Online Partially Observable Markov Decision Process Planning via Simplification

Ori Sztyglic

Online Partially Observable Markov Decision Process Planning via Simplification

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Ori Sztyglic

Submitted to the Senate
of the Technion — Israel Institute of Technology
Tishrei 5782 Haifa October 2021

This research was carried out under the supervision of Associate Prof. Vadim Indelman, in the Faculty of Computer Science.

Some results in this thesis have been published or submitted as articles by the author and research collaborators in conferences and journals during the course of the author's research period, the most up-to-date versions of which being:

O. Sztyglic, A. Zhitnikov, and V. Indelman. Simplified belief-dependent reward mcts planning with guaranteed tree consistency. Technical report, 2021.

Ori Sztyglic and Vadim Indelman. Online pomdp planning via simplification. *arXiv preprint arXiv:2105.05296*, 2021.

Acknowledgements

I would like to thank my advisor Prof. Vadim Indelman for teaching the highest standards of how to do research. I would also like to thank my close family Meirav, Nitzan, Yoram and Roxana for being there for me all those years (and still are).

And finally I would like to thank my wife Yael who has been there for me daily.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Tables

Abstract

Partially Observable Markov Decision Process (POMDPs) are notoriously hard to solve. In this work we consider online planning in partially observable domains. Solving the corresponding POMDP problem is a very challenging task, particularly in an online setting. Our key contribution is a novel algorithmic approach, *Simplified Information Theoretic Belief Space Planning* (SITH-BSP), which aims to speed up POMDP planning considering belief-dependent rewards, without compromising on the solution's accuracy. We do so by mathematically relating the simplified elements of the problem to the corresponding counterparts of the original problem. Specifically, we focus on belief simplification and use it to formulate bounds on the corresponding original belief-dependent rewards. These bounds in turn are used to perform branch pruning over the belief tree, in the process of calculating the optimal policy. We further introduce the notion of adaptive simplification, while re-using calculations between different simplification levels, and exploit it to prune, at each level in the belief tree, all branches but one. Therefore, our approach is guaranteed to find the optimal solution of the original problem but with substantial speedup. As a second key contribution, we derive novel analytical bounds for differential entropy, considering a sampling-based belief representation, which we believe are of interest on their own. We validate our approach in simulation using these bounds and where simplification corresponds to reducing the number of samples, exhibiting a significant computational speedup while yielding the optimal solution. Finally, we embed the paradigm of simplification into the MCTS algorithm. In particular, we present *Simplified Information-Theoretic Particle Filter Tree* (SITH-PFT), a novel variant to the MCTS algorithm that considers information-theoretic rewards but avoids the need to calculate them completely. Our approach is general; namely, any converging to the reward bounds can be easily plugged-in to achieve substantial speedup without any loss in performance.

Abbreviations and Notations

BSP	:	Belief Space Planning
POMDP	:	Partially Observable Markov Decision Process
MDP	:	Markov Decision Process
MCTS	:	Monte Carlo Tree Search
PFT	:	Particle Filter Tree
DPW	:	Double Progressive Widening
DEPOST	:	Determinized Sparse Partially Observable Tree
POMCP	:	Partially Observable Monte Carlo Planning
IPFT	:	Information Particle Filter Tree
SITH	:	Simplified Information Theoretic
UCB	:	Upper Confidence Bound
KF	:	Kalman Filter
KDE	:	Kernel Density Estimation
$\mathcal{X}$	:	State space
$\mathcal{A}$	:	Action space
$\mathcal{T}$	:	Probabilistic transition model
$r(\cdot)$	:	Reward function (over state or belief)
$\rho(\cdot)$	:	Reward function over the belief
$\mathcal{Z}$	:	Observation space
$\mathcal{O}$	:	Probabilistic observation model
$b[x]$	:	Belief - Posterior distribution over the state x
b_k	:	Belief at time index k
b_0	:	Initial/Prior belief (time index $k = 0$)
h_k	:	History of all actions observations and prior belief up to time k
η	:	Normalization term
x_k	:	State at time index k
a_k	:	Action at time index k
z_k	:	Observation at time index k
π_k	:	Policy at time index k that maps a belief to an action
$\square_{k:k+L}$	:	Sequence from time index k to time index $k + L$
π_{k+}	:	Policy sequence from time index k to the predefined horizon

$J(b_k, \pi_{k+})$	:	Objective (Value) function over the belief at time k given a policy
$J^\star(b_k)$	:	Optimal objective function
$\pi^\star$	:	Optimal Policy
$V^\pi(b_k)$	:	Value (objective) function over the belief at time k given a policy
$Q^\pi(b_k, a_k)$	:	Belief-Action value function given a policy
γ	:	Discount factor
$\mathcal{H}(\cdot)$	:	Differential Entropy
$\hat{\mathcal{H}}(\cdot)$	:	Differential Entropy approximation
λ	:	Information weight constant
c	:	UCB exploration constant
$\square^x, \square^I$	:	multi objective split to over the state and information respectively
$N(h)$	:	Visitation count over some belief node
$N(ha)$	:	Visitation count over some belief-action node
b^s	:	Simplified belief
lb or ℓ	:	Lower bound over the reward
ub or u	:	upper bound over the reward
$\mathcal{LB}$, $\mathcal{UB}$	:	Lower and upper bounds over cumulative reward functions
$\mathbb{T}$	:	Belief tree
s_i	:	Simplification level i
s^l	:	Simplification level of the lth belief node in the belief tree
m	:	Number of particles approximating the non parametric belief
n_z	:	Number of observations acquired from some belief action node
$\overline{\Delta}^s$, $\underline{\Delta}^s$	:	Distance between reward and it's corresponding upper and lower bounds
	:	w.r.t simplification level s
A^s	:	Set of particles indexes w.r.t simplification level s
w^i	:	Weight of particle i
N	:	Number of particles in the particle set representing the belief
$\underline{\text{UCB}}, \overline{\text{UCB}}$	:	Lower and upper UCB bounds
$g(ha)$	:	Bounds difference w.r.t belief action node ha
d	:	Depth of some tree node (root depth is maximal i.e. maximal height d_{max})
$\mathcal{N}(\cdot, \cdot)$	:	Gaussian distribution
Σ	:	Covariance matrix

Chapter 1

Introduction

1.1 Planning under uncertainty

In the world of autonomous agents operating in an uncertain environment, a Partially Observable Markov Decision Process (POMDP) provides a principled mathematical framework for planning under uncertainty. Solving a POMDP is proven to be PSPACE-Complete Papadimitriou and Tsitsiklis [1987] giving rise to many algorithms trying to approach the optimal solution without having to solve the entire problem. This difficulty is more keenly felt when considering an online-setting of such autonomous tasks, i.e., when a robot has few seconds in every time step to execute the action it deems to be 'optimal'. In a partially observable setting, the robot maintains a *belief*, a posterior distribution over the state of interest, given actions and observations history the robot has executed and gathered so far. At each planning session, given this belief the robot determines the optimal policy (or action sequence) by constructing and traversing a belief tree, as illustrated in Fig. 1.1, which models how the POMDP can evolve into the future considering some finite horizon of time steps. When constructing the tree in planning time, the tree branches when taking an action and again when acquiring an observation.

This setting presents two main difficulties. The first is the *curse of dimensionality*. The size of the state space grows exponentially with the number of state variables and correspondingly so does the belief. The second is the *curse of history*. Planning into the future requires building the belief tree, which grows exponentially with the action and observation spaces. These two problems gave rise to many works trying to reduce computation time when solving the POMDP such as Silver and Veness [2010] and Smith and Simmons [2004]. In recent years many works began to used POMDPs to model more realistic problems, such as continuous or huge observation and action spaces e.g. Sunberg and Kochenderfer [2018], Garg et al. [2019], Lim et al. [2020b]. Another example is works considering information-theoretic rewards (e.g. Fischer and Tas [2020]). Information gathering rewards has proven to be extremely beneficial in various robotics tasks such as exploration, efficient sensing etc. Stachniss et al. [2005], Singh et al. [2009].

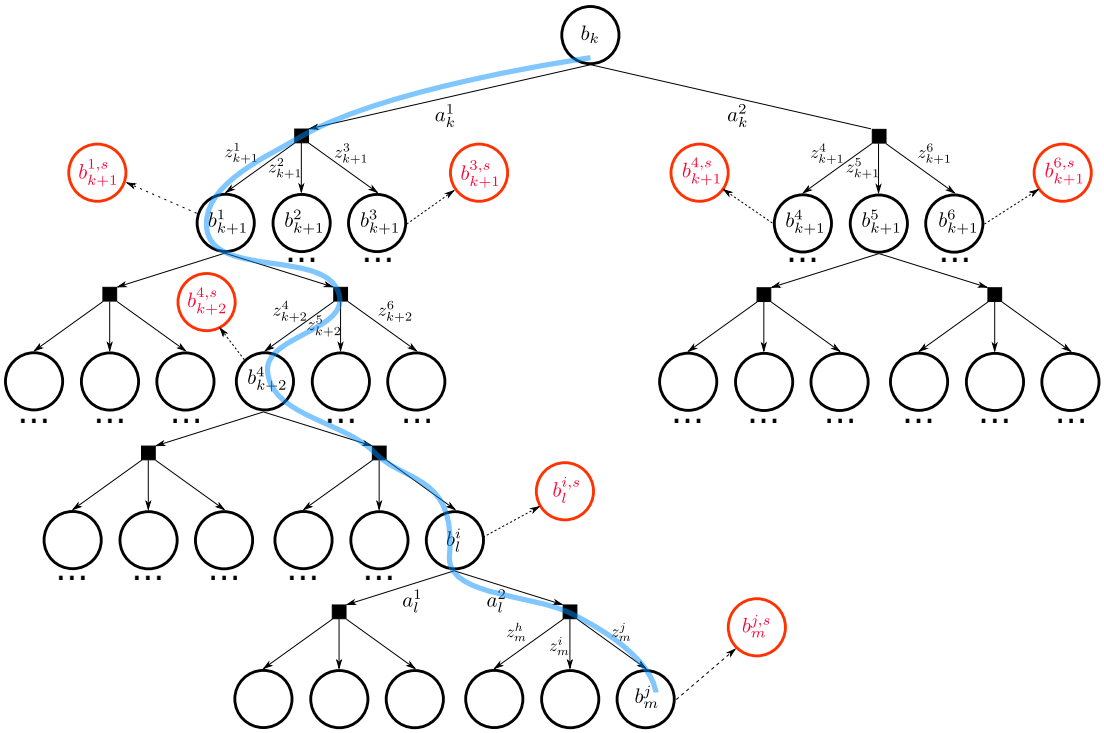


Figure 1.1: A belief tree is built starting from the current time belief b_k (the root). Tree branches when considering different future actions and observations. Round nodes represent belief nodes and square nodes represent belief-action nodes. For each belief node, b in a given belief tree a simplified belief b^s is calculated and used to formulate bounds over the corresponding belief-dependent reward. These bounds are then used to prune branches while calculating the optimal policy.

However, they can also be very costly (computationally speaking), and accounting for each node in the belief tree the problem quickly becomes even more intractable.

In this work we consider this challenging setting of continuous state and observation spaces, and the use of information-theoretic rewards. We utilize a notion called *Simplification*, and show how it can be a *Complementary* method to POMDP planning algorithms, speeding them up. We present novel mathematical simplification-based derivations that form a new fundamental way suggesting how to speed up calculations when planning is done by traversing the belief tree. Further, we consider non-parametric, sample-based belief representation and we derive novel analytical bounds for the particle-based differential entropy approximation. We verify that our approach can yield significant speedup via experimenting on a continuous state and observation setting and using the differential entropy approximation as a reward function.

1.2 Related Work

Accurately solving huge (or continuous) state and observations spaces POMDP is time-consuming. Early methods, tackling MDPs with huge state space, such as Kearns et al. [2002] build the belief tree up to a predetermined planning horizon. Next, they choose the optimal action at the root by utilizing the Bellman operator Bellman et al. [1957]

from the leaves up to the root of the tree, updating needed estimations along the way. The purpose is to avoid iterating the entire belief space and only consider belief elements that are achievable via sampled actions-observations sequences executed from the root of the tree. However, building the tree in full is still highly expensive. Silver and Veness [2010] introduced POMCP, an algorithm that applies Monte Carlo Tree Search over the POMDP's equivalent Belief-MDP. This method and its numerous expansions (Sunberg and Kochenderfer [2018], Hoerger et al. [2019], Lim et al. [2020b] etc.) avoid building the full belief tree. They do so by building it incrementally, exploring only the "promising" parts. Commonly, they make use of a strategy to balance exploration and exploitation, such as UCB Kocsis and Szepesvári [2006]. However, most of these algorithms are not suitable for information-theoretic rewards since they require the belief to be represented as a complete set of state samples commonly denoted as 'particles'. This last demand can be hardly met throughout the tree when the observation space is continuous because they (the mentioned algorithms) each time simulate only a single state sample (particle). Exceptions are PFT-DPW Sunberg and Kochenderfer [2018] and IPFT Fischer and Tas [2020]. These algorithms represent the belief nodes as a set of particles and each time a belief node is added to the tree, they propagate all the particles using a 'particle filter', which is a common sampling-based approach that can be used to preform belief updates given current belief, action and observation. Hence, the name Particle Filter Tree.

Another paradigm meant to speed up planning is the use of upper and lower bounds throughout the tree nodes Kochenderfer et al. [2022]. The gap induced by the bounds is used as a heuristic to choose "promising" sections of the tree to expand and when the bounds at the root are close enough they serve as stopping criteria to the algorithm. Smith and Simmons presented HSVI Smith and Simmons [2004], which is an early seminal work that makes use of bounds over the belief tree in the context of POMDP planning. However, their approach is not suitable for the setting we consider in this work. A notable assumption they make is that the observation space is finite where we assume it is infinite. Ye et al. introduced DESPOT Ye et al. [2017], which is one of the state-of-the-art algorithms for POMDP planning. It uses MCTS, utilizes upper and lower bounds, and adds regularizations to avoid over-fitting while planning. However, this work is also inadequate for information-theoretic rewards since again, like others, it simulates a single particle at a time which can lead to belief nodes containing a single particle. Following work, DESPOT-α Garg et al. [2019], indeed simulates a complete set of particles that potentially can open the door to information-theoretic rewards. However, the unique exploration strategy of this algorithm makes use of α-vectors which assumes a specific rewards structure that is not general enough and does not settle with information-theoretic rewards.

It is worth mentioning additional methods such as belief compression that can speed up planning Roy et al. [2005]. Yet, these kinds of methods are driven by error minimization such as belief representation error induced by the relaxation they carry out. In turn, these errors may result in a sub-optimal solution. On the other hand,

simplification Elimelech and Indelman [2021], Shienman et al. [2021] strives to perform relaxation of the decision-making problem while assuring the same solution as the original (non-relaxed) problem. When it is not feasible, the potential objective error (of executing one action over the optimal one) is bounded as part of the simplification scheme.

Finally, incorporation of information-theoretic reward into POMDPs is a long standing effort. Araya et al. [2010] were the first to consider rewards over the belief for POMDP planning. They introduced ρ-POMDP and extended the exact α-vectors method and a family of point based approximation algorithms to considering convex belief-dependent reward functions. However their formulation does not suggest how to deal with continuous state and observation spaces. Following work Fehr et al. [2018] extended their work further to Lipschitz-continuous reward functions and provide an HSVI-like algorithm which is as mentioned not suitable for continuous spaces. Thomas et al. [2021] extended POMCP such that it can handle belief dependent rewards (including information-theoretic ones) but their convergence proof does not hold when considering continuous action or observation spaces. Additional attempts such as Dressel and Kochenderfer [2017] were tackling offline solvers.

1.3 Contributions

We derive a new algorithmic approach, *Complementary* to existing POMDP planning algorithms. The approach is meant to speed up planning when considering information-theoretic rewards and continuous state and observation spaces. Our method follows the general sparse sampling planning scheme and thus lays the foundations to expanding it to additional planning techniques. Further, to demonstrate our approach we derive novel bounds over the particle-based approximation of the differential entropy. The bounds are easy to calculate, converge to the actual entropy approximation on-demand, and can be efficiently updated incrementally. Subsequently, our approach demonstrates substantial speedup while securing an identical to the baseline solution.

Further we show how our approach can be Incorporated in a state-of-the-art MCTS POMDP planner and provide a novel algorithmic framework based on our converging bounds on the belief-dependent reward. This method is guaranteed to yield the same action and belief tree as the most general algorithm suitable for such belief-dependent rewards (PFT-DPW). Our approach is general; namely, any bounds that are converging to the reward can be easily plugged-in to achieve substantial speedup without any loss in performance.

