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

TODO: list

1. THTS section, as it defines notation. Put macros in text/abreviations.tex
2. Copy DENTS and BTS into thesis and copy into common notation
3. Define the toy problems, D-Chain and the one with the entropy trap

Acknowledgements

TODO: acknowledgements here

Abstract

TODO: abstract here

Contents

List of Figures

List of Tables

List of Notation

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

Multi-Armed Bandits (Section 2.1)

x . . . . . TODO: define here and move into correct place in the list

y . . . . . TODO: define here and move into correct place in the list

f_i . . . . . TODO: define here and move into correct place in the list

x^m . . . . . TODO: define here and move into correct place in the list

y^m . . . . . TODO: define here and move into correct place in the list

$\bar{y}_i^m$ . . . . . TODO: define here and move into correct place in the list

N . . . . . TODO: define here and move into correct place in the list

μ_i . . . . . TODO: define here and move into correct place in the list

μ^* . . . . . TODO: define here and move into correct place in the list

$\texttt{cum\_regr}_{\text{MAB}}$. TODO: define here and move into correct place in the list

$\pi(m)$ . . . . . TODO: define here and move into correct place in the list. Maybe keep it as this and ask question

m . . . . . TODO: define here and move into correct place in the list

$\pi_{\text{UCB}}(m)$ . . . . TODO: define here and move into correct place in the list

π^m . . . . . TODO: define here and move into correct place in the list

ψ^m . . . . . TODO: define here and move into correct place in the list

w . . . . . TODO: define here and move into correct place in the list

w^m . . . . . TODO: define here and move into correct place in the list

$\mu_{w,i}$ . . . . . TODO: define here and move into correct place in the list

μ_w^* . . . . . TODO: define here and move into correct place in the list

W . . . . . TODO: define here and move into correct place in the list

. . . . . TODO: define here and move into correct place in the list

. . . . . TODO: define here and move into correct place in the list

. . . . . TODO: define here and move into correct place in the list

. . . . . TODO: define here and move into correct place in the list

Markov Decision Processes and Reinforcement Learning (Sect

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

$\mathcal{A}$ . . . . . Set of actions in an MDP.

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

$\mathcal{H}$ . . . . . Shannon entropy, of a policy.

$J(\pi)$ . . . . . TODO: objective funciton

$J_{\text{sft}}(\pi)$ . . . . . TODO: soft objective funciton

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

π . . . . . A policy, TODO: state that this is just for this chapter, see sect 2.2 defn for rest of thesis

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

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

Q^π . . . . . The Q-value of a policy π. $Q^\pi(s,a;t)$ denotes the expected cumulative reward that policy π will achieve, starting from state $s_t = s$, starting by taking action $a_{t+1} = a$.

Q^* . . . . . The optimal value function $Q^*(s,a;t)$ denotes the expected maximal value that can be achieved from state $s_t = s$, starting with action $a_{t+1} = a$ by any policy.

Q^π_{sft} . . . . . The Q-value of a policy π. $Q^\pi_{\text{sft}}(s,a;t)$ denotes the expected cumulative reward that policy π will achieve, starting from state $s_t = s$, starting by taking action $a_{t+1} = a$, TODO: plus the entopy of the policy weighted by the temperature alpha. TODO: consider just writing max pi over the soft values here instead?

Q^*_{sft} . . . . . The optimal value function $Q^*_{\text{sft}}(s,a;t)$ denotes the expected maximal value that can be achieved from state $s_t = s$, starting with action $a_{t+1} = a$ by any policy, TODO: plus the entopy of the (optimal soft) policy weighted by the temperature alpha.

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

$\mathcal{S}$ . . . . . Set of states in an MDP.

τ . . . . . TODO: trajectory

$\tau_{:h}$ . . . . . TODO: trajectory

t . . . . . TODO: free variable for current timestep? And generally add a list of free variables?

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

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

V^π_{sft} . . . . . The soft value of a policy π. $V^\pi_{\text{sft}}(s; t)$ denotes the expected cumulative reward that policy π will achieve, starting from state $s_t = s$, TODO: plus the entopy of the policy weighted by the temperature alpha.

V^*_{sft} . . . . . The optimal soft value function $V^*(s; t)$ denotes the expected maximal value that can be achieved from state $s_t = s$ by any policy, TODO: plus the entopy of the (optimal soft) policy weighted by the temperature alpha.TODO: consider just writing max pi over the soft values here instead?

$\mathcal{M}$ . . . . . TODO: define here and move into correct place in the list

Trial Based Heuristic Tree Search (Section ??)

$\mathcal{B}_Q$ . . . . . TODO: define backup

$\mathcal{B}_V$ . . . . . TODO: define backup

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

`mcts_mode` . . . TODO: definition of `mcts_mode`

$N(s)$ . . . . . 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).

n . . . . . Number of trials run.

`node` . . . . . A mapping from states and state-action pairs to their corresponding decision and chance nodes respectively.

$\texttt{node}(s).V$ . . . TODO: write this

$\texttt{node}(s, a).Q$ . . TODO: write this

π . . . . . A search policy, TODO: define, this is a parameter of thts++. TODO: handle π^k

T . . . . . Computation time limit.

$\mathcal{T}$ . . . . . A THTS search tree. TODO: With: $\mathcal{T} \subseteq \mathcal{S} \cup \mathcal{S} \times \mathcal{A}$. TODO: handle $\mathcal{T}^k$

$\mathcal{T}^k$ . . . . . TODO: search tree on after k trials

V_{init} . . . . . The initialisation function used in `THTS++`, used to initialise the value of a new decision node.

$\bar{V}_{\text{UCT}}$ . . . . . TODO: define here and move into correct place in the list

$\bar{Q}_{\text{UCT}}$ . . . . . TODO: define here and move into correct place in the list

π_{UCT} . . . . . TODO: define here and move into correct place in the list

b_{UCT} . . . . . TODO: define here and move into correct place in the list

$\bar{V}_{\text{MCTS}}$ . . . . . TODO: define here and move into correct place in the list

$\bar{Q}_{\text{MCTS}}$ . . . . . TODO: define here and move into correct place in the list

π_{MCTS} . . . . . TODO: define here and move into correct place in the list

b_{MCTS} . . . . . TODO: define here and move into correct place in the list

$\hat{V}_{\text{MENTS}}$ . . . . . TODO: define here and move into correct place in the list

$\hat{Q}_{\text{MENTS}}$ . . . . . TODO: define here and move into correct place in the list

π_{MENTS} . . . . . TODO: define here and move into correct place in the list

α_{MENTS} . . . . . TODO: define here and move into correct place in the list

λ_s . . . . . TODO: define here and move into correct place in the list

ϵ . . . . . TODO: define here and move into correct place in the list

Multi-Objective Reinforcement Learning (Section 2.4)

$\boldsymbol{\mathcal{M}}$ . . . . . TODO: define here and move into correct place in the list

τ . . . . . TODO: define here and move into correct place in the list

$\tau_{:h}$ . . . . . TODO: define here and move into correct place in the list

$\mathbf{V}^{\pi}$ . . . . . TODO: define here and move into correct place in the list

$\mathbf{Q}^{\pi}$ . . . . . TODO: define here and move into correct place in the list

Δ^d . . . . . TODO: define here and move into correct place in the list

u . . . . . TODO: define here and move into correct place in the list

$\mathbf{w}$ . . . . . TODO: define here and move into correct place in the list

d . . . . . TODO: define here and move into correct place in the list

Π . . . . . TODO: define here and move into correct place in the list

$U(\Pi; u)$ . . . . . TODO: define here and move into correct place in the list

u_{lin} . . . . . TODO: define here and move into correct place in the list

$CH(\Pi)$ . . . . . TODO: define here and move into correct place in the list

$CS(\Pi; u)$ . . . . TODO: define here and move into correct place in the list

$CCS(\Pi)$ . . . . TODO: define here and move into correct place in the list

. . . . . TODO: define here and move into correct place in the list

. . . . . TODO: define here and move into correct place in the list

. . . . . TODO: define here and move into correct place in the list

List of Abbreviations

Multi-Armed Bandits (Section 2.1)

MAB . . . . . TODO: define here and move into correct place in the list

UCB . . . . . TODO: define here and move into correct place in the list

EMAB . . . . TODO: define here and move into correct place in the list

CMAB . . . . TODO: define here and move into correct place in the list

Markov Decision Processes and Reinforcement Learning (S

MDP . . . . . Markov Decision Process.

