Previous Knowledge Utilization In Online and Non-Parametric Belief Space Planning

Michael Novitsky

Previous Knowledge Utilization In Online and Non-Parametric Belief Space Planning

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Autonomous Systems
and Robotics

Michael Novitsky

Submitted to the Senate
of the Technion — Israel Institute of Technology
Shevat 5785 Haifa February 2025

This research was carried out under the supervision of Associate Prof. Vadim Indelman, in the Technion Autonomous Systems and Robotics Program.

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

M. Novitsky, M. Barenboim, and V. Indelman. "Previous Knowledge Utilization In Online Belief Space Planning." In: *IEEE Robotics and Automation Letters (RA-L). Submitted.* 2024. arXiv: 2412.13128.

Acknowledgements

I wish to express my gratitude to my advisor, Associate Professor Vadim Indelman, for his invaluable guidance, support, and patience throughout my Master's studies. His ability to provide direction while allowing me the freedom to explore has been instrumental. Additionally, I am profoundly thankful to my family, and especially to my wife Anaelle, for their unwavering support and encouragement throughout this journey.

The Technion's funding of this research is hereby acknowledged.

Contents

List of Figures

Abstract

Online planning under uncertainty remains a critical challenge in robotics and autonomous systems. In real-world scenarios, autonomous agents often lack direct access to the true state of the environment and instead maintain a belief (distribution over possible states). Uncertainty in such systems arises from various sources, including imperfect perception due to sensor noise and limited information. Partially Observable Markov Decision Processes (POMDPs) provide a well-established mathematical framework for such problems.

Tree search techniques are widely used to create partial future trajectories within computational limits. However, a significant limitation of most existing methods is their tendency to discard information from previous planning sessions and start planning from scratch each time. This study introduces a novel, computationally efficient approach that incorporates historical planning data into the current decision-making process. We address a broad category of problems with continuous state, action, and observation spaces, where rewards are general functions of belief. Our theoretical framework for information reuse is based on Multiple Importance Sampling (MIS) and demonstrates how it can be applied to estimate action-value functions using expert demonstrations without planning. We build upon this approach by developing our algorithm, IR-PFT, which leverages Monte Carlo Tree Search (MCTS) and incorporates trajectory reuse from previous planning sessions into the ongoing planning process. Experimental results reveal that our method not only significantly reduces computation time but also maintains high performance levels. Our findings suggest that the integration of historical planning information can substantially enhance the efficiency of online decision-making in uncertain environments, ultimately leading to more responsive and adaptive autonomous systems.

Notation and Abbreviations

S	State Space
A	Action Space
O	Observation Space
T	State Transition Model
Z	Observation Model
$r(\cdot)$	Reward function over state or belief
$\rho(\cdot)$	Reward function over belief
b_0	Initial belief at time index 0
b_k	Posterior Belief at time index k
b_k^-	Propagated belief at time index k
s_k	State at time k
a_k	Action at time k
o_k	Observation at time k
H_k	History of prior belief actions and observations up to time k
H_k^-	History of prior belief actions and observations up to time $k-1$
π	Policy function which maps belief to action
$\cdot_{k:k+d}$	Sequence from time index k to time index $k+d$
$\pi_{k:k+d-1}$	Sequence of policy functions for horizon d which map belief to action
$V^\pi(\cdot)$	Value function given policy π
$Q^\pi(\cdot, a)$	Action value function given policy π and action a
G_k	Return following time k
B	Belief space
$\hat{b}_k$	Posterior belief based on state samples at time index k
$\hat{b}_k^-$	Propagated belief based on state samples at time index k
s_k^i	The i-th state particle in belief $\hat{b}_k$
$\hat{E}_p^{IS}[f(x)]$	Importance sampling estimator of $E[f(x)]$
x^i	i-th sample in the importance sampling estimator of $E[f(x)]$
w_{IS}^i	i-th weight in the importance sampling estimator of $E[f(x)]$
$\hat{E}_p^{MIS}[f(x)]$	Multiple Importance Sampling estimator of $E[f(x)]$
$x^{i,m}$	The i-th sample in m-th importance sampling distribution

w_{MIS}^m	m-th weight in the multiple importance sampling estimator of $E[f(x)]$
q_m	m-th distribution in the multiple importance sampling estimator of $E[f(x)]$
n_m	Number of samples from m-th importance sampling distribution
n_{avg}	Average number of samples in importance sampling distributions
$G_{PF(m)}(b, a)$	Particle filter belief update performed with m particles and a simulated observation
$N(b_k)$	Number of times belief b_k was visited in the MCTS tree
$N(b_k, a_k)$	Number of Executions of action a_k from belief b_k in the MCTS tree
$N(b_k^-)$	Number of times propagated belief b_k^- was visited in the MCTS tree
$Q(\cdot, a)$	Action value function in MCTS algorithm
$\hat{Q}^\pi(\cdot, a)$	Sample-based action value function estimator given policy π and action a
$\hat{Q}_{IS}^\pi(\cdot, a)$	Importance sampling action value function estimator given policy π and action a
$\hat{Q}_{MIS}^\pi(\cdot, a)$	Multiple Importance Sampling action value function estimator given policy π and action a
D	Dataset of trajectories and returns
τ^i	i-th trajectory in dataset D
τ_{suffix}^i	i-th trajectory suffix
τ'^i	i-th reused trajectory
$G_{k_i}^i$	i-th return in dataset D following time k_i
$\tilde{G}_k^i$	Adjusted i-th return following time k_i
$b_{k_m}^m$	Posterior belief at time index k_m marking the beginning of trajectory τ^m
$a_{k_m}^m$	Action at time index k_m of τ^m
$\tau^{l,m}$	l-th trajectory from m-th distribution
$G_k^{l,m}$	l-th return from m-th distribution
$\tilde{G}_k^{l,m}$	Adjusted l-th return from m-th distribution following time k
$b_{k_m}^{-l,m}$	l-th Propagated belief from m-th distribution at time index k_m
$C(b_{k_m}, a_{k_m})$	Set of propagated belief children of belief b_{k_m} and action a_{k_m}
$\tilde{G}_k^{m,l,y}$	Extended y-th return from l-th propagated belief child from m-th distribution following time k

$\tilde{G}_k^{m,l,y}$	Adjusted y-th return from l-th propagated belief child from m-th distribution following time k	
$d_{prev}^{l,m}$	Horizon of propagated belief $b_{k_m+1}^{-l,m}$ before reuse	
π_b	Behavioral policy in off-policy evaluation	
π_t	Target policy in off-policy evaluation	
$\pi_{rollout}$	Rollout policy in MCTS	
b_{MLE}^{-}	Maximum likelihood propagated belief	
$f_D(\cdot,\cdot)$	Distance function between propagated beliefs	
AI-FSSS	Adaptive Information Forward Search Sparse Sampling	
BSP	Belief Space Planning	7
DAG	Directed Acyclic Graph	10
DESPOT	Determinized Sparse Partially Observable Tree	
DPW	Double Progressive Widening	14
IPFT	Information Particle Filter Tree	
IR-PFT	Incremental Reuse Particle Filter Tree	8
IS	Importance Sampling	13
iX-BSP	Incremental Belief Space Planning	
KDE	Kernel Density Estimation	7
LABECOP	Lazy Belief Extraction for Continuous Observation POMDPs	
MCTS	Monte Carlo Tree Search	8
MDP	Markov Decision Process	12
MIS	Multiple Importance Sampling	8
PFT	Particle Filter Tree	
PFT-DPW	Particle Filter Tree - Double Progressive Widening	14
POMCP	Partially Observable Monte Carlo Planning	
POMCPOW	Partially Observable Monte Carlo Planning With Observation Widening	
POMDP	Partially Observable Markov Decision Process	7
SITH	Simplified Information Theoretic	
UCB	Upper Confidence Bound	9
UCD	Upper Confidence Bound For Directed Acyclic Graphs	10
UCT	Upper Confidence Tree	9

Chapter 1

Introduction

1.1 Planning Under Uncertainty

Autonomous robots and systems heavily rely on effective planning and decision-making processes. In real-world scenarios, these agents often lack direct access to the true state of the environment and instead maintain a belief (distribution over possible states). Uncertainty in such systems arises from various sources, including imperfect perception due to sensor noise and limited information. To tackle this challenge, Partially Observable Markov Decision Process (POMDP) provides a well-established mathematical framework. Solving POMDPs exactly is computationally demanding, as it falls into the PSPACE-complete complexity class [18]. This computational hardness primarily stems from two main factors: the curse of history and the curse of dimensionality. Furthermore, many real-world problems involve continuous state and observation spaces, further exacerbating the computational challenge. Recent advancements have led to the proposal of online algorithms [19] [20] [12] for solving POMDPs. These algorithms operate within limited budget constraints, such as restricted time or number of iterations, and employ a sampling-based approach to construct partial trees and search for the optimal action that maximizes the expected cumulative reward. By sampling a subset of the belief space, these algorithms effectively address both the curse of history and the curse of dimensionality, which are key obstacles in solving POMDPs.

In POMDPs, the reward function of a belief node is typically formulated as the expected reward over states. However, this formulation may be insufficient for certain problems, such as information gathering and active sensing. In such cases, the problem is commonly addressed as Belief Space Planning (BSP) or ρ-POMDP [1], where the reward is defined over the belief itself.

Information-theoretic measures, such as information gain and differential entropy, are commonly used to quantify uncertainty in the decision-making process [10]. However, exact calculation of information-theoretic rewards becomes intractable for general distributions, as it requires integrating over all possible states. To address this challenge, approximation methods such as Kernel Density Estimation (KDE) and particle

filter estimation [3] have been proposed in the literature. Nonetheless, these methods still incur significant computational expenses, with computation complexity scaling quadratically with the number of samples. As reward calculation is performed for each node in the tree, it becomes the primary source of computational complexity in online planning algorithms.