Chapter 2

Background

2.1 POMDPs

We model the Partially Observable Markov Decision Process (POMDP) for the finite horizon case, as a 7-tuple: $M = (\mathcal{X}, \mathcal{A}, T, r, \mathcal{Z}, \mathcal{O}, b_0)$, where $\mathcal{X}, \mathcal{A}$ and $\mathcal{Z}$ are the state, action and observation spaces, respectively. $T(x, a, x') \triangleq \mathbb{P}(x' \mid x, a)$ is the probabilistic transition model from past state $x \in \mathcal{X}$ to state $x' \in \mathcal{X}$ via action $a \in \mathcal{A}$. $\mathcal{O}(x, z) \triangleq \mathbb{P}(z \mid x)$ is the observation model expressing the measurement likelihood $z \in \mathcal{Z}$ for a given state. b_0 is the initial belief we have on the state at planning time. The *belief* is a posterior distribution over the state given all actions and measurements so far. It can be updated recursively via Bayes rule as $b[x'] = \eta \int \mathbb{P}(z' \mid x')\mathbb{P}(x' \mid x, a)b[x]dx$, where η is a normalization constant. Let $h_k = \{b_0, a_0, z_1, ...a_{k-1}, z_k\}$ denote *history* of actions and observations obtained by the agent up to time instance k and the prior belief.

In this research we consider a belief-dependent reward function $r(b, a)$. It allows one to use information-theoretic costs such as (differential) entropy, information gain and mutual information, thereby reasoning about future posterior uncertainty within the decision making process.

We denote the posterior belief at planning time k as $b[x_k] \triangleq \mathbb{P}(x_k \mid a_{0:k-1}, z_{1:k})$. Further, we denote by π_{k+j} a policy for time step $k+j$, i.e. $\pi_{k+j}(b[x_{k+j}])$ determines the action a_{k+j}. Let $\pi_{k+} \triangleq \pi_{k:k+L-1}$ represent a sequence of policies for the entire planning horizon of L steps that starts at time instant k. To shorten notations, we shall also use in the sequel $\pi_{(k+j)+} \triangleq \pi_{k+j:k+L-1}$, as well as $b_{k+j} \triangleq b[x_{k+j}]$. When solving a POMDP, one is trying to find the optimal policy that maximizes the objective (value) function,

$$J(b_k, \pi_{k+}) = \mathop{\mathbb{E}}_{z_{k+1:k+L}} \{ \sum_{i=k}^{k+L-1} r(b_i, \pi_i(b_i)) + r(b_{k+L}) \}, \tag{2.1}$$

where $r(b_{k+L})$ is the terminal reward. We may also consider a more general reward structure of $r(b_i, b_{i-1}, a_i)$, which is required, for example, to support information-theoretic reward functions such as information gain, and a specific sampling-based

approximation of differential entropy Boers et al. [2010] that we shall use in Section 3.1.3. As earlier, action a_i is determined by $\pi_i(b_i)$. The optimal policy $\pi^\star_{k+} \triangleq \pi^\star_{k:k+L-1}$ and the corresponding objective function are given by

$$\pi^\star_{k+} = \arg\max_{\pi_{k+}} J(b_k, \pi_{k+}),\ \ J^\star(b_k) = \max_{\pi_{k+}} J(b_k, \pi_{k+}). \tag{2.2}$$

Further, the objective function (2.1) can be written recursively, i.e., the Bellman equation.

$$J(b_k, \pi_{k+}) = r(b_k, a_k) + \mathop{\mathbb{E}}_{z_{k+1}} \{J(b_{k+1}, \pi_{(k+1)+})\}. \tag{2.3}$$

2.2 Planning using reward bounds

Usually when planning into the future a planning tree, or a belief tree in the more general case, is built in some manner. Tree nodes represent the different future beliefs that were acquired by considering future actions and observations. Each such node induces some reward that can be calculated using the reward model which is given as part of the POMDP tuple. This tree approximates the expectation of cumulative future rewards given different possible policies. In order to decide which action should be taken at the root of the tree, rewards should be summed bottom up (leaves to root). This weighted summation for the different routes in the tree, is nothing but the objective function (2.1). Once the rewards are propagated up the tree, the action (at the root) that present greater future cumulative reward should be chosen, i.e. choose the most promising subtree of the original tree (illustration in Fig. 2.1a). Due to the recursive nature of (2.1), (2.3) this formulation is also recursive and is applied in each belief node of the belief tree. I.e., in each node we propagate up the action that has the biggest corresponding subtree cumulative reward. Thus we get the optimal policy.

A possible way to improve this setting is by bounding the tree branches. Meaning, each belief node b_{k+j} in the belief tree has child subtrees corresponding to the different actions that can be taken from b_{k+j}. Each child subtree has it's own upper and lower bound $\{\mathcal{LB}^m, \mathcal{UB}^m\}_{m=1}^{|\mathcal{A}|}$ that we somehow got. So, according to the bounds, when some actions (subtrees) seem to be less promising than their sibling action, we can avoid expanding this tree branch in the first place. An alternative way to speedup the process is eliminating existing branches (subtrees or actions) according to these bounds. It becomes possible when for two sibling subtrees m', m'' corresponding to two different actions, we get $\mathcal{LB}_{m'} > \mathcal{UB}_{m''}$ or $\mathcal{LB}_{m''} > \mathcal{UB}_{m'}$. E.g., in Fig. 2.1c the lower bound of π'''' is higher than all other policies upper bounds. However this becomes problematic if (a) the bounds are not cheaper to calculate than the original objective of some tree. (b) We cannot eliminate all actions but one since the bounds are not tight enough. E.g. in Fig. 2.1b one cannot say for sure that policy π'''' is better than policy π'' since the latter upper bound is higher than the former lower bound. As explained later in 2.3, when using incremental methods for planning, this straightforward scheme just explained can't

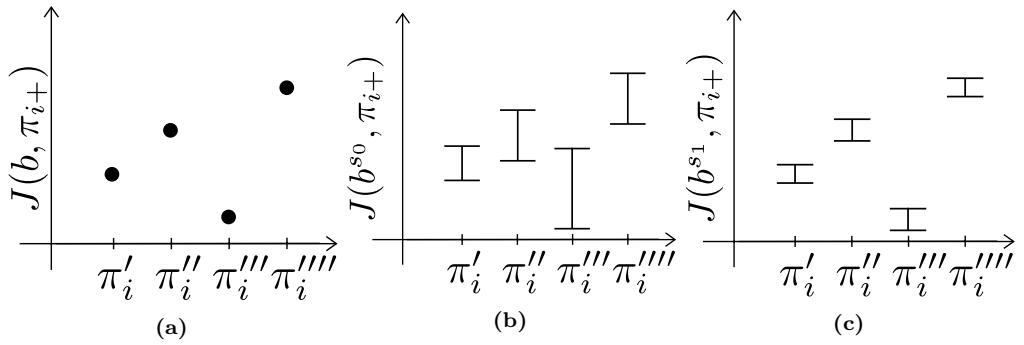


Figure 2.1: Action elimination using bounds. **(a)** Objective for some belief b and candidate policies; **(b)** Objective bounds given belief *coarse* abstraction; **(c)** Objective bounds given belief *fine* abstraction.

be carried out naively since it will output a different solution compared to the baseline (which in this work is PFT-DPW). One of our core novelties (see Sec. 3.2) shows how we can incorporate converging bounds to incremental methods without changing the baseline solution.

2.3 MCTS over Belief-MDP (PFT-DPW)

As mentioned before, the *curse of history* and the *curse of dimensionality* have been a great challenge in the context of POMDP planning. MCTS (Monte Carlo Tree Search) based algorithms tackle those problems by (a) building the belief tree incrementally and exploring only the "promising" parts of the tree, and (b) representing the belief as a fixed size set of weighted state samples (particles). The common practice to apply MCTS on partially observable environments is to convert POMDP to Belief-MDP, see e.g. Sunberg and Kochenderfer [2018]. Most of the MCTS based algorithms use the belief-action value function notation, also known as the Q-function, $V^{\pi^*}(b) = \arg\max_{a_k} Q^{\pi}(b_k, a_k)$, where

$$Q^{\pi}(b_k, a_k) = \mathop{\mathbb{E}}_{z_{k+1}} \{\rho(b_k, a_k, z_{k+1}, b_{k+1}) | b_k, a_k\} + \mathop{\mathbb{E}}_{z_{k+1}} \{V^{\pi^*}(b_{k+1}) | b_k, a_k\} \tag{2.4}$$

and where the notation of the objective (value function) is slightly different,

$$V^{\pi}(b_k) = \mathop{\mathbb{E}}_{z_{k+1:k+L}} \{ \sum_{i=k}^{k+L-1} \gamma^{i-k} \rho(b_i, \pi_i(b_i), z_{i+1}, b_{i+1}) | \pi, b_k \}. \tag{2.5}$$

We will use the notations interchangeably to settle with common MCTS litterateur notations.

In our generalized formulation, the reward $\rho(b_k, a_k, z_{k+1}, b_{k+1}) = r^x(b_k, a_k) + \lambda r^I(b_k, a_k, z_{k+1}, b_{k+1})$ is a function of two subsequent beliefs, an action and an ob-

servation. Specifically, our reward is

$$\rho(b_k, a_k, z_{k+1}, b_{k+1}) = \mathop{\mathbb{E}}_{x_k \sim b_k}\{r(x_k, a_k)\} - \lambda \hat{\mathcal{H}}(b_k, a_k, z_{k+1}, b_{k+1}), \tag{2.6}$$

where $r(x_k, a_k)$ is state and action dependent reward, and $r^x(b_k, a_k)$ is the expectation with regard to the state. $r^I(b_k, a_k, z_{k+1}, b_{k+1})$ is an information-theoretic reward, which in general can be dependent on consecutive beliefs and the elements relating them (e.g. information gain). $-\hat{\mathcal{H}}(b_k, a_k, z_{k+1}, b_{k+1})$ is an estimator of our information-theoretic reward weighted by λ. Yet, since such estimators do not commonly have a closed-form expression for non-parametric beliefs represented by a set of samples, one has to consider an estimator $\hat{\mathcal{H}}$ of $\mathcal{H}$ (e.g., Boers et al. [2010]). As shall be seen, our chosen estimator requires also previous belief b_k, chosen action a_k, and received observation z_{k+1}. Depending on the estimation method, the inputs can vary. Using the structure of (2.6),

$$Q(b_k, a_k) = Q^x(b_k, a_k) + \lambda Q^I(b_k, a_k), \tag{2.7}$$

where Q^x is induced by state dependent rewards and Q^I by the information-theoretic rewards. They are constituted by L elements of the form $\mathop{\mathbb{E}}_{x_i \sim b_i}\{r(x_i, a_i)\}$ and $-\mathcal{H}(b_i, a_i, z_{i+1}, b_{i+1})$, respectively. The Q^x element is easy to calculate, thus out of our focus, whereas the Q^I is computationally expensive to compute. From here on, for the sake of clarity, we will use the notation h and b interchangeably.

An inherent part of MCTS based algorithms is the Upper Confidence Bound (UCB) technique Kocsis and Szepesvári [2006] designed to balance exploration and exploitation while building the belief tree. This technique assumes that calculating the reward over the belief node does not pose any computational difficulty. Information-theoretic rewards violate this assumption. The algorithm constructs the policy tree by executing multiple simulations. Each simulation adds a single belief node to the belief tree or terminates by terminal state or action. To steer towards more deep and beneficial simulations, MCTS chooses action $a^\dagger$ at each belief node according to following rule

$$a^\dagger = \mathop{\arg\max}_{a \in \mathcal{A}} \text{UCB}(ha) \quad \text{UCB}(ha) = Q(ha) + c \cdot \sqrt{\frac{\log(N(h))}{N(ha)}}, \tag{2.8}$$

where $N(h)$ is the visitation count of belief node defined by the history h, $N(ha)$ is the visitation count of belief-action node defined by the history h and following action a, c is the exploration parameter and, $Q(ha)$ is the approximation of the belief-action value function Q for node ha obtained by simulations. When the action is selected, a question arises either to open a new branch in terms of observation and posterior belief or to continue through one of the existing branches. In continuous spaces, this is resolved by the Progressive Widening technique Sunberg and Kochenderfer [2018]. If a new branch is expanded, an observation o is created from state x drawn from the belief b.

Chapter 3

Approach

3.1 SITH-BSP a general planning with simplification scheme

3.1.1 Simplification

Simplification is any sort of relaxation of the POMDP tuple elements. In this work we consider a specific instantiation of this general simplification framework; namely, we suggest to use a simplified belief b^s to bound the reward instead of calculating it in full. Note, in this setting, simplification does not impact the distribution over which the expectation is taken. Thus, a given belief tree of the original problem corresponds also to the simplified problem. We assume the belief tree was built in some manner and it is given. This setting settles well with *Sparse Sampling* approaches Kearns et al. [2002] which are extremely general and widely used. Hence, we will derive our novel approach over a similar setting. Extensions to additional approaches such as MCTS are also presented in this research later on.

As mentioned, we aim to simplify the reward calculations. Namely, the original reward model is bounded using the simplified belief and takes the form

$$\mathbf{lb}(b^s, b, a) \leq r(b, a) \leq \mathbf{ub}(b^s, b, a), \tag{3.1}$$

where **lb** and **ub** are the corresponding lower and upper bounds, respectively. A key requirement is reduced computational complexity of these bounds compared to the complexity of the original reward. Furthermore, our formulation can be extended straightforwardly to support also information-theoretic rewards of the form $r(b_{i-1}, b_i)$, which involve two (consecutive) beliefs b_{i-1} and b_i, such as information gain. In such a case, the corresponding bounds would be

$$\mathbf{lb}(b^s_{i-1}, b^s_i, b_{i-1}, b_i) \leq r(b_{i-1}, b_i) \leq \mathbf{ub}(b^s_{i-1}, b^s_i, b_{i-1}, b_i). \tag{3.2}$$

In this section we formulate our approach considering the general form of the bounds (3.1). In Section 3.1.3 we derive novel bounds of the form (3.2).

Given a belief tree of the original problem M, Instead of calculating the computationally expensive reward $r(b,a)$ for each belief node b in this belief tree, we first calculate the corresponding simplified belief b^s, as illustrated in Fig. 1.1, and then formulate the bounds **lb** and **ub** from (3.2). Moreover, we can now traverse the belief tree from the leafs upwards and calculate recursively bounds over the objective (value) function at each node b_i via Bellman equation (2.3) as described below for $i \in [k, k+L-1]$.

$$\begin{aligned}\mathcal{UB}(b_i, \pi_{i+}) &= \mathbf{ub}(b_i^s, b_i, a)) + \mathop{\mathbb{E}}_{z_{i+1}} \{\mathcal{UB}(b_{i+1}, \pi_{(i+1)+})\} \\ \mathcal{LB}(b_i, \pi_{i+}) &= \mathbf{lb}(b_i^s, b_i, a) + \mathop{\mathbb{E}}_{z_{i+1}} \{\mathcal{LB}(b_{i+1}, \pi_{(i+1)+})\},\end{aligned} \tag{3.3}$$

with $\pi_{i+} = \{\pi_i, \pi_{(i+1)+}\}$ and $a = \pi_i(b_i) \in \mathcal{A}$, and where the expectation is taken with respect to $\mathbb{P}(\cdot \mid b_i, a)$, and the bounds are initialized at the terminal rewards (Lth time step in the planning horizon) as $\mathcal{LB}(b_{k+L}) = \mathbf{lb}(r^s(b_{k+L}))$ and $\mathcal{UB}(b_{k+L}) = \mathbf{ub}(r^s(b_{k+L}))$. This recursive procedure is common and practiced in many works (e.g. Ye et al. [2017]), yet a key difference is that our bounds (3.2) are obtained by relating the simplified POMDP elements to the original problem. Eq. (3.3) is a recursive update considering *some* trajectory down the belief tree determined by some policy π_{i+}. In contrast, we now consider upper and lower bounds for the optimal policy $\pi^\star_{i+}$. We denote these bounds as

$$\mathcal{UB}^\star(b_i) \triangleq \mathcal{UB}(b_i, \pi^\star_{i+}), \quad \mathcal{LB}^\star(b_i) \triangleq \mathcal{LB}(b_i, \pi^\star_{i+}). \tag{3.4}$$

Updating the bounds (3.4) is done recursively in two steps. First by considering the expansion of the already-calculated bounds $\mathcal{UB}^\star(b_{i+1})$ and $\mathcal{LB}^\star(b_{i+1})$ via (3.3) we have,

$$\begin{aligned}\mathcal{UB}(b_i, \{a, \pi^\star_{(i+1)+}\}) &= \mathbf{ub}(b_i^s, b_i, a) + \mathop{\mathbb{E}}_{z_{i+1}} \{\mathcal{UB}^\star(b_{i+1})\} \\ \mathcal{LB}(b_i, \{a, \pi^\star_{(i+1)+}\}) &= \mathbf{lb}(b_i^s, b_i, a) + \mathop{\mathbb{E}}_{z_{i+1}} \{\mathcal{LB}^\star(b_{i+1})\}.\end{aligned} \tag{3.5}$$

In practice, the expectation over observations is approximated by a parametric number of samples, n_z, which yields

$$\begin{aligned}\mathcal{UB}(b_i, \{a, \pi^\star_{(i+1)+}\}) &= \mathbf{ub}(b_i^s, b_i, a) + \frac{1}{n_z} \sum_l \mathcal{UB}^\star(b_{i+1}^l) \\ \mathcal{LB}(b_i, \{a, \pi^\star_{(i+1)+}\}) &= \mathbf{lb}(b_i^s, b_i, a) + \frac{1}{n_z} \sum_l \mathcal{LB}^\star(b_{i+1}^l)\end{aligned} \tag{3.6}$$

where superscript l is the belief node index corresponding to the z^l observation. The above is defined for each $a \in \mathcal{A}$.