Trial Based Heuristic Tree Search (Section 2.3)

MCTS . . . . . Monte Carlo Tree Search.

MENTS . . . . Maximum ENtropy Tree Search.

THTS . . . . . Trial-based Heuristic Tree Search.

`THTS++` . . . . . TODO: thts++

UCT . . . . . Upper Confidence Bound applied to Trees.

`node` . . . . . TODO: define here and move into correct place in the list

Multi-Objective Reinforcement Learning (Section 2.4)

CHVI . . . . . Convex Hull Value Iteration

MOMDP . . . Multi-Objective Markov Decision Process

1 Introduction

Contents

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

1.1 Overview

TODO: list

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

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

1.2 Contributions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.3 Structure of Thesis

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

1.4 Publications

TODO: update final publication when submit

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

- Painter, M; Lacerda, B; and Hawes, N. "Convex Hull Monte-Carlo Tree-Search." In *Proceedings of the international conference on automated planning and scheduling. Vol. 30. 2020*, ICAPS, 2020, (see Appendix **??**).
- 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, (see Appendix **??**).
- Painter, M; Hawes, N; and Lacerda, B. "Simplex Maps for Multi-Objective Monte Carlo Tree Search." In *TODO*, **Under Review at conf_name**, (see Appendix **??**).

2
Background

Contents

TODO: Introduce that going to introduce notation and give the building blocks this thesis builds off

TODO: Nicks comment to revisit: In Sections 2 and 3 you present RL before tree search. This was a surprise to me. Wouldn't it be better to start with single objective MDPs, then introduce methods of solving them (planning, bandits, rl) and the different assumptions they make? Then move to MO versions. I'd also be

open to you presenting MCTS then generalising to MCTS rather than the other way around, but I think the way you present it is probably the most efficient.

TODO: Add a regret or multi-armed bandit section?

2.1 Multi-Armed Bandits

TODO: Introduce tree search using multi-armed bandits?

- Would like to think a bit about some of the bandits work that sample actions (from adversarial I think), because they were similar to boltzmann search but I hadn't seen details about those works when writing dents
- Also the gradient based MAB stuff in sutton and barto book? Looks relevant? Maybe consider that as update to DENTS paper? Either way, another idea for getting good Go results.

TODO: list

- $R(s, a)$ is a random variable in MAB literature, but we're assuming it's a fixed value in RL
- Multi-Armed Bandits routines algos
- Exploring Bandits routines and algos
- Contextual Bandits routines and algos

TODO: some more waffelly shit about MABs

In the K-armed bandit problem, an agent is tasked with iteratively selecting one of K arms, originally introduced by TODO: author TODO: cite MAB orig. In each iteration, once an arm is selected, say $x \in \{1, ..., K\}$, and a reward y is returned according to an unknown probability distribution f_x. Let n be the number of rounds of this game that are played, and let $\mu_i = \mathbb{E}[y|x = i]$ be the expected reward of pulling arm i.

When analysing an algorithmic strategy for Multi-Armed Bandits (MABs), a quantity known as *regret* is commonly used, which compares the cumulative

reward obtained, compared to the maximal reward that could be obtained with full knowledge of $\{f_i\}$.

TODO: work out how to deal with the issue of not having policies defined yet. thinhk this needs to be moved to after ch2.1

TODO: some sentence, the process of a MAB is as follows:

- for m in $\{1, ..., n\}$:
- agent selects arm x^m according to policy π
- agent recieves reward $y^m \sim f_{x^i}$

Definition 2.1.1. *The* (cumulative) regret *of the agent in the above process is defined as:*

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

where $\mu^* = max_i \mu_i$.

To theoretically analyse algorithms for MAB problems, the quantity of expected regret, $\mathbb{E}[\texttt{cum\_regr}_{\text{MAB}}]$ is considered. In TODO: cite it is shown using information theory that there is a lower bound on the expected regret that an agent can achieve of $\Sigma(\log N)$, and in TODO: cite ucb the Upper Confidence Bound (UCB) algorithm is introduced, achieving a matching upper bound of $O(\log N)$.

Let $N^m(j)$ be the number of times that arm i has been pulled after m rounds, $\bar{y}_i^m$ is the sample average of rewards recieved when pulling arm i:

$$\bar{y}_i^m = \frac{1}{m} \sum_{j=1}^{m} y^j \mathbb{1}[x^j = i]. \tag{2.2}$$

Then the arm selected by the UCB algorithm on the mth round is given by:

$$\pi_{\text{UCB}}(m) = \underset{j \in \{1,..,K\}}{\arg\max}\, \bar{y}_j^m + b_{\text{UCB}} \sqrt{\frac{\log(m)}{N(j)}} \tag{2.3}$$

TODO: define b_{UCB}

TODO: talk about explortaiton-exploitation trade off – see Edwin thesis too because said it nicely

2.1.1 Exploring Bandits

In the pure exploration K-armed bandit problem TODO: cite, the game is changed slighly. The agent still gets to pull an arm each round, but after it recieves a reward each round it is given the opportunity to output a *recommendation*. In exploring multi-armbed bandits (EMABs), 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 change seperates 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.

TODO: some sentence, the process of a EMAB is as follows:

- for m in $\{1, ..., n\}$:
- agent selects arm x^m according to policy π^m
- agent recieves reward $y^m \sim f_{x^m}$
- agent outputs a recommendation policy ψ^m

Under this regime, 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 an *instantaneous regret*, TODO: define instantaneous regret which would come from following the recommendation policy.

Definition 2.1.2. *The* simple regret *of following the recommendation policy ψ^m on the mth round is:*

$$\texttt{sim\_regr}_{\text{EMAB}}(m) = \mathbb{E}_{i \sim \psi^m}[\mu^* - \mu_i]. \tag{2.4}$$

TODO: talk about the bounds they show in their work

TODO: describe the algorithm which is pulling arms 1,2,...,K,1,2,...,K,1,..., and so on, and then recommending the arm that has the best empirical average.

2.1.2 Contextual Bandits

TODO: sleepy and rushed this section a bit

In the contextual K-armed bandit problem TODO: cite, on each round the algorithm is given a context $w \in \mathcal{W}$, which the algorithm does not choose. If contextual multi-armed bandit problems (CMABs), the distribution that the rewards are drawn from now depend on w, written $f_{w,i}$ for context w for arm i.

