2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Online Planning for Interactive-POMDPs using Nested Monte Carlo Tree Search

Jonathon Schwartz, Ruijia Zhou and Hanna Kurniawati
School of Computing, Australian National University
{jonathon.schwartz, ruijia.zhou, hanna.kurniawati}@anu.edu.au

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981713

***Abstract*— The ability to make good decisions in partially observed non-cooperative multi-agent scenarios is important for robots to interact effectively in human environments. A robust framework for such decision-making problems is the Interactive Partially Observable Markov Decision Processes (I-POMDPs), which explicitly models the other agents' beliefs up to a finite reasoning level in order to more accurately predict their actions. This paper proposes a new online approximate solver for I-POMDPs, called Interactive Nested Tree Monte-Carlo Planning (I-NTMCP), that combines Monte Carlo Tree Search with the finite nested-reasoning construction of I-POMDPs. Unlike existing full-width I-POMDP planners, I-NTMCP focuses planning on the set of beliefs at each nesting level which are reachable under an optimal policy and uses sampling to construct and update policies at each nesting level, online. This strategy enables I-NTMCP to plan effectively in significantly larger I-POMDP problems and to deeper reasoning levels than has previously been possible. We demonstrate I-NTMCP's effectiveness on two competitive environments. The results indicate that I-NTMCP can generate substantially better policies up to more than 50× faster than I-POMDP Lite – one of the fastest I-POMDP solvers today. In the *pursuit-evasion* domain, we show I-NTMCP can plan effectively in a complex problem with over 88K states, which is two orders of magnitude larger than existing I-POMDP planning benchmark problems.**

I. Introduction

To operate and interact effectively in human environments, robots need to be able to account for uncertainty in the their own actions and perceptions as well as uncertainty about the other agents' behaviours and reasoning. Partially Observable Markov Decision Processes (POMDPs) [1], [2] provide a mathematical framework for handling uncertainty in both the effect of actions and the agent's perceptions. However, they are designed for single-agent problems. Interactive-POMDPs (I-POMDPs) generalize POMDPs to multi-agent domains by incorporating behavioural models of the other agents [3]. I-POMDPs provide a framework of recursive reasoning which allows an agent to explicitly model the other agents, which in turn model other agents, and so on down to some finite depth. Similar to POMDPs, an I-POMDP agent never knows its exact state and must infer the best action to perform with respect to beliefs, where a belief is a distribution over the possible physical states. However, in an I-POMDP, the set of possible states is the joint product between the set of physical states and the set of possible models of the other agents. This makes I-POMDPs ideal for modelling decision making problems involving uncertainty about the other agents in the environment, including non-cooperative agents [4], [5], [6].

While I-POMDPs possess many desirable properties, their application has been restricted to relatively small problems due to their high computational complexity. I-POMDPs inherit the intractability of POMDPs [7] while introducing additional complexity, namely the curse of nested reasoning, due to the inclusion of other agent models. This difficulty has motivated research into more tractable methods of solving I-POMDPs [8], [9], [10], [11], [12]. However, the best solvers today are still limited to problems with only hundreds of states. This is a far contrast to existing online POMDP solvers [13] which are capable of handling problems with millions of discrete states and even continuous state spaces with many dimensions.

Motivated by recent advances in POMDP planning, this paper proposes an online I-POMDP planner based on Monte Carlo Tree Search (MCTS) [14], called Interactive Nested Tree Monte-Carlo Planning (I-NTMCP). I-NTMCP is designed for the subclass of I-POMDPs where there is common knowledge of the initial belief and models among agents. This sub-class captures many problems of practical interest and has been a focus of prior I-POMDP solvers [10], [12].

I-NTMCP focuses on computing the best action to perform from the current belief, which allows I-NTMCP to minimise the computation required in estimating the policy of the other agents. To find the best action from a belief, I-NTMCP constructs and maintains a sequence of inter-related belief trees, where each tree encodes an approximately optimal policy for an agent operating at a particular nesting level. This sequence of trees is built bottom-up, from the lowest to the highest nesting level, using MCTS.

