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.

Markov Decision Processes (Section 2.1)

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.

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?

Trial Based Heuristic Tree Search (Section 2.2)

a_t . . . . . A random variable for the tth action sampled in a trial of `THTS++`.

n . . . . . Number of trials run.

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

r_t . . . . . A random variable for the tth reward sampled in a trial of `THTS++`.

s_t . . . . . A random variable for the tth state sampled in a trial of `THTS++`.

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 $\mathcal{T}_k$

τ . . . . . A trajectory for a trial of `THTS++`. I.e. $\tau = (s_0, a_1, s_1, ..., s_{h-1}, a_{h-1}, s_h)$. TODO: handle τ_k

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

List of Abbreviations

Markov Decision Processes and Reinforcement Learning (S

MDP . . . . . Markov Decision Process.

Trial Based Heuristic Tree Search (Section 2.2)

`CNODE` . . . . . The chance node class.

`cnode` . . . . . An instance of `CNODE`, i.e. a chance node instance.

`DNODE` . . . . . The decision node class.

`dnode` . . . . . An intance of `DNODE`, i.e. a decision node instance.

MCTS . . . . . Monte Carlo Tree Search.

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

MENTS . . . . Maximum ENtropy Tree Search.

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

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

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

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

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 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 2.1.1. *A* Markov Decision Process *(MDP) is a tuple* $\mathcal{M} = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{R}, p, H)$*, 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* $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$*,* $p(\cdot|s, a)$ *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 2.1.2. *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 2.1.3. *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, that is* $\pi(s) = a$ *where* $a \in \mathcal{A}$ *is such that* $\pi(a|s) = 1$.

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.

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 2.1.4. *A* trial *or* 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.*

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

Definition 2.1.5. *The* value *of a policy π from state s at time t is:*

$$V^\pi(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} r_t \middle| s_t = s\right]. \tag{2.1}$$

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

$$Q^\pi(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim\text{Succ}(s,a)}[V^\pi(s';t+1)]. \tag{2.2}$$

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

Definition 2.1.6. *The* Optimal (Q-)Value *of a state(-action pair) is defined as:*

$$V^*(s;t) = \max_\pi V^\pi(s;t) \tag{2.3}$$

$$Q^*(s,a;t) = \max_\pi Q^\pi(s,a;t). \tag{2.4}$$

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

Definition 2.1.7. *The* (standard) reinforcement learning objective function $J(\pi)$ *is defined as:*

$$J(\pi) = V^\pi(s_0;0). \tag{2.5}$$

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*:

$$V^*(s;t) = \max_{a \in \mathcal{A}} Q^*(s,a;t), \tag{2.6}$$

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

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:

$$V^{k+1}(s;t) = \max_{a \in \mathcal{A}} Q^{k+1}(s,a;t), \tag{2.8}$$

$$Q^{k+1}(s,a;t) = \mathbb{E}_{s' \sim \text{Succ}(s,a)}[R(s,a) + V^k(s';t+1)]. \tag{2.9}$$

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: add optimal policy from Q values

2.1.1 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:

$$\mathcal{H}(\pi(\cdot|s)) = \mathbb{E}_{a \sim \pi(\cdot|s)}[-\log \pi(a|s)]. \tag{2.10}$$

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

$$V^{\pi}_{\text{sft}}(s;t) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=t}^{H-1} r_t + \alpha\mathcal{H}(\pi(\cdot|s_i))\middle| s_t = s\right]. \tag{2.11}$$

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

$$Q^{\pi}_{\text{sft}}(s,a;t) = R(s,a) + \mathbb{E}_{s'\sim p(\cdot|s,a)}[V^{\pi}_{\text{sft}}(s';t+1)]. \tag{2.12}$$

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

Definition 2.1.9. *The* Optimal soft (Q-)Value *of a state(-action pair) is defined as:*

$$V^{*}_{\text{sft}}(s;t) = \max_{\pi} V^{\pi}_{\text{sft}}(s;t) \tag{2.13}$$

$$Q^{*}_{\text{sft}}(s,a;t) = \max_{\pi} Q^{\pi}_{\text{sft}}(s,a;t). \tag{2.14}$$