TODO: some more words around when this is useful and why, and define CMABs

TODO: some sentence, the process of a MAB is as follows:

- for m in $\{1, ..., n\}$:
- agent recieves context w^m
- agent selects arm x^m according to policy π
- agent recieves reward $y^m \sim f_{w^m, x^m}$

Similar to MABs and EMABs, the notion of regret is used to analyse CMABs. Specifically, *contextual regret* is defined similarly to cumulative regret TODO: ref, while taking into account the contexts drawn.

Definition 2.1.3. *The* (cumulative) contextual regret *of the agent in the above process is defined as:*

$$\texttt{ctx\_regr}_{\text{CMAB}}(\pi, n) = \sum_{i=1}^{n} \mu^*_{w^i} - y^i, \tag{2.5}$$

where $\mu^*_w = \max_i \mu_{w,i}$.

TODO: add definitions for $\mu_{w,i}$

TODO: Talk about contextual zooming precursor (need to read and add to litrev)

TODO: Write about contextual zooming, using CHMCTS paper, and use opportunity to add to litrev

2.2 Markov Decision Processes and Reinforcement Learning

TODO: list

- Typical agent interacting with environment diagram
- Agent planning with simulator
- ~~MDPs definition~~
- ~~Policies~~
- ~~Value functions (single (and multi-objective?))~~
- ~~Basic results and definitions we use (tabular planning algorithms)~~
- ~~Talk about entropy and some of that work (probably a subsection)~~

TODO: Split this section into MDPs definition and move RL below monte carlo tree search? Probably if do the multi-armed bandit stuff.

In this section Markov Decision Processes are introduced, as well as some fundamental concepts in Reinforcement Learning such as *policies* and *value functions* which will be useful in the following. Afterwards, the *maximum entropy objective* is discussed, as it will be relevant to some of the work in this thesis and other closely related work.

In this thesis we will only consider discrete and finite-horizon Markov Decision Processes, TODO: but we will point to works that make adaptions for non-finite horizons and continuous and partially observable environments, and briefly describe why the ideas could be used in parallel with the concepts in this thesis.

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.*

Where it is more convenient, we will use the following notation to refer to the set of successor states when discussing MDPs:

Definition 2.2.2. *The set of* successor states $\text{Succ}(s, a)$ *of a state-action pair* (s, a) *is defined as* $\text{Succ}(s, a) = \{s' | p(s'|s, a) > 0\}$. *Additionally, let* $s' \sim \text{Succ}(s, a)$ *be a shorthand for* $s' \sim p(\cdot|s, a)$.

TODO: decide if should keep p(cdot|s,a), and just swap out the succ command as shorthand for that?

Formally, we will consider a policy to be a mapping from a state in $\mathcal{S}$ to a distribution over actions $\mathcal{A}$. TODO: For this thesis we will consider policies to be stochastic, so when a policy is used to generate actions to follow, it is done by sampling from the distribution.

TODO: So im using pi for the definition here. But for rest of thesis it will be used as explicitly the "search policy"

In some cases it will be necessary to talk about deterministic policies. TODO: When ambiguous we will state when a policy is deterministic. A deterministic policy is written as a one hot stochastic distribution

Definition 2.2.3. *A* (stochastic) policy $\pi : \mathcal{S} \to \mathcal{A} \to [0, 1]$ *is a mapping from states to distributions over actions, where for all* $s \in \mathcal{S}$ *we have* $\sum_{a \in \mathcal{A}} \pi(a|s) = 1$. *We will use the following notations:* $\pi(\cdot|s)$ *is used to denote the entire distribution over the set of actions,* $\pi(a|s)$ *denotes the probability that action* a *is sampled by the policy at state* s.

Additionally, for deterministic policies *we will use the shorthand notation* $\pi(s)$ *to denote the action taken by the policy. Formally, we will write* $\pi(s) = a'$ *where* $a' \in \mathcal{A}$ *is such that* $\pi(a'|s) = 1$.

TODO: want to add a statement saying something like $\pi(s) = a'$ *is shorthand for* $\pi(a|s) = \mathbb{1}[a = a']$.

TODO: Actually want to say something like: a deterministic policy is special case of the stochastic, with a delta distribution, such as $\pi(a|s) = \mathbb{1}[a = a']$, *and in*

such cases we will use $\pi(s) = a'$ as a short hand to state $\pi(a|s) = \mathbb{1}[a = a']$ and $\pi(s)$ as a shorthand for a'.

TODO: Explicitly note that we are using pi as a free variable (is free variable correct word here?) here, and that in section - todo ref - we will use pi to refer to a specific type of policy (search policies) in the rest of the thesis after this section.

TODO: clean up policy definition with shorthand stuff

TODO: actually state the the values of a policy need to add to one

In reinforcement learning it is common to refer to a sequence of states, actions and rewards as a *trajectory*. In the tree search literature it is more common to refer to this as a *trial* TODO: cite thts?, so we will use these terms interchangeably.

Definition 2.2.4. *A* trajectory*, is a sequence of state, action and rewards, that is induced by a policy π and MDP $\mathcal{M}$ pair. Let the trials/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 $s_{t+1} \sim \text{Succ}(s_t, a_t)$. Notationally, we will write $\tau \sim \pi$ to denote a sampled trial/trajectory with respect to a policy, where the MDP is implicit.*

Sometimes it will be necessary to reason about trajectories with a horizon $h < H$, which will be denoted $\tau_{:h} = (s_0, a_0, r_0, s_1, a_1, r_1, ..., s_{h-1}, a_{h-1}, r_{h-1}, s_h)$.

Next the value and Q-value of a policy is defined, which is the expected cumulative reward that a policy will obtain:

Definition 2.2.5. *The* value *of a policy π from state s at time t is:*

$$V^\pi(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} r_t \middle| s_t = s\right]. \tag{2.6}$$

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

$$Q^\pi(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim\text{Succ}(s,a)}[V^\pi(s';t+1)]. \tag{2.7}$$

From the definition of the values functions the optimal value functions can be defined:

Definition 2.2.6. *The* Optimal (Q-)Value *of a state(-action pair) is defined as:*

$$V^*(s;t) = \max_\pi V^\pi(s;t) \tag{2.8}$$

$$Q^*(s,a;t) = \max_\pi Q^\pi(s,a;t). \tag{2.9}$$

In reinforcement learning, the objective is to find a policy with maximal value:

Definition 2.2.7. *The* (standard) reinforcement learning objective function $J(\pi)$ *is defined as:*

$$J(\pi) = V^\pi(s_0;0). \tag{2.10}$$

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

It can be shown TODO: refs that the optimal (Q-)value functions satisfy the *Bellman equations*:

$$V^*(s;t) = \max_{a\in\mathcal{A}} Q^*(s,a;t), \tag{2.11}$$

$$Q^*(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim \mathrm{Succ}(s,a)}[V^*(s';t+1)]. \tag{2.12}$$

In tabular reinforcement learning, a table of values for $V(s;t)$ TODO: havent actually defined V without any superscript is kept for each s,t. Given any initial value function $V^{(0)}$ let the *Bellman backup* operations be:

$$V^{k+1}(s;t) = \max_{a\in\mathcal{A}} Q^{k+1}(s,a;t), \tag{2.13}$$

$$Q^{k+1}(s,a;t) = \mathbb{E}_{s'\sim \mathrm{Succ}(s,a)}[R(s,a) + V^k(s';t+1)]. \tag{2.14}$$

Using this *dynamic programming* approach is known as *value iteration*. It can be shown that the Bellman backups are contraction operators TODO: add cite, which can be used to show that $V^k \to V^*$ as $k \to \infty$. In the discrete TODO: what the actual conditions are case we are considering, there will always be some $N < \infty$ such that $V^N = V^*$.