Second, we perform branch pruning using Alg. 3.1 and as explained in Section 2.2: For each action $a \in \mathcal{A}$ we have corresponding bounds acquired via (3.6). For sufficiently tight bounds, all branches but one can be pruned (w.l.o.g. the branch corresponding to action $a^\star \in \mathcal{A}$). Thus, the bounds corresponding to action $a^\star$ hold: $\mathcal{UB}^\star(b_i) = \mathcal{UB}(b_i, \{a^\star, \pi^\star_{(i+1)+}\}), \mathcal{LB}^\star(b_i) = \mathcal{LB}(b_i, \{a^\star, \pi^\star_{(i+1)+}\})$, and $\pi^\star_{i+}(b_i) = \{a^\star, \pi^\star_{(i+1)+}\}$. As a

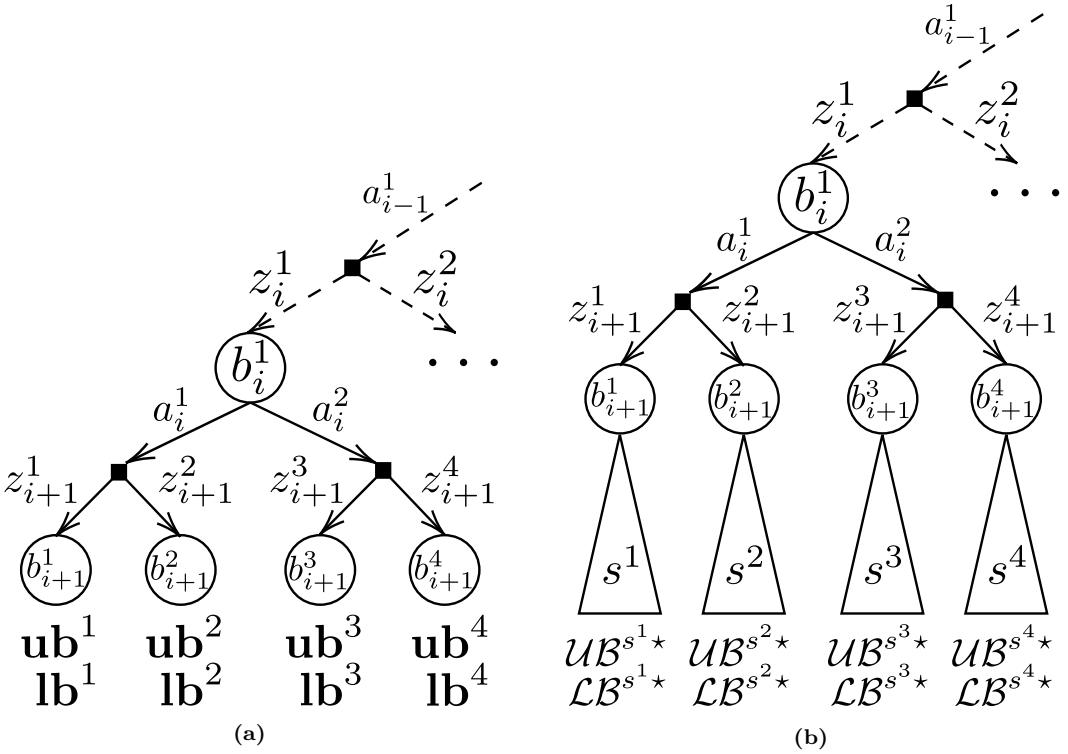


Figure 3.1: **(a)** Leaf nodes are bounded using (3.2). **(b)** Adaptive Simplification, Subtrees are bounded using (3.8) and Alg. 3.1.

consequence, we get upper and lower bounds on the *optimal* objective (value) function $J^\star(b_i)$,

$$\mathcal{LB}^\star(b_i) \leq J^\star(b_i) \leq \mathcal{UB}^\star(b_i), \tag{3.7}$$

and the optimal policy $\pi^\star_{i+}(b_i)$. See illustration in Fig. 3.1b.

Algorithm 3.1 Prune Branches

1: **procedure** PRUNE
2: **Input:** (belief-tree root, b; bounds of root's children, $\{\mathcal{LB}^m,\mathcal{UB}^m\}_{m=1}^C$) $\triangleright$ C is the number of child branches going out of b.
3: $\mathcal{LB}^\star \leftarrow \max_m \{\mathcal{LB}^m\}_{m=1}^C$
4: **for all** children of b **do**
5: **if** $\mathcal{LB}^\star > \mathcal{UB}^m$ **then**
6: prune child m from the belief tree
7: **end if**
8: **end for**
9: **end procedure**

Yet, this formulation presents a difficulty. It is generally not guaranteed that after using Alg. 3.1 we are left with a single branch in each belief node since the bounds might overlap (see illustration in Fig. 2.1b). We discuss how we overcome this difficulty in the next section.

3.1.2 Adaptive Simplification

We address the problem of bounds overlapping by extending the definition of simplification as we envision it to be an *adaptive paradigm*. We denote *level of simplification* as how 'aggressive' the suggested simplification is. Assume that the belief is represented by a set of samples (particles), as we do in Section 3.1.3. Taking a small subset of particles to represent the simplified belief corresponds to *coarse* simplification. Taking many of them will correspond to *fine* simplification. Naturally, with this setting, we can define many discrete levels. We denote subscript i for s_i as the simplification level, where s_0 and s_n correspond, respectively, to coarsest and finest simplification levels. Additionally we denote superscript j for s^j as the index corresponding to the belief's tree index. E.g. in Fig. 1.1 for tree node b^4_{k+1} the corresponding simplification index is s^4 and it may assume any value of simplification: $s^4 \in \{s_0, s_1, ...s_n\}$.

Further, in order to assure we can prune all branches but one, we assume bounds monotonically become tighter as the simplification level is increased and that the bounds for the finest simplification level s_n converge to the original problem. More formally denote $\overline{\Delta}^s(b,a) \triangleq \mathbf{ub}(b^s_i, b_i, a) - r(b_i, a)$ and $\underline{\Delta}^s(b,a) \triangleq r(b_i, a) - \mathbf{lb}(b^s_i, b_i, a)$.

Assumption 3.1.1. $\forall s \in [0, n-1]$ we get: $\overline{\Delta}^s(b,a) \geq \overline{\Delta}^{s+1}(b,a)$ and $\underline{\Delta}^s(b,a) \geq \underline{\Delta}^{s+1}(b,a)$.

Assumption 3.1.2. $\forall b_i, a$ we get: $\mathbf{ub}(b^{s_n}, b_i, a) = \mathbf{lb}(b^{s_n}, b_i, a) = r(b_i, a)$.

In the sequel we provide bounds that indeed satisfy these assumptions. A key question is how can we decide the appropriate level of simplification beforehand? We would like the coarsest level s_i that will enable eliminating actions/branches, i.e. lead us to the situation depicted in Fig. 2.1c and not Fig. 2.1b. In Alg. 3.2 our adaptive simplification approach is summarized. The general idea is to break down recursively a given belief tree $\mathbb{T}$ into its sub-problems (subtrees), denoted as $\{\mathbb{T}_m\}_{m=1}^{|\mathcal{A}|}$, and solve each sub-problem with its simplification level s_i. Ultimately this would lead to the solution of the entire problem via (3.6). A potential computational issue is that it may not be worth it to increase the simplification level repeatedly, since the overall time for all levels is suppressing the time it takes to solve the original problem. Fortunately, as discussed next, this is not an issue with our adaptive simplification.

Our adaptive simplification approach is based on two key observations. The *first key observation* is that we can compare bounds from *different* levels of simplification when pruning. Our *second key observation* is that we can re-use calculations between different simplification levels, and thus avoid re-calculating simplification from scratch. In the following sections, we elaborate on each of these crucial aspects.

Algorithm 3.2 Simplified Information Theoretic Belief Space Planning (SITH-BSP)

procedure FIND OPTIMAL POLICY(belief-tree: $\mathbb{T}$)
  $s \leftarrow s_0$
  return ADAPT SIMPLIFICATION($\mathbb{T}$,s)
end procedure
procedure ADAPT SIMPLIFICATION(belief-tree: $\mathbb{T}, s_i$)
  if $\mathbb{T}$ is a leaf **then**
    return $\{\mathbf{lb}, \mathbf{ub}\}$ ▷ Corresponds to immediate reward bounds over the leaf (3.1).
  end if
  Set simplification level: $s \leftarrow s_i$
  for all subtrees $\mathbb{T}'$ in $\mathbb{T}$ **do**
    ADAPT SIMPLIFICATION($\mathbb{T}'$,s)
    Calculate $\mathcal{LB}^{s^j}$,$\mathcal{UB}^{s^j}$ according to s and (3.8)
  end for
  Using $\{\mathcal{LB}^{s^j},\mathcal{UB}^{s^j}\}_{j=1}^{|\mathcal{A}|}$ and Alg. 3.1 prune branches
  while not all $\mathbb{T}'$ but 1 in $\mathbb{T}$ pruned **do**
    Increase simplification level: $s \leftarrow s+1$
    ADAPT SIMPLIFICATION($\mathbb{T}$,s)
  end while
  Update $\{\mathcal{LB}^{s^j\star},\mathcal{UB}^{s^j\star}\}$ according to (3.11)
  return optimal action branch that left $a^\star$ and $\{\mathcal{LB}^{s^j\star},\mathcal{UB}^{s^j\star}\}$.
end procedure

Comparing Bounds with Different Simplification Levels

Consider again some belief node b_i in the belief tree, and assume recursively for *each* of its children belief nodes b_{i+1} we already calculated the optimal policy $\pi^\star_{(i+1)+}(b_{i+1})$ and the corresponding upper and lower bounds $\mathcal{UB}^{s^l\star}(b_{i+1})$ and $\mathcal{LB}^{s^l\star}(b_{i+1})$, where s^l indicates the simplification level, and l corresponds to the belief tree nodes indexing notation. In general, the bounds for each belief node b_{i+1} can correspond to different simplification levels, as illustrated in Fig. 3.1b.

We now discuss how the simplification level is updated recursively, and revisit the process to calculate the optimal policy and the corresponding bounds for belief node b_i, previously described by Eqs. (3.6) and (3.7). Incorporating adaptive simplification, Eq. (3.6) is modified to

$$\begin{aligned}\mathcal{UB}^{s^j}(b_i,\{a_i^j,\pi^\star_{(i+1)+}\}) &= \mathbf{ub}(b_i^s,b_i,a_i^j)+\frac{1}{n_z}\sum_l \mathcal{UB}^{s^l\star}(b_{i+1}^l)\\ \mathcal{LB}^{s^j}(b_i,\{a_i^j,\pi^\star_{(i+1)+}\}) &= \mathbf{lb}(b_i^s,b_i,a_i^j)+\frac{1}{n_z}\sum_l \mathcal{LB}^{s^l\star}(b_{i+1}^l).\end{aligned} \tag{3.8}$$

Note this equation applies for each $a_i^j \in \mathcal{A}$, and as mentioned, each belief node b_{i+1}^l (one for each observation z_{i+1}^l) has its own simplification level s^l. In other words, for each b_{i+1}^l, s^l is the simplification level that was sufficient for calculating the bounds $\{\mathcal{UB}^{s^l\star}(b_{i+1}^l), \mathcal{LB}^{s^l\star}(b_{i+1}^l)\}$ and the corresponding optimal policy $\pi^\star_{(i+1)+}(b_{i+1}^l)$. Thus, when addressing belief node b_i in (3.8), for each belief node b_{i+1}^l and its corresponding simplification level s^l, these bounds are already available. Yet, we may still need to adapt the simplification level as we further discuss in Section 3.1.2.

Further, as seen in (3.8), the immediate reward and the corresponding bounds **ub** and **lb**, in general, can be calculated with their own simplification level s. In particular, when starting calculations, s could correspond to a default coarse simplification level, e.g. coarsest level s_0.

To define simplification level s^j of the bounds (3.8) we remind the reader that the belief tree is a discrete approximation to the expectation taken w.r.t. future observations $z_i, i \in \{k, k+L\}$. We account for some number n_z of observations made in tree nodes (e.g., in Fig. 3.1b, $n_z = 2$)

$$s^j \triangleq \min\{s, s^{l_1}, s^{l_2}, ...s^{l_{n_z}}\}, \tag{3.9}$$

where $\{s^{l_1}, s^{l_2}, ...s^{l_{n_z}}\}$ represents the (generally different) simplification levels of belief nodes b_{i+1}^l considered in the expectation approximation in (3.8). We explain the reason to define s^j as such in Section 3.1.2.

As mentioned earlier, we wish to decide which action $a_i^\star \in \mathcal{A}$ is optimal from belief node b_i; the corresponding optimal policy would then be $\pi_{i+}^\star = \{a_i^\star, \pi^\star_{(i+1)+}\}$, where $\pi^\star_{(i+1)+}$ is the already-calculated optimal policy for belief node b_{i+1}^l that $a_i^\star$ leads to. See illustration in Fig. 3.1b.

Determining $a_i^\star$ requires eliminating all other candidate actions $a^j \in \mathcal{A}$, which involves comparing their corresponding bounds (3.8). Importantly, the bounds are analytical, i.e. they are *valid for all simplification levels.*

As earlier, we can compare bounds for different candidate actions and if the bounds do not overlap, perform pruning. For example, if for $a_i^1, a_i^2 \in \mathcal{A}$,

$$\mathcal{UB}^{s^1}(b_i, \{a_i^1, \pi^\star_{(i+1)+}\}) < \mathcal{LB}^{s^2}(b_i, \{a_i^2, \pi^\star_{(i+1)+}\}), \tag{3.10}$$

we can prune the a_i^1 branch. If the bounds are sufficiently tight and all branches but one were pruned, then the remaining action, in this case a_i^2, is announced as $a_i^\star$, and s^2 is announced as $s^\star$. Thus the above-mentioned optimal policy $\pi_{i+}^\star$ is constructed.

We now recall b_i itself has an index in the belief tree, with respect to the previous level. We denote it as b_i^l, considering the father node of b_i is b_{i-1}, and the lth corresponding

observation z_i^l. At this point we get

$$\begin{aligned}\mathcal{UB}^{s^l\star}(b_i^l) &= \mathcal{UB}^{s^\star}(b_i^l, \{a^\star, \pi^\star_{(i+1)+}\}) \\ \mathcal{LB}^{s^l\star}(b_i^l) &= \mathcal{LB}^{s^\star}(b_i^l, \{a^\star, \pi^\star_{(i+1)+}\}).\end{aligned} \tag{3.11}$$

As in (3.7), (3.11) leads to bounds over the objective function

$$\mathcal{LB}^{s^l\star}(b_i^l) \leq J^\star(b_i^l) \leq \mathcal{UB}^{s^l\star}(b_i^l), \tag{3.12}$$

where b_i^l corresponds to the same notation as in (3.8), recursively. In general, pruning as done in (3.10) will not always hold, and we need to adapt level of simplification. We discuss in the next section how we do this, including re-use of calculations.