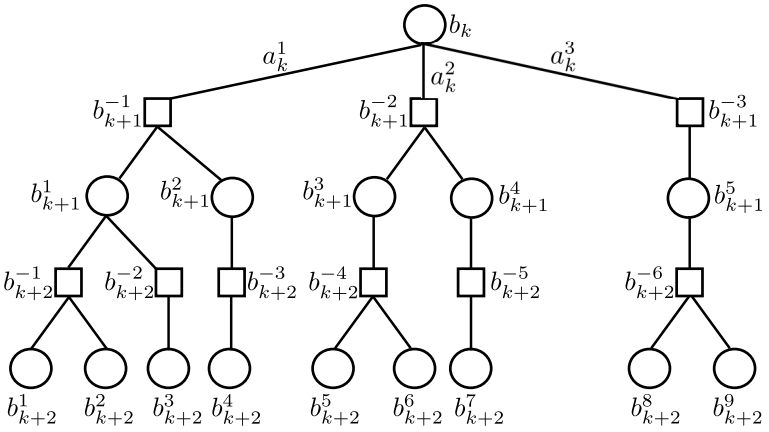

Figure 1.1: Illustration of a belief tree.

The main objective of this paper is to improve planning efficiency within a non-parametric setting, continuous state, action and observation spaces, and general reward functions. To address these challenges, we propose a novel approach that leverages the Multiple Importance Sampling (MIS) framework [27] to tackle the problem of reusing information from previous planning sessions. Figure 1.1 illustrates a depth-2 belief tree generated at time step k. The circles represent beliefs, while the squares denote propagated beliefs. In the proposed approach, this tree will be utilized in subsequent planning sessions to accelerate the estimation of the action-value function at a future time $k' > k$.

Our approach introduces a new algorithm specifically designed to utilize knowledge gathered during prior planning sessions. We demonstrate how our method can be integrated with Monte Carlo Tree Search (MCTS) to create a novel online algorithm called Incremental Reuse Particle Filter Tree (IR-PFT).

To assess the effectiveness of our algorithm, we conduct experiments in an online planning setting and evaluate its performance. Our results show a substantial decrease in planning time without compromising overall performance, underscoring the effectiveness of our proposed approach.

1.2 Related Work

1.2.1 Online Algorithms

The problem of solving POMDPs is notoriously challenging. Nevertheless, recent years have seen remarkable progress in this field. One notable development is the introduc-

tion of the POMCP algorithm [19], which extends the Upper Confidence Tree (UCT) algorithm [14] to handle partially observable settings. In the POMCP algorithm, during each simulation, a state particle is randomly sampled from the current belief. This particle is then propagated through the search tree, and at each belief node, information such as visitation count and average accumulated reward statistics is recorded. The action selection process employs a Upper Confidence Bound (UCB) formulation [17]. The authors assume discrete state, action and observation spaces, and a state-based reward function. The number of samples in each belief node in POMCP is determined by the number of simulations in which it participated and makes less visited nodes to be composed of fewer samples.

POMCPOW [21] extends POMCP to continuous state, action and observation spaces by incorporating progressive widening and representing beliefs as sets of weighted particles. An access to observation likelihood model is assumed to be given and at each step the simulated state is inserted into the weighted belief and then a new state is sampled from that belief according to its weight (observation likelihood).

In PFT-DPW [21] the authors adopt particle filter formulation for belief update and each belief is represented with a constant number of samples and enables calculation of information-theoretic rewards.

[24] introduce $\rho-$POMCP which propagates a set particles in each simulation using particle filter and adds it to existing particles in visited nodes. Nodes that are visited more often will have an increasingly better representation and the authors provide an asymptotical proof of convergence when ρ is continuous and bounded.

[10] introduce IPFT algorithm that extends PFT [21]. They consider reward that is a linear combination of differential entropy and expected state reward. At each simulation paticle set is sampled from root belief and propagated through the tree. Entropy estimates are then averaged over different particle sets in each belief node and the average value is used to estimate differential entropy.

[12] propose LABECOP - an algorithm that deals with continuous observations space. At each visit to new belief node b, state particle s is sampled, action a is chosen according to modified UCB formulation and new observation o is sampled from s. Previously generated states from b, a are rewieghted according to o and their values are used to get better estimation of the current value function $\hat{Q}(b, a)$.

SITH-BSP[23, 30] and AI-FSSS[2] make use of simplification of reward function calculation and observation space sampling accordingly, while preserving action consistency.

[29] quantify the effect of applying simplification and extend ρ-POMDP to $\mathbb{P}\rho-$POMDP, while providing stochastic bounds on the return.

Despot [20] and the follow up works [28], [11], [5] propose a different family of algorithms that utilize determinized random sampling to build the search tree incrementally and recent works also tackle large observation spaces [11]. The usage of $\alpha-$vectors in [28], [11], [5] limits the application to POMDPs with state dependent reward functions.

All of the aforementioned methods start each planning session from scratch without utilizing information from previous planning sessions.

1.2.2 Information Reuse In Online Algorithms

In iX-BSP [9], [8]the authors propose reuse algorithm, however they assume an open loop setting and do not address non-parametric beliefs.

In [6], the authors investigate transpositions in MCTS by representing states as a Directed Acyclic Graph (DAG) to improve state reuse through modified backpropagation strategies. They propose the Upper Confidence Bound For Directed Acyclic Graphs (UCD) framework, which enables the aggregation of information across shared states (transpositions). This approach assumes a fully observable state and aggregates information only when a state is revisited exactly during the current planning session.

1.3 Contributions

In our work, we consider a partially observable setting where the exact state is unknown, and information is reused even when the belief is not reached exactly during planning. We address continuous state, action, and observation spaces, incorporating general belief-dependent rewards, a non-parametric framework, and a closed-loop control setting. Our work makes three key contributions.

Incremental Multiple Importance Sampling Update

We present an efficient method for updating the MIS estimator incrementally upon the receipt of new samples by utilizing additional memory.

Experience-Based Value Function Estimation

We demonstrate the application of the MIS estimator for estimating the action-value function. By leveraging trajectories generated by an expert agent, we show that the action-value function can be estimated directly from pre-existing data, without additional planning.

IR-PFT Algorithm

We build upon the efficient MIS update and experience-based value function estimation to introduce IR-PFT, an online algorithm based on MCTS. This approach accelerates computations by reusing data from prior planning sessions.

Chapter 2

Background

2.1 POMDPs

POMDP is a 7-tuple (S, A, O, T, Z, r, b_0), where S, A and O correspond to state, action and observation spaces.

$$T \triangleq \mathbb{P}(s_{k+1}|s_k, a_k) \tag{2.1}$$

is the state transition density function,

$$Z \triangleq \mathbb{P}(o_{k+1}|s_{k+1}) \tag{2.2}$$

is the observation density function, $r(b, a, b')$ represents the reward function based on the current belief b, the action a, and the subsequent belief b', while b_0 denotes the current belief over states. We denote by $H_k = (b_0, a_0, o_1, a_1, o_2, ..., a_{k-1}, o_k) = \{b_0, o_{1:k}, a_{1:k-1}\}$ the history up to time k, which consists of a series of actions made and observations received. Similarly, we define $H_k^- = (b_0, a_0, o_1, a_1, o_2, ..., a_{k-1}) = H_k \setminus \{o_k\}$. Since the exact state of the world is not known and we only receive observations, a probability distribution (belief) over states is maintained $b_k = \mathbb{P}(s_k|H_k)$. It is assumed that the belief is sufficient statistics for the decision making and a Bayesian update is used to update the belief recursively:

$$b_{k+1} = \eta \mathbb{P}(o_{k+1}|s_{k+1}) \int_{s_k} \mathbb{P}(s_{k+1}|s_k, a_k) b_k ds_k. \tag{2.3}$$

where η is a normalization term.

The belief update can be split into two main steps, first we make a prediction using the transition model and calculate a propagated belief b_{k+1}^-,

$$b_{k+1}^- = \mathbb{P}(s_{k+1}|H_{k+1}^-) = \int_{s_k} \mathbb{P}(s_{k+1}|s_k, a_k) b_k ds_k. \tag{2.4}$$

and then we use the observation model tocalculate a posterior belief b_k,

$$b_{k+1} = \mathbb{P}(s_{k+1}|H_{k+1}) = \eta\mathbb{P}(o_{k+1}|s_{k+1})b_{k+1}^{-}. \tag{2.5}$$

A policy $\pi \in \Pi$ is a mapping from belief space to action space $\pi : b \to a$. We define the value function V^{π} for any policy π and horizon d as

$$V^{\pi}(b_k) = \mathop{\mathbb{E}}_{b_{k+1:k+d}}[G_k|b_k, \pi]. \tag{2.6}$$

where $\pi \triangleq \pi_{k:k+d-1}$ represents a sequence of policies for horizon d and

$$G_k = \sum_{i=k}^{k+d-1} r(b_i, \pi_i(b_i), b_{i+1}) \tag{2.7}$$

is the return. Similarly, we define the action value function Q^{π} as

$$Q^{\pi}(b_k, a) = \mathbb{E}_{b_{k+1}}[r(b_k, a, b_{k+1}) + V^{\pi}(b_{k+1})]. \tag{2.8}$$

2.2 Belief MDP

Each POMDP problem can be viewed as and Markov Decision Process (MDP) (B, A, τ, r, b_0) over the belief space assuming markovian belief state where B is the space of all possible beliefs over states, A, r and b_0 are same as in POMDP definition and $\tau(b_{k+1}|b_k, a_k)$ is the belief transition function [13]

$$\tau(b_{k+1}|b_k, a_k) \triangleq \mathbb{P}(b_{k+1}|b_k, a_k) \tag{2.9}$$

2.3 Non-Parametric Setting

In our work we assume a non-parametric setting, where we use collections of state particles to estimate complex belief distributions. We leverage the particle filter method [25] to update our approximations of posterior distributions as we receive new observations from the environment.