TODO: cite [Bellman 1957] Bellman, R. 1957. Dynamic Programming. Princeton, NJ, USA: Princeton University Press, 1 edition.

TODO: add optimal policy from Q values

2.2.1 Maximum Entropy Reinforcement Learning

In *Maximum Entropy Reinforcement Learning*, the objective function is altered to include the addition of an entropy term. Let $\mathcal{H}$ denote the (Shannon) entropy function TODO: cite:

$$\mathcal{H}(\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi(\cdot|s)}[-\log \pi(a|s)]. \tag{2.15}$$

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.

In the maximum entropy objective, the relative weighting of entropy terms is included using a coefficient α, which is 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 2.2.8. *The* soft value *of a policy π from state s at time t is:*

$$V^{\pi}_{\text{sft}}(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} r_t + \alpha\mathcal{H}(\pi(\cdot|s_i))\middle| s_t = s\right]. \tag{2.16}$$

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

$$Q^{\pi}_{\text{sft}}(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}[V^{\pi}_{\text{sft}}(s';t+1)]. \tag{2.17}$$

Similarly, optimal soft (Q-)values can be defined:

Definition 2.2.9. *The* Optimal soft (Q-)Value *of a state(-action pair) is defined as:*

$$V^{*}_{\text{sft}}(s;t) = \max_{\pi} V^{\pi}_{\text{sft}}(s;t) \tag{2.18}$$

$$Q^{*}_{\text{sft}}(s,a;t) = \max_{\pi} Q^{\pi}_{\text{sft}}(s,a;t). \tag{2.19}$$

Equations similar to the Bellman equations, aptly named the *Soft Bellman equations* can be defined, which differ to equations TODO: ref by the replacement of the max operation with the *softmax* operation (which is why the maximum entropy analogues are referred to as the *soft* versions of their standard reinforcement learning counterparts).

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

Definition 2.2.10. *The* maximum entropy (or soft) reinforcement learning objective function $J_{\text{sft}}(\pi)$ *is defined as:*

$$J_{\text{sft}}(\pi) = V^{\pi}_{\text{sft}}(s_0; 0). \tag{2.20}$$

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

Similarly to standard reinforcement learning, it can be shown TODO: refs that the optimal soft (Q-)value functions satisfy the *soft Bellman equations*:

$$V^{*}_{\text{sft}}(s; t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(Q^{*}_{\text{sft}}(s, a; t)/alpha\right), \tag{2.21}$$

$$Q^{*}_{\text{sft}}(s, a; t) = R(s, a) + \mathbb{E}_{s' \sim \text{Succ}(s,a)}[V^{*}_{\text{sft}}(s'; t + 1)]. \tag{2.22}$$

Again, similarly to standard reinforcement learning, we can define *soft Bellman backups* that admit an analogous algorithm to value iteration:

$$V^{k+1}_{\text{sft}}(s; t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(Q^{k+1}_{\text{sft}}(s, a; t)/alpha\right), \tag{2.23}$$

$$Q^{k+1}_{\text{sft}}(s, a; t) = R(s, a) + \mathbb{E}_{s' \sim \text{Succ}(s,a)}[V^{k}_{\text{sft}}(s'; t + 1)]. \tag{2.24}$$

Finally, given the optimal soft value and soft Q-value functions, the optimal soft policy is known TODO: cite:

$$\pi^{*}_{\text{sft}}(a|s; t) = \exp\left(\left(Q^{*}_{\text{sft}}(s, a; t) - V^{*}_{\text{sft}}(s; t)\right)/\alpha\right). \tag{2.25}$$

2.2.2 Remaining todos for this chapter after first draft

TODO: Add a comment similar to DENTS paper where we will drop the t in notation. Make it quite bold somehow

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

TODO: list

- Give high level overview of MCTS (why use it etc)
- Outline that I'll present this as here is THTS, and then here's the THTS routines for MCTS

TODO: double check the intro to 2.2, as wrote this a while ago

TODO: try to make sure specific about using thts vs mcts

In this section we introduce `THTS++` [3], which is an open-source, parallelised extension of the Trial-based Heuristic Tree Search schema [2] (THTS). This schema is a generalisation of Monte Carlo Tree Search (MCTS), as presented in Section **??**. In `THTS++` trees consist of *decision nodes* and *chance nodes*. Decision nodes output actions that can be taken by the agent, and chance nodes output *outcomes* that may be random and may depend on the action taken. As such, each decision node has an associated *state* and each chance node has an associated *state-action pair*. In this work, we are considering fully-observable environments, but `THTS++` can be generalised to consider *partially-observable* environments. We give `THTS++` implementations of the standard Upper Confidence Bound applied to Trees (UCT) algorithm and Maximum ENtropy Tree Search (MENTS) in Sections **??** and **??** respectively.

TODO: comments about how most MCTS algorithms are using a multi-armed bandit algorithm at each node

In MCTS we run trials, either for some fixed number of trials, or some timelimit, where each trial is split into four stages: (1) selection, which samples states and actions for the trial, corresponding to a path down the tree; (2) expansion, which creates any new nodes in the tree; (3) initialisation, which initialises values at any new leaf nodes in the tree; (4) backup, which updates values at all nodes visited on the trial.

TODO: add MCTS figure here?

```
1 def foo(bar):
2         print("helloworld!")
```

2.3.1 Trial Based Heuristic Tree Search

TODO: list

- Copy DENTS MCTS section presentation, make a notation `node`(s_t) for the node at state s_t
- Present thts++
- Indicate what parts are new versus the original paper (context function, optionally running `mcts_mode` and mutli-threading)
- Small comment about multi-threading and two-phase locking used to avoid deadlock
- TODO: probably not necessary to say - but thought of nice/concise way of explaining it (a node can lock children, not parent, if need info from parent, then it has to put a thread safe copy in the context)
- Define terms precisely and consistently, for example `mcts_mode` (say that notation and terminology varies widely in literature, e.g. does uct run in mcts mode or not?)
- Mention that V_{init} can be implemented as V_θ to be used with deep RL methods
- TODO: Find the best place to talk about deep RL? Maybe in the RL section?

TODO: this section relly really needs diagrams

TODO: add psuedo code

TODO: need to talk about contexts here again...

In this section we will present `THTS++`schema, which is TODO: adaptation? of the Trial-Based Heuristic Tree Search (THTS) schema TODO: cite. After we have

presented THTS++, we will use the schema to define tree search algorithms that are relevant in this thesis, namely Upper Confidence Bound Applied to Trees (UCT) TODO: cite in subsection TODO: ref and Maximum ENtropy Tree Search (MENTS) in subsection TODO: Ref. Finally we will briefly point out the differences between THTS++and the original THTS schema in subsection TODO: ref.

TODO: this is already a subsection, so update above

In THTS++ a search tree $\mathcal{T}$ is built using Monte Carlo trials. Each trial is split into two phases: the *selection phase* where a trajectory is sampled using a *search policy*; and the *backup phase* where value estimates stored in the tree structure are updated. In THTS++ the selection phase encumpasses the expansion and initialisation phases TODO: of the common presentation of MCTS, where new nodes are added to the tree and the values of any new leaf node is initialised.

TODO: be more presise about trajectory vs trial, and update for sect 2.1.

