TODO: Use the simple example of reward in dim 1 for action 1 and reward in dim 2 for action 2

TODO: Also make a simple 3 step example demonstrating what the vanmoffaert paper does, where there is a joint state on the two optimal paths. Which is an example of just picking actions from even the optimal convex hull at each step does not mean that you are following an optimal policy.

BD: So the main things to highlight with these two examples. One, that taking the objective to compute each local convex hull (the full convex hulls at not root node) is not necessary, and leads to suboptimal exploration. Two, that even if you are given all of the optimal convex hulls at every state, that some form of context or history is required to follow a globally optimal policy.

TODO: Argue that in the case of decision support, if a context/weight vector can be obtained from the selection of a policy by the user, then the weight vector is a natural choice (and sufficient) for the context. As once given a context, the optimal choice is defined at every step. (And reference the theory section (and quote the result here).)

TODO: Front load results on the 2 action tree search. Demonstrate that the prior works dont do well on it, and show that ours does.

TODO: Introduce contextual (simple) regret

TODO: Below is brain dumping this section on 22 feb. Didn't look at CHMCTS paper at all when writing it.

This subsection discusses the use of *contextual tree search* for planning in multi-objective settings. BD: And highlights why any multi-objective tree search algorithm needs to be contextual to run effectively through an example MOMDP.

TODO: Example MOMDP where the rewards are a circle. Have an image of the rewards, have an example of the 2 action 2D, and 3 action 2D envioronments. Also want to cite the EUM point generation that we used to generate points uniformly on a hypersphere, which allows the environment to be generate for an arbitrary (num actions, dimension) pair

In previous work TODO: refs back to litrev, the rough intuition used is at every node in the tree that a full convex hull is desirable, and maps action selection in a variety of ways to try and achieve this.

Using this example MOMDP TODO: reffering to the 3d one?, where there are M actions, it is quite clear from inspection what policies form the Convex Hull.

$\phi^i(s) = a_{\mathcal{M}(i)}$ is the optimal policy for the BD: weight vector $\mathbf{w}^i = \mathbf{R}(\cdot, a_{\mathcal{M}(i)})$ TODO: probably want to fix up notation. The convex hull is from inspection $\{\phi^i | 1 \leq i \leq M\}$.

BD: The point of this MOMDP is that really is that its easy to know optimal CH from inspection, and should be relatively easy to solve for any tree search algorithm.

To motivate why contextual tree search is a necessary component of MOMCTS, consider $s_0 \rightarrow a_0 \rightarrow s_1$ TODO: clean notation and that from state s_1 onwards, we know that the optimal action must be to continue taking a_0 until the trial ends. In some sense, the actions for the remainder of the trial should be consistent with any actions previously taken in the trial. Otherwise, the search policy will be essentially flipping between trying to optimse for different combinations of objectives and in effect optimising for none of them. TODO: This paragraph very brain dump.

The proposal in this thesis is for every trial to sample a *context* weight vector, where for now it will be assumed that $\mathbf{w}^i \sim U(\Delta^D)$ TODO: just realised the Delta in BTS section might be overloaded notation? ALSO double check what notation used for simplex. ALso clean this equation up. This context is then used to allow algorithms to follow a consistent objective over all actions in a trial. TODO: There may be other ways to achieve this, but the proposal of contextual tree search solves this problem relatively simply

TODO: Some experiments showing that the prior work on MOMCTS fails horribly on this MOMDP.

BD: As the work in this chapter will follow the typical pattern of taking a multi-armed bandit problem and applying it to a sequential decision making problem, the theoretical quantity that will be considered in this chapter is contextual (cumulative) regret.

TODO: Define the typical MAB problem figure, and contextual regret equation.

TODO: HERE HERE HERE HERE. NExt = write up the typical MAB problem figure and defn contextual regret. With new command local. Then move onto defining contextual zooming for trees and CHMCTS. Doesn't matter if the notation is a bit off, as long as the equations which are slow to type out are typed out.

5.2 Convex Hull Monte Carlo Tree Search

TODO: part 1: give generalised CHMCTS backups

TODO: part 2: say that this generalises the prior works from ch3

TODO: part 3: state the search policies of the prior works, and label them

TODO: part 4: dont know if can be bothered to implement, but updating N(s) by w and making it a visitation vector now makes more sense than just a count.

This section introduces *Convex Hull Monte Carlo Tree Search* (CHMCTS), which generalises the prior work in multi-objective MCTS TODO: ref ch3 to handle stochastic environments by using backups over *Convex Hull Value Sets* from Chapter 2 TODO: ref ch2. First the backups, which will be common across all of the following algorithms.

Let ALG be a substitute acronym for any of the following algorithms. At each decision node s a value set is maintained $\hat{\boldsymbol{\mathcal{V}}}_{\text{ALG}}(s)$, and at chance nodes (s,a) the value set $\hat{\boldsymbol{\mathcal{Q}}}_{\text{ALG}}(s)$ is maintained. The search policy, $\pi_{\text{ALG}}(\cdot|s,\mathbf{w})$, is *contextual* and will depend on a randomly sampled $\mathbf{w}\in\Delta^D$ for each trial. BD: Assume for now that the search policy is defined, as it will differ for each algorithm.

Given a context objective $\mathbf{w}\in\Delta^D$ and a trajectory $\tau=(s_0,a_0,\mathbf{r}_0,...,s_{h-1},a_{h-1},\mathbf{r}_{h-1},s_h)\sim\pi_{\text{ALG}}$ TODO: want the w in the pialg bit somehow? Maybe make it superscript? But dont want it to mess with notation for theory. the value estimates are updated for $t = h-1,...,0$:

$$\hat{\boldsymbol{\mathcal{V}}}_{\text{ALG}}(s_t) \leftarrow \texttt{cvx\_prune}\left(\bigcup_{a_t\in\mathcal{A}}\hat{\boldsymbol{\mathcal{Q}}}_{\text{ALG}}(s_t,a_t)\right), \tag{5.1}$$

$$\hat{\boldsymbol{\mathcal{Q}}}_{\text{ALG}}(s_t,a_t) \leftarrow \mathbf{R}(s_t,a_t) + \sum_{s'\in\text{Succ}(s_t,a_t)}\frac{N(s')}{N(s_t,a_t)}\hat{\boldsymbol{\mathcal{V}}}_{\text{ALG}}(s'). \tag{5.2}$$

where the `cvx_prune` operation is defined in (2.68), BD: but repeated here for clarity over an arbitrary set of vectors $\boldsymbol{\mathcal{V}}$:

$$\texttt{cvx\_prune}(\boldsymbol{\mathcal{V}}) = \{\mathbf{v}\in\boldsymbol{\mathcal{V}}|\exists\mathbf{w}\in\Delta^D.\forall\mathbf{v}'\in\boldsymbol{\mathcal{V}}-\{\mathbf{v}\}.\mathbf{w}^\top\mathbf{v}>\mathbf{w}^\top\mathbf{v}'\}. \tag{5.3}$$

Additionally, because the search policies are contextual TODO: in the following subsections (todo ref), it will be useful to define *contextual value estimates*, given a context objective $\mathbf{w} \in \Delta^D$ as follows:

$$\hat{V}^{\mathbf{w}}_{\text{ALG}}(s) = \max\{\mathbf{w}^\top \mathbf{v} | \mathbf{v} \in \hat{\boldsymbol{\mathcal{V}}}_{\text{ALG}}(s)\}, \tag{5.4}$$

$$\hat{Q}^{\mathbf{w}}_{\text{ALG}}(s,a) = \max\{\mathbf{w}^\top \mathbf{q} | \mathbf{q} \in \hat{\boldsymbol{\mathcal{Q}}}_{\text{ALG}}(s)\}. \tag{5.5}$$

TODO: define the policies for the algorithms in chapter 3. also think want to justify the equivalence, because they will use some $\bar{\mathbf{R}}$ TODO: in the updates to the convex hulls. Decompose $\bar{\mathbf{R}}(s_t) = \mathbf{R}(s_t, a_t) + \bar{\mathbf{R}}(s_{t+1})$, and say that this would only get added to CHVS at s_t iff at $t+1$ the second part was added to the convex hull.

TODO: should just write up the ch3 stuff before writing this tidbit

The algorithms described in Chapter 3 TODO: ref can be realised using the above backups, with the search policies taking the following form:

$$\pi_{\text{ALG}}(s|\mathbf{w}) = \arg\max_{a\in\mathcal{A}} \left[\zeta_{\text{ALG}}(\boldsymbol{\mathcal{Q}}(s,a)) + c_{\text{ALG}}\sqrt{\frac{\log(N(s))}{N(s,a)}} \right] \tag{5.6}$$

TODO: write up the policies for the ch3 algorithms by defining zeta for the different algorithms.

TODO: correct that pareto ucb uses a different search policy too

Below = previous writing on this

TODO: list

- Give convex hull monte carlo tree search
- Contextual zooming with the convex hull backups

TODO: Discuss property of extracting from convex hull

TODO: A range of action selection methods will be used.

Now that contextual tree search has been motivated, *Convex Hull Monte Carlo Tree Search* is given in a form that generalises the works of TODO: cite lots.

TODO: CHMCTS backups

TODO: Contextual Search policy

TODO: Restate and name the prior works search policies

5.2.1 CH-UCT

TODO: part 1: give action selection for CH-UCT and talk about it

TODO: part 2: discuss more?

A natural way to extend the UCT search policy for contextual tree search is to use the contextual value estimates (5.4) and (5.5) in the UCT search policy. Call this algorithm *UCT with Convex Hulls* (CH-UCT). This gives the following search policy:

$$\pi_{\text{CH-UCT}}(s|\mathbf{w}) = \arg\max_{a\in\mathcal{A}} \left[\hat{Q}^{\mathbf{w}}_{\text{CH-UCT}}(s,a) + c_{\text{CH-UCT}}\sqrt{\frac{\log(N(s))}{N(s,a)}} \right]. \tag{5.7}$$

BD: One issue is that the confidence interval terms are shared between all possible context objectives, and may assign a high confident, or tight bound, for contexts that haven't been explored down this path much. A hueristic way to solve this is to a vectorised visit counts $\mathbf{M}(s)$, $\mathbf{M}(s,a)$. These values are backed up after a trial for context $\mathbf{w}$ as follows:

$$\mathbf{M}(s) \leftarrow \mathbf{M}(s) + \mathbf{w}, \tag{5.8}$$

$$\mathbf{M}(s,a) \leftarrow \mathbf{M}(s,a) + \mathbf{w}. \tag{5.9}$$

Then a contextual visit count can be defined, and used in the search policy as follows:

$$M(s;\mathbf{w}) = \mathbf{w}^\top \mathbf{M}(s) \tag{5.10}$$

$$M(s,a;\mathbf{w}) = \mathbf{w}^\top \mathbf{M}(s,a) \tag{5.11}$$

$$\pi_{\text{CH-UCT}}(s|\mathbf{w}) = \arg\max_{a\in\mathcal{A}} \left[\hat{Q}^{\mathbf{w}}_{\text{CH-UCT}}(s,a) + c_{\text{CH-UCT}}\sqrt{\frac{\log(M(s;\mathbf{w}))}{M(s,a;\mathbf{w})}} \right]. \tag{5.12}$$

Note that the original value of $N(s)$ is still needed in the convex hull backups for Q-value sets, to empirically estimate the transition distribution $p(s'|s,a) \approx \frac{N(s')}{N(s,a)}$, however this can be recovered from the vectorised visit count by summing over the values of the vector:

$$N(s) = \sum_{i=1}^{D} M(s)_i \tag{5.13}$$

$$N(s,a) = \sum_{i=1}^{D} M(s,a)_i. \tag{5.14}$$

TODO: notation above isn't clean.