Since we use a finite number of samples to represent each belief, we do not have access to the theoretical belief b_k. Instead, we rely on an approximation, assuming resampling at each step of the particle filter. m is the total number of particles, and each state s_k^i represents the i-th particle. Since we assume resampling, the weights are uniform and equal to $\frac{1}{m}$.

$$\hat{b}_k = \frac{1}{m}\sum_{i=1}^{m} \delta(s - s_k^i). \tag{2.10}$$

Given resampled belief $\hat{b}_k$, action a_k, and propagated belief $\hat{b}_{k+1}^{-}$, calculating the prob-

ability $\mathbb{P}(\hat{b}^-_{k+1}|\hat{b}_k, a_k)$ involves determining all the matchings between the states in $\hat{b}_k$ and those in $\hat{b}^-_{k+1}$ which is $\sharp P$-complete [26]. We assume, similar to [15], that the beliefs are not permutation invariant, meaning particle beliefs with different particle orders are not considered identical. This assumption simplifies the derivation of the propagated belief likelihood. Consequently, we can express $\hat{b}_k$ as $\{s^i_k, \frac{1}{m}\}^m_{i=1}$ and $\hat{b}^-_{k+1}$ as $\{s^{-i}_{k+1}, \frac{1}{m}\}^m_{i=1}$

$$\mathbb{P}(\hat{b}^-_{k+1}|\hat{b}_k, a_k) = \frac{1}{m}\prod_{i=1}^{m}\mathbb{P}(s^{-i}_{k+1}|s^i_k, a_k). \tag{2.11}$$

In the rest of the paper we assume a non-parametric case and for the ease of notation we remove the hat sign ˆ from all beliefs.

2.4 Multiple Importance Sampling

Importance Sampling (IS) is a statistical technique that allows estimating properties of some target distribution $p(x)$ by sampling from a different proposal distribution $q(x)$. In this technique, weights are assigned to the samples drawn from $q(x)$ in order to adjust the contribution of each sample according to $p(x)$:

$$\hat{E}^{IS}_p[f(x)] = \frac{1}{N}\sum_{i=1}^{N} w^i_{IS}\cdot f(x^i),\ w^i_{IS} = \frac{p(x^i)}{q(x^i)},\ x^i \sim q. \tag{2.12}$$

and the distribution q must satisfy $q(x^i) = 0 \Rightarrow p(x^i) = 0$. When there are M proposal distributions $\{q_m\}^M_{m=1}$, Multiple Importance Sampling formulation can be used [27]:

$$\hat{E}^{MIS}_p[f(x)] = \sum_{m=1}^{M}\frac{1}{n_m}\sum_{i=1}^{n_m} w^m_{MIS}(x^{i,m})\frac{p(x^{i,m})}{q_m(x^{i,m})}f(x^{i,m}). \tag{2.13}$$

Here, n_m denotes the number of samples that originate from distribution q_m, $x^{i,m}$ denotes the ith sample that originates from distribution q_m and the weights w^m_{MIS} must satisfy

$$\begin{aligned} q_m(x^{i,m}) = 0 &\Rightarrow w^m_{MIS}(x)f(x)p(x) = 0. \\ f(x^{i,m}) \neq 0 &\Rightarrow \sum_{m=1}^{M} w^m_{MIS}(x^{i,m}) = 1. \end{aligned} \tag{2.14}$$

We assume that the weights w^m_{MIS} are determined using the balance heuristic [27] which bounds the variance of the estimator and in this case, the MIS estimator is

$$\hat{E}^{MIS}_p[f(x)] = \sum_{m=1}^{M}\sum_{i=1}^{n_m}\frac{p(x^{i,m})}{\sum_{j=1}^{M} n_j\cdot q_j(x^{i,m})}f(x^{i,m}). \tag{2.15}$$

2.5 PFT-DPW Algorithm

The Particle Filter Tree - Double Progressive Widening (PFT-DPW) algorithm [21] is based on the UCT algorithm [14] and expands its application to a continuous state, action and observation setting. It utilizes Monte-Carlo simulations to progressively construct a policy tree for the belief MDP. At every belief node b_k and action a_k it sets up visitation counts $N(b_k, a_k)$ and $N(b_k)$, where $N(b_k) = \sum_{a_k} N(b_k, a_k)$ and action-value function is calculated incrementally

$$Q(b_k, a_k) \triangleq \frac{1}{N} \sum_{i=1}^{N} G_k^i, \tag{2.16}$$

by averaging accumulated reward upon initiating from node b_k and taking action a_k within the tree. Notably, $Q(b_k, a_k)$ (2.16) is not equal to $Q^\pi(b_k, a_k)$ (2.8) as the policy varies across different simulations within the tree, causing the distribution of the trajectories to be nonstationary, hence the absence of the π superscript.

Employing the particle filter method involves generating a new propagated belief b_{k+1}^- and posterior belief b_{k+1} from b_k and a_k, during which the observation o_{k+1} is sampled and the reward r is computed

$$b_{k+1}, b_{k+1}^-, o_k, r \leftarrow G_{PF(m)}(b_k, a_k). \tag{2.17}$$

In addressing the continuous state, action, and observation spaces, Double Progressive Widening (DPW) is implemented, wherein the number of children of a node is constrained artificially to kN^α, where N represents the number of times the node has been visited, and k and α serve as hyperparameters [21].

Chapter 3

Approach

3.1 Incremental Multiple Importance Sampling Update

We consider an MIS setting in which we estimate some target distribution $p(x)$ by getting samples that arrive incrementally in batches, with each batch originating from $q_i \in \{q_m\}_{m=1}^{M}$. A straightforward computation of (2.15) would necessitate a complexity of $O(M^2 \cdot n_{avg})$, where M denotes the number of different distributions and n_{avg} denotes the average sample count across all distributions. We develop an efficient way to update the estimator (2.15) incrementally in the theorem below.

Theorem 3.1. *Consider an MIS estimator* (2.15) *with M different distributions and n_m samples for each distribution $q_m \in \{q_1, ..., q_M\}$. Given a batch of L I.I.D samples from distribution $q_{m'}$, where $q_{m'}$ could be one of the existing distributions or a new, previously unseen distribution, $\hat{E}_p^{MIS}[f(x)]$* (2.15) *can be efficiently updated with a computational complexity of $O(M \cdot n_{avg} + M \cdot L)$ and memory complexity $O(M \cdot n_{avg})$.*

Proof. see Appendix A.1.

3.2 Experience-Based Value Function Estimation

We assume that we have access to a dataset

$$D \triangleq \{\tau^i, G_{k_i}^i\}_{i=1}^{|D|} \tag{3.1}$$

of trajectories executed by an agent that followed a policy π. Each trajectory is defined as the sequence

$$\begin{aligned} \tau^i \triangleq (b_{k_i}^i, a_{k_i}^i) \rightarrow (b_{k_i+1}^{-i}, o_{k_i+1}^i, b_{k_i+1}^i a_{k_i+1}^i) \rightarrow ... \\ \rightarrow (b_{k_i+d}^{-i}, o_{k_i+d}^i, b_{k_i+d}^i), \end{aligned} \tag{3.2}$$

where k_i represents the starting time index and is used to differentiate between different steps in trajectory τ^i and d is the horizon length. We assume that the agent applied a

particle filter with resampling at each step of the trajectory. The return G^i associated with trajectory τ^i is defined as the accumulated reward,

$$G^i_{k_i} \triangleq \sum_{j=0}^{d-1} r(b^i_{k_i+j}, a^i_{k_i+j}, b^i_{k_i+j+1}). \tag{3.3}$$

In this section, we evaluate $V^\pi(b_k)$ for the current belief b_k using only the dataset D (3.1), without planning. Such estimation is important in data-expensive domains like autonomous vehicles [4] and robotic manipulation tasks [16]. In the next section, we will expand our methodology to include planning.

Reusing trajectories where the initial belief is set to b_k presents no challenge - we can aggregate all trajectories that begin with belief b_k and action a_k and assuming we have N such trajectories, we define a sample-based estimator

$$\hat{Q}^\pi(b_k, a_k) \triangleq \frac{1}{N} \sum_{i=1}^{N} G^i_k. \tag{3.4}$$

However, in continuous state, action and observation spaces, the probability of sampling the same belief twice is zero. Consequently, each trajectory in the dataset D (3.1) will have an initial belief that is different from b_k.