To simplify notation in the presentation of THTS++ we will assume that states and state-action pairs have a one to one correspondance with nodes in the search tree $\mathcal{T}$. 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. Given this assumption, we can state that the search tree is a subset of the state and state-action spaces, that is $\mathcal{T} \subseteq \mathcal{S} \cup \mathcal{S} \times \mathcal{A}$. TODO: rephrase last sentance for defn

Definition 2.3.1. *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 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* $(s', a') \in \mathcal{T}$*.*

TODO: clean above defn up

TODO: probably want to explicitly define what it means for s or s,a to be in a trajectory

TODO: words about decision and chance nodeS?

Definition 2.3.2. *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. When it is not clear from context if an* s *or* (s, a) *refers to a state(-action pair), the notation* $\texttt{node}(s)$ *and* $\texttt{node}(s, a)$ *will be used. TODO: fix node notation here*

Additionally, each decision and chance node will generally store value estimates that are algorithm dependent. To specify this we will use $\texttt{node}(s).V$ to denote the set of values stored at node $\texttt{node}(s)$, and $\texttt{node}(s, a).Q$ for the set of value stored at node $\texttt{node}(s, a)$. TODO: make this a defn?

TODO: define N(s) and N(s,a)

The initial search tree consists of a single root node that corresponds to the initial state of the MDP: $\mathcal{T}^0 = \{s_0\}$. And let $\mathcal{T}^k$ denote the search tree of `THTS++` after k trials have been run.

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

Search policy: A distribution π^k for the $(k+1)$th trial, which can use values in the current search tree $\mathcal{T}^k$;

Heuristic function: A function V_{init} used as a heuristic to initialise values for new desicion nodes added to the tree;

Backup function: Two functions $\mathcal{B}_V$ and $\mathcal{B}_Q$ which updates values; TODO: Clewan this up, and word better pls

MCTS mode: A boolean `mcts_mode`specifying if `THTS++` should operate in MCTS mode.

TODO: actually define the above things properly somewhere

The $k+1$th trial of the `THTS++` schema operates as follows: TODO: this is probably better written as psuedocode...

1. sample a trajectory $\tau_{:h}$ using the search policy π^k;
   - If `mcts_mode` is False, then $h = H$;
   - If `mcts_mode` is True, then h is such that $s_{h-1} \in \mathcal{T}^k$ and $s_h \notin \mathcal{T}^k$, or $h = H$.

2. Any new nodes nodes that need to be added from this trajectory are added to the tree, $\mathcal{T}^{k+1} = \mathcal{T}^k \cup \tau_{:h}$;

3. If $s_h \notin \mathcal{T}^k$ then $\texttt{node}(s_h).V$ is initialised using V_{init};

4. The backup functions are used to update values in the tree:

   - For $i = h-1, h-2, ..., 1, 0$:
     - $\texttt{node}(s_i, a_i).Q \leftarrow \mathcal{B}_Q(\{\texttt{node}(s').V | s' \in \texttt{node}(s_i, a_i).\texttt{children}\})$
     - $\texttt{node}(s_i).V \leftarrow \mathcal{B}_V(\{\texttt{node}(s_i, a').Q | a' \in \texttt{node}(s_i).\texttt{children}\})$

TODO: converting the above into some psuedocode. Should probably define children as a property of nodes. Should also just state that $\texttt{node}(s_i, a_i).Q$ TODO: and so on are just scalar values for now. We can make them vectors when needed later. TODO: also make sure define $N(s)$ and $N(s, a)$

```
1  def run_trial(search_policy: π, heuristic_fn: V_init):
2      τ_:h = sample_trajectory(pi)
3      if s_h ∉ T:
4          initialise_values(s_h, V_init)
5      for i in {h-1,h-2,...,1,0}:
6          backup_q(s_i,a_i)
7          backup_v(s_i)
8
9  def sample_trajectory(search_policy: π):
10     pass
11
12 def initialise_values(s_h, V_init):
13     pass
14
15 def backup_q(s_i, a_i):
16     pass
17
18 def backup_v(s_i):
19     pass
```

TODO: define $\texttt{node}(s_i).\texttt{children}$ and $\texttt{node}(s_i, a_i).\texttt{children}$.

TODO: Check how thts deals with using value estimates when children dont exist. There should be some form of using heuristic. Maybe this needs to be added to THTSpp todo list

TODO: Basically want to say that the 0th version of any estimate is filled using the Vinit and Qinit functions where necessary. So for example if a chance node doesnt exist yet then use Qinit.

TODO: should also be more clear that bottom node is initialised s_h TODO: using Vinit in the backup phase, and the rest use the backups

TODO: make sure neurips paper writing integrated (commented out, and below this comment in the .tex) - read the actual pdf for thesis and neurips papers and compare cover same info?

TODO: talk about the differences between thtspp and thts

2.3.2 Upper Confidence Bounds Applied to Trees (UCT)

TODO: list

- Define UCT here

TODO: I'm feeling ill writing this section, so just going to word vomit this shit out and make it sound not shit later

TODO: add Quct commands like Qments

Upper Confidence Bounds Applied to Trees (UCT) TODO: cite is a commonly used tree search algorithm, which is based on the Upper Confidence Bounds (UCB) TODO: cite both papers algorithm for Multi-Armed Bandit problems TODO: cite original MAB and a review.

In the literature, UCT and MCTS are often used synonomously, however this leaves some of the specifics of the algorithms used as ambiguous. In this thesis, we will present UCT as it was originally presented in TODO: cite. And in subsection TODO: ref we will specify the variant of UCT which is commonly referred to as MCTS.

UCT can be defined using the THTS schema outlined in section TODO: ref as follows:

Firstly, UCT as originally presented is run with `mcts_mode`set to False. As such, all sampled trajectories are sampled until timestep H, the finite horizon of the MDP.

At each node a the sampled averages $\bar{V}_{\text{UCT}}$ or $\bar{Q}_{\text{UCT}}$ for value estimates.

The search policy that UCT follows is:

$$\pi_{\text{UCT}}(s) = \arg\max_{a \in \mathcal{A}} Q_{\text{UCT}}(s,a) + b_{\text{UCT}} \sqrt{\frac{\log(N(s))}{N(s,a)}} \tag{2.26}$$

TODO: add labels for equations

In TODO: ref above eqn, when $N(s,a) = 0$ there is a division by zero, which is taken as inf, and ties are broken randomly, which effectively implements the "every arm is initialised by pulling it once" TODO: actually quote the paper, and cite UCT paper. TODO: define the bias param

After sampling a trajectory $\tau_{:H} \sim \pi_{\text{UCT}}$ are updated as follows:

$$\bar{Q}_{\text{UCT}}(s_t, a_t) \leftarrow \frac{1}{N(s,a)} \left((N(s,a)-1)\bar{Q}_{\text{UCT}}(s_t,a_t) + \sum_{i=t}^{H-1} R(s_i,a_i) \right) \tag{2.27}$$

$$\bar{V}_{\text{UCT}}(s_t) \leftarrow \frac{1}{N(s,a)} \left((N(s,a)-1)\bar{V}_{\text{UCT}}(s_t,a_t) + \sum_{i=t}^{H-1} R(s_i,a_i) \right) \tag{2.28}$$

TODO: add labels for equations

TODO: Some note about the V values not actually being used in the algorithm

TODO: some comment about it can be implemented as backups (copy equations from THTS), but typically implemented as above. OR, just define backup functions to take the trajectory too

Because UCT is planning in a finite horizon MDP, the heuristic function will only be called on states that are at the time horizon H. As such, for UCT we can set $V_{\text{init}}(s) = 0$.