TODO: Why doesnt a CH-UCT withe confidence bounds around the DP values you get from the convex hulls still work? Couldn't we do a k-d tree of visit counts for contexts, or some other way of counting how often a context has been searched for? There would be some lemma that the number of visits, is correct, for some epsilon ball around the weight, which the epsilon gives a small enough error in the value to be fine, except for the visit from the nodes above it in the kd tree. ANd then require exp many more trials to count for each level of the tree.

TODO: Add to ORIMenu to just submit some arxiv shit for something like "an addendum to convex hull monte carlo tree search", where we basically copy past this chapter. Actually, I think just take this chapter and publish it with the definitions cut down. Say for litreview either look at the CHMCTS or Thesis literature reviews.

5.2.2 CH-DENTS

TODO: part 1: give action selection for CH-DENTS and talk about it

TODO: part 2: say CH-BTS is same without entropy backup

TODO: part 2: discuss entropy, say that it's just an approximate exploration bonus now

5.3 Contextual Zooming for Trees

TODO: part 1: say that a common approach from tree search is to think about each node as solving a (non-stationary) multi-armed bandit problem

TODO: part 2: following a similar approach, contextual zooming is an approach for contextual bandits

TODO: part 3: have a diagram of a box, UCB right to UCT, UCB down to CZ, UCT down to CZT, CZ right arrow to CZT

TODO: part 4: describe the algorithm. say that it's for a fixed number of trials. talk about doubling trick and how it can be used to bound contextual regret for unknown number of trials. In practise just use the equations without thinking about it.

TODO: part 5: subsection on CH-CZT, which is just using the convex hull backups for the recommendation policy.

TODO: this should mostly be copying out from the chmcts paper.

Below is old writing on this section

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)

TODO: Old todo from contextual tree search section, but more appropriate here: Discuss how UCT = running a non-stationary UCB at each node, so given above discussion, there is work in contextual MAB

BD: taking a slight tangent from the exploration reinforcment learning setting, this section introduces Contextual Zooming for Trees, which will be used as a baseline contextual algorithm in experiments and considered for an action selection mechanism.

BD: Following a similar methodology to UCT — which runs UCB on a non-stationary multi-armed bandit problem at each node in the search tree — *Contextual Zooming for Trees* (CZT) runs *Contextual Zooming* (Chapter 2, TODO REF) on a non-stationary contextual multi-armed bandit problem at each node in the search tree.

5.4 Results

TODO: part 0: discuss the evaluation prodceedure, and how the results are evaluated.

TODO: part 1: run algorithms on the tree envs that show the context dependent stuff better

TODO: part 1a: run algorithms with and without vectorised visit counts and show this helps with exploration?

TODO: part 2: run on suite of mo-gym environments

TODO: part 3: find environments where the convex hull blows up to motivate ch6. Give the CH fully and plot it out.

TODO: part 3a: also consider showing some of the optimal convex hulls where possible

TODO: list

- Results from CHMCTS paper
- Get same plots from C++ code, but compare expected utility, rather than the confusing hypervolume ratio stuff

BD: A range of algorithms will be considered in this experimental section. Firstly, for baselines, the convex hull backup versions of algorithms from Ch3 (REF) will be used. Additionally, CZT, CH-CZT, CH-UCT, CH-BTS and CH-DENTS will be considered

This section provides an empirical evalutation of the algorithms developed in this chapter. The baseline algorithms considered will also be CHMCTS algorithms, using the action selection methods outline in Sections TODO: ref litrev and above.

5.4.1 Evaluation Proceedure

Firstly, given a context $\mathbf{w}$ and CHMCTS search tree $\mathcal{T}$ that defines a partial recommendation poilicy ψ_{alg}, the policy will be made complete similarly to Section TODO: ref eval proceedure in ch4:

$$\texttt{complete}(\psi_{\text{alg}})(a|s;\mathbf{w}) = \begin{cases} 1 & \text{if } s \in \mathcal{T} \text{ and } a = \psi_{\text{alg}}(s;\mathbf{w}), \\ 0 & \text{if } s \in \mathcal{T} \text{ and } a \neq \psi_{\text{alg}}(s;\mathbf{w}), \\ \frac{1}{|\mathcal{A}|} & \text{otherwise.} \end{cases} \tag{5.15}$$

TODO: should reword to account for CZT and how it does the same thing

The Convex Hull (of policies) can then be written as

$$\Psi_{\text{alg}} = \{\texttt{complete}(\psi_{\text{alg}}) \mid \mathbf{w} \in \Delta^D\}. \tag{5.16}$$

This Convex Hull of policies is then evaluated using the *Expected Utility Metric* (EUM), which is defined as:

$$\mathrm{EUM}(\Psi_{\mathrm{alg}}) = \mathbb{E}_{\mathbf{w}\sim U(\Delta^D)}\left[\max_{\psi\in\Psi_{\mathrm{alg}}} \mathbf{w}^\top \mathbf{V}^{\psi}(s_0)\right], \tag{5.17}$$

where $U(\Delta^D)$ is the uniform distribution over vectors in Δ^D. A monte-carlo estimate of $\mathrm{EUM}(\Psi_{\mathrm{alg}})$ will be computed, following TODO: cite the deep MORL paper, to evaluate the algorithms:

$$\mathrm{EUM}(\Psi_{\mathrm{alg}}) \approx \sum_{\mathbf{w}\in E(\Delta^D)} \mathbf{w}^\top \mathbf{V}^{\tau(\mathbf{w})}, \tag{5.18}$$

where $E(\Delta^D)$ is an evenly spaced set of points in Δ^D, $\tau(\mathbf{w}) \sim \texttt{complete}(\psi_{\mathrm{alg}})$ is a sampled trajectory for the point $\mathbf{w} \in \Delta^D$, and $\mathbf{V}^{\tau(\mathbf{w})}$ is the cumulative return from the trajectory. That is, if TODO: double check indices are consistent with how been doing it before the trajectory is $\tau(\mathbf{w}) = (s_0, a_0, \mathbf{r}_0, ..., s_{h-1}, a_{h-1}, \mathbf{r}_{h-1}, s_h)$, then $\mathbf{V}^{\tau(\mathbf{w})} = \sum_{i=0}^{h-1} \mathbf{r}_i$.

For the results given in TODO: ref, the number of sampled used to approximate the EUM was XXX. That is $|E(\Delta^D)| = XXX$.

TODO: double check this is consistent with the litrev paper on MORL and also cite it. Also what are the other metrics?

Experiments are run with `mcts_mode` turned off, as no memory issues were encountered during the evalutation.

Similarly to the gridworld environments in Chapter TODO: ref, the BayesOpt package [24] was used to tune the hyperparameters of all algorithms in this chapter. Full details on the hyperparameter tuning and the selected parameters are given in Appendix TODO: new appendix? join the appendix?.

5.4.2 Environments

The MO-Gymnasium TODO: cite suite of environments was used to evaluate the algorithms.

TODO: Pick a couple envs to describe in more detail, and refer to cite for full details

Additionally, the Deep Sea Treasure is used, and the TODO: extended? (whatever the updated one is) are used. TODO: Also mention that we can generate maps of varying size to perform a scalability analysis

TODO: have noticed that inproceedings etc in references.bib not works. Need to fix.

TODO: Also double declared rave. Should look where that's cited, change the combining online and offline knowledge in UCT to another name, and update the refs that are intended to point to that entry. ALSO, think combining online and offline is what using to cite mctsmode as a development?

5.5 Theory

TODO: part 1: all the (relevant) search policies will select actions infinitely often.

TODO: part 1a: BTS and DENTS follows from ch4. For UCT can argue that if select one argument a finite number of times. Then use the confidence interval term to arrive at contradiction

TODO: part 2: convex hull backups perform the scalarised backups for all possible weights over the objective.

TODO: write up proofs wrote up on ipad. Used this gpt convo in cleaning up details: https://chatgpt.com/c/67fa9295-0cb8-8007-88fd-5bec1ac5393b

TODO: part 2a: refer back to convergence of backups in ch4, clean up and DKW stuff need to

TODO: part 3: if values converge to optimal, then contextual simple regret tends to zero

TODO: part 4: mention that CZT proofs would be much more complex, and given the other algorithms do better. Mention a lower bound which comes from the CZ regret bound. And that this algorithm could be preferable in not in exploring context.

TODO: part 4a: similar with CH-UCT to CZT

Below is what we initially wrote in this section.

TODO: See if with chatgpt help can crack out a rough proof outline? Because assuming no noise in the reward function, the leaf nodes clearly converge with same rate. Just really need the induction step. Feel like it should be very do-able, by bootstrapping on the CZT proofs.

TODO: Can easily show that CH converges, by using every action sampled infinitly often still, and bootstrap off the convex hull convergence proof.

TODO: One thing need to prove is that our form of backups converges, because we're using the sample averages our backups are slightly different. Should

TODO: Remainder of properties bootstrap off the proofs in Ch4. Every action is sampled inf often, hence the convex hull backups converge. Don't bother showing convergence rates. Just show that they are consistent.

TODO: Also want to show that given any convex hull value estimates, that following the policy extracted from convex hulls does give the same multi-objective value.

basically this would say

$$\pi(s;w) = \arg\max_{a\in\mathcal{A}} \circ\mathbf{Q}(s,a;w) \tag{5.19}$$

$$w^\top \mathbf{V}^\pi(s;w) = w^\top \mathbb{E}_{a\sim\pi}[\mathbf{Q}^\pi(s,a;w)] \tag{5.20}$$

$$= \circ\mathbf{Q}(s,a;w) \tag{5.21}$$

$$w^\top \mathbf{Q}^\pi(s,a;w) = ... \tag{5.22}$$

So say that the value of the policy by taking the maimum from value estimates, will achieve the value estimates. We've been handwavy around empirical distriubtion vs actual transition distribution. But basically the argument is inductive, and holds trivially for leaf nodes.