The results are promising. Tests on two competitive two-agent problems indicate that I-NTMCP can plan effectively in a problem that is two orders of magnitude larger than existing I-POMDP benchmark problems.

II. Background on I-POMDPs

I-POMDPs generalize POMDPs [1], [2] to multi-agent settings by including models of other agents in the belief state space [3]. While I-POMDPs can be used to model any finite number of agents, in this paper we restrict ourselves to environments with two agents. With this is mind, and for readability, we limit the following background to I-POMDPs with only two agents.

Formally, a finitely-nested I-POMDP for an agent i with nesting level l interacting with another agent j is a tuple $\langle IS_i, \mathcal{A}, \mathcal{O}_i, \mathcal{T}_i, \mathcal{Z}_i, \mathcal{R}_i, \gamma \rangle$, where:

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.

- $IS_{i,l}$ denotes the set of *interactive states* defined as, $IS_{i,l} = S \times M_{j,l-1}$ for $l > 0$, and $IS_{i,0} = S$, where S is the set of physical states and $M_{j,l-1}$ is the set of possible models for agent j and nesting level $l-1$.
- $A = A_i \times A_j$ is the set of joint actions of all agents. Where $a = \langle a_i, a_j \rangle$ denotes a joint action.
- $\mathcal{O}_i$ is the finite set of observations for agent i. Where o_i denotes an observation for agent i.
- $\mathcal{T}_i : S \times A \times S \rightarrow [0, 1]$ is the Markovian state transition function, defining $Pr(s'|a, s)$.
- $\mathcal{Z}_i : S \times A \times \mathcal{O}_i \rightarrow [0, 1]$ is the Markovian observation function, defining $Pr(o_i|a, s)$.
- $\mathcal{R}_i : S \times A \rightarrow \mathbb{R}$ is the reward function.

In an I-POMDP, the environment interaction unfolds in discrete time steps. At each step, all agents simultaneously perform an action a_i and receive an observation o_i and reward r_i. The interaction can continue for a finite or infinite number of steps. A sequence of actions and observations of an agent i is an *agent history*, $h_i^t = \{a_i^1, o_i^1, ..., a_i^t, o_i^t\}$[1]. Similarly, a *joint history* is a sequence of joint actions and joint observations, $h^t = \{a^1, o^1, ..., a^t, o^t\}$, where $o = \langle o_i, o_j \rangle$ denotes a joint observation. $\mathcal{H}_i^t$ is the set containing all time t agent i histories, while $\mathcal{H}^t$ is the set containing all time t joint histories. For this paper we assume the optimality criteria for each agent is to maximise their expected discounted *return*, which for an agent i is the total discounted reward accumulated from time t onwards, $R_i^t = \sum_{k=t}^{\infty} \gamma^{k-t} r_i^k$, where γ is a discount factor.

Similar to [10], [11], [12], I-NTMCP focuses on the class of I-POMDPs where the frame, $\hat{\theta}_j = \langle A, \mathcal{O}_j, \mathcal{T}_j, \mathcal{Z}_j, \mathcal{R}_j \rangle$, for each agent is common knowledge and is fixed, the transition function is the same for both agents, $\mathcal{T} = \mathcal{T}_i = \mathcal{T}_j$, and at each level the modelling agent only considers the intentional models of the other agent that are one level below. Hence, we simplify the definition of the set of intentional models to be the set of possible beliefs of the other agent over the interactive states, $\Theta_{j,l-1} = \{b_{j,l-1} : b_{j,l-1} \in \Delta(IS_{j,l-1})\}$. Level 0 beliefs correspond to beliefs that do not explicitly consider other agents, and instead model them as part of the environment. A level 1 belief of agent i, $b_{i,1}$, is a belief over the state of the environment and the level 0 beliefs of agent i. This recursive definition can be extended to any desired level of nesting. Using this well defined hierarchy of nested beliefs, I-POMDPs can be solved via a finite recursion, where the solution at each belief level is built using the solution of the level directly below it [3].