Adapting Simplification Level with Calculation Re-Use

For some belief node b_i in the belief tree, consider the bounds $\mathcal{UB}^{s^j}(b_i, \{a_i^j, \pi^\star_{(i+1)+}\})$ and $\mathcal{LB}^{s^j}(b_i, \{a_i^j, \pi^\star_{(i+1)+}\})$ from (3.8) for different actions $a_i^j \in \mathcal{A}$, that partially overlap and therefore could not be pruned. Each such action a_i^j can generally have its own simplification level s^j. We now iteratively increase the simplification level by 1. This can be done for each of the branches, if s^j is identical for all branches, or only for the branch with the coarsest simplification level. Consider now any such branch whose simplification level needs to be adapted from s^j to $s^j + 1$. Recall, that at this point, the mentioned bounds were already calculated, thus their ingredients, in terms of $\mathbf{ub}(b_i^s, b_i, a_i^j)$, $\mathbf{lb}(b_i^s, b_i, a_i^j)$ and $\{\mathcal{UB}^{s^l\star}(b_{i+1}), \mathcal{LB}^{s^l\star}(b_{i+1})\}_{l=1}^{n_z}$, involved in approximating the expectation in (3.8), are available. Recall also $s^j \triangleq \min\{s, s^{l_1}, s^{l_2}, ...s^{l_{n_z}}\}$ from (3.9), i.e. each element in $\{s, s^{l_1}, s^{l_2}, ...s^{l_{n_z}}\}$ is either equal or larger than s^j. We now discuss both cases, starting from the latter.

As we assumed bounds to improve monotonically as simplification level increases, see Assump. 3.1.1, for any $s^l > s^j + 1$ we already have readily available bounds $\{\mathcal{UB}^{s^l\star}(b_{i+1}), \mathcal{LB}^{s^l\star}(b_{i+1})\}$ which are tighter than those that would be obtained for simplification level $s^j + 1$. Thus, we can *safely skip* the calculation of the latter and use the existing bounds from level s^l as is.

For the former case, i.e. $s^l = s^j$, we now have to adapt the simplification level to $s^j + 1$ by calculating the bounds $\{\mathcal{UB}^{(s^l+1)\star}(b_{i+1}), \mathcal{LB}^{(s^l+1)\star}(b_{i+1})\}$. Here, our *key insight* is that, instead of calculating these bounds from scratch, we can re-use calculations between different simplification levels, in this case, from level s^l. As the bounds from that level are available, we can identify only the incremental part that is "missing" to get from simplification level s^l to $s^l + 1$, and update *analytically* the existing bounds $\{\mathcal{UB}^{s^l\star}(b_{i+1}), \mathcal{LB}^{s^l\star}(b_{i+1})\}$ to recover $\{\mathcal{UB}^{(s^l+1)\star}(b_{i+1}), \mathcal{LB}^{(s^l+1)\star}(b_{i+1})\}$ exactly. The same argument applies also for bounds over momentary rewards. In Section 3.1.4 we apply this approach to a specific simplification and reward function.

We can repeat iteratively the above process of increasing the simplification level

until we can prune all branches but one. This means each subtree will be solved maximum once, per simplification level. Since we assumed the simplification converges to the original problem for the finest level s_n, see Assump. 3.1.2, we are guaranteed to eventually disqualify all sub-optimal branches. Moreover, due to the discussed-above calculation re-use, in the worst case, adapting the simplification all the way up to the finest level s_n, is roughly equivalent to solving the original problem. We address this aspect explicitly in Section 3.1.4. For a detailed illustrative example w.r.t. Fig. 3.1b see Sec. 6.2.

In the following sections, we present a specific simplification along with novel derivations that show it holds the mentioned mathematical properties. Our described approach is summarized in Alg. 3.1, 3.2.

3.1.3 Bounds

We now delve deeper to the novel bounds our chosen simplification suggests.

Belief Representation and Chosen Simplification

As mentioned, we use Particle Filter for belief update. This means the belief is represented as a set of weighted particles,

$$b \triangleq \{x^i, w^i\}_{i=1}^N, \tag{3.13}$$

where N is a tune-able parameter specifying the desired number of particles.

Suggested Simplification: Given the belief representation (7.2), the simplified belief is a subset of N^s particles, sampled from the original belief, where $N^s \leq N$. More formally:

$$b_k^s \triangleq \{(x^i, w^i) \mid i \in A_k^s, A_k^s \subseteq \{1, 2, \ldots, N\}, |A_k^s| = N_k^s\}, \tag{3.14}$$

where A_k^s is the set of particle indices comprising the simplified belief b_k^s for time k.

Increasing the level of simplification is done *incrementally*. Specifically, consider $|A^s| = N^s$ and to get to the next simplification level we add m particles with indices $j \in B,\ |B| = m$. Then the following holds: $A^s \cap B = \emptyset,\ A^{s+1} = A^s \cup B,\ N^{s+1} = N^s + m$.

Bounding the Differential Entropy

We consider a common reward function, the differential entropy. As one of our key contributions, in this section we derive novel analytical upper and lower bounds **lb** and **ub** considering this reward function, assuming a sampling-based belief representation (7.2) and the corresponding simplified belief (3.14). These bounds can then be used within our general simplification framework presented in Sections 3.1.1 and 3.1.2.

Under this setting approximating differential entropy of the belief is not an easy task. To calculate $\mathcal{H}(b[x_k]) = -\int b[x_k] \cdot \log\left(b[x_k]\right) dx_k$, one must have access to the manifold representing the belief. Several approaches exist. One of them is using Kernel

Density Estimation as done, e.g., by Fischer and Tas [2020]. Here, we consider the method proposed by Boers et al. [2010]:

$$\hat{\mathcal{H}}(b_{k+1}) \triangleq \underbrace{\log\left[\sum_i \mathbb{P}(z_{k+1} \mid x^i_{k+1}) w^i_k\right]}_{\text{(a)}} \tag{3.15}$$
$$\underbrace{-\sum_i w^i_{k+1} \cdot \log\left[\mathbb{P}(z_{k+1} \mid x^i_{k+1}) \sum_j \mathbb{P}(x^i_{k+1} \mid x^j_k, a_k) w^j_k\right]}_{\text{(b)}},$$

where i indexes particles and their corresponding weight as in (7.2) and (3.14). One can observe this method requires access to samples representing both b_k and b_{k+1}; thus, this corresponds to an information-theoretic reward of the form $r(b_k, b_{k+1})$. Utilizing the chosen simplification (3.14) we can now upper and lower bound 3.15. We do so below by bounding term (b) which is the source of complexity. First, as the models are known, we define $m \triangleq \max\{\mathbb{P}(x_{k+1} \mid x_k, a_k)\}$.

Theorem 1. Term (b) in (3.15) can be upper and lower bounded via simplification as

$$(b) \geq -\sum_{i \in \neg A^s_{k+1}} w^i_{k+1} \cdot \log\left[m \cdot \mathbb{P}(z_{k+1} \mid x^i_{k+1})\right]$$
$$-\sum_{i \in A^s_{k+1}} w^i_{k+1} \cdot \log\left[\mathbb{P}(z_{k+1} \mid x^i_{k+1}) \sum_j \mathbb{P}(x^i_{k+1} \mid x^j_k, a_k) w^j_k\right]$$
$$(b) \leq -\sum_i w^i_{k+1} \cdot \log\left[\sum_{j \in A^s_k} \mathbb{P}(z_{k+1} \mid x^i_{k+1}) \mathbb{P}(x^i_{k+1} \mid x^j_k, a_k) w^j_k\right]$$

See proof in Sec. 6.1.

Finally, bounding (3.15) using Theorem 1 corresponds, in our general framework from Sections 3.1.1 and 3.1.2, to (3.2).

3.1.4 Bounds Analysis

Convergence

We now analyze convergence of the simplification described in Section 3.1.3. Since simplifying to the level of s_n means the simplified belief is just the original belief, we get: $N^s = N \Rightarrow b = b^s \Rightarrow r(b^s, a) = r(b, a)$. Furthermore, it can be easily seen the upper and lower bounds in Theorem 1 coincide and equal to term (b) from (3.15). Meaning, under our chosen simplification, the bounds converge to the original rewards for the non-simplified original belief. Where we consider as lower bounds term (a) from (3.15) with lower bound to term (b), and similarly we build the upper bounds (using (a) and upper bound to (b)).

Complexity Analysis

The original formulation of Boers et al. [2010] suggests complexity of $O(N^2)$ where N is as in (7.2). The derivations in Section 3.1.3 suggest complexity of $O(N^s \cdot N)$, where N^s is the maximal number of particles needed for pruning, for some belief node. Altogether, time saved for all belief nodes in the tree will result in the total speedup of our approach.

Re-use of Calculations

The unique structure of the simplification allows us to cache the calculation from a previous simplification level. Specifically, moving from simplification level s to level $s+1$, corresponds to adding some m additional particles to b^s in order to get b^{s+1}. For the lower bound in Theorem 1 we can just cache the final result and augment the sums with the calculations corresponding to the m missing particles. For the upper term in Theorem 1, we need to cache the inner sums of the log, augment them with the missing m particles calculations, and re-weight it using the outer sum weights. Note space complexity remains the same since we are already caching the particles and weights for each belief. This re-use of calculations results in time complexity, going up from simplification level s_0 to level s_n being the same order as solving the original problem in the first place. Thus, making simplification is worthwhile always (up to some constant overhead).

3.2 SITH-PFT incorporation of simplification into state of the art MCTS planner

SITH-PFT (Alg. 3.3) follows the same algorithmic baseline as PFT. We adhere to the conventional notations Sunberg and Kochenderfer [2018] and denote by $G_{\mathrm{PF}(m)}(bao)$ a generative model receiving as input the belief b, an action a and an observation o, and producing the posterior belief b' and the mean reward over the state $r^x(b,a)$. For belief update, we use a particle filter based on m belief samples. Instead of calculating the immediate information-theoretic rewards and the corresponding Q^I function estimates, we calculate low-cost lower and upper bounds ℓ, u over the information-theoretic rewards and corresponding bounds $\mathcal{LB}, \mathcal{UB}$ over the Q^I function. These bounds are adaptive and can be tightened on demand. We call the process of tightening "resimplification". In turn, these bounds induce bounds over UCB. As we discuss in detail next, an essential aspect of our approach is using these bounds to achieve the exact same action-selection as UCB without exactly calculating the Q function and UCB. To this end we present a novel action-selection method (Alg. 3.4). Crucially, by tightening the bounds only to a minimal needed extent (Alg. 3.5), we guarantee the same tree connectivity and calculated optimal action compared to PFT-DPW, but faster. We devote the subsequent section to the bounds and explain how they pertain to SITH-PFT.

3.2.1 Information theoretic bounds

In the setting of continuous state space and nonparametric belief represented by m weighted particles $b \triangleq \{w^i, x^i\}_{i=1}^m$, the estimation of differential entropy is not a simple task. Typically, such estimators' complexity is squared in the number of particles Fischer and Tas [2020], Boers et al. [2010]. We use Boers et al. [2010] as a reward function and utilize the bounds over it, developed by Sztyglic and Indelman [2021]. The bounds can be tightened on demand incrementally without an overhead. Namely, after a few bounds-tightening iterations they are just the reward itself and the entire calculation is time-equivalent to calculating the original reward. We define the bounds over the minus differential entropy estimator for b_{k+1} as (see the appendix for the full terms)

$$\ell(b_k, a_k, z_{k+1}, b_{k+1}; A_k^s, A_{k+1}^s) \leq -\hat{\mathcal{H}}(b_k, a_k, z_{k+1}, b_{k+1}) \leq u(b_k, a_k, z_{k+1}, b_{k+1}; A_k^s, A_{k+1}^s), \tag{3.16}$$

where s is the discrete level of simplification $s \in \{1, 2, \ldots, M\}$. Higher levels of simplification correspond to tighter, and lower levels of simplification correspond to looser bounds. A_k^s, A_{k+1}^s are the simplification level corresponding sets of indices. Specifically, b_k, b_{k+1} are each represented as a set of m weighted particles. We keep track over the indices of particles that were chosen for the bounds calculation. Namely, $A^s \subseteq \{1, 2, \ldots, m\}$ and $|A^s| = m^s$. Each subsequent level (low to high) defines a larger set of indices. Sometimes the bounds are not close enough to select the same action as UCB. In this case, our modified action selection routine triggers the resimplification process. When resemplification is carried out, new indices are drawn from the sets $\{1, 2, \ldots, m\} \setminus A_k^s$ and $\{1, 2, \ldots, m\} \setminus A_{k+1}^s$ respectively, and added to the sets A_k^s and A_{k+1}^s. This operation promotes the simplification level to $s+1$ and defines A_k^{s+1} and A_{k+1}^{s+1}. Importantly, increasing simplification level is done incrementally (as introduced by Sztyglic and Indelman [2021]). Thus, when we refine the bounds ℓ, u (Alg. 3.5 lines 3,12,18), from simplification level $s = 1$ all the way to $s = M$ (worst case scenario) the time complexity is equivalent to calculation of $\hat{\mathcal{H}}(\cdot)$. When $s = M$, it holds that $\ell(\cdot) = -\hat{\mathcal{H}}(\cdot) = u(\cdot)$. Importantly, by caching the shared calculations of the two bounds, we never repeat the calculation of these values and obtain maximal speedup. The immediate bounds (3.16) induce bounds over $Q^I(\cdot)$. In MCTS, the Q approximation is a mean over simulations. Each simulation yields a sum of discounted cumulative rewards. Therefore, if we replace the reward $-\hat{\mathcal{H}}(\cdot)$ with the bounds from (3.16) we will get corresponding discounted cumulative upper and lower bounds. Averaging the simulations, in the same manner (Alg. 3.3 lines 29-30), yields

$$\mathcal{LB}(\cdot) \leq Q^I(\cdot) \leq \mathcal{UB}(\cdot). \tag{3.17}$$

3.2.2 UCB bounds

Since the MCTS tree is built upon (2.8), using (2.7) and (3.17) we denote UCB upper and lower bounds as

$$\underline{\mathrm{UCB}}(ha) \triangleq Q^x(ha) + \lambda \mathcal{LB}(ha) + c \cdot \sqrt{\frac{\log(N(h))}{N(ha)}}, \tag{3.18}$$

$$\overline{\mathrm{UCB}}(ha) \triangleq Q^x(ha) + \lambda \mathcal{UB}(ha) + c \cdot \sqrt{\frac{\log(N(h))}{N(ha)}}. \tag{3.19}$$

3.2.3 Guaranteed belief tree consistency

In this section, we define the Tree Consistency and explain and prove the equivalence of our algorithm to PFT-DPW.

Definition 1 (Tree consistent algorithms). Consider two algorithms, constructing a belief tree. Assume every common sampling operation for the two algorithms uses the same seed. The two algorithms are *tree consistent* if two belief trees constructed by the algorithms are identical in terms of actions, observations, and visitation counts.