TODO: add polynomial UCT here? and or prioritised UCT from alpha go here?

TODO: make sure neurips paper writing integrated in UCT and MCTS section (commented out, and below this comment in the .tex) - read the actual pdf for thesis and neurips papers and compare cover same info?

TODO: add stuff about regret here

2.3.3 Monte-Carlo Tree Search

TODO: list

- Give overview of MCTS
- Give UCT in terms of THTS schema
- Define terms precisely and consistently in terms of THTS functions, maybe `mcts_mode` should go here
- Define the value initialisation of THTS using a rollout policy for MCTS
- 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?

TODO: I'm feeling ill writing this section, so just going to word vomit this shit out and make it sound not shit later

TODO: add Qmcts commands like Qments

In this thesis we will refer to any algorithm that only adds one decision node to the search tree on each trial as an MCTS algorithm. That is any THTS algorithm with `mcts_mode`set to True is an MCTS algorithm.

In this section we will present what is commonly referred to as MCTS in the literature, where the heuristic function is either in the form of a *rollout*, using a *rollout policy* TODO: cite papers that do this, including some that just call it UCT, or use a function V_θ that aims to approximate the true optimal value function V^* from Equation TODO: ref TODO: cite papers that do this.

These algorithms follow the same

These algorithms use the similar value functions to UCT, $\bar{V}_{\text{MCTS}}$ or $\bar{Q}_{\text{MCTS}}$.

The search policy corresponds to the UCT search policy, using the new $\bar{Q}_{\text{MCTS}}$ values:

$$\pi_{\text{MCTS}}(s) = \arg\max_{a \in \mathcal{A}} Q_{\text{MCTS}}(s,a) + b_{\text{MCTS}} \sqrt{\frac{\log(N(s))}{N(s,a)}} \tag{2.29}$$

TODO: add labels for equations

A trajectory $\tau_{:h}$ is sampled until a new decision node not in the tree is reached, as we are now running with `mcts_mode`set to True.

If the algorithm uses a function approximation V_θ, then it is used directly for the heuristic function V_{init}. If a rollout is used for the heuristic function, then the algorithm needs to define a *rollout policy* π_{rollout}, which is used to sample a Monte Carlo estimate of the value function $V^{\pi_{\text{rollout}}}$ as follows. The sampled trajectory $\tau_{:h} \sim \pi_{\text{MCTS}}$ is extended with the rollout trajectory $\tau_{h:H} \sim \pi_{\text{rollout}}$ to give the Monte Carlo estimate of the value at s_h:

$$V^{\pi_{\text{rollout}}}(s_h) \approx \sum_{i=h}^{H-1} r_i. \tag{2.30}$$

Letting $\tilde{r} = V_{\text{init}}(s_h)$, the value estimates (or sample averaqes) are updated as follows:

$$\bar{Q}_{\text{MCTS}}(s_t, a_t) \leftarrow \frac{1}{N(s,a)} \left((N(s,a)-1)\bar{Q}_{\text{MCTS}}(s_t,a_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i \right) \tag{2.31}$$

$$\bar{V}_{\text{MCTS}}(s_t) \leftarrow \frac{1}{N(s,a)} \left((N(s,a)-1)\bar{V}_{\text{MCTS}}(s_t,a_t) + \tilde{r} + \sum_{i=t}^{h-1} r_i \right) \tag{2.32}$$

TODO: add labels for equations

2.3.4 Maximum Entropy Tree Search

TODO: list

- Define MENTS here

TODO: I'm feeling ill writing this section, so just going to word vomit this shit out and make it sound not shit later

Maximum ENtropy Tree Search (MENTS) TODO: cite, in contrast to UCT, focuses on the maximum-entropy objective. 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

$$\pi_{\text{MENTS}}(a|s) = (1-\lambda_s)\exp\left(\frac{1}{\alpha_{\text{MENTS}}}\left(\hat{Q}_{\text{MENTS}}(s,a) - \hat{V}_{\text{MENTS}}(s)\right)\right) + \frac{\lambda_s}{|\mathcal{A}|}, \quad (2.33)$$

TODO: add labels for equations where α_{MENTS} is the temperature paramter used for Equation TODO: ref in MENTS, and $\lambda_s = \min(1, \epsilon/\log(e + N(s)))$, with $\epsilon \in (0, \infty)$ is an exploration parameter.

The value estimates are updated using the soft Bellman backups (TODO: ref) as follows:

$$\hat{Q}_{\text{MENTS}}(s_t, a_t) \leftarrow R(s_t, a_t) + \sum_{s' \in \text{Succ}(s,a)} \left(\frac{N(s')}{N(s_t, a_t)} \hat{V}_{\text{MENTS}}(s')\right), \quad (2.34)$$

$$\hat{V}_{\text{MENTS}}(s_t) \leftarrow \alpha \log \sum_{a \in \mathcal{A}} \exp\left(\frac{1}{\alpha} \hat{Q}_{\text{MENTS}}(s_t, a)\right). \quad (2.35)$$

TODO: add labels for equations

TODO: talk about initialistations, for Vinit its the same as MCTS, think about how to integrate Qinit properly into the thesis using below stuff (commented out)

TODO: make sure neurips paper writing integrated (commented out, and below this comment in the .tex) - read the actual pdf for thesis and neurips papers and compare cover same info?

THINGS WROTE before for THTS section TODO: read through and see if anything want to keep, otherwise delete

In THTS++ we run trials for either some fixed number of trials n, or some time limit T. Each trial consists of three steps: (1) sample a context, which is used to store variables that are associated with a specific trial, and is passed to the following three functions; (2) selection, which samples states, actions and outcomes for the trial, corresponding to a path down the tree; (3) initialisation, which creates any new nodes in the tree and initialises their values; (4) backup, which updates values at all nodes visited on the trial.

Decision nodes follow the interface:

```
1 class DNODE:
2     # children : dictionary[A] -> DNODE
3     def initialise(state (s_t), depth (t), context)
4     def select_action(context)
5     def backup(trial_return (R_t), context)
```

And chance nodes:

```
1 class CNODE:
2     # children : dictionary[S] -> DNODE
3     def initialise(state (s_t), action (a_t), depth (t), context)
4     def sample_outcome(context)
5     def backup(trial_return (R_t), context)
```

The `run_trial`function can be written as:

```
1 def run_trial:
2     # root_node : DNODE
3     # mcts_mode : bool
4     t = 0
5     state = root_node.state
6     while (not selection_phase_ended(t,mcts_mode)):
7
8 def selection_phase_ended(t,mcts_mode):
9     if
```

urgh BRAIN POOP

TODO - copy the descriptions from DENTS, and adapt and add the psuedocode

2.4 Multi-Objective Reinforcement Learning

TODO: list

- MOMDP definition
- (Expected) utility
- Define an interface for pareto front and convex hull objects
- Define CHVI
- Should talk about multi-objective and/or contextual multi-armed bandits here?
- I'm planning on aligning this section with the recent MORL survey [1]
- Mention some deep MORL stuff, say that this work (given AlphaZero) is adjacent work

TODO: Follow CHMCTS and https://arxiv.org/abs/2103.09568

In this thesis we will follow a utility based approach to Multi-Objective Reinforcement learning similar to TODO: cite. For a full review of Multi-Objective Reinforcement Learning see TODO: cite. In this work we will specifically consider *linear utility* functions and the decision support scenario, which will be defined more precisely below.