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
Additional Algorithm Details For Chapter 4

TODO: Currently a pasting ground for currting average returns

This appendix covers any additional details for algorithms covered in Chapter 4 where

A.0.1 AR-BTS

TODO: Write this out in full

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 $\alpha_{\text{AR-BTS}}(x) > 0$:

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

$$\rho_{\text{BTS}}(a|s) \propto \exp\left(\frac{1}{\alpha_{\text{AR-BTS}}(N(s))}\left(\hat{Q}_{\text{AR-BTS}}(s, a)\right)\right). \tag{A.2}$$

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 $t = h-1, ..., 0$:

$$\hat{Q}_{\text{AR-BTS}}(s_t, a_t) \leftarrow \frac{1}{N(s_t, a_t)} \left((N(s_t, a_t) - 1)\hat{Q}_{\text{AR-BTS}}(s_t, a_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i \right) \tag{A.3}$$

$$\hat{V}_{\text{AR-BTS}}(s_t) \leftarrow \frac{1}{N(s_t)} \left((N(s_t) - 1)\hat{V}_{\text{AR-BTS}}(s_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i \right). \tag{A.4}$$

Similarly to BTS, either the Q-value estimates can be used for a recommendation policy or the most visited child node can be used:

$$\psi_{\text{AR-BTS}}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_{\text{BTS}}(s, a), \tag{A.5}$$

$$\texttt{mv}_{\text{AR-BTS}}(s) = \arg\max_{a \in \mathcal{A}} N(s, a). \tag{A.6}$$

A.0.2 AR-DENTS

TODO: Write this out in ful

A.0.3 MENTS, RENTS and TENTS with Average Returns

TODO: Its DENTS, with alpha b̄eta c̄onst, and different entropy functions.

TODO: Justify using DENTS with these settings to apprixmate AR-MENTS etc by pointing at the DNETS mimicing MENTS search policy empirically, Appendix B.2.

TODO: Below is c and p of original writing on this in the neurips paper

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 B.3). The Bellman backups were sensitive to and propogated noise from the neural network evaluations. We use the prefix ‘AR’ to denote the algorithms using average returns, such as AR-DENTS. Full details for these algorithms are given in Appendix **??**.

B
Additional Experimental Details for Chapter 4

B.1 Grid world hyper-parameter search and additional results

TODO: This is copy and pasted from neurips at the moment. It doesn't account for the changes made to the algorithms or now using BayesOpt. Need to describe the parameter space searched over for each algorithm, how many Bayes trials were used and how each bayes tirla evaluated. And also give the final values used in tables. Manually setting Qinit function, but no prior knowledge used other than this.

To select hyper-parameters for the experiments detailed in Section ??, we performed a hyper-parameter search. The search was run on the 8x12 Frozen Lake environment from Figure 4.9b, and the results in Section ?? were run on the 8x12 Frozen Lake envrionment from Figure ??. For the Sailing problem, we performed the search using an initial wind direction of North, and the result in Section ?? used an initial wind direction of South-East.

To avoid the search space from becoming too large, we set some parameters manually. A good rule of thumb for initial values is to assure that $Q^{\text{init}}_{\text{sft}}(s,a) < Q^*_{\text{sft}}(s,a)$ and $Q^{\text{init}}(s,a) < Q^*(s,a)$. Explicitly this means that an initial value of zero is *not* a good choice for the Sailing problem, as rewards are negative (i.e.

it has costs). In the Sailing environment, we actually set the initial values to -200, so that they were equal to the lowest possible return from a trial (the trial length was set to 50, and an agent can incur a cost of at most -4 per timestep). To simplify the search space, we initially set the decay function in DENTS to $\beta(m) = \alpha/\log(e+m)$ and tune it after.

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

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

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

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

The final set of hyperparameters is given in Tables B.1 and B.2, which were used in the gridworld experiments in Section **??**. Not included in the tables: TODO: citewas selected for the UCT bias in both Frozen Lake and the Sailing problem.

B.2 DENTS with a constant β

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

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

Table B.1: Final hyperparameters used for Frozen Lake in Section **??**. Not included in the table: TODO: citewas selected for the bias in UCT, TODO: citewas selected for the bias and 3000 for the UCT budget in HMCTS and $\beta_{\text{init}} = 1.0$ was selected as the initial entropy temperature for DENTS.

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

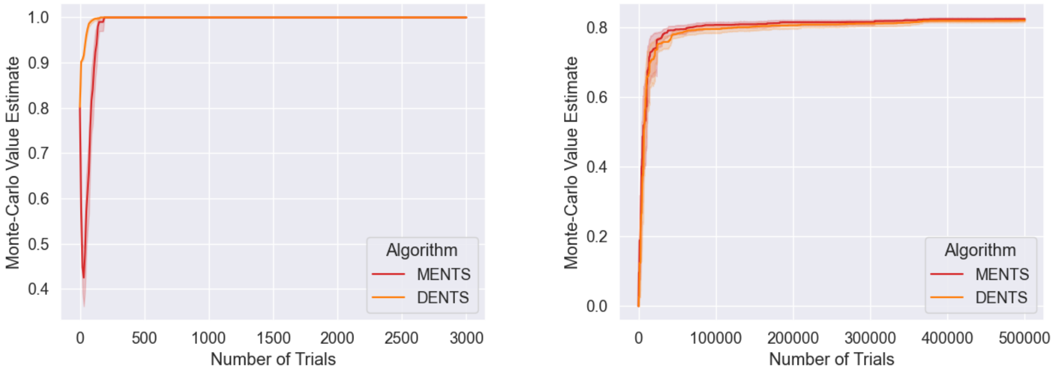
Table B.2: Final hyperparameters used for the Sailing Problem in Section **??**. Not included in the table: TODO: citewas selected for the bias in UCT, TODO: citewas selected for the bias and 30 for the UCT budget in HMCTS and $\beta_{\text{init}} = 10.0$ was selected as the initial entropy temperature for DENTS.

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 B.1. 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.

B.3 Additional Go details, results and discussion

TODO: Everything in this section this in this appendix is C and P from NeruIPS Go parameter tuning section.

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



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

Figure B.1: 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?

and -100, which means that any parameters selected are about 100 times larger than they would be if we had not have used this scaling.

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

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

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

B.3.1 Parameter selection for Go and supplimentary results

In this section we work through the process used to select parameters for the Go round robin tournament used in Section ??. Predominantly parameters were chosen

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 B.3: 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 B.4: 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.

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 B.3 and B.4 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 B.5 and B.6 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 B.7 and B.8 we give results of tuning the exploration parameter for BTS and AR-BTS. Although we selected the lowest value we tried here, we note that with such a value that a random action would have been sampled very few times, so the result of the

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 B.5: 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 B.6: Results for round robin to select the weighting of the prior policy $\epsilon_{\tilde{\lambda}}$ for AR-BTS. The value of 1.0 won the most matches so was selected.

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

Table B.7: Results for round robin to select the exploration parameter ϵ for BTS. The value of 0.003 won the most matches so was selected.

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 B.9, B.10,B.11, B.12,B.13 and B.14.

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 B.15 and B.16 for DENTS and AR-DENTS respectively.

After tuning all of the algorithms, we compared each algorithm to their AR version in table B.17, and the AR versions universally outperformed their counterparts. We used all of the selected parameters in 19x19 Go to run our final experiments given in Table 4.1.

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 B.8: 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 B.9: 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 B.10: 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 B.11: 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 B.12: Results for round robin to select the temperature parameter α for AR-RENTS. The value of 0.3 won the most matches so was selected.

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

Table B.13: 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 B.14: Results for round robin to select the temperature parameter α for AR-TENTS. The value of 3.0 won the most matches so was selected.

Black \White	0.1	0.03	0.01	0.003	0.001
0.1	-	10-5	12-3	8-7	11-4
0.03	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 B.15: 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 B.16: 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 B.17: Results for the matches of each algorithm against its AR version.

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.

C
Additional Theoretical Results For Chapter 4

TODO: Currently a pasting ground for things cutting from theory section in Ch4 to make story not sound rubbish.

TODO: Go through captions and make sure that we have short versions for all

C.1 The MCTS Stochastic Process

TODO: Words about how this is a lengthier version of the notation and preliminaries section? Make this section consistent with that section too

TODO: <beg> In main prose, but edited heavily. Keep some amount of this? Definitely keep the T and N stuff.

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

$$T(s_t, m) = \{i | i \leq m, s_t^i = s_t\} \tag{C.1}$$

$$T(s_t, a_t, m) = \{i | i \leq m, s_t^i = s_t, a_t^i = a_t\}. \tag{C.2}$$

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:

$$N(s_t, m) = \sum_{i=1}^{m} \mathbb{1}[s_t^i = s_t] = |T(s_t, m)|, \tag{C.3}$$

$$N(s_t, a_t, m) = \sum_{i=1}^{m} \mathbb{1}[s_t^i = s_t, a_t^i = a_t] = |T(s_t, a_t, m)|, \tag{C.4}$$

$$N(s_t, a_t, m) = \sum_{i \in T(s_t, m)} \mathbb{1}[a_t^i = a_t], \tag{C.5}$$

$$N(s_{t+1}, m) = \sum_{i \in T(s_t, a_t, m)} \mathbb{1}[s_{t+1}^i = s_{t+1}]. \tag{C.6}$$

Additionally, the assumption that for any two states $s, s' \in \mathcal{S}$ that $s = s'$ if and only if the trajectories leading to them are identical 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: <end> In main prose, but edited heavily. Keep some amount of this?

TODO: do we do anything at all with the UCT process? TODO: Maybe just write it out for completeness

TODO: Also add the backups for the number of visit function N into the backups in the full definitions in the appendices.

TODO: Also add the updates to the search tree. (Union of search tree with the trials.)

TODO: Handle setting the value at the end of the trial to Vinit in these processes too

C.1.1 The UCT process.