To be able to reuse trajectories from (3.1), we discard the initial belief and action of the trajectory, instead linking the current belief and action to the remainder of the trajectory. Formally, given a trajectory $\tau^i \in D$, $\tau^i = (b^i_{k_i}, a^i_{k_i}) \rightarrow \tau^i_{suffix}$ where

$$\begin{aligned} \tau^i_{suffix} \triangleq & (b^{-i}_{k_i+1}, o^i_{k_i+1}, b^i_{k_i+1}, a^i_{k_i+1}) \rightarrow ... \\ & \rightarrow (b^{-i}_{k_i+d}, o^i_{k_i+d}, b^i_{k_i+d}). \end{aligned} \tag{3.5}$$

and the current belief b_k and action a_k, we construct a new trajectory τ'_i (see Figure 3.1), where τ^i represents a trajectory executed by an agent following policy π, while τ^i_{suffix} denotes the segment of τ^i reused for the current belief b_k and action a_k.

$$\tau'_i \triangleq (b_k, a_k) \rightarrow \tau^i_{suffix}. \tag{3.6}$$

To estimate $Q^\pi(b_k, a_k)$ using the information within trajectory τ^i, two adjustments are required. Firstly, we need to modify the initial term in the return G^i to be equal to $r(b_k, a_k, b^i_{k_i+1})$, recognizing that $b_k \neq b^i_{k_i}$ and $a_k \neq a^i_{k_i}$. Consequently, we define the return of trajectory τ'_i

$$\tilde{G}^i_k \triangleq G^i_{k_i} - r(b^i_{k_i}, a^i_{k_i}, b^i_{k_i+1}) + r(b_k, a_k, b^i_{k_i+1}). \tag{3.7}$$

Secondly, we need to adjust the weight of $\tilde{G}^i$ due to the disparity between the distribution $\mathbb{P}(\tau^i_{suffix}|b^i_{k_i}, a^i_{k_i}, \pi)$ and the distribution $\mathbb{P}(\tau^i_{suffix}|b_k, a_k, \pi)$, which is acheived

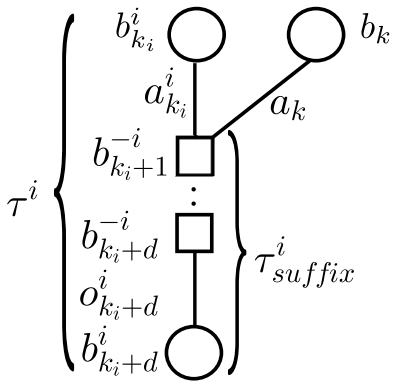

Figure 3.1: Illustration of trajectory reuse.

through importance sampling. The distribution $\mathbb{P}(\cdot|b^i_{k_i}, a^i_{k_i}, \pi)$ of partial trajectory τ^i_{suffix} is determined by the initial belief $b^i_{k_i}$ and action $a^i_{k_i}$. Given N_{IS} partial trajectories sampled from the same distribution $\mathbb{P}(\cdot|b^i_{k_i}, a^i_{k_i}, \pi)$, we define an Importance Sampling estimator

$$\hat{Q}^{\pi}_{IS}(b_k, a_k) \triangleq \frac{1}{N_{IS}} \sum_{i=1}^{N_{IS}} w^i \cdot \tilde{G}^i_k. \tag{3.8}$$

where $w^i \triangleq \frac{\mathbb{P}(\tau^i_{suffix}|b_k,a_k,\pi)}{\mathbb{P}(\tau^i_{suffix}|b^i_{k_i},a^i_{k_i},\pi)}$.

As a result of our approach to constructing reusable trajectories as described in (3.6), we can efficiently calculate the weights w^i utilizing the theorem presented below.

Theorem 3.2. *Given belief node b_k, action a_k and trajectory $\tau^i = (b^i_{k_i}, a^i_{k_i}) \to \tau^i_{suffix}$ where τ^i_{suffix} is defined in* (3.5)*, the following equality holds:*

$$\frac{\mathbb{P}(\tau^i_{suffix}|b_k, a_k, \pi)}{\mathbb{P}(\tau^i_{suffix}|b^i_{k_i}, a^i_{k_i}, \pi)} = \frac{\mathbb{P}(b^{-i}_{k_i+1}|b_k, a_k)}{\mathbb{P}(b^{-i}_{k_i+1}|b^i_{k_i}, a^i_{k_i})}. \tag{3.9}$$

Proof. see appendix A.2.

We denote by M the number of unique distributions of partial trajectories

$$\{\mathbb{P}(\cdot|b^m_{k_m}, a^m_{k_m}, \pi)\}_{m=1}^{M}, \tag{3.10}$$

where each distribution is defined by the initial belief $b^m_{k_m}$ and action $a^m_{k_m}$. Additionally, we denote the sample count from each distribution as n_m. Consequently, we can reformulate the dataset D (3.1) as follows:

$$D \triangleq \{\tau^{l,m}, G^{l,m}_k\}^{M,n_m}_{m=1,l=1}. \tag{3.11}$$

Using this formulation, we define a multiple importance sampling estimator assuming

the balance heuristic (2.15),

$$\hat{Q}^{\pi}_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{n_m} \frac{\mathbb{P}(\tau^{l,m}_{suffix}|b_k, a_k, \pi)\tilde{G}^{l,m}_k}{\sum_{j=1}^{M} n_j \cdot \mathbb{P}(\tau^{l,m}_{suffix}|b^j_{k_j}, a^j_{k_j}, \pi)}. \tag{3.12}$$

where $\tau^{l,m}_{suffix}$ represents the lth partial trajectory that was sampled from the distribution $\mathbb{P}(\cdot|b^m_{k_m}, a^m_{k_m}, \pi)$ and $\tilde{G}^{l,m}_k$ is the adjusted accumulated reward (3.7).

Using Theorem 3.2, we can re-write the MIS estimator (3.12)

$$\hat{Q}^{\pi}_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{n_m} \frac{\mathbb{P}(b^{-l,m}_{k_m+1}|b_k, a_k)}{\sum_{j=1}^{M} n_j \cdot \mathbb{P}(b^{-l,m}_{k_m+1}|b^j_{k_j}, a^j_{k_j})} \cdot \tilde{G}^{l,m}_k. \tag{3.13}$$

Since each element in the second sum of (3.13) corresponds to a propagated belief, which might appear more than once, we can rewrite the sum in a more compact form. Specifically, we group the terms based on unique propagated beliefs and account for their multiplicity:

$$\hat{Q}^{\pi}_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{|C(b_{k_m}, a_{k_m})|} W(b^{-l,m}_{k_m+1}) \cdot \sum_{y=1}^{N(b^{-l,m}_{k_m+1})} \tilde{G}^{m,l,y}_k. \tag{3.14}$$

The weights $W(b^{-l,m}_{k_m+1})$ are defined by:

$$W(b^{-l,m}_{k_m+1}) = \frac{\mathbb{P}(b^{-l,m}_{k_m+1}|b_k, a_k)}{\sum_{j=1}^{M} n_j \cdot \mathbb{P}(b^{-l,m}_{k_m+1}|b^j_{k_j}, a^j_{k_j})} \tag{3.15}$$

$C(b_{k_m}, a_{k_m})$ denotes the set of reused propagated belief children associated with b_{k_m} and a_{k_m}. The term $N(b^{-l,m}_{k_m+1})$ represents the visitation count of $b^{-l,m}_{k_m+1}$, indicating the number of trajectories that pass through the propagated belief $b^{-l,m}_{k_m+1}$ and $\tilde{G}^{m,l,y}_k$ is the return of the y-th trajectory passing through $b^{-l,m}_{k_m+1}$. Note that $b^{-l,m}_{k_m+1}$ in (3.14) represents unique propagated beliefs, which differs from (3.13), where it denotes the propagated belief associated with a single trajectory. Figure 3.2 illustrates the estimator from (3.14). For the current belief b_k and action a_k, three prior trajectories are incorporated: two from $(b^i_{k_i}, a^i_{k_i})$ and one from $(b^j_{k_j}, a^j_{k_j})$. Light green edges show the connections between (b_k, a_k) and the reused nodes for estimating $\hat{Q}^{\pi}_{MIS}(b_k, a_k)$. Further in this work, we consider a framework where the dataset D (3.11) expands over time with trajectory samples from an agent following policy π. Theorem 3.1 is used to efficiently update the estimator (3.13) with new samples.

To clarify, our framework differs from standard off-policy evaluation methods. Traditional importance sampling for experience-based value estimation operates within the off-policy paradigm [22], where trajectories originate from the current belief b_k, using behavioral (π_b) and target (π_t) policies to estimate $V^{\pi_t}(b_k)$. In contrast, we estimate $V^{\pi}(b_k)$ for the current belief and a specified policy π, with trajectories drawn form

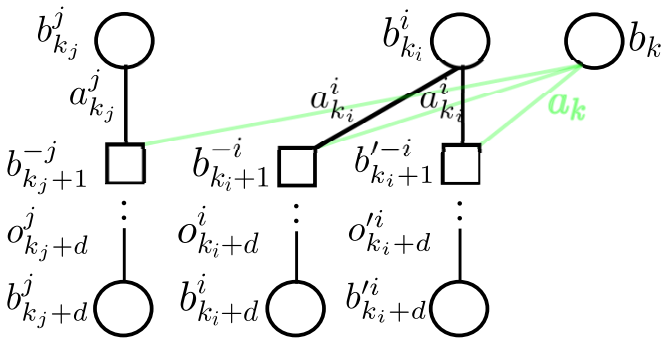

Figure 3.2: Illustration of reuse of three trajectories.

different beliefs in the dataset D (3.1). To our knowledge, such a setting has not been addressed before in the context of action-value function estimation in POMDPs.

3.3 Our POMDP Planning Algorithm: IR-PFT

Up to this point, we considered a specific single policy, denoted as π, and utilized previously-generated trajectories by an agent following π to estimate the action-value function $Q^\pi(b_k, a_k)$. In this section, we present an anytime POMDP planning algorithm that uses trajectories from the dataset D, which includes data from previous planning sessions, to accelerate current planning.

We name our algorithm Incremental Reuse Particle Filter Tree (IR-PFT). Instead of calculating $Q(b_k, a_k)$ from scratch in each planning session, we use previous experience to speed up the calculations.