Various exact and approximate I-POMDP solvers have been proposed, including methods based on model equivalence [15], particle filtering [9], value iteration [8], structural problem reduction [10], policy iteration [11], modelling other agents as finite state-automata [16], and function approximation using deep learning [12]. Furthermore, I-POMDPs solvers when transition and observation models are initially unknown have also been proposed [17], [18].

[1]Here and in the remainder of this paper the superscript notation is used to specify the time index.

Despite the above advances, existing I-POMDP solvers are still practically limited to problems with a few hundred states. This is in contrast to POMDP solvers which can now handle problems with millions of discrete states and continuous states spaces [19], [20], [21], [22], [23]. In this work we extend advances in online planning for POMDPs, specifically methods based on Monte Carlo Tree Search (MCTS) [20], to the multi-agent setting using the I-POMDP formalism.

Most closely related to our work is [16] which presents a method for online Monte Carlo planning using learned subintentional models of the other agent. We take this research a step further by using online planning with intentional models of the other agent to an arbitrary reasoning depth.

III. Overview of I-NTMCP

A. Problem Setting and Assumptions

I-NTMCP is an online approximate I-POMDP solver for the subclass of I-POMDPs where there is a common initial belief over states, b_0, and where all agents have knowledge of the other agents' frames and share a transition function. While the observation space, observation function, and reward functions of each agent are common knowledge, no assumptions are made about the form of the reward and observation functions. The assumptions we make are used in other I-POMDP solvers, however, unlike many other I-POMDP solvers which require an explicit representation of the I-POMDP functions, $\langle \mathcal{T}, \mathcal{Z}, \mathcal{R} \rangle$, I-NTMCP only requires a generative model $\mathcal{G}$ of the joint environment dynamics. This requirement of only a generative model is especially useful for domains that are difficult to model or where a compact representation of the transition or observation probabilities may not be available, as is often the case for large problems.

Similarly to the majority of prior work on I-POMDPs, in this paper we restrict ourselves to environments with two agents. Conceptually, it is possible to extend I-NTMCP to environments with more than two agents, however for N agents this would require maintaining $(N-1)^l$ belief trees for a given nesting level l. This complexity is not unique to I-NTMCP and problems with more than two-agents are a challenge for I-POMDP solvers in general, especially when dealing with deeper nesting levels. Conversely, the online sample-based nature of I-NTMCP opens up a number of avenues of investigation for efficiently scaling up I-NTMCP to more than two agents - a possibility for future work.

B. Algorithm Overview

At a high level, I-NTMCP constructs and maintains a sequence of inter-related belief trees. Each tree represents the set of beliefs reachable under an optimal policy and encodes an approximately optimal policy for an agent with a particular nesting level. We denote the sequence of inter-related belief trees for agent i as $\mathcal{T}_{i,L}, \mathcal{T}_{j,L-1}, \mathcal{T}_{i,L-2}, \cdots, \mathcal{T}_{k,0}$, where L is the nesting level being considered, and where $k = i$ if L is even and $k = j$ if L is odd. The notation $\mathcal{T}_{k,l}$ refers to the belief tree of agent k, where $k \in \{i, j\}$, operating at nesting level $l \in [0, L]$. Each node of $\mathcal{T}_{k,l}$ is associated with an agent history, h_k, and represents a belief

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.