Our approach leans on a specific action selection procedure inside the tree, which differs from the PFT. At every belief node we mark as a candidate action the one that maximizes the lower bound $\underline{\mathrm{UCB}}$ as such

$$\tilde{a} = \arg\max_{a \in \mathcal{A}} \underline{\mathrm{UCB}}(ha). \tag{3.20}$$

If $\forall a \neq \tilde{a}$, $\underline{\mathrm{UCB}}(h\tilde{a}) \geq \overline{\mathrm{UCB}}(ha)$, there is no overlap (Fig. 3.2 **(c)**) and we can announce $\tilde{a}$ is identical to $a^\dagger$, i.e., the action that would be returned by PFT using (2.8) and the tree consistency was not compromised. Else, the bounds need to be tightened, so we may guarantee the tree consistency. We examine the ha siblings of $h\tilde{a}$, fulfilling $a \neq \tilde{a} : \underline{\mathrm{UCB}}(h\tilde{a}) < \overline{\mathrm{UCB}}(ha)$ (Fig. 3.2 **(a)**). Our next step is to tighten the bounds via resimplification (Fig. 3.2 **(b)**) until there is no overlap. When some sibling nodes have overlapping bounds, we strive to avoid tightening all of them at once since fewer resimplifications lead to a greater speedup. Thus, among them we pick a single ha node that induces the biggest "gap", denoted by g, between its bounds (see Alg. 3.4 lines 20-28), where

$$g(ha) \triangleq \mathcal{UB}(ha) - \mathcal{LB}(ha). \tag{3.21}$$

Further, we tighten the bounds down the branch of the chosen node (see Alg. 3.4 lines 7-9) for every member of $C(ha)$, the set of children of ha. Since the bounds converge to the actual information reward we can guarantee the algorithm will pick a single action after a finite number of "bounds-tightening" iterations (resimplification); thus,

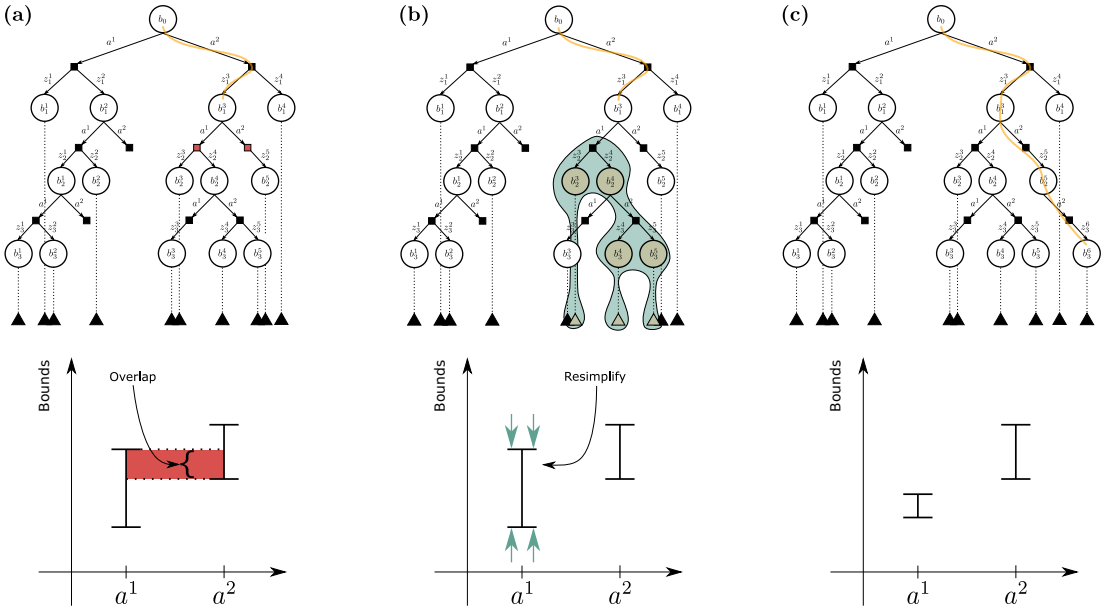


Figure 3.2: Illustration of our approach. The circles denote the belief nodes, and the rectangles represent the belief-action nodes. Rollouts, emanating from each belief node, are indicated by dashed lines finalized with triangles. **(a)** The simulation starts from the root of the tree, but at node b_1^3 it can not continue due to an overlap of the child nodes (colored red) bounds. **(b)** One of the red colored belief-action nodes is chosen, and resimplification is triggered from it down the tree to the leaves (shaded green area in the tree). The beliefs and rollouts inside the green area (colored by light brown) undergo resimplification if decided so. This procedure results in tighter bounds. **(c)** After the bounds got tighter, nothing prevents the SITH-PFT from continuing down from node b_1^3 guaranteeing the Tree Consistency. If needed, additional resimplification can be commenced.

tree consistency is assured. In the following section, we delve into the resimplification procedure.

3.2.4 Resimplification

In this section, we explain how resimplification is done. The algorithmic scheme is formulated in a general manner. However, it is guided by a specific strategy meant to minimize the number of times we tighten the bounds (as mentioned in Sec. 3.2.3). We denote this strategy as *Resimplification Strategy*. We assume this strategy satisfies two conditions to guarantee tree consistency.

Convergence 3.2.1. When using a *converging strategy*, each call to resimplify on the children of ha, tightens the $\underline{\mathrm{UCB}}(ha), \overline{\mathrm{UCB}}(ha)$ bounds (unless they are already equal).

Finite-time 3.2.2. When using a *finite-time strategy*, after a finite number of calls to resimplify on the children of ha, it holds $\underline{\mathrm{UCB}}(ha) = \mathrm{UCB}(ha) = \overline{\mathrm{UCB}}(ha)$.

Resimplification algorithmic scheme

Consider a belief-action node ha at level d with $\mathcal{LB}(ha), \mathcal{UB}(ha)$. Assume the algorithm chooses it for bounds tightening, as described in Sec. 3.2.3 and Alg. 3.4 line 3. All

tree nodes that ha is an ancestor to them, contribute their immediate ℓ, u bounds to $\mathcal{LB}(ha)$,$\mathcal{UB}(ha)$ calculation. Thus, to tighten $\mathcal{LB}(ha)$,$\mathcal{UB}(ha)$, we can potentially choose any candidate nodes in the subtree of ha. Every child belief node of ha is sent to the resimplification routine (Alg. 3.4 lines 7-9), which performs four tasks. Firstly, it chooses the action (Alg. 3.5 line 7) that will participate in the subsequent resimplification call and sends all its children beliefs nodes to the recursive call down the tree (Alg. 3.5 line 8-10). Secondly, it refines the belief node ℓ, u according to the specific *resimplification strategy* (Alg. 3.5 lines 3,12,18). Thirdly, it reconstructs $\mathcal{LB}(ha), \mathcal{UB}(ha)$ once all the children belief nodes of ha have returned from the resimplification routine (Alg. 3.5 line 11). Fourthly, it engages the rollout resimplification routine according to the specific *resimplification strategy* (Alg. 3.5 lines 4, 13). Upon completion of this resimplification call initiated at ha, we get tighter immediate bounds of some of ha descendant belief nodes (including rollouts nodes). Accordingly, all of ha descendant belief-action nodes bounds ($\mathcal{LB}$,$\mathcal{UB}$) were updated.

Specific resimplification strategy

Specifically, we decide to refine ℓ', u' of a belief node h' with depth d' if

$$\gamma^{d-d'} \cdot (u' - \ell') > \frac{1}{d} g(ha), \tag{3.22}$$

where $g(ha)$ corresponds to the gap (3.21) of the belief-action node ha that initially triggered resimplification in Alg. 3.4 line 24. The explanation to (3.22) resimplification strategy is rather simple. The right hand side of (3.22) is the mean gap per depth/level in the sub-tree with ha as its root and spreading downwards to the leaves. Naturally, some of the nodes in this subtree have $u - \ell$ above the mean gap, and some under. We wish to locate and refine all the ones above. For the left-hand side of (3.22); the rewards are accumulated and discounted according to their depth. Thus, when comparing ha node with depth d to belief node h' with depth d', we must account for the relative proper discount factor. Note the depth identified with the root is $d_{\max}$ as seen in Alg. 3.3 line 4, and the leafs are distinguished by depth $d = 0$. For each rollout originating from the tree belief node, we find the rollout node with the biggest $u - \ell$ fulfilling (3.22) term locally in the rollout and resimplify it (Alg. 3.5 lines 4,13). To choose the action to continue resimplification down the tree, we take the action corresponding to the belief-action node with the largest gap weighted by its visitation count (Alg. 3.5 line 7). With this strategy, we aim to leave the belief tree at the lowest possible simplification levels whilst still guarantee tree consistency.

Reconstructing the bounds

If the action selection procedure triggered a resimplification, it modified the bounds through the tree. Since the resimplification works recursively, it reconstructs the belief-

action node bounds coming back from the recursion (Alg. 3.5 line 11). Similarly, the action dismissing procedure reconstructs $\mathcal{LB}$, and $\mathcal{UB}$ of the belief-action node at which the action dismissing is performed (Alg. 3.4 line 10). Moreover, on the way back from the simulation, we shall update the ancestral belief-action nodes of the tree. Specifically, we are required to reconstruct each $\mathcal{LB}$ and $\mathcal{UB}$ higher than the deepest starting point of the resimplification (Alg. 3.3 line 23-25). Reconstruction is essentially a double loop. To reconstruct $\mathcal{UB}(ha), \mathcal{LB}(ha)$ we first query for all belief children nodes hao. We then query all belief-action nodes that are children to the hao, i.e., $haoa'$. The possibly modified immediate bounds ℓ and u are taken from hao nodes and the $\mathcal{UB}(\cdot)$, $\mathcal{LB}(\cdot)$ bounds are taken from the $haoa'$ nodes. Importantly, each of the bounds is weighted according to the proper visitation count.

3.2.5 Guarantees

Assuming a *converging and finite-time resimplification strategy*, the following theorems are satisfied:

Theorem 2. The SITH-PFT and PFT are Tree Consistent Algorithms.

Theorem 3. The SITH-PFT provides the same solution as PFT.

Theorem 4. The specific resimplification strategy from Sec. 3.2.4 is a converging and finite-time resimplification strategy.

See full proofs of the theorems and time complexity analysis using the specific bounds in the appendix. Note other resimplification strategies are possible, see the appendix.

3.2.6 Algorithms

Algorithm 3.3 SITH-PFT

procedure PLAN(belief: b)
  $h \leftarrow \emptyset$
  for $i \in 1 : n$ **do**
    SIMULATE(b, $d_{\max}$, h)
  end for
  return ACTION SELECTION(b, h) ▷ called with nullified exploration constant c
end procedure
procedure SIMULATE(belief: b, depth: d, history: h)
  if $d = 0$ **then**
    return 0
  end if
  $a \leftarrow$ ACTION SELECTION(b, h)
  if $|C(ha)| \leq k_o N(ha)^{\alpha_o}$ **then**
    $o \leftarrow$ sample x from b, generate o from (x, a)
    $b', r^x \leftarrow G_{\mathrm{PF}(m)}(bao)$
    Calculate initial u', ℓ' for b' based on $s \leftarrow 1$ ▷ minimal simp. level
    $C(ha) \leftarrow C(ha) \cup \{(r^x, \ell', u', b', o)\}$
    $R, L, U \leftarrow r^x, \ell', u' + \gamma$ ROLLOUT(b', hao, $d-1$)
  else
    $(r^x, \ell', u', b', o) \leftarrow$ sample uniformly from $C(ha)$
    $R, L, U \leftarrow r^x, \ell', u' + \gamma$ SIMULATE(b', hao, $d-1$)
  end if
  if deepest resimplification depth $< d$ **then** ▷ accounting for updated deeper in the tree bounds. See section 3.2.4
    reconstruct $\mathcal{LB}(ha)$,$\mathcal{UB}(ha)$
  end if
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $Q^x(ha) \leftarrow Q^x(ha) + \frac{R - Q^x(ha)}{N(ha)}$
  $\mathcal{LB}(ha) \leftarrow \mathcal{LB}(ha) + \frac{L - \mathcal{LB}(ha)}{N(ha)}$
  $\mathcal{UB}(ha) \leftarrow \mathcal{UB}(ha) + \frac{U - \mathcal{UB}(ha)}{N(ha)}$
  return R, L, U
end procedure

Algorithm 3.4 Action Selection

```
1: procedure ACTION SELECTION(b, h)
2:     while true do
3:         Status, a ← SELECT BEST(b, h)
4:         if Status then
5:             break
6:         else
7:             for all b', o ∈ C(ha) do
8:                 RESIMPLIFY(b', hao)
9:             end for
10:            reconstruct LB(ha),UB(ha)
11:        end if
12:    end while
13:    return a
14: end procedure
15: procedure SELECT BEST(b, h)
16:    Status ← true
17:    ã ← arg max{UCB(ha)}
18:    gap ← 0^a
19:    child-to-resimplify ← ã
20:    for all ha children of b do
21:        if UCB(hã) < ¯UCB(ha) ∧ a ≠ ã then
22:            Status ← false
23:            if UB(ha) − LB(ha) > gap then
24:                gap ← UB(ha) − LB(ha)
25:                child-to-resimplify ← a
26:            end if
27:        end if
28:    end for
29:    return Status, child-to-resimplify
30: end procedure
```

Algorithm 3.5 Resimplification

procedure RESIMPLIFY(b, h)
  if b is a leaf **then**
    REFINE$_{\{\ell,u\}}(b)$
    RESIMPLIFY ROLLOUT(b, h)
    return
  end if
  $\tilde{a} \leftarrow \arg\max\{N(ha) \cdot (\mathcal{UB}(ha) - \mathcal{LB}(ha))\}$
  for all $b', o \in C(h\tilde{a})$ **do**
    RESIMPLIFY(b', $h\tilde{a}o$)
  end for
  reconstruct $\mathcal{LB}(h\tilde{a})$,$\mathcal{UB}(h\tilde{a})$
  REFINE$_{\{\ell,u\}}(b)$
  RESIMPLIFY ROLLOUT(b, h)
  return
end procedure
procedure RESIMPLIFY ROLLOUT(b, h)
  $b^{\text{rollout}} \leftarrow$ find weakest link in rollout
  REFINE$_{\{\ell,u\}}(b^{\text{rollout}})$
end procedure
procedure REFINE$_{\{\ell,u\}}(b)$
  if (3.22) holds for b, refine its ℓ, u and promote its simplification level
end procedure

Chapter 4

Experiments and Results

4.1 SITH-BSP Evaluation

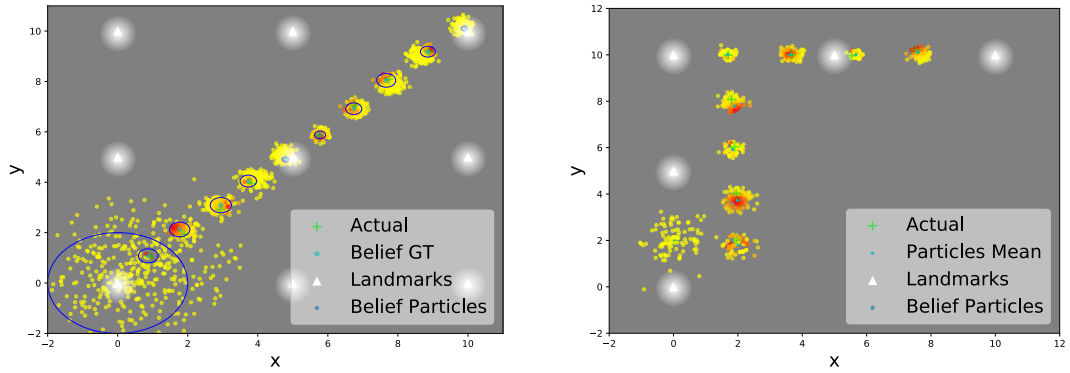
We consider an autonomous navigation to goal scenario with continuous state and observation spaces. First, we experiment with a passive case, i.e. a given policy, to demonstrate our novel entropy bounds behavior. Second, we show how the bounds can use us to speed up POMDP planning as explained in Section 3.1. All experiments were conducted on a laptop with Intel i7-9850H CPU 2.59GHz with 16 GB RAM.

4.1.1 Experimental Setting - 2D Continuous Light-Dark

We consider a *2D continuous Light-Dark problem.* The robot starts at some unknown point $x_0 \in \mathbb{R}^2$. In this world, there are spatially scattered beacons with known locations. Near the beacons, the attained observations are less 'noisy'. The goal is to get to the goal $x^t \in \mathbb{R}^2$ (upper right corner of the world). Initial belief is $b[x_0] = \mathcal{N}(x_0, I \cdot \sigma_0)$, motion and observation models are $T = \mathbb{P}(x' \mid x, a) = \mathcal{N}(x + a, I \cdot \sigma_T)$, $\mathcal{O} = \mathbb{P}(z \mid x) = \mathcal{N}(x - x^b, I \cdot \sigma_{\mathcal{O}} \cdot \max\{r, r_{min}\})$ respectively, where r is the robot's distance to the nearest beacon whose known location is x^b, and r_{min} is a tune-able parameter. For all experiments, the belief is approximated by a set of N weighted samples as in (7.2). This setting implies the belief at each time step is Gaussian and can be inferred exactly using Kalman Filter (KF). Thus, the differential entropy has a closed-form and can be calculated across the simulation. Note we consider this Gaussian case only as a reference point and our approach is applicable to any distribution.

4.1.2 Differential Entropy Approximations

To verify our novel bounds behavior we consider a passive scenario over the setting described in 4.1.1. The robot moves diagonally to the goal (Fig. 4.1a). Along the way, it passes close by two beacons. Consequentially, the robot's uncertainty decreases. In Fig 4.2a we plot the bounds along with the original approximation (3.15), a KDE approximation (as done by Fischer and Tas [2020]), the actual differential entropy and



(a) **(b)**

Figure 4.1: **(a)** Localization with a predefined policy in an environment represented by known landmarks. Particles are colored according to weight (high-red, small-yellow). Blue ellipses correspond to estimated uncertainty covariance by a KF. **(b)** Planning simulation, setting II. At each time step the robot is planning L step into the future and then executes the first action from the calculated optimal policy. The plot shows belief evolution in terms of particles sets along the actual trajectory taken by the robot.

naive approximation using discrete entropy of the particles weights: $h(b) = -\sum_i w^i \cdot \log w^i$. We experiment with a changing number of particles and it is clear the bounds converge to the original reward (differential entropy approximation) (3.15).