We adopt the same approach as in Section 3.2 to reuse trajectories, with three key modifications: first, the propagated belief nodes from the previous planning session in dataset D have a shorter planning horizon. We extend the horizon of these nodes before reusing them; second, the policy varies across different simulations (as in standard MCTS), resulting in a non-stationary distribution of reused trajectories in D; and third, we integrate the planning and generation of new trajectories with the reuse of previous trajectories within an anytime MCTS setting.

Figure 3.3 visually illustrates the horizon alignment process, where a propagated belief node $b^-_{k_i}$ with horizon d_{prev} must be extended by Δd to match the current horizon d. We analyze the complexity of the correction of belief nodes from previous planning sessions in case of using the MCTS [7] algorithm in Corollary 3.3.

Corollary 3.3. *Given an MCTS tree T with horizon d_{prev}, number of simulations m and N nodes, extending its horizon by Δd will require adding at most $m \cdot \Delta d$ nodes and reward calculations.*

The proof is straightforward: after m simulations, the MCTS tree contains at most m leaves and we need to extend each leaf by Δd and for each new node we calculate a reward.

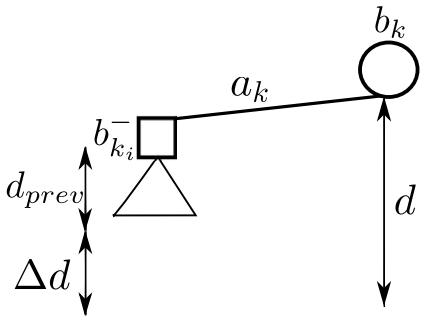

Figure 3.3: Illustration of horizon gap.

We now define the estimator

$$\hat{Q}_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{|C(b_{k_m}, a_{k_m})|} W(b^{-l,m}_{k_m+1}) \cdot \sum_{s=1}^{N(b^{-l,m}_{k_m+1})} \bar{G}^{m,l,y}_k, \tag{3.16}$$

where the weights $W(b^{-l,m}_{k_m+1})$ are defined in (3.15). $\bar{G}^{m,l,y}_k$ is the extended return defined as

$$\bar{G}^{m,l,y}_k = \tilde{G}^{m,l,y}_k + \sum_{i=k+d^{l,m}_{prev}}^{k+d^{l,m}_{prev}+\Delta d-1} r(b_i, \pi_{rollout}(b_i), b_{i+1}). \tag{3.17}$$

$d^{l,m}_{prev}$ is the horizon of propagated belief $b^{-l,m}_{k_m+1}$, $\bar{G}^{m,l,y}_k$ shares the same values in the summation as the return $\tilde{G}^{m,l,y}_k$ (3.7), but it also includes additional terms from the extended trajectory due to the horizion extension using the rollout policy $\pi_{rollout}$. Therefore, only the rewards for these additional terms need to be computed when extending the horizon, while all shared terms ($\tilde{G}^{m,l,y}_k$) can be reused.

Since the tree policy varies between simulations, the update represented by (3.16) operates in a heuristic manner, with its convergence yet to be established. We intend to explore this aspect in a future work.

After extending the horizon of a reused propagated belief node b^- and reusing its action-value function, the counter $N(b_k, a_k)$ is incremented by the visitation count $N(b^-)$ using the relation

$$N(b_k, a_k) = \sum_{b^-_k \in C(b_k, a_k)} N(b^-_k). \tag{3.18}$$

This approach accelerates our algorithm for a given number of simulations, offering a speedup over PFT-DPW [21].

To summarize, here is the high-level logical flow of our algorithm: At each iteration, we either reuse a propagated belief node by extending its horizon by Δd, as illustrated in Figure 3.3, and compute the extended return for the subtree rooted at the reused

node b^-, or create a new node. Subsequently, the Multiple Importance Sampling (MIS) estimator (3.16) is employed to evaluate the action-value function. To avoid the computational expense of a naive calculation, we leverage Theorem 3.1 to perform efficient incremental updates of the estimator in (3.16).

3.3.1 Algorithm Description

The complete algorithm is outlined across multiple methods - Algs. 3.1, 3.2, 3.3 and 3.4. Alg. 3.1 illustrates a general planning loop wherein the agent iteratively plans and executes actions until the problem is solved. After each planning session, reuse candidates are updated based on the preceding planning tree. The main algorithm is

Algorithm 3.1 General Planning Loop

1: **Procedure**: SOLVE(b, D)
2: **while** ProblemNotSolved() **do**
3: $a \leftarrow Plan(b, D)$
4: $o \leftarrow ReceiveObservation(b, a)$
5: $b', b'^-, r \leftarrow G_{PF(m)}(b, a, o)$
6: UpdateReuseCandidates(a, D, b, b')
7: $b \leftarrow b'$
8: **end while**

detailed in Alg. 3.2 with key modifications compared to the PFT-DPW algorithm [21] highlighted in red. The ActionProgWiden method (line 5) is implemented following the same approach as described in [21]. ShouldReuse method (line 7) evaluates three conditions: current node b is the root, the balance between reused and new nodes, and the availability of reuse candidates. The second criterion is important because, while acquiring estimates from prior partial trajectories is runtime-efficient, generating new trajectory samples from the correct distribution is essential. Currently, our algorithm only applies reuse to the root node, as it promises the most significant computational savings. Since the root node typically has the shallowest depth in the tree, we can optimize by conserving numerous reward computations for most of its descendants. While extending reuse to nodes at other depth levels is feasible, it falls outside the scope of this work.

The GetReuseCandidate method (line 8) selects a reuse candidate propagated belief b'^- from the dataset D based on a distance function f_D (line 2). An example of f_D is $||\mathbb{E}[b^- - b^-_{MLE}]||_2^2$ where b^-_{MLE} represents the maximum likelihood propagated belief, given belief b and action a which can be calculated using (2.11) with $O(m)$ complexity where m is the number of samples. Since f_D is applied to the entire dataset, it needs to be computationally efficient. Additionally, reusing nodes with high visitation counts will further reduce the overall runtime of the algorithm.

Algorithm 3.2 IR-PFT

1: **Procedure**: PLAN(b, D)
2: $i = 0$
3: **while** $i < n$ **do**
4: $\quad Simulate(b, d_{max}, D)$
5: **end while**
6: $a = argmax_a\{Q(b, a)\}$
7: return a

1: **Procedure**: SIMULATE(b, d, D)
2: **if** $d = 0$ **then**
3: $\quad$ return 0
4: **end if**
5: $a \leftarrow$ ActionProgWiden(b)
6: **if** $|C(ba)| \leq k_o N(ba)^{\alpha_o}$ **then**
7: $\quad$ **if** $ShouldReuse(b, a, D)$ **then**
8: $\quad\quad b'^{-} \leftarrow GetReuseCandidate(b, a, D)$
9: $\quad\quad$ FillHorizonPropagated($b'^{-}, d - d_{b^-}$)
10: $\quad\quad N(b) \leftarrow N(b) + N(b'^{-})$
11: $\quad\quad N(ba) \leftarrow N(ba) + N(b'^{-})$
12: $\quad\quad i \leftarrow i + N(b'^{-})$ {update simulation counter}
13: $\quad\quad Q(ba) \leftarrow MISUpdate()$ {update using (3.16)}
14: $\quad\quad C(b, a) \leftarrow C(b, a) \cup \{(b'^{-})\}$
15: $\quad\quad$ return $total$
16: $\quad$ **else**
17: $\quad\quad b', b'^{-}, r \leftarrow G_{PF(m)}(b, a)$
18: $\quad\quad C(b, a) \leftarrow C(b, a) \cup \{(b'^{-})\}$
19: $\quad\quad C(b'^{-}) \leftarrow C(b'^{-}) \cup \{(b', r)\}$
20: $\quad\quad total \leftarrow r + \gamma ROLLOUT(b', d - 1)$
21: $\quad$ **end if**
22: **else**
23: $\quad b'^{-} \leftarrow$ sample uniformly from $C(ba)$
24: $\quad b', r \leftarrow$ sample uniformly from $C(b'^{-})$
25: $\quad total \leftarrow r + \gamma Simulate(b', d - 1, T)$
26: **end if**
27: $N(b) \leftarrow N(b) + 1$
28: $N(ba) \leftarrow N(ba) + 1$
29: $Q(ba) \leftarrow MISUpdate()$ {update using (3.16)}
30: return $total$

The FillHorizonPropagated method (line 9), addresses discrepancies in horizon

Algorithm 3.3 Reuse Functions

```
1: Procedure: UpdateReuseCandidates(a,D,b_k,b_{k+1}^{real})
2: ReuseDict dict ← {}
3: for b_{k+1}^- ∈ C(b_k, a) do
4:    for b_{k+1} ∈ C(b_{k+1}^-) do
5:       for a' ∈ Actions(b_{k+1}) do
6:          for b_{k+2}^- ∈ C(b_{k+1}, a') do
7:             if n(b_{k+2}^-) > n_min then
8:                D.append(b_{k+2}^-)
9:             end if
10:         end for
11:      end for
12:   end for
13: end for
```

```
1: Procedure: ShouldReuse(b,a,D)
2: if not b.IsRoot() then
3:    return false
4: end if
5: if NumReused(b, a) > Total(b,a)/2 then
6:    return false
7: end if
8: candidates ← D.GetReuseCandidatesDict()
9: return not(candidates.empty())
```

```
1: Procedure: GetReuseCandidate(b,a,D)
2: b^- ← argmin_{b^-}{f_D(b^-, b, a)}
3: return b^-
```

lengths when reusing nodes from the previous planning sessions. Algorithm 3.4 performs recursive traversal of the subtree defined by propagated belief b'^{-} and extends its depth by d using the rollout policy.