The UCT search policy can be defined as:

$$\pi^n_{\text{UCT}}(s_t) = \max_{a \in \mathcal{A}} \text{UCB}^n(s_t, a), \tag{C.7}$$

$$\text{UCB}^n(s_t, a) = \begin{cases} \infty & \text{if } N(s_t, a) = 0 \\ \bar{Q}^{N(s_t,a)}(s_t, a) + c\sqrt{\frac{\log N(s_t)}{N(s_t,a)}} & \text{if } N(s_t, a) > 0 \end{cases} \tag{C.8}$$

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

$$\bar{V}^{N(s_t)+1}(s_t) = \bar{V}^{N(s_t)}(s_t) + \frac{\bar{R}(t) - \bar{V}^{N(s_t)}(s_t)}{N(s_t)+1}, \tag{C.9}$$

$$\bar{Q}^{N(s_t,a_t)+1}(s_t,a_t) = \bar{Q}^{N(s_t,a_t)}(s_t,a_t) + \frac{\bar{R}(t) - \bar{Q}^{N(s_t,a_t)}(s_t,a_t)}{N(s_t,a_t)+1}, \tag{C.10}$$

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

C.1.2 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

$$\pi^m_{\text{MENTS}}(a|s) = (1-\lambda^m(s,\epsilon_{\text{MENTS}}))\rho^m_{\text{MENTS}}(a|s) + \frac{\lambda^m(s,\epsilon_{\text{MENTS}})}{|\mathcal{A}|}, \tag{C.11}$$

$$\rho^m_{\text{MENTS}}(a|s) = \exp\left(\frac{1}{\alpha_{\text{MENTS}}}\left(\hat{Q}^{(N(s,a,m-1))}_{\text{MENTS}}(s,a) - \hat{V}^{(N(s,m-1))}_{\text{MENTS}}(s)\right)\right). \tag{C.12}$$

$$\lambda^m(s,x) = \min\left(1, \frac{x}{\log(e+N(s,m-1))}\right), \tag{C.13}$$

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

$$\hat{V}^{(N(s_t,m))}_{\text{MENTS}}(s_t) = \alpha_{\text{MENTS}} \log \sum_{a\in\mathcal{A}} \exp\left(\frac{1}{\alpha_{\text{MENTS}}}\hat{Q}^{(N(s_t,a,m))}_{\text{MENTS}}(s_t,a)\right), \tag{C.14}$$

$$\hat{Q}^{(N(s_t,a_t,m))}_{\text{MENTS}}(s_t,a_t) = R(s_t,a_t) + \sum_{s'\in\text{Succ}(s,a)} \left(\frac{N(s',m)}{N(s_t,a_t,m)}\hat{V}^{(N(s',m))}_{\text{MENTS}}(s')\right). \tag{C.15}$$

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?

$$\psi^m_{\text{MENTS}}(s) = \arg\max_{a\in\mathcal{A}} \hat{Q}^{(N(s,a,m))}_{\text{MENTS}}(s,a), \tag{C.16}$$

$$\texttt{mv}^m_{\text{MENTS}}(s) = \arg\max_{a\in\mathcal{A}} N(s,a,m). \tag{C.17}$$

C.1.3 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.

$$\pi^m_{\text{DNTS}}(a|s) = (1-\lambda^m(s,\epsilon_{\text{DNTS}}))\rho^m_{\text{DNTS}}(a|s) + \frac{\lambda^m(s,\epsilon_{\text{DNTS}})}{|\mathcal{A}|}, \tag{C.18}$$

$$\rho^m_{\text{DNTS}}(a|s) \propto \exp\left(\frac{1}{\alpha_{\text{DNTS}}(N(s,m))}\left(\hat{Q}^{(N(s,a,m-1))}_{\text{DNTS}}(s,a) + \beta_{\text{DNTS}}(N(s,m))\bar{\mathcal{H}}^{(N(s,a,m-1))}_{Q,\text{DNTS}}(s,a)\right)\right). \tag{C.19}$$

$$\rho^m_{\text{DNTS}}(a|s) = \frac{\exp\left(\frac{1}{\alpha_{\text{DNTS}}(N(s,m))}\left(\hat{Q}^{(N(s,a,m-1))}_{\text{DNTS}}(s,a) + \beta_{\text{DNTS}}(N(s,m))\bar{\mathcal{H}}^{(N(s,a,m-1))}_{Q,\text{DNTS}}(s,a)\right)\right)}{\sum_{a'\in\mathcal{A}}\exp\left(\frac{1}{\alpha_{\text{DNTS}}(N(s,m))}\left(\hat{Q}^{(N(s,a',m-1))}_{\text{DNTS}}(s,a') + \beta_{\text{DNTS}}(N(s,m))\bar{\mathcal{H}}^{(N(s,a',m-1))}_{Q,\text{DNTS}}(s,a')\right)\right)}. \tag{C.20}$$

$$\lambda^m(s,x) = \min\left(1, \frac{x}{\log(e+N(s,m-1))}\right), \tag{C.21}$$

The mth trajectory is sampled using the search policy: $\tau^m \sim {\pi_{\text{DNTS}}}^m$. 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

$$\hat{Q}^{(N(s_t,a_t,m))}_{\text{DNTS}}(s_t,a_t) = R(s_t,a_t) + \sum_{s'\in\text{Succ}(s_t,a_t)} \left(\frac{N(s',m)}{N(s_t,a_t,m)} \hat{V}^{(N(s',m))}_{\text{DNTS}}(s') \right), \tag{C.22}$$

$$\hat{V}^{(N(s_t,m))}_{\text{DNTS}}(s_t) = \max_{a\in\mathcal{A}} \hat{Q}^{(N(s_t,a,m))}_{\text{DNTS}}(s_t,a), \tag{C.23}$$

$$\bar{\mathcal{H}}^{(N(s_t,a_t,m))}_{Q,\text{DNTS}}(s_t,a_t) = \sum_{s'\in\text{Succ}(s_t,a_t)} \frac{N(s',m)}{N(s_t,a_t,m)} \bar{\mathcal{H}}^{(N(s',m))}_{V,\text{DNTS}}(s'), \tag{C.24}$$

$$\bar{\mathcal{H}}^{(N(s_t))}_{V,\text{DNTS}}(s_t) = \mathcal{H}(\pi^m_{\text{DNTS}}(\cdot|s_t)) + \sum_{a\in\mathcal{A}} \pi^m_{\text{DNTS}}(a_t|s_t) \bar{\mathcal{H}}^{(N(s_t,a_t,m))}_{Q,\text{DNTS}}(s_t,a_t). \tag{C.25}$$

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

TODO: recommendation policies word stuff

$$\psi^m_{\text{DNTS}}(s) = \arg\max_{a\in\mathcal{A}} \hat{Q}^{(N(s,a,m))}_{\text{DNTS}}(s,a), \tag{C.26}$$

$$\texttt{mv}^m_{\text{DNTS}}(s) = \arg\max_{a\in\mathcal{A}} N(s,a,m). \tag{C.27}$$

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

$$\begin{aligned}\hat{V}^{(N(s))}_{\rho,\text{DNTS}}(s) =& \alpha_{\text{DNTS}}(N(s,m)) \log \left[\sum_{a'\in\mathcal{A}} \exp \left(\frac{1}{\alpha_{\text{DNTS}}(N(s,m))} \Big(\hat{Q}^{(N(s,a'))}_{\text{DNTS}}(s,a') \right.\right. \\ &\left.\left. + \beta_{\text{DNTS}}(N(s,m)) \bar{\mathcal{H}}^{(N(s,a'))}_{Q,\text{DNTS}}(s,a') \Big) \right) \right],\end{aligned} \tag{C.28}$$

and notice that the value of $\exp(\hat{V}^{(N(s,m))}_{\rho,\text{DNTS}}(s)/\alpha_{\text{DNTS}}(N(s,m)))$ is equal to the denominator in Equation (C.20), and so by rearranging we can write ρ^m_{DNTS} as

$$\rho^m_{\text{DNTS}}(a|s) = \exp \left(\frac{1}{\alpha_{\text{DNTS}}(N(s,m))} \Big(\hat{Q}^{(N(s,a,m))}_{\text{DNTS}}(s,a) + \beta_{\text{DNTS}}(N(s,m)) \bar{\mathcal{H}}^{(N(s,a,m))}_{Q,\text{DNTS}}(s,a) - \hat{V}^{(N(s,m))}_{\rho,\text{DNTS}}(s) \Big) \right) \tag{C.29}$$

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

$$\pi^m_{\text{DNTS}}(a|s) = (1-\lambda(s,m)) \exp \left(\frac{1}{\alpha_{\text{DNTS}}(N(s,m))} \Big(\hat{Q}^{(N(s,a,m))}_{\text{DNTS}}(s,a) + \beta_{\text{DNTS}}(N(s,m)) \bar{\mathcal{H}}^{(N(s,a))}_{Q,\text{DNTS}}(s,a) - \hat{V} \right. \tag{C.30}$$

C.1.4 The AR-DENTS Process

TODO: Copy DENTS after its written. Swap out notation and backups.

C.1.5 The (AR-)BTS process.

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

C.2 Exponential Convergence Results

TODO: This section give proofs for exponential convergence results for DENTS and MENTS, it is structured as follows:

1. First, Appendix C.2.1 provides a set of Lemmas that are used later in the proofs for exponential convergence bounds;
2. Second, Appendix C.2.2 gives Lemmas that provide sufficient conditions for the maximum entropy objective and standard objective to be aligned;
3. Third, in Appendix C.2.3 it is shown in a general way that is Q-values are computed as a sample average of child value functions, and those child value functions admit a concentration inequality, then the Q-value also admits a concentration inequality;
4. Fourth, in Appendix C.2.4, exponential convergence bounds (requiring restrictive conditions on the temperature parameter) are shown for MENTS;
5. Finally, in Appendix C.2.5, Theorem 4.2.3 and 4.2.7 are

TODO: Make sure that

TODO: Make sure assumptions and conditions stated. Anything that depends on the min prob needs to have the lower/upper bounds on the decay functions for example.

C.2.1 Preliminaries

This subsection contains preliminary lemmas that are used for the building blocks to prove.