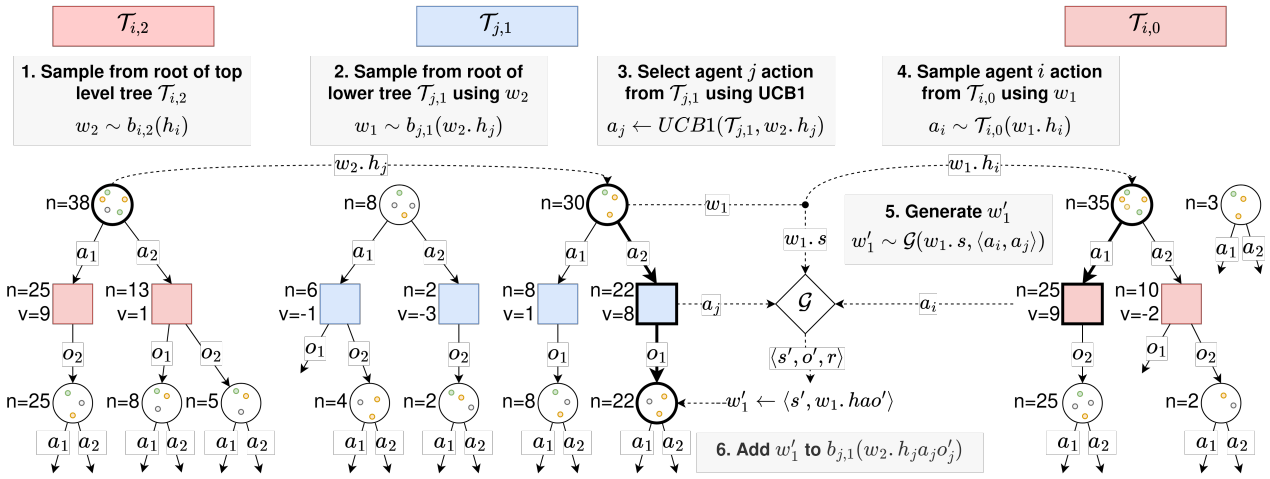


Fig. 1. Top-down construction of the I-NTMCP Nested Trees Data structure. Shown here is the first step of a single simulation for I-NTMCP using a nesting level of two for an environment involving two agents i (red) and j (blue) who both have two actions and observations. In this case the planning is being done for agent i, and the level 1 tree of agent j is being expanded. Following step 6, the simulation continues until a leaf node is reached, at which point a rollout is used to estimate the value of the leaf node.

$b_{k,l}$ for agent k when operating with nesting level l. Along with the belief, each node maintains an information tuple $\langle N_{k,l}, V_{k,l} \rangle$, where $N_{k,l}$ is the number of times the node has been visited and $V_{k,l}$ is the value of the belief represented by the node. Each edge of $\mathcal{T}_{k,l}$ corresponds to an action–observation pair $\langle a_k, o_k \rangle$ where $a_k \in \mathcal{A}_k$ and $o_k \in \mathcal{O}_k$. An edge from $b_{k,l}$ to $b'_{k,l}$ implies that there is a pair of action–observation $\langle a_k, o_k \rangle$ such that, if at belief $b_{k,l}$, agent k performs action a_k and perceives observation o_k, agent k's belief will become $b'_{k,l}$.

Now, the question is how do we represent the beliefs. I-NTMCP follows the nested-belief idea of I-POMDP, but represents the belief of the other agent using the agent's histories. The use of histories to represent beliefs is aligned with the representation used in many state-of-the-art online POMDP solvers. Specifically, we define a history-state w^t as a tuple $w^t = \langle s^t, h^t \rangle$, where $s^t \in \mathcal{S}$ is the physical state and $h^t = h_i^t \times h_2^t$ is the joint history of agent i and agent j. $\mathcal{W}^t$ denotes the set of all time t history-states [2] . A node in belief tree $\mathcal{T}_{k,l}$ that is associated with time t history h_k^t is then defined as a belief $b_{k,l}(w^t, h_k^t) = P(w^t \mid h_k^t)$, where $k \in \{i, j\}$ and $w^t \in \mathcal{W}^t$. I-NTMCP represents each belief node as a set of unweighted particles of history-states.