At lines 10 and 11, we increment counters, where $N(b'^{-})$ represents the count of trajectories passing through reuse candidate propagated belief node b'^{-}. At line 12, we increment the Plan procedure counter by $N(b'^{-})$.

At lines 13 and 29 we utilize (3.16) to update $Q(b, a)$, leveraging efficiency through the application of Theorem (3.1). At line 14 we store the propagated belief b'^{-}.

Lines 17 - 19 are executed when we choose not to reuse and instead initialize a new propagated belief from scratch. A new belief is generated using the particle filter method [25], after which the propagated belief and posterior belief are saved, and a rollout is performed.

At lines 23 and 24 we sample uniformly both propagated and posterios beliefs.

UpdateReuseCandidates method in Algorithm 3.3 inserts new reuse candidates that have a visitation count larger than a threshold n_{min}, as we aim to reuse nodes with higher visitation counts, which leads to a greater speedup.

Algorithm 3.4 Fill Horizon Gap

Procedure: FillHorizonPropagated(b^-, d)
$Q_{new}(b^-) \leftarrow 0$
for $b' \in C(b^-)$ **do**
  $Q_{new}(b^-) \leftarrow Q_{new}(b^-) + FillHorizonPosterior(b')$
end for
return $\frac{Q_{new}(b^-)}{|C(b^-)|}$

Procedure: FillHorizonPosterior(b, d)
if $IsLeaf(b)$ **then**
  $a \leftarrow DefaultPolicy(b)$
  $b', b'^-, r \leftarrow G_{PF(m)}(b, a)$
  $N(b, a) \leftarrow 1$
  $N(b'^-) \leftarrow 1$
  $Q(b'^-) \leftarrow r$
  $Q(b, a) \leftarrow r$
  return r
end if
$Q(b) \leftarrow 0$
for $a \in Actions(b)$ **do**
  for $b'^- \in C(b, a)$ **do**
    $Q(b) \leftarrow Q(b) + FillHorizonPropagated(b'^-, d-1)$
  end for
end for
$Q(b) \leftarrow \frac{Q(b)}{N(b)}$
return $Q(b)$

Chapter 4

Results

4.1 Setup

4.1.1 IR-PFT Evaluation

We assess the performance of the IR-PFT algorithm by comparing it to the PFT-DPW algorithm [21]. Our evaluation focuses on two main aspects: runtime and accumulated reward, with statistics measured for each. The beliefs are approximated with a finite number of state particles and each algorithm was evaluated using different quantities of particles—specifically, $5, 10, 15$, and 20, while maintaining a constant horizon length of $d = 10$. In all experiments, the solvers were limited to 1000 iterations for each planning phase. The code for both algorithms IR-PFT and PFT was implemented in the Julia programming language and is available at https://github.com/miken1990/ir-pft.

4.1.2 Continuous Light-Dark 2D

All experiments were conducted using the standard Light-Dark 2D benchmark, where the agent's objective is to reach a predefined goal $g \in \mathbb{R}^2$ while minimizing localization uncertainty through the use of beacons distributed across the map (refer to Figure 4.1).

The state space $S \subseteq \mathbb{R}^2$ is continuous, representing the agent's position. The action space $A \subseteq \mathbb{R}^2$ is also continuous, constrained to movements within the unit circle, $\|\mathbf{a}\| = 1$, where $\mathbf{a} \in \mathcal{A}$. The observation space $O \subseteq \mathbb{R}^2$ is continuous, providing noisy measurements of the agent's location. The initial belief $b_0(s)$ is modeled as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, with mean $\boldsymbol{\mu}_0$ and covariance $\boldsymbol{\Sigma}_0$.

The state transition model is defined as a Gaussian distribution:

$$T(s' \mid s, \mathbf{a}) = \mathcal{N}(\mathbf{s} + \mathbf{a}, \boldsymbol{\Sigma}_T),$$

where $\boldsymbol{\Sigma}_T$ represents the process noise covariance.

The observation model is also a Gaussian distribution:

$$p(o \mid s) = \mathcal{N}(\mathbf{o} \mid \mathbf{s}, \Sigma_O \cdot \max(r, r_{\max})),$$

where $r = \|\mathbf{s} - \mathbf{b}\|$ is the Euclidean distance to the nearest beacon $\mathbf{b} \in \mathbb{R}^2$, $r_{\max}$ is the maximum distance for decreasing localization accuracy and Σ_O is the observation noise covariance.

The reward function is defined as the weighted sum:

$$R(b, a, b') = \mathbb{E}_{s' \sim b'}[\|(s' - g)\|] - \lambda \cdot H(b, b'),$$

where: $\mathbb{E}_{s' \sim b'}[\|(s' - g)\|]$ is the expected Euclidean distance between states in b' and the goal $g \in \mathcal{S}$, $H(b, b')$ is the differential entropy of the belief transition from b to b' calculated using [3] and $\lambda \in \mathbb{R}^+$ is a scaling factor that balances the contribution of the differential entropy term. The first term encourages minimization of the expected distance to the goal, while the second term promotes belief updates that reduce uncertainty in the agent's localization. The goal of the agent is to reach the goal while minimizing uncertainty.

Although the experiment involves Gaussian transitions, our approach is generalizable and remains applicable to any non-parametric settings.

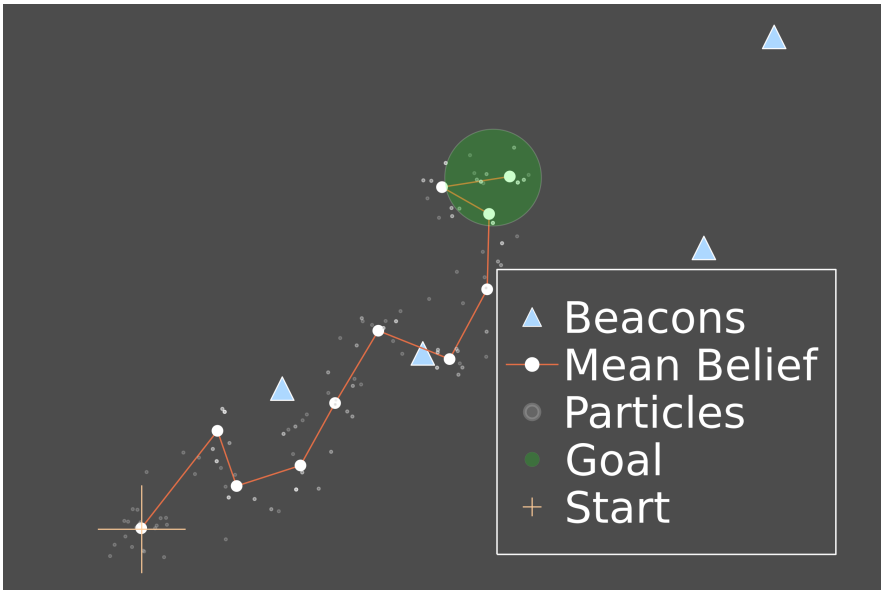

Figure 4.1: Illustration of Continuous Light-Dark 2D problem.

4.2 Runtime Analysis

We compared the runtime of IR-PFT vs PFT-DPW. The results are depicted in Figure 4.2 as a function of number of particles. Additionally, we included a speedup chart, which provides more insightful information, in Figure 4.3. The runtime of IR-PFT consistently outperformed that of PFT, with a notable saturation in speedup at approximately 1.5×. This improvement is primarily attributed to the reduction in

computational overhead associated with reward calculations, which represent the most resource-intensive component of the algorithm. As the number of particles increases, the computation of the differential entropy reward becomes the dominant factor in runtime, scaling quadratically with the number of samples.

Additionally, node reuse was managed using the `ShouldReuse` method (line 7), as detailed in the previous section. This method limits the number of reused nodes in the tree, ensuring a balance between reusing existing nodes and exploring new nodes to maintain algorithmic efficiency and exploration capability.

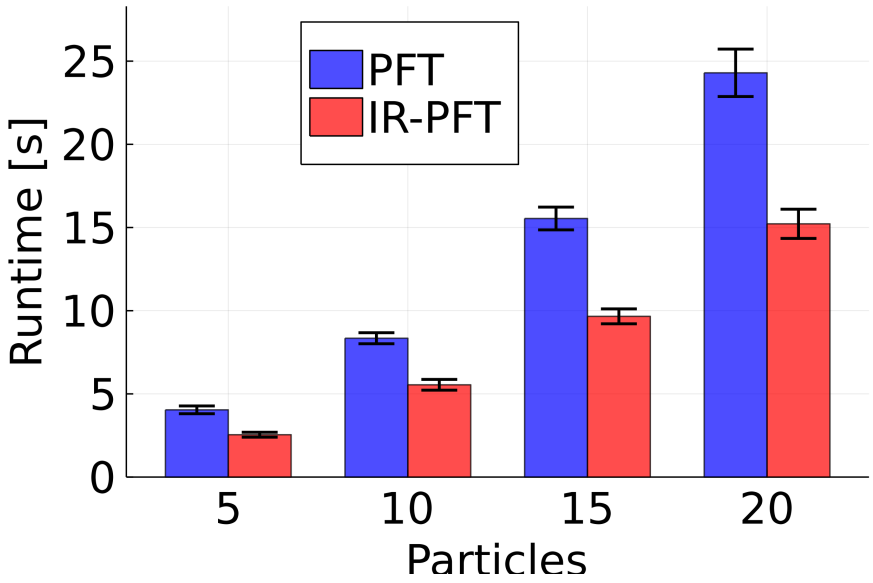

Figure 4.2: Runtime comparison.