This section defines the multi-objective conterparts to various definitions found in TODO: ref.

To specify problems with multiple objectives, the reward function of an MDP now outputs a vector of rewards, rather than a scalar reward.

Definition 2.4.1. *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,* $\mathbf{V}(s, a)$ *is a 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. Outside of this section, we may drop the prefix "multi-objective" where it should be clear from context, however we will continue to use bold typeface to denote any vector variables or functions.

Definition 2.4.2. *A* multi-objective trajectory*, is a sequence of state, action and rewards, that is induced by a policy π and MOMDP $\boldsymbol{\mathcal{M}}$ pair. Let the trials/-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 $s_{t+1} \sim \text{Succ}(s_t, a_t)$. Notationally, we will write $\tau \sim \pi$ to denote a sampled trial/trajectory with respect to a policy, where the MOMDP is implicit.*

Sometimes it will be necessary to reason about trajectories with a horizon $h < H$, which will be denoted $\tau_{:h} = (s_0, a_0, \mathbf{r}_0, s_1, a_1, \mathbf{r}_1, ..., s_{h-1}, a_{h-1}, \mathbf{r}_{h-1}, s_h)$.

Similarly, multi-objective variants of the value of a policy needs to be defined:

Definition 2.4.3. *The* value *of a policy π from state s at time t is:*

$$\mathbf{V}^\pi(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} \mathbf{r}_t \middle| s_t = s\right]. \tag{2.36}$$

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

$$\mathbf{Q}^\pi(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim\text{Succ}(s,a)}[\mathbf{V}^\pi(s';t+1)]. \tag{2.37}$$

Now, in the corresponding single-objective section TODO: ref we have reached the point where we defined the optimal (Q-)value functions and the objective of single-objective reinforcement learning, where the maximum is taken over all possible policies. 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". To resolve this issue, a *utility function* or *scalarisation function* is used to map multi-objective values to scalars.

Definition 2.4.4. *The* (D-dimensional) Simplex *consists of the set of D-dimensional vectors, whose entries are non-negative and sum to one. We denoted this as $\Delta^D = \{\mathbf{w} \in \mathbb{R}^D | w_i > 0, \sum_i w_i = 1\}$.*

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

Definition 2.4.5. *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* $u(\mathbf{v}; \mathbf{w})$.

TODO: 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

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

Definition 2.4.6. *The* linear utility function u_{lin} *is the utility function defined by:*

$$u_{\text{lin}}(\mathbf{v}; \mathbf{w}) = \mathbf{w}^\top \mathbf{v}. \tag{2.38}$$

Equiped with a scalarisation function and a weight vector any set of multi-objective values can be ordered. Letting Π be the set of all possible policies we can now define solution sets.

Definition 2.4.7. *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 scalarised value is maximised:*

$$U(\Pi; u) = \left\{ \pi \in \Pi \;\middle|\; \exists \mathbf{w} \in \Delta^D . \forall \pi' \in \Pi : u(\mathbf{V}^\pi(s_0; 0); \mathbf{w}) \geq u(\mathbf{V}^{\pi'}(s_0; 0); \mathbf{w}) \right\}. \tag{2.39}$$

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

TODO: write something here?

TODO: change d to D for the dimension, going to use d for a distance function

Definition 2.4.8. *A set* $CS(\Pi; u) \subseteq \Pi$*, is a* coverage set *with respect to a utility function* u*, if for every weight vector* $\mathbf{w} \in \Delta^D$*, there is a policy* $\pi \in CS(\Pi; u)$ *that*

maximises the value of $u(\cdot; \mathbf{w})$. *That is, for* $CS(\Pi; u)$ *to be a coverage set, the following statement must be true:*

$$\forall \mathbf{w} \in \Delta^D . \exists \pi \in CS(\Pi; u) . \forall \pi' \in \Pi : u(\mathbf{V}^{\pi}(s_0; 0); \mathbf{w}) \geq u(\mathbf{V}^{\pi'}(s_0; 0); \mathbf{w}). \quad (2.40)$$

It can be shown that any $CS(\Pi; u) \subseteq U(\Pi; u)$ TODO: cite?.

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}.

TODO: some comments about undominated sets often being an infite set of policies, but coverage sets often being finite and more feasible to compute

TODO: some comment about often computing the value set:

Definition 2.4.9. *The* (multi-objective) value set *with respect to a set of policies* $\Pi' \subseteq \Pi$ *is defines as:*

$$\boldsymbol{\mathcal{V}}(\Pi') = \{\mathbf{V}^{\pi}(s_0; 0) | \pi \in \Pi'\}. \quad (2.41)$$

TODO: Should acknowledge some things. Often we actually compute the value set of a convex coverage set. Often we compute a very specific convex coverage set, which is the extreme points of the convex hull. Also that the term convex hull is typically used to refer to any of the previous three sets (value set, convex coverage set, convex hull). And finally, say that often with methods that compute the value set can often use tagging to compute the policies after the fact, and cite some of the pomdp algorithms from LPK that actually explain the tagging

Additionally, this thesis focusses on the decision support scenario as outlined in TODO: cite, with a linear utility function. In the decision support scenario the true weight vector is unknown, and so the objective is to compute a convex coverage set. When a convex coverage set is produced, it is then provided to a user that picks their most preferred policy or value from the coverage set. After this policy is selected, it can be used as a single solution to the problem that the user was trying to solve.

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.

TODO: make the above couple paragraphs not read like poo, maybe chatgpt it

2.4.1 Convex Hull Value Iteration

TODO: label for subsection

Convex Hull Value Iteration (CHVI) TODO: cite is a tabular dynamic programming algorithm similar to Value Iteration TODO: ref. In CHVI the value functions of value iterations are replaced by sets of vectors, which represent convex hulls (really they are the value set of a convex coverage set).

CHVI maintains estimates of convex hulls (actually convex coverage sets) at each state $\hat{\boldsymbol{\mathcal{V}}}_{\text{CHVI}}(s)$.

One important operation for CHVI is `cvx_prune`, which returns that undominated set of vectors from a given set of vectors $\boldsymbol{\mathcal{V}}$:

$$\texttt{cvx\_prune}(\boldsymbol{\mathcal{V}}) = \{\mathbf{v} \in \boldsymbol{\mathcal{V}} | \exists \mathbf{w} \in \Delta^D . \forall \mathbf{v}' \in \boldsymbol{\mathcal{V}} - \{\mathbf{v}\} . \mathbf{w}^\top \mathbf{v} > \mathbf{w}^\top \mathbf{v}'\}. \tag{2.42}$$

TODO: handle overloading of caligraphic V

Additionally, to be able to define a multi-objective version of value iteration, we need to define an arithmetic over sets of vectors. Given the sets of vectors $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ we define multiplication by a scalar s and addition as follows:

$$s\boldsymbol{\mathcal{V}} = \{s\mathbf{v} | \mathbf{v} \in \boldsymbol{\mathcal{V}}\} \tag{2.43}$$

$$\boldsymbol{\mathcal{U}} + \boldsymbol{\mathcal{V}} = \{\mathbf{u} + \mathbf{v} | \mathbf{u} \in \boldsymbol{\mathcal{U}}, \mathbf{v} \in \boldsymbol{\mathcal{V}}\}. \tag{2.44}$$