TODO: <beg> dump

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 C.2.1. *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:*

$$\Pr\left(\bigcup_{i=1}^{\ell} \neg A_i\right) \leq C \exp(-k), \tag{C.31}$$

$$\Pr\left(\bigcap_{i=1}^{\ell} A_i\right) = 1 - \Pr\left(\bigcup_{i=1}^{\ell} \neg A_i\right) \geq 1 - C \exp(-k), \tag{C.32}$$

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

Proof outline. Lemma C.2.1 is a consequence of the union bound, using $\exp(k_i) \leq \exp(k)$ and simplifying. Inequality (C.32) is just a negation of Inequality (C.31). □

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 C.2.2. *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:

$$Q^{\min}(s_t, a_t) = \min_{s_{t+1},a_{t+1},\ldots,s_H,a_H} \sum_{i=t}^{H} \min(0, Q^{\text{init}}_{\text{sft}}(s_i), R(s_i, a_i)). \tag{C.33}$$

TODO: clean up with respect to init function

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

$$V^{\max}(s_t) = \alpha_{\text{MENTS}} \log \sum_{a\in\mathcal{A}} \exp(Q^{\max}(s_t, a)/\alpha_{\text{MENTS}}), \tag{C.34}$$

$$Q^{\max}(s_t, a_t) = R(s_t, a_t) + \max_{s_{t+1}\in\text{Succ}(s_t,a_t)} V^{\max}(s_{t+1}). \tag{C.35}$$

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:

$$\pi^{\min} = \inf_{\lambda\in[0,1]} \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} (1-\lambda) \exp\left(\left(Q^{\min}(s, a) - V^{\max}(s)\right)/\alpha\right) + \frac{\lambda}{|\mathcal{A}|}. \tag{C.36}$$

Because the value of an exponential is positive, as is $1/|\mathcal{A}|$, it follows that $\pi^{\min} > 0$. Recall the MENTS policy (Equation (C.11)). By the monotonicity of the exponential function, it follows that for any $s_t \in \mathcal{S}, a_t \in \mathcal{A}, n \in \mathbb{N}$:

$$\pi^m_{\text{MENTS}}(a_t|s_t) = (1-\lambda(s_t,m)) \exp\left(\left(\frac{1}{\alpha_{\text{MENTS}}}\left(\hat{Q}^{(N(s_t,a_t,m))}_{\text{MENTS}}(s_t,a_t) - \hat{V}^{(N(s_t,m))}_{\text{MENTS}}(s_t)\right)\right)\right) + \frac{\lambda(s_t,m)}{|\mathcal{A}|} \tag{C.37}$$

$$\geq (1-\lambda(s_t,m)) \exp\left(\left(Q^{\min}(s_t,a_t) - V^{\max}(s_t)\right)/\alpha\right) + \frac{\lambda(s_t,m)}{|\mathcal{A}|} \tag{C.38}$$

$$\geq \pi^{\min}. \tag{C.39}$$

□

TODO: this whole lemma just needs cleaning up really

TODO: this lemma requires $\beta_{\text{DNTS}}(x) < U$ for all x, or β_{DNTS} is bounded. AND it requires $\alpha_{\text{DNTS}}(x) < U'$ so α_{DNTS} 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 C.2.3. *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* $\pi^m_{\text{DNTS}}(a_t|s_t) \geq \pi^{\min}$.

Proof outline. This proof outline follows similar reasoning to Lemma C.2.2. The same definition for $Q^{\min}$ can be used:

$$Q^{\min}(s_t, a_t) = \min_{s_{t+1},a_{t+1},\dots,s_H,a_H} \sum_{i=t}^{H} \min(0, Q^{\text{init}}_{\text{sft}}(s_i), R(s_i, a_i)). \tag{C.40}$$

TODO: clean up with respect to init function

However, the definition of $V^{\max}$ from Equation (C.34) needs to altered to:

$$V^{\max}(s_t) = \max_{a\in\mathcal{A}} Q^{\max}(s_t, a) \tag{C.41}$$

$$Q^{\max}(s_t, a_t) = R(s_t, a_t) + \max_{s'\in\text{Succ}(s_t,a_t)} V^{\max}(s') \tag{C.42}$$

$$V^{\max}_\rho(s_t) = \alpha \log \sum_{a\in\mathcal{A}} \exp\left(\left(Q^{\max}(s_t, a) + \beta_{\max} H \log|\mathcal{A}|\right)/\alpha\right), \tag{C.43}$$

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 $Q^{\min}(s_t, a_t) \leq \hat{Q}^{(N(s_t,a_t,m))}_{\text{DNTS}}(s_t, a_t)$ and $V^{\max}_\rho(s_t) \geq V^{N(s_t,m)}_\rho(s_t)$, where the latter implicitly uses that $0 \leq \mathcal{H}^{N(s_t,a_t)}_Q(s_t, a_t) \leq (H - t)\log|\mathcal{A}| \leq H \log|\mathcal{A}|$ 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 (C.36), with the updated definition of $V_\rho^{\max}$:

$$\pi^{\min} = \inf_{\lambda\in[0,1]} \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} (1-\lambda)\exp\left(\left(Q^{\min}(s,a) - V_\rho^{\max}(s)\right)/\alpha_{\max}\right) + \frac{\lambda}{|\mathcal{A}|}. \tag{C.44}$$

where $\alpha_{\max} = \inf_x \alpha_{\mathrm{DNTS}}(x)$ 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 (C.30)), and using similar reasoning to before, as well as $\beta_{\mathrm{DNTS}}(N(s,m))\bar{\mathcal{H}}_{Q,\mathrm{DNTS}}^{(N(s,a))}(s,a) \geq 0$, the result follows:

$$\pi^m_{\mathrm{DNTS}}(a_t|s_t) = (1-\lambda(s,m))\exp\Bigg(\frac{1}{\alpha_{\mathrm{DNTS}}(N(s,m))}\Big(\hat{Q}_{\mathrm{DNTS}}^{(N(s,a,m))}(s,a) + \beta_{\mathrm{DNTS}}(N(s,m))\bar{\mathcal{H}}_{Q,\mathrm{DNTS}}^{(N(s,a))}(s, \tag{C.45}$$

$$\geq (1-\lambda(s,m))\exp\left(\frac{1}{\alpha_{\mathrm{DNTS}}(N(s,m))}\left(\hat{Q}_{\mathrm{DNTS}}^{(N(s,a))}(s,a) - \hat{V}_{\rho,\mathrm{DNTS}}^{(N(s))}(s)\right)\right) + \frac{\lambda(s,m)}{|\mathcal{A}|} \tag{C.46}$$

$$\geq (1-\lambda(s,m))\exp\left(\left(Q^{\min}(s_t,a_t) - V_\rho^{\max}(s_t)\right)/\alpha_{\max}\right) + \frac{\lambda(s,m)}{|\mathcal{A}|} \tag{C.47}$$

$$\geq \pi^{\min}. \tag{C.48}$$

□

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

Theorem C.2.4. *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:*

$$\Pr(|S_k - \mathbb{E}S_k| > \varepsilon) \leq 2\exp\left(-\frac{2\varepsilon^2}{k}\right). \tag{C.49}$$

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 C.2.5. *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:*

$$\Pr(E) \leq C' \exp(-k' N(s_t, a_t, m)). \tag{C.50}$$

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

$$\Pr(E) \leq C \exp(-k N(s_t, m)). \tag{C.51}$$

Proof. TODO: Define m again somewhere? Recall from Equation (C.5) 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:

$$\begin{aligned}
&\Pr\left(N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) && \text{(C.52)}\\
\leq &\Pr\left(N(s_t, a_t, m) < \mathbb{E}N(s_t, a_t, m) - \frac{1}{2}\eta N(s_t, m)\right) && \text{(C.53)}\\
= &\Pr\left(\mathbb{E}N(s_t, a_t, m) - N(s_t, a_t, m) > \frac{1}{2}\eta N(s_t, m)\right) && \text{(C.54)}\\
\leq &\Pr\left(|\mathbb{E}N(s_t, a_t, m) - N(s_t, a_t, m)| > \frac{1}{2}\eta N(s_t, m)\right) && \text{(C.55)}\\
\leq &2\exp\left(-\frac{1}{2}\eta^2 N(s_t, m)\right). && \text{(C.56)}
\end{aligned}$$

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 (C.54) implying the inequality in (C.55), and the final line uses Theorem C.2.4, 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:

$$\begin{aligned}\Pr(E) =& \Pr\left(E \middle| N(s_t, a_t, m) \geq \frac{1}{2}\eta N(s_t, m)\right) \Pr\left(N(s_t, a_t, m) \geq \frac{1}{2}\eta N(s_t, m)\right) \\ &+ \Pr\left(E \middle| N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) \Pr\left(N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) \end{aligned} \tag{C.57}$$

$$\begin{aligned}\leq& \Pr\left(E \middle| N(s_t, a_t, m) \geq \frac{1}{2}\eta N(s_t, m)\right) \cdot 1 \\ &+ 1 \cdot \Pr\left(N(s_t, a_t, m) < \frac{1}{2}\eta N(s_t, m)\right) \end{aligned} \tag{C.58}$$

$$\leq C' \exp\left(-\frac{1}{2}k'\eta N(s_t, m)\right) + 2\exp\left(-\frac{1}{2}\eta^2 N(s_t, m)\right) \tag{C.59}$$

$$\leq C \exp(-kN(s_t, m)), \tag{C.60}$$

where (C.59) uses the assumed Inequality (C.50) with the condition $N(s_t, a_t) \geq \eta N(s_t)/2$, and also uses the bound from (C.56). The final inequality (C.60) follows from Lemma C.2.1, 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 C.2.5, 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 C.2.6. *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:*

$$\Pr(E) \leq C' \exp(-k' N(s'_t)). \tag{C.61}$$

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

$$\Pr(E) \leq C \exp(-kN(s_t, a_t)). \tag{C.62}$$

Proof outline. Proof is similar to Lemma C.2.5. 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 (C.6)) as the sum of indicator random variables will give the result. □

TODO: <end> dump

C.2.2 Maximum Entropy Reinforcement Learning Results

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 rearranged, to give a relation between the soft Q-value, the standard Q-value and the entropy of the policy:

$$Q^{\pi}_{\text{sft}}(s,a;t) = Q^{\pi}(s,a;t) + \alpha\mathbb{E}_{\pi}\left[\sum_{i=t+1}^{H} \mathcal{H}\left(\pi(\cdot|s_i)\right)\middle| s_t = s, a_t = a\right], \tag{C.63}$$

$$= Q^{\pi}(s,a;t) + \alpha\mathcal{H}_{t+1}(\pi|s_t = s, a_t = a), \tag{C.64}$$

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 C.2.7. $Q^*(s,a;t) \leq Q^*_{\text{sft}}(s,a;t)$.