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

$$J_{\text{sft}}(\pi) = V^{\pi}_{\text{sft}}(s_0;0). \tag{2.15}$$

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*:

$$V^*_{\text{sft}}(s;t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(Q^*_{\text{sft}}(s,a;t)/alpha\right), \tag{2.16}$$

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

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

$$V^{k+1}_{\text{sft}}(s;t) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(Q^{k+1}_{\text{sft}}(s,a;t)/alpha\right), \tag{2.18}$$

$$Q^{k+1}_{\text{sft}}(s,a;t) = R(s,a) + \mathbb{E}_{s' \sim \text{Succ}(s,a)}[V^k_{\text{sft}}(s';t+1)]. \tag{2.19}$$

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

$$\pi^*_{\text{sft}}(a|s;t) = \exp\left(\left(Q^*_{\text{sft}}(s,a;t) - V^*_{\text{sft}}(s;t)\right)/\alpha\right). \tag{2.20}$$

2.1.2 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

2.2 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

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 2.2.2. 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 2.2.2 and 2.2.3 respectively.

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!")
```

2.2.1 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: To simplify notation we're going to assume that states and state-action pairs have a one to one correspondance with decision and chance nodes respectively. This is just to make notation clean, results generalse. AND were going to use `node` as a mapping from $\mathcal{S} \cup \mathcal{S} \times \mathcal{A}$, to `THTS++` decision and chance nodes. So $\texttt{node}(s_i)$ refers to the decision node associated with $\texttt{node}(s_i)$ and $\texttt{node}(s_i, a_{i+1})$ is a chance node. TODO: Basically want to say that I'll use it without the node bit, unless I want to distinguish between the node, the state and state-action pair.

TODO: Define some node stuff here, mabye define $\texttt{node}(s_i).V$ and so on

TODO: Add parameters of nodes in tables, and define the interfaces somewhere.

TODO: The initial tree is $\mathcal{T}_0 = \{\texttt{node}(s_0)\}$.

TODO: Define the value which could be a convex hull and so on. TODO: Going to have to make below consistent with this

An algorithm following the `THTS++` schema needs to define the following functions TODO: clean up this list:

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

Initialisation function: A function V_{init} 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, maybe just accept overloeaded notation?

In `THTS++` a search tree $\mathcal{T}$ is built using random trials. The kth trial is split into four phases:

Selection: A *search policy* is used to sample actions and the transition distribution p is used to sample successor states TODO: until new node not in tree, or some depth D. From this we get a trajectory for the trial $\tau = (s_0, a_1, s_1, ..., s_{h-1}, a_h, s_h)$;

Expansion: New nodes corresponding to the sampled trial trajectory are added to the tree: $\mathcal{T}_k = \mathcal{T}_{k-1} \cup \texttt{node}(\tau)$, where $\texttt{node}(\tau)$ is the set of nodes sampled in the trial (i.e. $\texttt{node}(\tau) = \{\texttt{node}(s_i)\}_{i=0}^{h} \cup \{\texttt{node}(s_i, a_{i+1})\}_{i=0}^{h-1}$);

Initialisation: If s_h is a new node added to the tree ($\texttt{node}(s_h) \notin \mathcal{T}_{k-1}$), then initialise the value of $\texttt{node}(s_h)$ using V_{init};

Backup: TODO: clean up value function notation. $V^k(s_t) = \mathcal{B}^V(\mathbf{Q}^k(s_t, \cdot))$ and $Q^k(s_t) = \mathcal{B}^Q(\mathbf{V}^k(\cdot))$.

THINGS WROTE before

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:

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

And chance nodes:

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

The `run_trial`function can be written as:

```
def run_trial:
    # root_node : DNODE
    # mcts_mode : bool
    t = 0
    state = root_node.state
    while (not selection_phase_ended(t,mcts_mode)):

def selection_phase_ended(t,mcts_mode):
    if
```

urgh BRAIN POOP

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

2.2.2 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?

2.2.3 Maximum Entropy Tree Search

TODO: list

- Define MENTS here

2.3 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

2.4 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

2.5 Sampling Random Variables

TODO: list

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

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?

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

3.6 Sampling Random Variables

TODO: Just citing the alias method papers?

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

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

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.