4.3 Accumulated Reward Analysis

We compare the accumulated rewards of IR-PFT vs PFT (Figure 4.4). The results show negligible differences, indicating that our method improves runtime without compromising on the accumulated reward received by the agent.

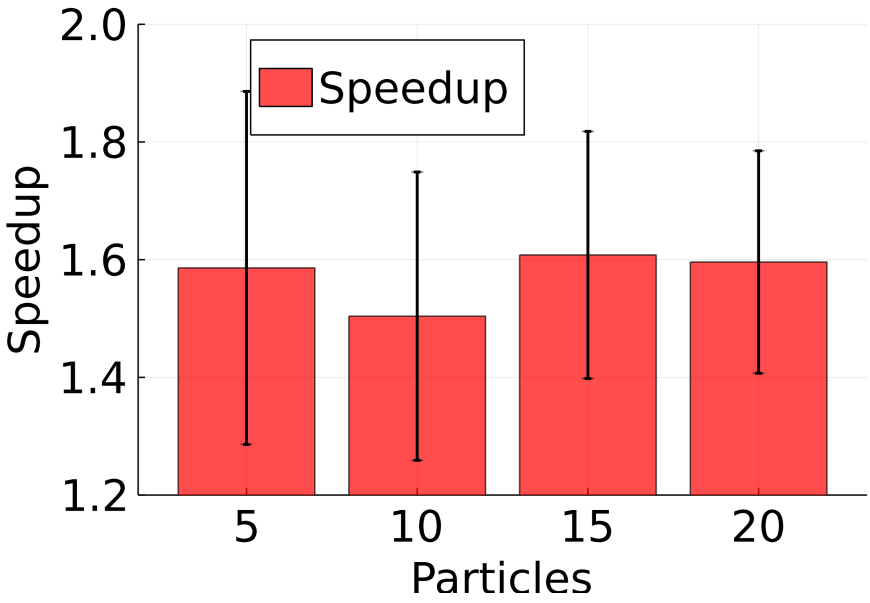

Figure 4.3: Speedup.

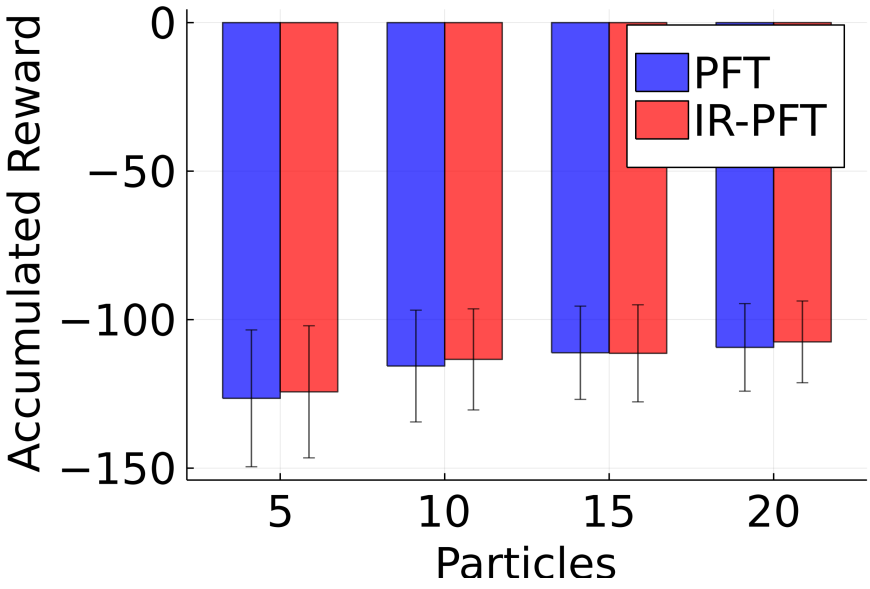

Figure 4.4: Accumulated Reward Comparison.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this paper, we have proposed a general framework which allows to reuse prior information during the current planning session in a non-parametric setting. We derived theoretical justification for reuse via MIS and introduced a new MCTS-like algorithm, IR-PFT which reuses information from previous planning session and allows to speed up calculations in current planning session. In order to evaluate IR-PFT algorithm, we conducted an empirical performance study. Specifically, we compared the performance of our approach to the PFT-DPW algorithm which doesn't reuse prior information and starts each planning iteration from scratch. We measured various performance metrics, including runtime, speedup and the accumulated reward. Our results clearly indicate a speed-up in the planning process when prior information is reused. Importantly, despite the speedup gain in computations, our approach maintains the same level of performance as the traditional planning approach without reuse. Incorporating prior information significantly boosts planning efficiency, delivering time savings while maintaining high-quality results. These findings underscore the effectiveness and potential of the proposed approach.

5.2 Future Work

Several potential extensions of this work can be envisioned. First, the reuse mechanism could be extended beyond the root node to other belief nodes in the planning tree. By allowing every node to reuse prior information when estimating the action-value function, the computational efficiency of the algorithm could be further enhanced.

Second, the convergence properties of the Multiple Importance Sampling (MIS) update in the IR-PFT algorithm present an interesting direction for further research. While this work provided theoretical justification for reuse in the context of experience-based value function estimation, the MIS update was employed as a heuristic within the IR-PFT algorithm. A more rigorous analysis of its performance and convergence

properties in the Monte Carlo Tree Search (MCTS) framework, particularly under the challenges posed by a non-stationary policy, could yield deeper insights and establish stronger theoretical guarantees.

Appendix A

Appendix

A.1 Proof of Theorem 3.1

Consider an MIS estimator (2.15) with M different distributions and n_m samples for each distribution $q_m \in \{q_1, ..., q_M\}$. Given a batch of L I.I.D. samples from distribution $q_{m'}$ which may be an existing or new distribution,

$$\hat{E}_p^{MIS}[f(x)] = \sum_{m=1}^{M} \sum_{i=1}^{n_m} \frac{p(x_{i,m})}{\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m})} f(x_{i,m}),$$

(2.15) can be efficiently updated with a computational complexity of $\mathrm{O}(M \cdot n_{avg} + M \cdot L)$ and memory complexity $\mathrm{O}(M \cdot n_{avg})$.
For every distribution q_m we have the inner sum term $\sum_{i=1}^{n_m} \frac{p(x_{i,m})}{\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m})} f(x_{i,m})$.
In case $m \neq m'$:
$\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) \leftarrow \sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) + L \cdot q_{m'}(x_{i,m})$ - $O(1)$ complexity. We have n_m samples and M distributions so the complexity of this update is $O(M \cdot n_m)$.
In case $m = m'$:
For existing samples $\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) \leftarrow \sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) + L \cdot q_{m'}(x_{i,m})$ - $O(1)$ complexity. We have n_m samples existing samples so in total $O(n_m)$ complexity.
For each new sample we need to calculate $\frac{p(x_{i,m})}{\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m})} f(x_{i,m})$ - $O(M)$ complexity. We have L new samples so in total $O(L \cdot M)$ complexity. The total complexity of the update is $O(M \cdot n_{avg} + n_m + M \cdot L) = O(M \cdot n_{avg} + M \cdot L)$.

A.2 Proof of Theorem 3.2

$$\frac{\mathbb{P}(\tau_{suffix}^i | b_k, a_k, \pi)}{\mathbb{P}(\tau_{suffix}^i | b_{k_i}^i, a_{k_i}^i, \pi)} = \frac{\mathbb{P}(b_{k_i+1}^{-i}, ..., b_{k_i+L}^i | b_k, a_k, \pi)}{\mathbb{P}(b_{k_i+1}^{-i}, ..., b_{k_i+L}^i | b_{k_i}^i, a_{k_i}^i, \pi)}. \tag{A.1}$$

Applying chain rule yields,

$$\frac{\mathbb{P}(b^{-i}_{k_i+1}|b_k, a_k)}{\mathbb{P}(b^{-i}_{k_i+1}|b^i_{k_i}, a^i_{k_i})} \cdot \cancel{\frac{\mathbb{P}(o^i_{k_i+1}, \ldots, b^i_{k_i+L}|b^{-i}_{k_i+1}, \pi)}{\mathbb{P}(o^i_{k_i+1}, \ldots, b^i_{k_i+L}|b^{-i}_{k_i+1}, \pi)}} = \frac{\mathbb{P}(b^{-i}_{k_i+1}|b_k, a_k)}{\mathbb{P}(b^{-i}_{k_i+1}|b^i_{k_i}, a^i_{k_i})} \tag{A.2}$$

Bibliography

[1] Mauricio Araya-López, Olivier Buffet, Vincent Thomas, and Francois Charpillet. "A POMDP Extension with Belief-dependent Rewards." In: *NIPS*. 2010, pp. 64–72.

[2] M. Barenboim and V. Indelman. "Adaptive Information Belief Space Planning." In: *the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI)*. Vienna, Austria, July 2022.

[3] Y. Boers, H. Driessen, A. Bagchi, and P. Mandal. "Particle filter based entropy." In: *2010 13th International Conference on Information Fusion*. 2010, pp. 1–8.

[4] H. Caesar, J. Kabzan, and K. Tan et al. "NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles." In: *CVPR ADP3 workshop*. 2021.

[5] Panpan Cai, Yuanfu Luo, David Hsu, and Wee Sun Lee. "HyP-DESPOT: A hybrid parallel algorithm for online planning under uncertainty." In: *Intl. J. of Robotics Research* 40.2-3 (2021), pp. 558–573.

[6] Tristan Cazenave, Jean Méhat, and Abdallah Saffidine. "UCD: Upper confidence bound for rooted directed acyclic graphs." In: *Knowledge-Based Systems* 34 (2012), pp. 26–33.