4.1.3 Planning in 2D environment

We demonstrate the simplification speedup when planning in a continuous 2D scenario. The setting is as in Section 4.1.1. However, now the robot is given the ability to plan a parametric number of L steps into the future. After the planning session is done, the robot executes the first action out of the calculated optimal policy, acquires a new observation, updates the belief, and performs planning again (and so on). The reward function is $-r(b,a) = \underset{x\sim b}{\mathbb{E}}\{\|x - x^t\|_1\} + \hat{\mathcal{H}}(b)$, where the first term is the expected distance to goal, and $\hat{\mathcal{H}}(b_k)$ is the approximation of the differential entropy (3.15).

We apply our approach considering different tree structures. Specifically, we evaluate our approach on a POMCP-like tree (deep and sparse) Silver and Veness [2010], DESPOT-like sparse tree (Ye et al. [2017]), and on a shallow and 'thick' tree like the one generated by Lim et al. [2020a]. We provide a full explanation of how these trees are built in the caption of Table 4.1. The reported results in Table 4.1 are the mean planning time (in seconds) of our approach, compared to calculating the objective using original rewards. We experiment with a changing number of particles and planning horizon.

We consider two settings: 'I' and 'II'. Setting 'II', illustrated in Fig. 4.1b, is more complex as the robot needs to move from the bottom left corner to the top right and the action space is {left, right, up, down}. Setting 'I' is easier: The robot needs to move from an initial point x_0 to the same height point x_L, which means the optimal path is just a straight line, while the action space is {left, right}. This easy setting allows the robot to utilize simplification to its full extent. Empty cells in Table 4.1 correspond

Table 4.1: Mean time per planning session. Each table entry is (*original objective time/simplified approach time*). Results are in Seconds. The second row indicates the number of particles used. DESPOT-like belief tree Ye et al. [2017] was built by expanding all actions in every level of the tree but only making a single observation in each level, i.e. $n_z = 1$. POWSS-like tree Lim et al. [2020a] was built by expending each belief node for all actions, and generating an observation for each particle of the belief, i.e. number of observations when branching is as the number of particles. POMCP-like tree Silver and Veness [2010] was built using five 'rollout's starting from the root of the tree. In each rollout down the tree, we randomly choose if to expand a new node by taking an action that was not taken previously from that node or to go down the tree using nodes that were already expanded from previous rollouts.

Simulation	Horizon	Ye et al. [2017] Tree		
		20	50	100
Setting I	1	0.124/**0.043**	0.741/**0.192**	2.892/**0.667**
	2	0.364/**0.129**	2.196/**0.584**	8.616/**2.042**
	3	0.853/**0.339**	5.059/**1.324**	19.899/**4.658**
Setting II	1	0.245/**0.099**	1.513/**0.4**	5.855/**2.018**
	2	1.209/**0.738**	7.195/**3.821**	30.638/**13.49**
	3	5.027/**3.212**	31.515/**18.288**	-

Simulation	Horizon	Lim et al. [2020a] Tree		
		10	20	30
Setting I	1	0.554/**0.287**	4.065/**1.437**	12.908/**3.953**
	2	11.02/**5.386**	-	-
	3	-	-	-
Setting II	1	1.112/**0.953**	8.501/**5.143**	26.375/**11.977**
	2	-	-	-
	3	-	-	-

Simulation	Horizon	Silver and Veness [2010] Tree		
		20	50	100
Setting I	5	1.13/**0.776**	6.625/**2.008**	28.19/**7.232**
	10	2.648/**2.555**	15.342/**8.214**	-
	15	4.2/**3.677**	26.205/**20.174**	-
Setting II	5	1.383/**0.733**	8.417/**3.864**	33.244/**10.97**
	10	2.985/**2.112**	17.293/**6.092**	-
	15	4.53/**3.701**	27.712/**11.385**	-

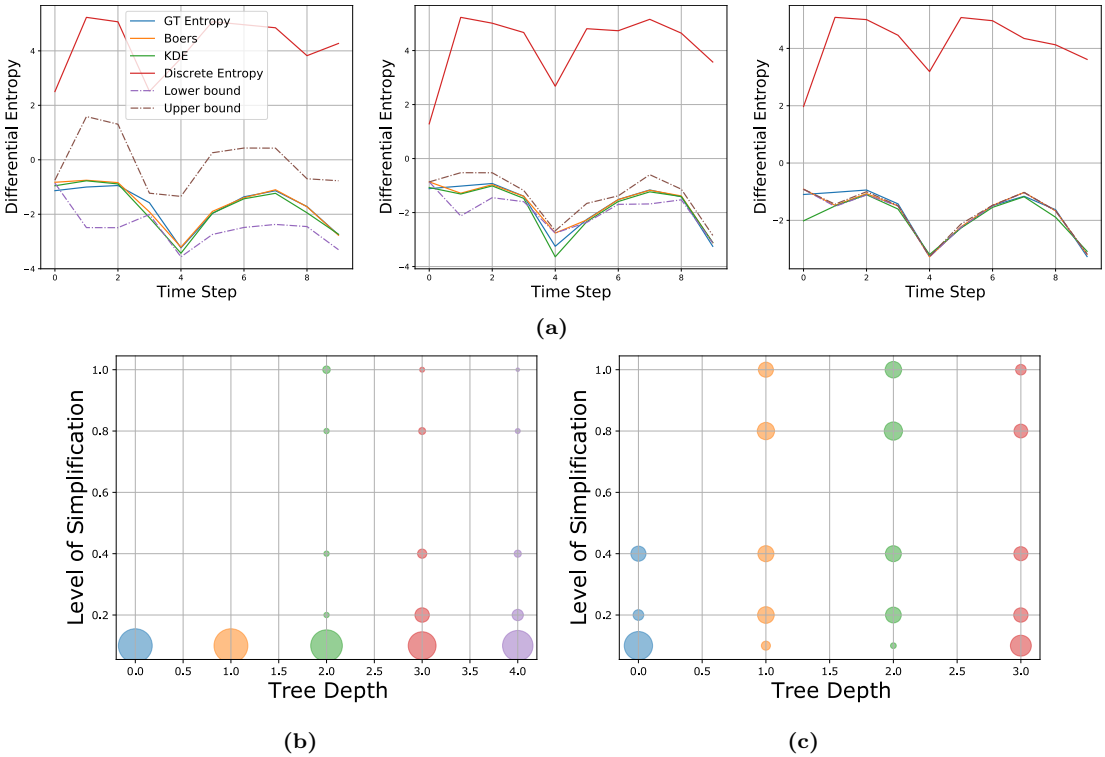
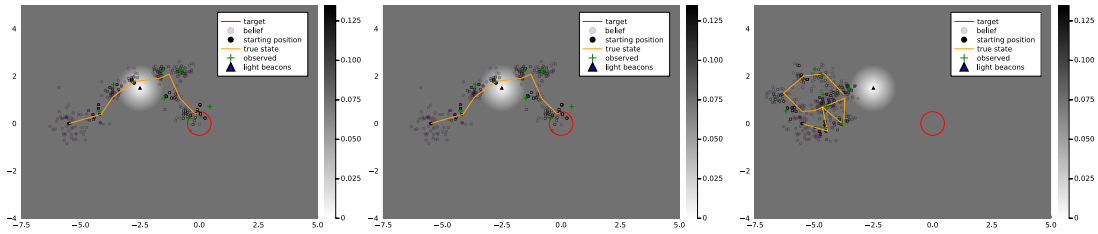


Figure 4.2: **(a)** Differential entropy approximations and bounds. Calculations were done using $N = 200$ particles. From left to right: Simplification is $N^s = \{0.1, 0.5, 0.9\} \cdot N$. **(b)** and **(c)** show simplification level histograms vs. tree depth for entire simulation (10 planning sessions). **(b)** Setting I using $N = 50$ and Horizon of 4. **(c)** Setting II using $N = 20$ and Horizon of 3. x-axis corresponds to tree nodes of some depth in the belief tree. y-axis corresponds to the simplification level needed to achieve pruning. The scale of the circles corresponds to how many nodes were in that category. Circle scales are normalized since the number of nodes grows exponentially going down the tree.

to runs that planning session (for a regular objective) took longer than 35 [sec] and was stopped. Initial simplification level was set to $s_0 = 0.1$, i.e., $N^{s_0} = 0.1 \cdot N$ and specifically the levels are $s_i \in \{0.1, 0.2, 0.4, 0.8, 1.0\}$.

It is clear from Table 4.1, using our suggested simplification is a favorable approach, leading to speedup in all of the conducted experiments. Since the simplification bounds are analytical and used for eliminating branches in the belief tree of the original problem, the *same optimal action* is obtained with or without our simplification. In other words, we demonstrate a significant speedup while obtaining the same solution.

In Fig. 4.2 we can get a glimpse into how our adaptive simplification performing in the tree depth for Settings I and II. The shown plots are histograms that tell us what are the levels of simplification in the tree needed for pruning. It can be seen that indeed for Setting I the simplification is performing extremely well thus saving a lot of time 4.2b. In the more difficult Setting II, indeed higher simplification levels are more common 4.2c. Nevertheless, as seen in Table 4.1, we still get a significant speedup, while the speedup for Setting I is even more drastic. This implies our simplification can identify by itself situations where we can save resources (computation time) and all this without compromising on the accuracy of the desired solution.



(a) SITH-PFT **(b)** PFT-DPW **(c)** IPFT

Figure 4.3: 2D Continuous Light Dark. The agent starts from an initial unknown location and is given an initial belief. The goal is to get to location $(0, 0)$ (circled in red) and execute the terminal action. Near the beacon (white light) the observations are less noisy. We consider multi-objective function, accounting for the distance to the goal and the differential entropy approximation (with the minus sign for reward notation). Executing the terminal action inside the red circle gives the agent a large positive reward but executing it outside it, will yield a large negative reward.

4.2 SITH-PFT Evaluation

In the continuous setting with information-theoretic rewards, many common POMDP benchmarks (e.g., rock sampling, laser tag) are inadequate. We turn to the challenging Continuous Light Dark Problem with a few modifications. We extend it to a 2D domain and place a single "light beacon" in the continuous world. The agent's goal is to get to location $(0, 0)$ and execute the terminal action - $Null$. Executing it within a small radius from $(0, 0)$ will give the agent a reward of 200, and executing it outside the radius will yield a negative reward of -200. The agent can move in eight evenly spread directions $\mathcal{A} = \{\rightarrow, \nearrow, \uparrow, \nwarrow, \leftarrow, \swarrow, \downarrow, \searrow, Null\}$. The multi-objective reward function is $r(b, a, z, b') = -\mathop{\mathbb{E}}_{x\sim b'}\{\|x\|_2\} - \lambda\hat{\mathcal{H}}(b, a, z, b')$. Motion, observation, and initial belief are $\mathbb{P}_T(\cdot|x, a) = \mathcal{N}(x + a, \Sigma_T)$, $\mathbb{P}_Z(z|x) = \mathcal{N}(x, \min\{1.0, \| x - x^b \|_2^2\} \cdot \Sigma_O)$, $b_0 = \mathcal{N}(x_0, \Sigma_0)$ respectively. x^b is the 2D location of the beacon and all covariance matrices are diagonal (i.e. $\Sigma = I \cdot \sigma^2$). Implementation is built upon the JuliaPOMDP package collection Egorov et al. [2017]. The code will become available upon acceptance of Sztyglic et al. [2021]. Extensive experiments confirm the advantage of our approach. We experiment with ten different configurations (rows of Table 4.2) that differ in m (number of particles), d (simulation depth), and #iter (number of simulation iterations per planning session). Each scenario comprises 10 planning sessions i.e. the agent performs up to 10 planning-action executing iterations. We repeat each of the experiments 25 times. In all different configurations, we obtained significant speedup while achieving the exact same solution compared to PFT. Results are summed up in Table 4.2. An illustration can be found in Fig. 4.3. Note that SITH-PFT 4.3a yields identical to PFT solution 4.3b while IPFT demonstrates severely degraded behavior. We remind the purpose of our work is to speed up the PFT approach when coupled with information-theoretic reward. Hence, since the two algorithms produce identical belief trees and action at the end of each planning session, there is no point reporting the algorithms *identical* performances (apart from planning time). For our simulations, we used an 8 cores Intel(R) Xeon(R) CPU E5-1620 v4 with 128 GB of RAM working at 3.50GHz.

Table 4.2: Runtimes of SITH-PFT versus PFT-DPW. The rows are different configurations of the number of belief particles m, maximal tree depth d, and the number of iterations per planning session. Reported values are averaged over 25 simulations 10 planning sessions each, and presented with the standard errors. In all simulations SITH-PFT and PFT-DPW declared *identical* actions as optimal and exhibited *identical* belief trees in terms of connectivity and visitation counts.

(m, d, #iter.)	Algorithm	planning time [sec]
(50, 30, 200)	PFT-DPW	3.54 ± 0.4
	SITH-PFT	2.96 ± 0.49
(50, 50, 500)	PFT-DPW	9.82 ± 1.31
	SITH-PFT	8.1 ± 1.33
(100, 30, 200)	PFT-DPW	13.42 ± 1.49
	SITH-PFT	10.77 ± 1.73
(100, 50, 500)	PFT-DPW	35.06 ± 4.44
	SITH-PFT	26.7 ± 4.37
(200, 30, 200)	PFT-DPW	55.89 ± 5.41
	SITH-PFT	39.46 ± 7.09
(200, 50, 500)	PFT-DPW	142.14 ± 12.39
	SITH-PFT	100.09 ± 14.67
(400, 30, 200)	PFT-DPW	211.86 ± 24.18
	SITH-PFT	160.36 ± 31.02
(400, 50, 500)	PFT-DPW	570.13 ± 45.48
	SITH-PFT	414.65 ± 53.37
(600, 30, 200)	PFT-DPW	503.78 ± 31.61
	SITH-PFT	374.0 ± 44.23
(600, 50, 500)	PFT-DPW	1204.78 ± 119.16
	SITH-PFT	912.92 ± 116.08

Chapter 5

Conclusions and Future Work

5.1 Conclusion

In this research we have introduced SITH-BSP, an algorithmic paradigm able to speedup calculations when performing online POMDP planning, considering general distributions and belief-dependent rewards. The lion share of our approach, definition of simplification over sparse sampling planning, is mathematically formulated in a general manner. It can be easily extended to many directions and as we showed applicable for existing or future approaches. The more specific aspects of our approach (assuming Particle Filter for belief approximation) provides novel derivations for bounds over the differential entropy approximation and may be taken to other areas in the vast field of robotics. We have shown how the approach assumes an adaptive form, while re-using calculations and thus gives the exact optimal solution to the original problem in an effective manner.

Further, we presented a novel method to accelerate information-theoretic reward planning using state of the art MCTS planner. Our approach is applicable with any converging to the reward bounds. We provide thorough proofs that our method is entirely equivalent to PFT-DPW, yielding the same solution and belief tree in each planning step. Our experiments demonstrate that the technique is paramount in terms of computation time compared to PFT-DPW. In the worst-case scenario, the computation time is approaching the baseline. The limitation of our algorithm is that it leans on converging bounds, which are not trivial to derive and specific for a particular reward function. In addition, it requires slightly more caching than the baseline.

5.2 Future work

Many possible extensions can be envisioned. The algorithmic baseline already exists, thus, deriving new converging to the reward bounds can provide a new algorithm by simply plugin them into our existing mechanism. Note this is true for both SITH-BSP and SITH-PFT. Importantly, these bounds don't have to be over information theoretic rewards function. Complex and 'expensive' reward functions can be envisioned.

Additionally, the existing bounds can be extended further to additional POMDP celebrated planners such as DESPOT-α. Finally, the math used in the development of the bounds can be modified into a linear approximation. While it is not tight enough for our proposed algorithm it can be thought of as a *heuristic* tool to guide the tree expansion in MCTS like methods.