Proof. Taking a maximum over policies in Equation (C.64), and considering that π^*, the optimal standard policy, is one of the possible policies considered in the maximisation, gives the result:

$$\begin{aligned} Q^*_{\text{sft}}(s,a;t) &= \max_{\pi} \left(Q^{\pi}(s,a;t) + \alpha \mathcal{H}_{t+1}(\pi | s_t = s, a_t = a) \right) && \text{(C.65)} \\ &\geq Q^{\pi^*}(s,a;t) + \alpha \mathcal{H}_{t+1}(\pi^* | s_t = s, a_t = a) && \text{(C.66)} \\ &\geq Q^*(s,a;t). && \text{(C.67)} \end{aligned}$$

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:

$$\begin{aligned} \Delta_{s,t} &= \min_{(a,a') \in \delta(s,t)} |Q^*(s,a;t) - Q^*(s,a';t)|, && \text{(C.68)} \\ \Delta_{\mathcal{M}} &= \min_{s,t} \Delta_{s,t}. && \text{(C.69)} \end{aligned}$$

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

$$Q^*(s,a;t) + \Delta_{\mathcal{M}} \leq Q^*(s,a';t). \tag{C.70}$$

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 C.2.8. *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. (⇐) First consider that the optimal soft Q-value is less than or equal to the optimal standard Q-value and maximum possible entropy:

$$\begin{aligned} Q^*_{\text{sft}}(s,a;t) &= \max_{\pi} \left(Q^{\pi}(s,a;t) + \alpha\mathcal{H}_{t+1}(\pi)\right) && \text{(C.71)} \\ &\leq \max_{\pi} Q^{\pi}(s,a;t) + \max_{\pi} \alpha\mathcal{H}_{t+1}(\pi) && \text{(C.72)} \\ &= Q^*(s,a;t) + \alpha(H-t)\log|\mathcal{A}| && \text{(C.73)} \\ &\leq Q^*(s,a;t) + \alpha H \log|\mathcal{A}|. && \text{(C.74)} \end{aligned}$$

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 C.2.7 gives the desired inequality:

$$\begin{aligned} Q^*_{\text{sft}}(s,a;t) &\leq Q^*(s,a;t) + \alpha H \log|\mathcal{A}| && \text{(C.75)} \\ &< Q^*(s,a;t) + \Delta_{\mathcal{M}} && \text{(C.76)} \\ &\leq Q^*(s,a';t) && \text{(C.77)} \\ &\leq Q^*_{\text{sft}}(s,a';t). && \text{(C.78)} \end{aligned}$$

(⇒) 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:

$$\begin{aligned} Q^*(s,a;t) &\geq Q^*(s,a';t) && \text{(C.79)} \\ \Rightarrow Q^*(s,a;t) &> Q^*(s,a';t) && \text{(C.80)} \\ \Rightarrow Q^*_{\text{sft}}(s,a;t) &> Q^*_{\text{sft}}(s,a';t) && \text{(C.81)} \\ \Rightarrow Q^*_{\text{sft}}(s,a;t) &\geq Q^*_{\text{sft}}(s,a';t), && \text{(C.82)} \end{aligned}$$

where the first implication uses that $(a,a') \in \delta(s)$, the second reuses the (⇐) 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. □

C.2.3 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 (C.15) and (C.22)). 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.

TODO: <beg> dump (stuff put in main text)

Theorem C.2.9. *Let $\{X_i\}_{i=1}^{m}$ be random variables drawn from a probability distribution with a cumulative distribution function of F. Let the empirical cumulative distribution function be $F_m(x) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}[X_i < x]$. Then the Dvoretzky-Kiefer-Wolfowitz inequality is:*

$$\Pr\left(\sup_x |F_m(x) - F(x)| > \varepsilon\right) \leq 2\exp\left(-2m\varepsilon^2\right). \tag{C.83}$$

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 C.2.9.1. *Consider any Boltzmann MCTS process. For all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ and for all $\varepsilon > 0$ we have:*

$$\Pr\left(\max_{s_{t+1}\in\text{Succ}(s_t,a_t)} \left|\frac{N(s_{t+1}, n)}{N(s_t, a_t, n)} - p(s_{t+1}|s_t, a_t)\right| > \varepsilon\right) \leq 2\exp\left(-\frac{1}{2}\varepsilon^2 N(s_t, a_t)\right). \tag{C.84}$$

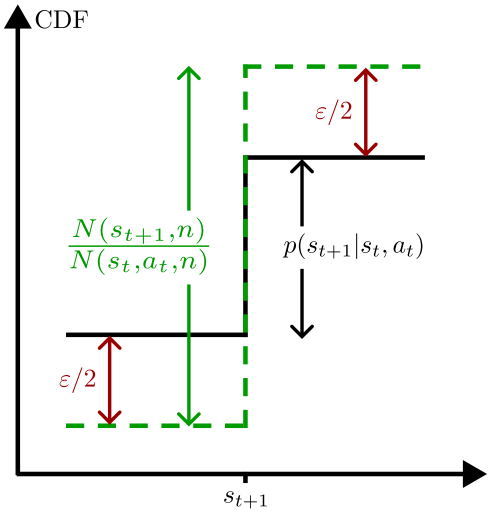


Figure C.1: Bounding the empirical transition probabilities to the true transition probabilities. The true cdf is shown as a solid black line, the empirical cdf is shown as a dashed green line, and a worst case error of $\varepsilon/2$, using Theorem C.2.9, is shown in red. The probability mass of $p(s_{t+1}|s_t,a_t)$ 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 C.2.9.1.

Proof outline. By considering some arbitrary ordering over the successor states in $\text{Succ}(s_t,a_t)$ and applying Theorem C.2.9, replacing ε by $\varepsilon/2$, the result follows.

To see why the factor of 1/2 is needed, consider Figure C.1. Because the distribution is discrete, the cumulative distribution function is a (piecewise constant) step function. As Theorem C.2.9 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. □

TODO: <end> dump (stuff put in main text)

Now that Corollary C.2.9.1 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 C.2.10. *Consider any Boltzmann MCTS process, and some state action pair* $(s_t,a_t)\in\mathcal{S}\times\mathcal{A}$*. Let* $\dot{V}^{N(s,n)}(s):\mathcal{S}\to\mathbb{R}$, $\dot{V}^*(s):\mathcal{S}\to\mathbb{R}$ *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* $\varepsilon_0>0$*:*

$$\Pr\left(\left|\dot{V}^{N(s_{t+1})}(s_{t+1})-\dot{V}^*(s_{t+1})\right|>\varepsilon_0\right)\leq C_{s_{t+1}}\exp(-k_{s_{t+1}}\varepsilon_0^2N(s')). \tag{C.85}$$

If the optimal and estimated Q-values are defined as follows:

$$\dot{Q}^*(s,a) = R(s,a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\dot{V}^*(s')\right], \tag{C.86}$$

$$\dot{Q}^{N(s,a,n)}(s,a) = R(s,a) + \sum_{s' \in \text{Succ}(s,a)} \left[\frac{N(s',n)}{N(s,a,n)} \dot{V}^{N(s',n)}(s')\right]. \tag{C.87}$$

Then there exists some $C, k > 0$, for all $\varepsilon > 0$ such that:

$$\Pr\left(\left|\dot{Q}^{N(s,a,n)}(s,a) - \dot{Q}^*(s,a)\right| > \varepsilon\right) \leq C \exp(-k\varepsilon^2 N(s,a,n)). \tag{C.88}$$

Proof. By the assumed bounds, Lemma C.2.1 and Lemma C.2.6, there is some $C_1, k_1 > 0$, such that for any $\varepsilon_1 > 0$:

$$\Pr\left(\forall s_{t+1}. \left|\dot{V}^{N(s_{t+1})}(s_{t+1}) - \dot{V}^*(s_{t+1})\right| \leq \varepsilon_1\right) > 1 - C_1 \exp(-k_1\varepsilon_1^2 N(s_t,a_t,n)). \tag{C.89}$$

And recall that for any $p(s_{t+1}|s_t,a_t) > \varepsilon_2 > 0$, using Corollary C.2.9.1 that:

$$\Pr\left(\max_{s_{t+1} \in \text{Succ}(s_t,a_t)} \left|\frac{N(s_{t+1},n)}{N(s_t,a_t,n)} - p(s_{t+1}|s_t,a_t)\right| \leq \varepsilon_2\right) > 1 - 2\exp\left(-\frac{1}{2}\varepsilon_2^2 N(s_t,a_t,n)\right). \tag{C.90}$$

If the events in Inequalities (C.89) and (C.90) hold, then the following inequalities must also hold:

$$\dot{V}^*(s_{t+1}) - \varepsilon_1 \leq \dot{V}^{N_n(s_{t+1},n)}(s_{t+1}) \leq \dot{V}^*(s_{t+1}) + \varepsilon_1 \tag{C.91}$$

$$p(s_{t+1}|s_t,a_t) - \varepsilon_2 \leq \frac{N(s_{t+1},n)}{N(s_t,a_t,n)} \leq p(s_{t+1}|s_t,a_t) + \varepsilon_2. \tag{C.92}$$

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 $\dot{Q}^{N(s_t,a_t,n)}(s_t,a_t)$:

$$\dot{Q}^{N(s_t,a_t,n)}(s_t,a_t) \tag{C.93}$$

$$= R(s_t,a_t) + \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \frac{N(s_{t+1},n)}{N(s_t,a_t,n)} \dot{V}^{N(s_{t+1},n)}(s_{t+1}) \tag{C.94}$$

$$\leq R(s_t,a_t) + \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} (p(s_{t+1}|s_t,a_t) + \varepsilon_2)(\dot{V}^*(s_{t+1}) + \varepsilon_1) \tag{C.95}$$

$$= R(s_t,a_t) + \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t,a_t)}[\dot{V}^*(s_{t+1})] + \varepsilon_1 + \varepsilon_2 \sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1}) + \varepsilon_1\varepsilon_2 \tag{C.96}$$

$$\leq R(s_t,a_t) + \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t,a_t)}[\dot{V}^*(s_{t+1})] + \varepsilon_2 \left|\sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1})\right| + \varepsilon_1 + \varepsilon_1\varepsilon_2 \tag{C.97}$$

$$= \dot{Q}^*(s_t,a_t) + \varepsilon_2 \left|\sum_{s_{t+1} \in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1})\right| + \varepsilon_1 + \varepsilon_1\varepsilon_2. \tag{C.98}$$

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.

$$\dot{Q}^{N(s_t,a_t,n)}(s_t,a_t) \geq \dot{Q}^*(s_t,a_t) - \varepsilon_2 \left| \sum_{s_{t+1}\in \text{Succ}(s_t,a_t)} \dot{V}^*(s_{t+1}) \right| - \varepsilon_1 - \varepsilon_1\varepsilon_2. \tag{C.99}$$

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:

$$\dot{Q}^*(s_t,a_t) - \varepsilon \leq \dot{Q}^{N(s_t,a_t,n)}(s_t,a_t) \leq \dot{Q}^*(s_t,a_t) + \varepsilon. \tag{C.100}$$