[7] Auger Couetoux and Teytaud. "Continuous Upper Confidence Trees with Polynomial Exploration - Consistency." In: *European conference on machine learning*. Springer. 2013.

[8] E. Farhi and V. Indelman. "iX-BSP: Incremental Belief Space Planning." In: (2021). arXiv: `2102.09539`.

[9] E. I. Farhi and V. Indelman. "iX-BSP: Belief Space Planning through Incremental Expectation." In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. May 2019.

[10] 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.

[11] 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.

[12] Marcus Hoerger and Hanna Kurniawati. "An On-Line POMDP Solver for Continuous Observation Spaces." In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 7643–7649.

[13] Mykel J. Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

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

[15] Michael H Lim, Tyler J Becker, Mykel J Kochenderfer, Claire J Tomlin, and Zachary N Sunberg. "Optimality guarantees for particle belief approximation of POMDPs." In: *Journal of Artificial Intelligence Research* 77 (2023), pp. 1591–1636.

[16] Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. *Scaling Robot Supervision to Hundreds of Hours with RoboTurk: Robotic Manipulation Dataset through Human Reasoning and Dexterity*. 2019. arXiv: `1911.04052 [cs.RO]`.

[17] Rémi Munos. *From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning*. 2014.

[18] C. Papadimitriou and J. Tsitsiklis. "The complexity of Markov decision processes." In: *Mathematics of operations research* 12.3 (1987), pp. 441–450.

[19] David Silver and Joel Veness. "Monte-Carlo planning in large POMDPs." In: *Advances in Neural Information Processing Systems (NIPS)*. 2010, pp. 2164–2172.

[20] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. "DESPOT: Online POMDP Planning with Regularization." In: *NIPS*. Vol. 13. 2013, pp. 1772–1780.

[21] 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*. Vol. 28. 1. 2018.

[22] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[23] Ori Sztyglic and Vadim Indelman. "Speeding up Online POMDP Planning via Simplification." In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2022.

[24] Vincent Thomas, Geremy Hutin, and Olivier Buffet. "Monte Carlo Information-Oriented Planning." In: *arXiv preprint arXiv:2103.11345* (2021).

[25] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[26] L. G. Valiant. "The Complexity of Computing the Permanent." In: *Theoretical Computer Science* 8.2 (Apr. 1979), pp. 189–201.

[27] Eric Veach and Leonidas J Guibas. "Optimally combining sampling techniques for Monte Carlo rendering." In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM. 1995, pp. 419–428.

[28] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. "DESPOT: Online POMDP planning with regularization." In: *JAIR* 58 (2017), pp. 231–266.

[29] A. Zhitnikov and V. Indelman. "Simplified Risk Aware Decision Making with Belief Dependent Rewards in Partially Observable Domains." In: *Artificial Intelligence, Special Issue on "Risk-Aware Autonomous Systems: Theory and Practice"* (2022).

[30] Andrey Zhitnikov, Ori Sztyglic, and Vadim Indelman. "No Compromise in Solution Quality: Speeding Up Belief-dependent Continuous POMDPs via Adaptive Multilevel Simplification." In: *Intl. J. of Robotics Research* (2024).

בשלב הראשון אנו מראים איך ניתן לשערך את טיב הפתרון ללא תכנון עבור אמונה ופעולות נתונות ובהינתן מידע מסוכן מומחה שביצע פעולות במרחב האמונה. הגישה שלנו לשיערוך ממידע קודם מתבססת על דגימה ממספר פילוגים על פי חשיבות Multiple Importance Sampling (MIS) המספקת הצדקה תיאורטית לגישתנו. בשלב השני, אנו בונים על הגישה מהשלב הראשון על מנת לפתח את האלגוריתם שלנו - IR-PFT המבוסס על אלגוריתם MCTS שמבצע תכנון מכוון יעיל יותר בעזרת שימוש במידע שנצבר בשלבי התכנון הקודמים שלו. תוצאות ניסיוניות מראות שהשיטה שלנו לא רק מצמצמת באופן משמעותי את זמן החישוב, אלא גם שומרת על רמות ביצוע דומות לאלגוריתם המקורי בלי השיטה שלנו. ממצאינו מצביעים על כך ששילוב של מידע תכנון היסטורי יכול לשפר באופן משמעותי את היעילות של קבלת החלטות מקוונת בסביבות לא ודאיות, ובסופו של דבר להוביל למערכות אוטונומיות תגובתיות ומסתגלות יותר.

תקציר

תכנון מקוון בסביבות לא ודאיות מהווה אתגר מרכזי ומתמשך בתחום הרובוטיקה והמערכות האוטונומיות. במציאות, סוכנים אוטונומיים נדרשים לקבל החלטות בתנאים בהם הם אינם מקבלים גישה ישירה למצב האמיתי של הסביבה, אלא במקום זאת מסתמכים על "אמונה" – פילוג על פניה מצבים האפשריים. אי-הוודאות במערכות אלו נובעת ממקורות שונים, כמו רעש מובנה בחיישנים, מידע מוגבל וחלקי ושינויים דינמיים בסביבה. תהליכי החלטה מרקוביים מובחנים למחצה (POMDPs) מספקים מסגרת מתמטית מבוססת היטב לבעיות כאלה. בעקבות אי הוודאות במרחב המצב, הסוכן מתחזק פילוג על פני כל המצבים האפשריים הידוע בשם אמונה (belief). מטרת הסוכן בהינתן הפילוג ההתחלתי היא לקחת בכל שלב בתכנון את ההחלטה האופטימאלית במובן של מיקסום פונקציית התגמול על ידי שימוש במדיניות (Policy) האופטימאלית. מציאת המדיניות האופטימאלית היא בעיה קשה חישובית (PSPACE-complete). הקושי של הבעיה נובע, בין היתר, מהעובדה שמספר האפשרויות שהסוכן צריך לבחון על מנת לקבל את החלטה האופטימאלית גדל באופן מעריכי עם כמות הפעולות וכמות התצפיות, התופעה ידועה בתור "קללת ההיסטוריה" (Curse of history). סיבה נוספת לקושי החישובי של הבעיה הוא העובדה שכל הוספת מימד למרחב המצב מעלה באופן מעריכי את כמות המצבים האפשריים. תופעה זו ידועה בשם "קללת המימדיות" (Curse of dimensionality). במרחבי מצב, פעולות ותצפיות רציפים, הבעיה קשה חישובית אף יותר ומציאת פתרון אופטימאלי אינה אפשרית מלבד בבעיות פשוטות מאוד.

אלגוריתמים מכוונים (online algorithms) מנסים למצוא פתרון מקורב לבעיה, הם נמצאים בשימוש נרחב ולוקחים בחשבון אילוצי "תקציב" כמו זמן ריצה או מספר איטרציות מוגבלים ומשפרים את הפתרון הקיים ככל שה"תקציב" גדל. הדגימה של מרחב החיפוש נעשית על ידי דגימת מסלולים עתידיים מתוך סך כל המסלולים האפשריים ובניית עצי חיפוש (belief trees). אחד מהאלגוריתמים המכוונים הנפוצים ביותר הוא Monte Carlo Tree Search (MCTS). הוא מתמודד עם "קללת ההיסטוריה" ו"קללת המימדיות" על ידי דגימה חכמה (לא אחידה) של מרחב החיפוש, בניית עץ אמונה ואיסוף סטטיסטיקה לכל צומת בעץ. האלגוריתם מאזן בין חקירה (exploration) לבין ניצול (exploitation) כדי לפתח צמתים מבטיחים, ומדי פעם גם צמתים לא מבטיחים שעלולים להניב תגמול גבוהה יותר.

מגבלה משמעותית של רוב השיטות הקיימות היא אי-השימוש במידע משלבי התכנון הקודמים בשלב התכנון הנוכחי. בעיית "איפוס" זו מובילה לבזבוז משאבים ונדרש "תקציב" גבוהה יותר על מנת למצוא פתרון מספק. מחקר זה מציע גישה חדשנית, חסכונית בחישוב, אשר כוללת שימוש בעצי חיפוש שנוצרו בשלבי התכנון הקודמים בתהליך קבלת ההחלטות הנוכחי. אנו מתמודדים עם קטגוריה רחבה של בעיות עם מרחבי מצב, פעולה ותצפית רציפים, שבהן התגמולים הם פונקציות כלליות של האמונה.

המחקר בוצע בהנחייתו של פרופסור חבר ואדים אינדלמן, בתוכנית הבין-יחידתית למערכות אוטונומיות ורובוטיקה.

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

M. Novitsky, M. Barenboim, and V. Indelman. "Previous Knowledge Utilization In Online Belief Space Planning." In: *IEEE Robotics and Automation Letters (RA-L). Submitted.* 2024. arXiv: 2412.13128.

תודות

אני רוצה להודות למנחה שלי, פרופסור חבר ואדים אינדלמן, על ההנחיה, הסבלנות והעזרה לאורך לימודי התואר השני שלי. היכולת שלו להנחות אותי תוך מתן חופש לחקור הייתה חיונית להצלחתי. אני מוקיר תודה למשפחתי, ובפרט לאשתי אנאל, על התמיכה והעידוד המתמיד שלהם לאורך כל הדרך.

הכרת תודה מסורה לטכניון על מימון מחקר זה.

ניצול ידע קודם בתכנון מקוון ולא פרמטרי במרחב האמונה

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים במערכות אוטונומיות ורובוטיקה

מיכאל נוביצקי

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
שבט התשפ"ה חיפה פברואר 2025

ניצול ידע קודם בתכנון מקוון ולא פרמטרי במרחב האמונה

מיכאל נוביצקי