Chapter 6

Appendix 1 SITH-BSP

6.1 Proof for Theorem 1

$$
\begin{aligned}
&-\sum_i w_{k+1}^i \cdot \log\left[\mathbb{P}(z_{k+1} \mid x_{k+1}^i)\sum_j \mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right] = \\
&-\sum_{i\in A_{k+1}^s} w_{k+1}^i \cdot \log\left[\mathbb{P}(z_{k+1} \mid x_{k+1}^i)\sum_j \mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right] - \\
&\sum_{i\in \neg A_{k+1}^s} w_{k+1}^i \cdot \log\left[\mathbb{P}(z_{k+1} \mid x_{k+1}^i)\sum_j \mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right] \\
&\geq -\sum_{i\in A_{k+1}^s} w_{k+1}^i \cdot \log\left[\mathbb{P}(z_{k+1} \mid x_{k+1}^i)\sum_j \mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right] - \\
&\sum_{i\in \neg A_{k+1}^s} w_{k+1}^i \cdot \log\left[m \cdot \mathbb{P}(z_{k+1} \mid x_{k+1}^i) \cdot \sum_j w_k^j\right] = \\
&-\sum_{i\in A_{k+1}^s} w_{k+1}^i \cdot \log\left[\mathbb{P}(z_{k+1} \mid x_{k+1}^i)\sum_j \mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right] - \\
&\sum_{i\in \neg A_{k+1}^s} w_{k+1}^i \cdot \log\left[m \cdot \mathbb{P}(z_{k+1} \mid x_{k+1}^i)\right]
\end{aligned}
\tag{6.1}
$$

$$
\begin{aligned}
&-\sum_i w_{k+1}^i \cdot \log\left[\mathbb{P}(z_{k+1} \mid x_{k+1}^i)\sum_j \mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right] = \\
&-\sum_i w_{k+1}^i \cdot \log\left[\sum_{j\in A_k^s} \mathbb{P}(z_{k+1} \mid x_{k+1}^i)\mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right] - \\
&-\sum_i w_{k+1}^i \cdot \log\left[1 + \frac{\sum_{j\in \neg A_k^s} \mathbb{P}(z_{k+1} \mid x_{k+1}^i)\mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j}{\sum_{j\in A_k^s} \mathbb{P}(z_{k+1} \mid x_{k+1}^i)\mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j}\right] \leq \\
&-\sum_i w_{k+1}^i \cdot \log\left[\sum_{j\in A_k^s} \mathbb{P}(z_{k+1} \mid x_{k+1}^i)\mathbb{P}(x_{k+1}^i \mid x_k^j, a_k) w_k^j\right]
\end{aligned}
\tag{6.2}
$$

6.2 Adaptive Simplification Illustrative Example

Consider Fig. 3.1b and assume the subtrees to b_i^1 were solved using simplification levels that hold $s^2 = s^1 + 1, s^2 < s^3, s^4$. Further assume the immediate reward simplification is $s = s^1$. According to definitions above this means that for b_i^1, $s^{j=1} = \min\{s^1, s^{l=1}, s^{l=2}\}$ and $s^{j=2} = \min\{s^1, s^{l=3}, s^{l=4}\}$. Now, we consider the case the existing bounds of the subtrees were not tight enough to prune, we adapt simplification level of the tree starting from $b_i^1 : s^1 \to s^1 + 1$. Since $s^1 < s^1 + 1$ we re-simplify the subtree corresponding to simplification level of s^1 to simplification level $s^1 + 1$, i.e. to a finer simplification.

However we do not need to re-simplify subtrees corresponding to s^2, s^3, s^4: The tree corresponding to s^2 is already simplified to the currently desired level thus we can use its existing bounds. For the two other trees, their current simplification levels, s^3 and s^4, are higher (finer) than the desired $s^1 + 1$ level, and since the bounds are tighter as simplification level increases we can use their existing tighter bounds without the need to 'go-back' to a coarser level of simplification. If we can now prune one of the actions, we keep pruning up the tree. If pruning is still not possible, we need to adapt simplification again with simplification level $s^1 + 2$.

6.3 Additional Entropy Results

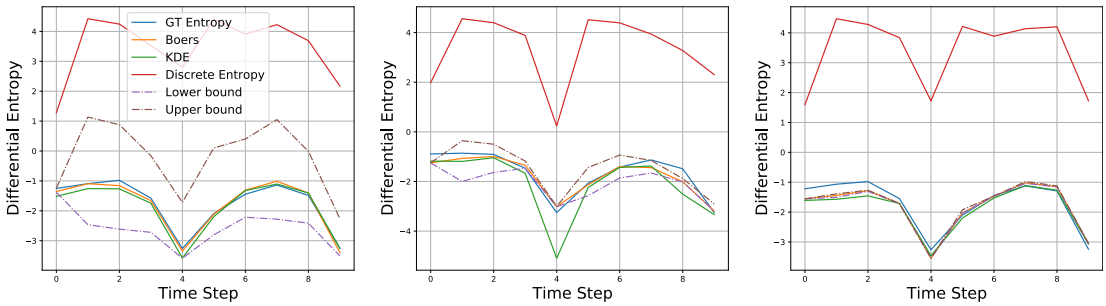


Figure 6.1: Differential Entropy Approximations ans Bounds. Calculations were done using 100 particles. From left to right: Simplification is $N^s = \{0.1, 0.5, 0.9\} \cdot N$

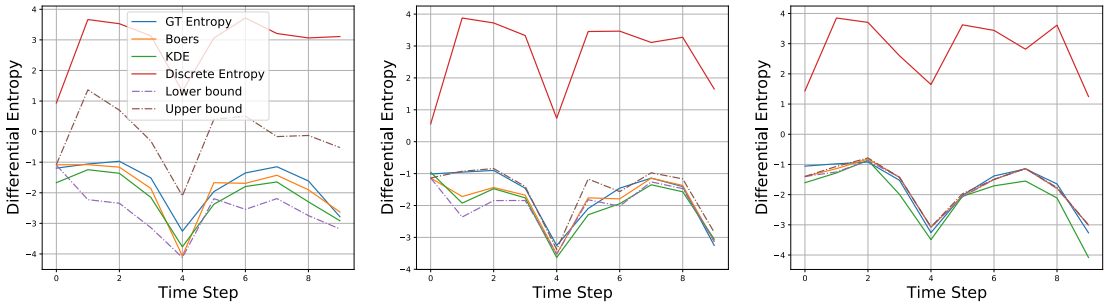


Figure 6.2: Differential Entropy Approximations ans Bounds. Calculations were done using 50 particles. From left to right: Simplification is $N^s = \{0.1, 0.5, 0.9\} \cdot N$

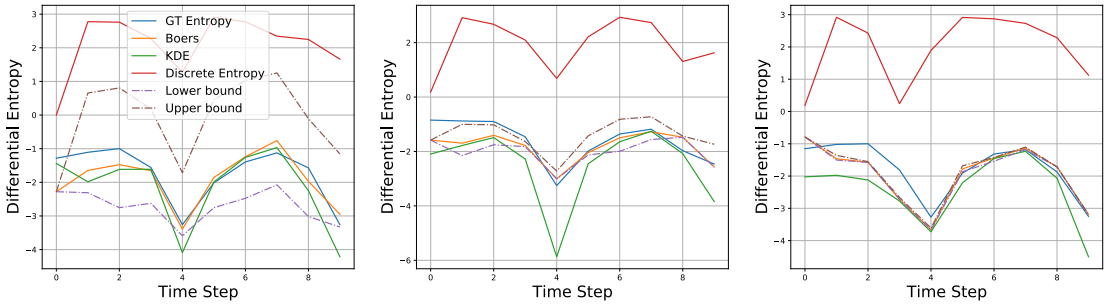


Figure 6.3: Differential Entropy Approximations ans Bounds. Calculations were done using 20 particles. From left to right: Simplification is $N^s = \{0.1, 0.5, 0.9\} \cdot N$

Chapter 7

Appendix 2 SITH-PFT

7.1 Information theoretic bounds

$$u \triangleq -\log\left[\sum_i \mathbb{P}_Z(z_{k+1}|x^i_{k+1})w^i_k\right] + \sum_{i\in\neg A^s_{k+1}} w^i_{k+1}\cdot\log\left[\text{const}\cdot\mathbb{P}_Z(z_{k+1}|x^i_{k+1})\right]$$
$$+\sum_{i\in A^s_{k+1}} w^i_{k+1}\cdot\log\left[\mathbb{P}_Z(z_{k+1}|x^i_{k+1})\sum_j \mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right]$$
$$\ell \triangleq -\log\left[\sum_i \mathbb{P}_Z(z_{k+1}|x^i_{k+1})w^i_k\right] + \sum_i w^i_{k+1}\cdot\log\left[\mathbb{P}_Z(z_{k+1}|x^i_{k+1})\sum_{j\in A^s_k}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right], \tag{7.1}$$

where $\text{const} = \max_{x'} \mathbb{P}_T(x'|x,a)$.

7.1.1 Proofs

Assumptions

For the following proofs (Secs. 7.1.1 and 7.1.1) assume we are using a *converging and finite-time resimplification strategy* that satisfies Assumptions 3.2.1,3.2.2.

Proof for Theorem 2

Proof. We provide proof by induction on the belief trees structure.
Base: Consider an initial given belief node b_0. No actions were taken and no observations were made. Thus, both PFT tree and SITH-PFT trees contain a single identical belief node and the claim holds.
Induction hypothesis: Assume we are given two identical trees with n nodes, generated by PFT and a SITH-PFT. The trees uphold the terms of Definition 1.
Induction step: Assume by contradiction that in the next simulation (expanding the belief tree by one belief node by definition) different nodes were added to the trees.

Thus, we got different trees.
Two different scenarios are possible:

Case 1. The same action-observation sequence $a_0, z_1, a_1, z_2 ... a_m$ was chosen in both trees, but different nodes were added.

Case 2. Different action-observation sequences were chosen for both trees and thus, we got different trees structure.

Case 1 is not possible. Since the Induction hypothesis holds, the last action a_m was taken from the same node denoted h' shared and identical to both trees. Next the same observation model is sampled for a new observation and a new belief node is added with a rollout emanating from it. The new belief nodes and the rollout are identical for both trees since both algorithms use the same randomization seed and the same observation and motion models.

Case 2 must be true since we showed Case 1 is false. There are two possible scenarios such that different action-observation sequences were chosen:

Case 2.1. At some point of the actions-observations sequence, different observations z_i, z_i' were chosen.

Case 2.2. At some point of the actions-observations sequence, PFT chose action $a^\dagger$ while SITH-PFT chose a different action, $\tilde{a}$, or even got stuck without picking any action.

Case 2.1 is not possible since if new observations were made, they are the same one by reasons contradicting Case 1 . If we draw existing observations (choose some observation branch down the tree) the same observations are drawn since they are drawn with the same random seed and from the same observations "pool". It is the same "pool" since the Induction hypothesis holds.

Case 2.2 must be true since we showed Case 2.1 is false, i.e., when both algorithms are at the identical node denoted as h PFT chooses action $a^\dagger$, while SITH-PFT chooses a different action, $\tilde{a}$, or even got stuck without picking any action. Specifically, PFT chooses action $a^\dagger = \arg\max_a$ UCB and SITH-PFT's candidate action is $\tilde{a} = \arg\max_{a \in \mathcal{A}} \underline{\text{UCB}}(ha)$.
Two different scenarios are possible:

Case 2.2.1. the $\overline{\text{UCB}}, \underline{\text{UCB}}$ bounds over $h\tilde{a}$ were tight enough and $\tilde{a}$ was chosen such that $a^\dagger \neq \tilde{a}$.

Case 2.2.2. SITH-PFT is stuck in an infinite loop. It can happen if the $\overline{\text{UCB}}, \underline{\text{UCB}}$ bounds over $h\tilde{a}$, and at least one of its sibling nodes ha, are not tight enough. However,

all of the tree nodes are at the maximal simplification level. Hence, resimplification is triggered over and over without it changing anything.

Case 2.2.1 is not possible since the bounds are analytical (always true) and converge to the actual reward ($\underline{\text{UCB}} = \text{UCB} = \overline{\text{UCB}}$) for the maximal simplification level.

Case 2.2.2 is not possible. If the bounds are not close enough to make a decision, resimplification is triggered. Each time some ha node - sibling to $h\tilde{a}$ and maybe even $h\tilde{a}$ itself is chosen in *Select Best* to over-go resimplification. According to Assumption. 3.2.1 and Assumption. 3.2.2, after some finite number of iterations for all of the sibling ha nodes (including $h\tilde{a}$) it holds $\underline{\text{UCB}}(ha) = \text{UCB}(ha) = \overline{\text{UCB}}(ha)$ and some action can be picked. If different actions have identical values we choose one by the same rule UCB picks actions with identical values (e.g. lower index/random).

Now, since Case 2.2.2 is false, after some finite number of resimplification iterations, SITH-PFT will stop with bounds sufficient enough to make a decision. And since Case 2.2.1 is false it holds that $a^{\dagger} = \tilde{a}$. Thus we get a contradiction and the proof is complete. □

Proof for Theorem 3:

Proof. Since the same tree is built according to Theorem 2, the only modification now is the final criteria at the end of the planning session at the root of the tree: $a^* = \arg\max_a Q(ha)$. Note we can set the exploration constant of UCB to $c = 0$ and we get that UCB is just the Q function. Thus if the bounds are not tight enough at the root to decide on an action, resimplification will be repeatedly called until SITH-PFT can make a decision. The action will be identical to the one chosen by UCB at PFT from similar arguments mentioned in the proof of Theorem 2, 7.1.1. Note that additional final criteria for action selection could be introduced, but it would not matter since tree consistency is kept according to Theorem 2 and the bounds converge to the actual immediate rewards and Q estimations. □

Proof for Theorem 4

We now prove the resimplification strategy described in section 3.2.4 is converging and finite-time resimplification strategy.

Proof: Converging resimplification strategy. Consider the condition for refinement of the bounds (3.22). Since $\frac{1}{d}g(ha)$ is the mean gap over all the nodes that are the descendants to ha, some of the nodes are above this mean gap, and some are under (accounting for the discount factor). We refine all the ones that are above. Further, for each descendant rollout, we refine one rollout node that is above the mean gap. If each time we refine all descendants belief nodes that are above the mean gap and one

rollout node per descendant rollout (if it satisfies (3.22)), after one iteration the mean gap must decrease since there exists a node above the mean gap that got tighter. If there is no such node above the mean gap that means all the values $u' - \ell'$ are the same throughout the sub-tree and those values must be zero since the immediate bounds converge. Thus, the mean gap (and consequentially so does $\overline{\mathrm{UCB}}(ha) - \underline{\mathrm{UCB}}(ha)$) is getting smaller in each iteration unless it is already zero. □

Proof: Finite-time resimplification strategy. Similar to previous proof, in each iteration there exists a node above the mean gap that is chosen for refinement. There are no nodes above the gap only if throughout the sub-tree all the values $u' - \ell'$ are zero. This happens after a finite number of iterations since there is a finite number of nodes and a finite number of simplification levels. Since the bounds converge, at the maximal simplification level it holds $u' = \ell' \Rightarrow u' - \ell' = 0$. Thus, after all nodes in the sub-tree got to the maximal simplification level it holds $\frac{1}{d} g(ha) = 0$ and consequentially so does $\overline{\mathrm{UCB}}(ha) - \underline{\mathrm{UCB}}(ha) = 0 \Rightarrow \overline{\mathrm{UCB}}(ha) = \mathrm{UCB} = \underline{\mathrm{UCB}}(ha))$. □

Time complexity analysis

We turn to analyze the time complexity of our method using the chosen bounds (7.1). We assume the significant bottleneck is querying the motion and observation models $\mathbb{P}_T(x'|x,a), \mathbb{P}_Z(z|x)$ respectively. Assume the belief is approximated by a set of m weighted particles,

$$b = \{x^i, w^i\}_{i=1}^m. \tag{7.2}$$

Consider the Boers et al. [2010] differential entropy approximation for belief at time $k+1$,

$$\hat{\mathcal{H}}(b_k, a_k, z_{k+1}, b_{k+1}) \triangleq \underbrace{\log \left[\sum_i \mathbb{P}_Z(z_{k+1}|x^i_{k+1}) w^i_k \right]}_{a} + \tag{7.3}$$

$$\underbrace{\sum_i w^i_{k+1} \cdot \log \left[\mathbb{P}_Z(z_{k+1}|x^i_{k+1}) \sum_j \mathbb{P}_T(x^i_{k+1}|x^j_k, a_k) w^j_k \right]}_{b} \tag{7.4}$$

Denote the time complexity to query the observation and motion models a single time as t_{obs}, t_{mot} respectively. It is clear from (7.2), (7.3) (term a) and, (7.4) (term b) that:

$$\forall b \text{ as in (7.2) } \Theta(\hat{\mathcal{H}}(b)) = \Theta(m \cdot t_{obs} + m^2 \cdot t_{mot}). \tag{7.5}$$

Since we share calculation between the bounds, the bounds' time complexity, for some level of simplification s, based on Sztyglic and Indelman [2021], is:

$$\Theta(\ell^s + u^s) = \Theta(m \cdot t_{obs} + m^s \cdot m \cdot t_{mot}), \tag{7.6}$$

where m^s is the size of the particles subset that is currently used for the bounds calculations, e.g. $m^s = |A^s|$ (A^s is as in (7.1)) and ℓ^s, u^s denotes the immediate upper and lower bound using simplification level s. Further, we remind the simplification levels are discrete, finite, and satisfy

$$s \in \{1, 2, \ldots, M\}, \quad \ell^{s=M} = \hat{\mathcal{H}} = u^{s=M}. \tag{7.7}$$

Now, assume we wish to tighten ℓ^s, u^s and move from simplification level s to $s+1$. Since the bounds are updated incrementally (as introduced by Sztyglic and Indelman [2021]), when moving from simplification level s to $s+1$ the only additional data we are missing are the new values of the observation and motion models for the newly added particles. Thus, we get that the time complexity of moving from one simplification level to another is:

$$\Theta(\ell^s + u^s \to \ell^{s+1} + u^{s+1}) = \Theta((m^{s+1} - m^s) \cdot m \cdot t_{mot}), \tag{7.8}$$

where $\Theta(\ell^s + u^s \to \ell^{s+1} + u^{s+1})$ denotes the time complexity of updating the bounds from one simplification level to the following one. Note the first term from (7.6), $m \cdot t_{obs}$, is not present in (7.8). This term has nothing to do with simplification level s and it is calculated linearly over all particles m. Thus, it is calculated once at the beginning (initial/lowest simplification level).