Using Lemma C.2.1 liberally, there is some $C_2, C_3, k_2, k_3 > 0$ such that:

$$\begin{aligned}
&\Pr\left(\left|\dot{Q}^{N(s_t,a_t,n)}(s_t,a_t) - \dot{Q}^*(s_t,a_t)\right| \leq \varepsilon\right) && \text{(C.101)}\\
>&\left(1 - C_1\exp(-k_1\varepsilon_1^2 N(s_t,a_t,n))\right)\left(1 - 2\exp\left(-\frac{1}{2}\varepsilon_2^2 N(s_t,a_t,n)\right)\right) && \text{(C.102)}\\
=&\left(1 - C_2\exp(-k_2\varepsilon^2 N(s_t,a_t,n)\right)\cdot\left(1 - C_3\exp(-k_3\varepsilon^2 N(s_t,a_t,n)\right) && \text{(C.103)}\\
>&1 - C_2\exp(-k_2\varepsilon^2 N(s_t,a_t,n)) - C_3\exp(-k_3\varepsilon^2 N(s_t,a_t,n)). && \text{(C.104)}
\end{aligned}$$

Finally, by negating and setting $C = C_2 + C_3$ and $k = \min(k_2, k_3)$ in Lemma TODO: ref the result follows:

$$\Pr\left(\left|\dot{Q}^{N(s_t,a_t,n)}(s_t,a_t) - \dot{Q}^*(s_t,a_t)\right| > \varepsilon\right) \leq C\exp(-k\varepsilon^2 N(s_t,a_t,n)). \tag{C.105}$$

□

C.2.4 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 [46] 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 C.2.11. *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 $\varepsilon_{s_{t+1}} > 0$:*

$$\Pr\left(\left|\hat{V}^{(N(s_{t+1},n))}_{\text{MENTS}}(s_{t+1}) - V^*_{\text{sft}}(s_{t+1})\right| > \varepsilon_{s_{t+1}}\right) \leq C_{s_{t+1}} \exp\left(-k_{s_{t+1}}\varepsilon^2_{s_{t+1}} N(s_{t+1}, n)\right), \tag{C.106}$$

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

$$\Pr\left(\left|\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) - V^*_{\text{sft}}(s_t)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 N(s_t, n)\right). \tag{C.107}$$

Proof. Given the assumptions and by Lemmas C.2.10, C.2.5 and C.2.1, there is some $C, k > 0$ such that for any $\varepsilon > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t) - Q^*_{\text{sft}}(s_t, a_t)\right| \leq \varepsilon\right) > 1 - C\exp(-k\varepsilon^2 N(s_t, n)). \tag{C.108}$$

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

$$Q^*_{\text{sft}}(s_t, a_t) - \varepsilon \leq \hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t) \leq Q^*_{\text{sft}}(s_t, a_t) + \varepsilon. \tag{C.109}$$

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 (C.14) gives:

$$\begin{align} \hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) &= \alpha \log \sum_{a\in\mathcal{A}} \exp\left(\frac{\hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t)}{\alpha}\right) \tag{C.110}\\ &\leq \alpha \log \sum_{a\in\mathcal{A}} \exp\left(\frac{Q^*_{\text{sft}}(s_t, a_t) + \varepsilon}{\alpha}\right) \tag{C.111}\\ &= \left[\alpha \log \sum_{a\in\mathcal{A}} \exp\left(\frac{Q^*_{\text{sft}}(s_t, a_t)}{\alpha}\right)\right] + \varepsilon \tag{C.112}\\ &= V^*_{\text{sft}}(s_t) + \varepsilon, \tag{C.113} \end{align}$$

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:

$$\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) \geq V^*_{\text{sft}}(s_t) - \varepsilon, \tag{C.114}$$

and hence:

$$|\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) - V^*_{\text{sft}}(s_t)| \leq \varepsilon. \tag{C.115}$$

This therefore shows: TODO: some comment about how if A implies B then pr(B) more than pr(A)

$$\Pr\left(\left|\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) - V^*_{\text{sft}}(s_t)\right| \leq \varepsilon_1\right) > 1 - C\exp(-k\varepsilon^2 N(s_t,n)), \tag{C.116}$$

and negating probabilities gives the result. □

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

Theorem C.2.12. *Consider a MENTS process, let $s_t \in \mathcal{S}$ then there is some $C,k>0$ for any $\varepsilon>0$:*

$$\Pr\left(\left|\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) - V^*_{\text{sft}}(s_t)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 N(s_t,n)\right). \tag{C.117}$$

Moreover, at the root node s_0:

$$\Pr\left(\left|\hat{V}^{(N(s_0,n))}_{\text{MENTS}}(s_0) - V^*_{\text{sft}}(s_0)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 n\right). \tag{C.118}$$

And hence $\hat{V}^{(N(s_t,n))}_{\text{MENTS}}(s_t) \xrightarrow{p} V^*_{\text{sft}}(s_t)$.

Proof. Consider that the result holds for $t=H+1$, because $\hat{V}^{(N(s_{H+1},n))}_{\text{MENTS}}(s_{H+1}) = V^*_{\text{sft}}(s_{H+1}) = 0$. Therefore the result holds for any $t=0,...,H+1$ by induction using Lemma C.2.11. Noting that $N(s_0)=n$ gives (C.118). 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 C.2.13. *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:*

$$\mathbb{E}\texttt{sim\_regr}(s_t, \psi^n_{\text{MENTS}}) \leq C' \exp\left(-k' N(s_t, n)\right). \tag{C.119}$$

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 C.2.8 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 C.2.12 and Lemmas C.2.10 and C.2.5 there exists $C_1, k_1 > 0$ such that for all $\varepsilon_1 > 0$: TODO: reword this paragraph, its a bit meh

$$\Pr\left(\forall a_t \in \mathcal{A} \left| \hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t) - Q^*_{\text{sft}}(s_t, a_t) \right| \leq \varepsilon_1 \right) > 1 - C_1 \exp(-k_1 \varepsilon_1^2 N(s_t, n)). \tag{C.120}$$

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 $a_t \in \mathcal{A}$:

$$Q^*_{\text{sft}}(s_t, a_t) - \Delta_{\mathcal{M}}/3 \leq \hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t, a_t) \leq Q^*_{\text{sft}}(s_t, a_t) + \Delta_{\mathcal{M}}/3. \tag{C.121}$$

And hence, with probability at least $1-C_1\exp(-k_2N(s_t,n))$, for all $a\in\mathcal{A}-\{a^*\}$ we have:

$$\begin{aligned}\hat{Q}^{(N(s_t,a,n))}_{\text{MENTS}}(s_t,a) \leq& Q^*_{\text{sft}}(s_t,a)+\Delta_{\mathcal{M}}/3 && \text{(C.122)}\\ \leq& Q^*(s_t,a)+\alpha H\log|\mathcal{A}|+\Delta_{\mathcal{M}}/3 && \text{(C.123)}\\ \leq& Q^*(s_t,a)+2\Delta_{\mathcal{M}}/3 && \text{(C.124)}\\ \leq& Q^*(s_t,a^*)-\Delta_{\mathcal{M}}/3 && \text{(C.125)}\\ \leq& Q^*_{\text{sft}}(s_t,a^*)-\Delta_{\mathcal{M}}/3 && \text{(C.126)}\\ \leq& \hat{Q}^{(N(s_t,a^*,n))}_{\text{MENTS}}(s_t,a^*). && \text{(C.127)}\end{aligned}$$

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 (C.72) in Lemma C.2.8); The third holds from the assumption on α; The fourth holds from the definition of $\Delta_{\mathcal{M}}$ (also see Inequality (C.70)); The fifth holds from the optimal soft value being greater than the optimal standard value (Lemma C.2.7); And the final line holds by using the lower bound on $\hat{Q}^{(N(s_t,a_t,n))}_{\text{MENTS}}(s_t,a_t)$ given above with $a_t=a^*$.

Negating the probability that (C.127) holds gives:

$$\Pr\left(\exists a_t\in\mathcal{A}-\{a^*\}.\left(\hat{Q}^{(N(s_t,a))}_{\text{MENTS}}(s_t,a)>\hat{Q}^{(N(s_t,a^*,n))}_{\text{MENTS}}(s_t,a^*)\right)\right)\leq C_1\exp(-k_2N(s_t)). \tag{C.128}$$

The expected immediate regret can be bounded as follows:

$$\begin{aligned}&\mathbb{E}\texttt{inst\_regr}(s_t,\psi^n_{\text{MENTS}}) && \text{(C.129)}\\ =&\sum_{a\in\mathcal{A}-\{a^*\}}(V^*(s_t)-Q^*(s_t,a))\Pr\left(\psi^n_{\text{MENTS}}(s_t)=a\right) && \text{(C.130)}\\ =&\sum_{a\in\mathcal{A}-\{a^*\}}(V^*(s_t)-Q^*(s_t,a))\Pr\left(a=\arg\max_{a'}\hat{Q}^{(N(s_t,a',n))}_{\text{MENTS}}(s_t,a')\right) && \text{(C.131)}\\ \leq&\sum_{a\in\mathcal{A}-\{a^*\}}(V^*(s_t)-Q^*(s_t,a))\Pr\left(\hat{Q}^{(N(s_t,a,n))}_{\text{MENTS}}(s_t,a)>\hat{Q}^{(N(s_t,a^*,n))}_{\text{MENTS}}(s_t,a^*)\right) && \text{(C.132)}\\ \leq&\sum_{a\in\mathcal{A}-\{a^*\}}(V^*(s_t)-Q^*(s_t,a))C_1\exp(-k_2N(s_t,n)), && \text{(C.133)}\end{aligned}$$

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 C.2.13, provided the sufficient conditions are met, MENTS is consistent and will converge to a simple regret of zero.