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

$$\hat{\boldsymbol{\mathcal{V}}}^{k+1}_{\text{CHVI}}(s;t) = \texttt{cvx\_prune}\left(\bigcup_{a \in \mathcal{A}} \hat{\boldsymbol{\mathcal{Q}}}^{k+1}_{\text{CHVI}}(s,a;t)\right), \tag{2.45}$$

$$\hat{\boldsymbol{\mathcal{Q}}}^{k+1}_{\text{CHVI}}(s,a;t) = \mathbb{E}_{s' \sim \text{Succ}(s,a)}[\mathbf{R}(s,a) + \hat{\boldsymbol{\mathcal{V}}}^{k}_{\text{CHVI}}(s';t+1)]. \tag{2.46}$$

This again parallels the Bellman backups use in single-objective value iteration TODO: ref, where the max operation is replaced by the `cvx_prune`operation over all possible Q-value vectors. TODO: write this sentence better

TODO: talk about the POMDP action tagging things

TODO: Talk about the better way of doing CHVI? https://www.jmlr.org/papers/volume13/lizot and Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. Also by Lizotte

2.5 Sampling From Catagorical Distributions

TODO: list

- Talk about the alias method here
- Reference to chapter 4 section where talk about using this with THTS

TODO: clean this up generally, wrote it in a rush. Also trying not to use notation that I may want to use later. Would like

Much of the work in this thesis will involve sampling from catagorical distributions. Let $f : \{1, ..., m\} \to \mathbb{R}$ be the probability mass function of a catagorical distribution with m categories. Suppose that we want to sample $i \sim f$. A naive method to sample from f will take $O(m)$ time, where a value is sampled from Uniform$(0, 1)$ is often used as follows:

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

However, the *Alias method* TODO: cite1, cite2 can instead be used, with $O(m)$ preprocessing time to construct an *Alias table*, and can sample from f in $O(1)$ time. In Figure TODO: ref we provide an example of an alias table. A value can be sampled

using the alias table by sampling two random numbers, one from Uniform($\{1, ..., m\}$) and one from Uniform$(0, 1)$. To sample from the alias table, one of the entries is sampled uniformly randomly using the sample from Uniform($\{1, ..., m\}$), each entry in the table contains three values, `threshold`, `cat_one` and `cat_two`, from which if we let $a \sim$ Uniform$(0, 1)$, we would then return `cat_one` if $a <$`threshold` and `cat_two` otherwise. Psuedocode for this is as follows:

```
def sample_from_alias_table(alias_table):
    index ~ Uniform({1, ..., m})
    cat_one, cat_two, threshold = alias_table[index]
    a ~ Uniform(0, 1)
    if (a < threshold):
        return cat_one
    return cat_two
```

In TODO: cite it is shown TODO: check that it's proven that an alias table can be constructed from an arbitrary probability mass function for a 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.

TODO: verify that the example alias table maintains the correct masses for each category

Following TODO: cite, we can construct an alias table as follows:

```
def build_alias_table(f):
    pass
```

TODO: write the build alias table psuedocode (use wiki it was good)

TODO: after finished chapter, make sure no errors from latex

TODO: after finished chapter, make sure all equations labelled

TODO: after finished chapter, make sure all sections correctly referenced, changed sec:2-5-sampling to sec:2-4-sampling, removed the sec:2-4-momcts, added sec:2-5-mabs

TODO: after finished chapter, make sure all abbreviations and accronyms added and correct

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

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)

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

3.3.1 Trial Based Heuristic Tree Search

TODO: THTS paper

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

3.3.3 Maximum Entropy Tree Search

TODO: MENTS

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 [1]
- Mention some deep MORL stuff, say that this work (given AlphaZero) is adjacent work

3.5 Multi-Objective Monte Carlo Tree Search

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

TODO: list

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

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

TODO: Write out implementations of prior works using THTS

TODO: define pareto front

TODO: perez algorithms

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

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

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

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

4

Monte Carlo Tree Search With Boltzmann Exploration

Contents

4.1 Introduction

TODO: list

- high level overview of DENTS work
- discuss how DENTS answers the research questions from introduction chapter
- state clearly that we're in single objective land here
- Comment about work exploring multi-armed bandits motivating this work

4.2 Boltzmann Search

TODO: list

- Recall MENTS
- Define BTS using THTS functions
- Define DENTS using THTS functions
- Discuss alias method variant (and complexity analysis) in a subsection?

4.3 Toy Environments

TODO: list

- Define D-chain stuff from the paper
- Define the D-chain with entropy trap
- Front load some results still

4.4 Theoretical Results

TODO: list

- add theoretical results

4.5 Empirical Results

TODO: list

- DChain
- GridWorlds
- Go

4.6 Full Results

TODO: there's a lot of figures for the D-chain environment, work out how to best fit them in? Or put them in this seperate section?

5 Convex Hull Monte Carlo Tree Search

Contents

5.1 Introduction

TODO: list

- high level overview of CHMCTS work
- discuss how CHMCTS answers the research questions from introduction chapter
- moving into multi-objective land now
- Comment about CHVI and prior MOMCTS work motivating this

5.2 Contextual Tree Search

TODO: list

- Discuss need for context when doing multi-objective tree Search
  - Use an example env where left gives (1,0) and right gives (0,1), optimal policy picks just left or just right, but hypervolume based methods wont
  - Use previous work on these examples and show they dont do well bad
- Discuss how UCT = running a non-stationary UCB at each node, so given above discussion, there is work in contextual MAB
- Introduce contextual regret here

5.3 Contextual Zooming for Trees

TODO: list

- Give contextual zooming for trees algorithm
- Discussion on the contextual MAB to non-stationary contextual MAB stuff (CZT is to CZ what UCT is to UCB) (and what theory carry over)

5.4 Convex Hull Monte Carlo Tree Search

TODO: list

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

5.5 Results

TODO: list

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

6

Simplex Maps for Multi-Objective Monte Carlo Tree Search

Contents

6.1 Introduction

TODO: list

- high level overview of simplex maps work
- discuss how simplex maps answer the research questions from introduction chapter
- staying in multi-objective land now
- Motivated by CHMCTS being slow

6.2 Simplex Maps

TODO: list

- Define simplex map interface
- Give details on how to efficiently implement the interface with tree structures
- (Good diagram is everything here I think)

6.3 Simplex Maps in Tree Search

TODO: list

- Come up with better title for section
- Use simplex maps interface to create algorithms from the dents work
- Give a high level idea of what δ parameter is (used in theory section)

6.4 Theoretical Results

TODO: list

- Convergence can build ontop of DENTS results
- Runtime bounds (better than $O(2^D)$ which is what using convex hulls has)
- Simplex map has a diameter δ (i.e. the furthest away a new context could be from a point in the map)
- Bounds can then come from that diameter (which is a parameter of the simplex map/algorithm) and DENTS results

6.5 Empirical Results

TODO: list

- Results from MO-Gymnasium
- Compare algorithms using expected utility

7 Conclusion

Contents

TODO: Something about we'll conclude by looking back at contributions and possible future work.

7.1 Summary of Contributions

TODO: go through each of the research questions and contributions, and write about how the work answers the research questions

7.2 Future Work

TODO: outline some avenues of potential future work

Appendices

A

List Of Appendices To Consider

- Multi Armed Bandits, maybe
- MAybe from of the things in background are more appropriate as appendices?

Bibliography

[1] 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.

[2] 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.

[3] Michael Painter. THTS++,https://github.com/MWPainter/thts-plus-plus.