We can now deduce using (7.6) and (7.8)

$$\Theta(\ell^{s+1} + u^{s+1}) = \Theta(\ell^s + u^s) + \Theta(\ell^s + u^s \to \ell^{s+1} + u^{s+1}). \tag{7.9}$$

Finally, using (7.5), (7.6), (7.7), (7.8), and (7.9), we come to the conclusion that if at the end of a planning session, a node's b simplification level was $1 \leq s \leq M$ than the time complexity saved for that node is

$$\Theta((m - m^s) \cdot m \cdot t_{mot}). \tag{7.10}$$

This makes perfect sense since if we had to resimplify all the way to the maximal level we get $s = M \Rightarrow m^{s=M} = m$ and by substituting $m^s = m$ in (7.10) we saved no time at all.

To conclude, the total speedup of the algorithm is dependent on how many belief nodes' bounds were not resimplified to the maximal level. The more nodes we had at the end of a planning session with lower simplification levels, the more speedup we get according to (7.10).

Additional resimplification strategies

We note that the proofs for Theorems 2, 3 depends on our resimplification strategy 3.2.4. That is, additional strategies can be introduced as long as they satisfy Assumptions 3.2.1, and 3.2.2. To clarify, a simple example of a converging and finite-time resimplification strategy would be to refine the bounds of all nodes (belief tree nodes and rollout nodes) that are descendants to the belief-action node ha that was chosen for resimplification at *Select Best* procedure. Naturally, there will always be a node that got tightened (unless all bounds are already equal); thus, Assumption. 3.2.1 is satisfied. Further, after a finite time, all nodes in the sub-tree got to the maximal level of simplification, and the bounds converged. Thus, Assumption. 3.2.2 is satisfied. Note that using this brute-force strategy can result in many unnecessary resimplifications. So, the potential speed-up may decrease but in the worst case, SITH-PFT will still yield the same time complexity as PFT.

Bibliography

Mauricio Araya, Olivier Buffet, Vincent Thomas, and Françcois Charpillet. A pomdp extension with belief-dependent rewards. In *Advances in Neural Information Processing Systems (NIPS)*, pages 64–72, 2010.

R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. ISBN 9780691079516.

Y. Boers, H. Driessen, A. Bagchi, and P. Mandal. Particle filter based entropy. In *2010 13th International Conference on Information Fusion*, pages 1–8, 2010. doi: 10.1109/ICIF.2010.5712013.

Louis Dressel and Mykel J. Kochenderfer. Efficient decision-theoretic target localization. In Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith, editors, *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*, pages 70–78. AAAI Press, 2017. URL `https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15761`.

Maxim Egorov, Zachary N. Sunberg, Edward Balaban, Tim A. Wheeler, Jayesh K. Gupta, and Mykel J. Kochenderfer. POMDPs.jl: A framework for sequential decision making under uncertainty. *Journal of Machine Learning Research*, 18(26):1–5, 2017. URL `http://jmlr.org/papers/v18/16-300.html`.

Khen Elimelech and Vadim Indelman. Simplified decision making in the belief space using belief sparsification. *Intl. J. of Robotics Research*, 2021. Accepted.

Mathieu Fehr, Olivier Buffet, Vincent Thomas, and Jilles Dibangoye. rho-pomdps have lipschitz-continuous epsilon-optimal value functions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6933–6943. Curran Associates, Inc., 2018.

Johannes Fischer and Omer Sahin Tas. Information particle filter tree: An online algorithm for pomdps with belief-based rewards on continuous domains. In *Intl. Conf. on Machine Learning (ICML)*, Vienna, Austria, 2020.

Neha P Garg, David Hsu, and Wee Sun Lee. Despot-α: Online pomdp planning with large state and observation spaces. In *Robotics: Science and Systems (RSS)*, 2019.

Marcus Hoerger, Hanna Kurniawati, and Alberto Elfes. Multilevel monte-carlo for solving pomdps online. In *Proc. International Symposium on Robotics Research (ISRR)*, 2019.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2): 193–208, 2002.

M. Kochenderfer, T. Wheeler, and K. Wray. *Algorithms for Decision Making*. MIT Press, 2022.

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

Michael H. Lim, Claire Tomlin, and Zachary N. Sunberg. Sparse tree search optimality guarantees in pomdps with continuous observation spaces. In *Intl. Joint Conf. on AI (IJCAI)*, pages 4135–4142, 7 2020a.

Michael H Lim, Claire J Tomlin, and Zachary N Sunberg. Voronoi progressive widening: Efficient online solvers for continuous space mdps and pomdps with provably optimal components. *arXiv preprint arXiv:2012.10140*, 2020b.

C. Papadimitriou and J. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

Nicholas Roy, Geoffrey J Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *J. Artif. Intell. Res.(JAIR)*, 23:1–40, 2005.

M. Shienman, A. Kitanov, and V. Indelman. Ft-bsp: Focused topological belief space planning. *IEEE Robotics and Automation Letters (RA-L)*, 6(3):4744–4751, July 2021.

David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2164–2172, 2010.

A. Singh, A. Krause, C. Guestrin, and W.J. Kaiser. Efficient informative sensing using multiple robots. *J. of Artificial Intelligence Research*, 34:707–755, 2009.

T. Smith and R. Simmons. Heuristic search value iteration for pomdps. In *Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 520–527, 2004.

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Robotics: Science and Systems (RSS)*, pages 65–72, 2005.

Zachary Sunberg and Mykel Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 2018.

O. Sztyglic, A. Zhitnikov, and V. Indelman. Simplified belief-dependent reward mcts planning with guaranteed tree consistency. Technical report, 2021.

Ori Sztyglic and Vadim Indelman. Online pomdp planning via simplification. *arXiv preprint arXiv:2105.05296*, 2021.

Vincent Thomas, Gérémy Hutin, and Olivier Buffet. Monte carlo information-oriented planning. *arXiv preprint arXiv:2103.11345*, 2021.

Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. *JAIR*, 58:231–266, 2017.

שעושה הנחה דומה נקראת חיפוש מונטה קרלו בעץ (MCTS - Monte Carlo Tree Search). שיטה זו ידועה כמוצלחת בהתמודדות עם שתי הקללות המדוברות שכן היא מייצגת את מרחב המצב על ידי מספר סופי של דגימות וכן אינה בונה את העץ במלואו אלא רק "חושפת" חלקים מבטיחים מעץ האמונה על ידי שימוש ביוריסטיקות ובנייה הדרגתית של העץ. ברוב העבודות העושות שימוש בשיטות של MCTS מניחים כי הגמול/פרס שהסוכן מקבל הוא תלוי מרחב המצב. זו הנחה מגבילה שכן היא אינה מאפשרת פרסי אינפורמציה תאורטית (information theoretic rewards). פרסים אלו כגון אנטרופיה דיפרנציאלית הוכיחו את עצמם כשימושיים ביותר במגוון רחב של אפליקציות בהן לסוכן חשוב להקטין את חוסר הוודאות שלו (כגון מיפוי מערה שקרסה, משימות חיפוש וחילוץ ועוד). אחד החסרונות של פרסים אלו הוא שהם יכולים להיות יקרים לחישוב. בעבודה זו שלנו אנו מבצעים תכנון במרחב האמונה ההסתברותי בשילוב עם פרסי אינפורמציה תאורטית. אנו עושים שימוש ברעיון ההפשטה (simplification) של חלקים מהבעיה המקורית על מנת לספק את אותו פתרון מייטבי כמו שיטות קודמות לנו רק בצורה מהירה יותר. פיתחנו חסמים אנליטיים מבוססי הפשטה עבור קירוב ידוע מבוסס חלקיקים לאנטרופיה הדיפרנציאלית ואנו מראים כיצד חסמים אלו הופכים להדוקים ואף מגיעים להתכנסות לקירוב המקורי על פי דרישה. בעזרת חסמים אלו אנו מציגים שני אלגוריתמים חדשים SITH-BSP ו- SITH-PFT. הקו המנחה בעבודה זו הוא שאנו יכולים להימנע מחישוב פרס האינפורמציה התאורטי היקר על ידי חישוב החסמים ובאופן זה לפסול ענפי העץ בכל פעם על ידי הידוק הגבולות עבור כל צומת בעץ רק על פי הנדרש. באופן זה אנו מייצרים פתרון זהה לחלוטין לפתרון המקורי שעושה שימוש בפרס היקר לחישוב, אלא שאנו עושים זאת מהר יותר. האלגוריתם הראשון בנוי על סכמה הידועה בשם דגימה דלילה (sparse sampling) ונועדה להוות אבן בסיס המכילה את כל הפיתוחים המתמטיים הבסיסיים המדגימים כיצד יש לשלב את החסמים בפתרון בעיית תכנון שכזו. האלגוריתם השני מדגים כיצד ניתן לקחת את השיטה כללית הזו ולשלב אותו באלגוריתם חדיש (state of the art) ולהגיע לשיפור בביצועים במובן זמן הריצה. שתי השיטות הללו נהנות מסכמת פעולה כללית שיכולה לקבל חסמים מתכנסים אחרים על פרסים נוספים. בנוסף שתי השיטות הן הדרגתיות, כלומר, מתאימות עצמן לפי הצורך לכמה שצריך להדק את החסמים ואינן מהדקות אותם עד כדי התכנסות בתחילת התהליך. לבסוף אנו מראים כיצד ניתן טרם הידוק החסמים להיעזר בחישובים שנעשו בפעם הקודמת שהחסמים הודקו ובכך לחסוך זמן ריצה יקר. אנו מוודאים כי הגישה שלנו עליונה במובן זמן ריצה וזהה במובן ביצועים על ידי סימולציות מחשב.

תקציר

בתחומי הבינה המלאכותית והרובוטיקה, ניווט אוטונומי בסביבה לא ידועה מהווה בעיה קשה ומרכזית. בבעיות מסוג זה, הנעלמים כגון מיקום הסוכן או הרובוט אינם ידועים בדיוק כתוצאה ממידע שמכיל אי וודאות המגיע מהסביבה (כגון מדידות רועשות). אחת הדרכים למדל בעיה שכזו המכילה אי וודאות היא הנחת מודל מסוג *תהליך החלטה מרקובי מובחן למחצה* (POMDP). תחת הנחת מודל זה, במקום גישה למרחב המצב של הסוכן (ואולי אף של העולם) לסוכן יש פילוג (distribution) על מרחב המצב הידוע בשם *אמונה* (belief). כעת תכנון (כגון תכנון מסלול תנועה) מתבצע במרחב ההסתברותי של האמונה (BSP – belief space planning). תכנון נעשה על ידי שימוש באמונה ההתחלתית שיש לסוכן ובמודלים הסתברותיים שלסוכן יש גישה אליהם – מודל התנועה ומודל המדידה. בעזרת אלו, ניתן על ידי ביצוע פעולה ולאחר מכן קבלת מדידה לעדכן את האמונה וכמו כן לקבל גמול/פרס עבור הפעולה שנבחרה. בעזרת התהליך המתואר, הסוכן יכול לתכנן קדימה אל העתיד על מנת לנסות למצוא את מדיניות הפעולה (*policy*) המיטבית, כלומר מדיניות שתמקסם את ערך הפרסים שהוא מקבל לאורך הדרך. ביכולתו לעשות זאת על ידי לקיחה בחשבון של רצפי פעולות – מדידות שונים. הדרך המקובלת לבחון רצפים עתידיים כאלו היא על ידי בניית עץ האמונה (belief tree) ששורשו הוא האמונה ההתחלתית וצמתי העץ במורדו מתאימות לפעולות ומדידות עתידיות שונות. פתרון POMDP בצורה מדויקת שייכת למחלקת הבעיות ה PSPACE-Complete. הקושי בפתירת בעיות שכאלו אף מורגש יותר כאשר על הסוכן לפתור בעיה שכזו בכל צעד תחת מגבלת זמן, קרי, תכנון מקוון (online planning). אי לכך מדענים רבים פיתחו מגוון שיטות יעילות יותר על מנת לקרב את הפתרון המדויק מבלי לפתור את הבעיה במלואה, בתקווה שהפתרון המקורב יהיה מספיק קרוב לפתרון המיטבי. כאמור בניית עץ אמונה הוא אחת השיטות הקיימות לקרב פתרון שכן במקום לבנות את העץ במלואו (על ידי התחשבות בכל רצפי הפעולות-מדידות האפשריים) ניתן לבנות אותו עד גודל מוגדר מראש ו\או תחת מגבלת זמן. כאשר בונים את העץ, ענפיו מתפצלים בכל פעם ששוקלים מספר פעולות ושוב כאשר שוקלים מספר מדידות שונות. אי לכך הגידול בצמתי העץ הוא מעריכי ככל שבונים אותו לעומק גדול יותר. בנוסף ככל שמימד מרחב המצב גדל כמות המצבים האפשריים גדלה מעריכית במימד ובאופן דומה גם האמונה. שתי הבעיות שתוארו ידועות בתור קיללת ההיסטוריה (curse of history) וקיללת המימד (curse of dimensionality). בעבודה זו אנו מניחים כי האמונה יכולה להיות מיוצגת על ידי מודל א-פרמטרי, ויכולה להיות מקורבת על ידי אוסף של דגימות ממושקלות ממרחב המצב הידועות בתור חלקיקים. שיטת פתרון מקורב פופולרית ומוצלחת

המחקר בוצע בהנחייתו של פרופסור חבר ואדים אינדלמן, בפקולטה למדעי המחשב.

חלק מן התוצאות בחיבור זה פורסמו או הוגשו כמאמרים מאת המחבר ושותפיו למחקר בכנסים ובכתבי-עת במהלך תקופת מחקר המאסטר של המחבר, אשר גרסאותיהם העדכניות ביותר הינן:

Ori Sztyglic and Vadim Indelman. Online pomdp planning via simplification. arXiv preprint arXiv:2105.05296, 2021. Submitted to RA-L 2022

O. Sztyglic, A. Zhitnikov, and V. Indelman. Simplified belief-dependent reward mcts planning with guaranteed tree consistency. Technical report, 2021. Submitted to AAAI 2022

תודות

אני רוצה להודות למנחה שלי פרופסור ואדים אינדלמן על שלימדת אותי כיצד לבצע מחקר בסטנדרטים הגבוהים ביותר. אני גם רוצה להודות למשפחתי הקרובה מירב, ניצן, יורם ורוקסנה על שתמכתם בי לאורך כל שנות לימודיי (ועדיין תומכים).

לסיום ברצוני להודות לאשתי יעל שהייתה שם בשבילי מדי יום.

אני מודה לטכניון על התמיכה הנדיבה במשך השתלמותי.

תכנון מקוון בתהליך החלטה מרקובי ניתן לצפייה חלקית ע״י הפשטה

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים במדעי המחשב

אורי שטיגליץ

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
תשרי התשפ״ב חיפה אוקטובר 2021

תכנון מקוון בתהליך החלטה מרקובי ניתן לצפייה חלקית ע״י הפשטה

אורי שטיגליץ