Theorem C.2.14. *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:*

$$\mathbb{E}\texttt{sim\_regr}(s_t, \psi^n_{\text{MENTS}}) \leq C' \exp\left(-k'N(s_t, n)\right). \tag{C.134}$$

And specifically, at the root node s_0:

$$\mathbb{E}\texttt{sim\_regr}(s_0, \psi^n_{\text{MENTS}}) \leq C' \exp\left(-k'n\right). \tag{C.135}$$

Proof. This theorem holds as a consequence of Corollary **??** and Lemma C.2.13, and noting that at the root node $N(s_0, n) = n$. □

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 C.2.14. □

C.2.5 DENTS (and BTS) 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 C.2.15. *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 $s_{t+1} \in \bigcup_{a \in \mathcal{A}} \text{Succ}(s_t, a)$ there is some $C_{s_{t+1}}, k_{s_{t+1}} > 0$ such that for any $\varepsilon_{s_{t+1}} > 0$:*

$$\Pr\left(\left|\hat{V}_{\text{DNTS}}^{(N(s_{t+1},n))}(s_{t+1}) - V^*(s_{t+1})\right| > \varepsilon_{s_{t+1}}\right) \leq C_{s_{t+1}} \exp\left(-k_{s_{t+1}} \varepsilon_{s_{t+1}}^2 N(s_{t+1}, n)\right), \tag{C.136}$$

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

$$\Pr\left(\left|\hat{V}_{\text{DNTS}}^{(N(s_t,n))}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 N(s_t, n)\right). \tag{C.137}$$

Proof. Given the assumptions and by Lemmas C.2.10, C.2.1 and C.2.5, for some $C, k > 0$ and for any $\varepsilon_1 > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}_{\text{DNTS}}^{(N(s_t,a_t,n))}(s_t, a_t) - Q^*(s_t, a_t)\right| \leq \varepsilon_1\right) > 1 - C\exp(-k\varepsilon_1^2 N(s_t)). \tag{C.138}$$

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:

$$Q^*(s_t, a_t) - \varepsilon_1 \leq \hat{Q}_{\text{DNTS}}^{(N(s_t,a_t,n))}(s_t, a_t) \leq Q^*(s_t, a_t) + \varepsilon_1. \tag{C.139}$$

Let $a^* = \max_{a \in \mathcal{A}} Q^*(s_t, a)$. Using $\varepsilon_1 \leq \Delta_{\mathcal{M}}/2$ in (C.139), then for any $a \in \mathcal{A} - \{a^*\}$:

$$\begin{aligned}
\hat{Q}_{\text{DNTS}}^{(N(s_t,a,n))}(s_t, a) \leq & Q^*(s_t, a) + \Delta_{\mathcal{M}}/2 && \text{(C.140)} \\
\leq & Q^*(s_t, a^*) - \Delta_{\mathcal{M}}/2 && \text{(C.141)} \\
\leq & \hat{Q}_{\text{DNTS}}^{(N(s_t,a^*,n))}(s_t, a^*), && \text{(C.142)}
\end{aligned}$$

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

$$\hat{V}_{\text{DNTS}}^{(N(s_t,n))}(s_t) = \max_a \hat{Q}_{\text{DNTS}}^{(N(s_t,a,n))}(s_t, a) = \hat{Q}_{\text{DNTS}}^{(N(s_t,a^*,n))}(s_t, a^*). \tag{C.143}$$

Then using (C.139) with $a_t = a^*$ (noting $V^*(s_t) = Q^*(s_t, a^*)$), using (C.143) and using $\varepsilon_1 \leq \varepsilon$ gives:

$$\begin{aligned} V^*(s_t) - \varepsilon \leq & V^*(s_t) - \varepsilon_1 && \text{(C.144)} \\ \leq & \hat{V}_{\text{DNTS}}^{(N(s_t,n))}(s_t) && \text{(C.145)} \\ \leq & V^*(s_t) + \varepsilon_1 && \text{(C.146)} \\ \leq & V^*(s_t) + \varepsilon. && \text{(C.147)} \end{aligned}$$

Hence:

$$\begin{aligned} \Pr\left(\left|\hat{V}_{\text{DNTS}}^{(N(s_t,n))}(s_t) - V^*(s_t)\right| > \varepsilon\right) &\leq C\exp(-k\varepsilon_1^2 N(s_t,n)) && \text{(C.148)} \\ &\leq C\exp(-k\varepsilon^2 N(s_t,n)), && \text{(C.149)} \end{aligned}$$

which is the result. □

Similarly to the MENTS section, Lemma C.2.15 provides an inductive step, which is used in Theorem C.2.16 to show concentration inequalities at all states that DENTS visits.

Theorem C.2.16. *Consider a DENTS process, let $s_t \in \mathcal{S}$ then there is some $C, k > 0$ for any $\varepsilon > 0$:*

$$\Pr\left(\left|\hat{V}_{\text{DNTS}}^{(N(s_t,n))}(s_t) - V^*(s_t)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 N(s_t,n)\right). \tag{C.150}$$

Moreover, at the root node s_0:

$$\Pr\left(\left|\hat{V}_{\text{DNTS}}^{(N(s_0,n))}(s_0) - V^*(s_0)\right| > \varepsilon\right) \leq C\exp\left(-k\varepsilon^2 n\right). \tag{C.151}$$

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 $\hat{V}_{\text{DNTS}}^{(N(s_{H+1}))}(s_{H+1}) = V^*(s_{H+1}) = 0$. 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 C.2.15. Noting that $N(s_0) = n$ gives (C.151). □

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 C.2.17. *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:*

$$\mathbb{E}\texttt{inst\_regr}(s_t, \psi^n_{\text{DNTS}}) \leq C' \exp\left(-k' N(s_t, n)\right). \tag{C.152}$$

Proof. Let a^* be the locally optimal standard action, so $a^* = \arg\max_{a\in\mathcal{A}} Q^*(s_t, a)$. By Theorem C.2.16 and Lemmas C.2.10 and C.2.5 there exists $C_1, k_1 > 0$ such that for all $\varepsilon_1 > 0$:

$$\Pr\left(\forall a_t \in \mathcal{A}. \left|\hat{Q}^{(N(s_t,a_t,n))}_{\text{DNTS}}(s_t, a_t) - Q^*(s_t, a_t)\right| \leq \varepsilon_1\right) > 1 - C_1 \exp(-k_1 \varepsilon_1^2 N(s_t, n)). \tag{C.153}$$

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:

$$Q^*(s_t, a) - \Delta_{\mathcal{M}}/2 \leq \hat{Q}^{(N(s_t,a,n))}_{\text{DNTS}}(s_t, a) \leq Q^*(s_t, a) + \Delta_{\mathcal{M}}/2. \tag{C.154}$$

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:

$$\begin{aligned}
\hat{Q}^{(N(s_t,a,n))}_{\text{DNTS}}(s_t, a_t) \leq& Q^*(s_t, a_t) + \Delta_{\mathcal{M}}/2 && \text{(C.155)}\\
\leq& Q^*(s_t, a^*) - \Delta_{\mathcal{M}}/2 && \text{(C.156)}\\
\leq& \hat{Q}^{(N(s_t,a^*,n))}_{\text{DNTS}}(s_t, a^*). && \text{(C.157)}
\end{aligned}$$

Negating the probability of (C.157) then gives:

$$\Pr\left(\hat{Q}^{(N(s_t,a,n))}_{\text{DNTS}}(s_t, a) > \hat{Q}^{(N(s_t,a^*,n))}_{\text{DNTS}}(s_t, a^*)\right) \leq C_1 \exp(-k_2 N(s_t, n)) \tag{C.158}$$

Finally, the bound on the expected immediate regret follows:

$$\begin{aligned}
&\mathbb{E}\texttt{inst\_regr}(s_t, \psi^n_{\text{DNTS}}) && \text{(C.159)}\\
&= \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\psi^n_{\text{DNTS}}(s_t) = a\right) && \text{(C.160)}\\
&= \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(a = \arg\max_{a'} \hat{Q}^{(N(s_t,a,n))}_{\text{DNTS}}(s_t,a)\right) && \text{(C.161)}\\
&\leq \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a)) \Pr\left(\hat{Q}^{(N(s_t,a,n))}_{\text{DNTS}}(s_t,a) > \hat{Q}^{(N(s_t,a^*,n))}_{\text{DNTS}}(s_t,a^*)\right) && \text{(C.162)}\\
&\leq \sum_{a\in\mathcal{A}-\{a^*\}} (V^*(s_t) - Q^*(s_t,a))\, C_1 \exp(-k_2 N(s_t,n)), && \text{(C.163)}
\end{aligned}$$

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 C.2.17, the bound on the immediate simple regret can be converted to a bound on the simple regret.

Theorem C.2.18. *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:*

$$\mathbb{E}\texttt{sim\_regr}(s_t, \psi^n_{\text{DNTS}}) \leq C' \exp\left(-k' N(s_t,n)\right). \tag{C.164}$$

Moreover, at the root node s_0:

$$\mathbb{E}\texttt{sim\_regr}(s_0, \psi^n_{\text{DNTS}}) \leq C' \exp\left(-k' n\right). \tag{C.165}$$

Proof. Theorem holds as a consequence of Corollary **??** and Lemma C.2.17. To arrive at (C.165) note that $N(s_0,n) = n$. □

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_{DNTS} is bounded above and $\beta_{\text{DNTS}}(m) \geq 0$ for all $m \in \mathbb{N}$, then there exists constants $C, k > 0$ such* *that for all* $\varepsilon > 0$ *it holds that* $\mathbb{E}[\texttt{sim\_regr}(s_0, \psi^n_{\text{DNTS}})] \leq C \exp(-kn)$*, and also* $\hat{V}^{(N(s_0,n))}_{\text{DNTS}}(s_0) \xrightarrow{p} V^*(s_0)$ *as* $n \to \infty$*.*

Proof. The simple regret bound follows immediately from Theorem C.2.18. Using Theorem C.2.16 it holds that $\Pr\left(\left|\hat{V}^{(N(s_0,n))}_{\text{DNTS}}(s_0) - V^*(s_0)\right| > \varepsilon\right) \leq C \exp\left(-k\varepsilon^2 n\right) \to 0$ as $n \to \infty$, and hence $\hat{V}^{(N(s_0,n))}_{\text{DNTS}}(s_0)$ converges in probability to $V^*(s_0)$. □

Finally, the BTS exponential convergence bound follows from it being a special case of DENTS. TODO: need to convert to new latex notation for thesis, so commented out below

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] David Auger, Adrien Couetoux, and Olivier Teytaud. Continuous upper confidence trees with polynomial exploration–consistency. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I 13*, pages 194–209. Springer, 2013.

[4] 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.

[5] 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.

[6] Richard Bellman. *Dynamic Programming.* Dover Publications, 1957. ISBN 9780486428093.

[7] 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.

[8] 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.

[9] Tristan Cazenave, Flavien Balbo, Suzanne Pinson, et al. Monte-carlo bus regulation. In *12th international IEEE conference on intelligent transportation systems*, volume 340345, 2009.

[10] Pierre-Arnaud Coquelin and Rémi Munos. Bandit algorithms for tree search. *arXiv preprint cs/0703062*, 2007.

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

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

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1:1–22, 2006.

[23] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[24] Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. Technical report, 2014.

[25] 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.

[26] Thomas M Moerland, Joost Broekens, Aske Plaat, and Catholijn M Jonker. A0c: Alpha zero in continuous action space. *arXiv preprint arXiv:1805.09613*, 2018.

[27] Mausam Natarajan and Andrey Kolobov. *Planning with Markov decision processes: An AI perspective*. Springer Nature, 2022.

[28] Michael Painter. THTS++, 2025. URL `https://github.com/MWPainter/thts-plus-plus`. Available at `https://github.com/MWPainter/thts-plus-plus`.

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

[30] Herbert Robbins. Some aspects of the sequential design of experiments. 1952.

[31] 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.

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

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[34] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[35] David Silver and Joel Veness. Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010.

[36] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda

Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[37] 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.

[38] 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.

[39] 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.

[40] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

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

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

[43] 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.

[44] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

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

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