I-NTMCP expands the tree, and also updates beliefs, by evolving each particle. To evolve a particle w in belief node $b_{i,l}(\cdot\ , h_i)$, I-NTMCP must estimate the action that will be taken by the other agent, i.e., agent j at belief $b_{j,l-1}(\cdot\ , w.h_j)$. Such an action can be inferred from the best action from the belief node $b_{j,l-1}(\cdot\ , w.h_j)$ in tree $\mathcal{T}_{j,l-1}$. This dependency forms the inter-relation between the belief trees. Figure 1 illustrates the representation of trees in I-NTMCP, while the details of tree construction and belief updates are presented in the next section.

[2]We specify agent k's history contained in the history-state w^t using the notation $w^t.h_k^t$. Similarly for the state $w^t.s^t$, joint history $w^t.h^t$, joint action $w^t.a^t$, and joint observation $w^t.o^t$, agent k action $w^t.a_k^t$, and agent k observation $w^t.o_k^t$. When t is clear, we omit the time superscript, in which case $w = w^t$.

Last but not least, given the true belief over the space of history-states for an agent, say agent i, $b_i(w, h_i)$, and the fixed history-based policy for the other agent, say π_j, I-NTMCP converges to the optimal value function. This convergence is a straightforward derivation of the proofs in [3], [24], and [20].

IV. Details of I-NTMCP

I-NTMCP takes as its inputs the joint action space $\mathcal{A}$, the initial belief over physical states $b^0 \in \Delta(\mathcal{S})$, a fixed policy for the other agent at level 0 $\pi_{-k,0}(a_{-k}|h_{-k})$ corresponding to a subintentional model (e.g. uniform random), and a generative model $\mathcal{G}$ of the joint environment dynamics, $\langle s^{t+1}, o^{t+1}, r^{t+1} \rangle \sim \mathcal{G}(s^t, a^{t+1})$. Given this information, I-NTMCP constructs the sequence of inter-related belief trees online and bottom-up. Each tree is constructed using Monte Carlo Tree Search (MCTS). After each real action and observation, I-NTMCP updates beliefs top-down using particle filtering.

A. Constructing the Sequence of Inter-Related Trees

To begin, at time $t = 0$, I-NTMCP initializes the root of each tree $\mathcal{T}_{i,L}, \mathcal{T}_{j,L-1}, \cdots, \mathcal{T}_{k,0}$ to represent the initial belief $b^0 = b_{k,l}(\langle s^0, \emptyset \rangle, \emptyset)$, while the information tuple $\langle N_{k,l}, V_{k,l} \rangle$ in each root node is initialised to 0 for $N_{k,l}(b^0)$ and the highest expected immediate reward over all possible actions for $V_{k,l}(b^0)$. Without loss of generality, we will assume the level $l \in [0, L]$ tree $\mathcal{T}_{k,l}$ corresponds to agent k and use $-k$ to denote the other agent.

The I-NTMCP search algorithm is described in Algorithm 1. At each step t, I-NTMCP constructs the sequence of inter-related trees by expanding each tree for some fixed number of simulations M, starting with the lowest level tree $\mathcal{T}_{k,0}$ and working up to the top level tree $\mathcal{T}_{i,L}$. Each individual expansion begins from a time t root node corresponding to a time t history, h_k^t, of the agent at the level being expanded. For the top level tree $\mathcal{T}_{i,L}$ this will be the real time t history observed from the environment by agent

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.

i, h_i^t. Trees at lower levels $l < L$ may have many possible time t histories, since the nodes of lower level trees represent the possible histories of the agent at lower levels, which are not directly observable by the agent one level above. I-NTMCP uses top-down recursive sampling to ensure that during expansion the number of simulations assigned to a given time t node in the lower level trees is proportional to how likely the history associated with that node is, within the time t beliefs of the higher level trees. In this way planning computation is directed to provide better estimates for the lower level beliefs that are more likely within the higher level beliefs.

Specifically, for expanding the tree $\mathcal{T}_{k,l}$ at level l each simulation starts by sampling a history-state from the root node of the level L tree corresponding to the true observed history of agent i, $w_L^t \sim b_{i,L}(\cdot, h_i^t)$. The history contained in w_L^t is then used to sample a history-state from the level below, $w_{L-1}^t \sim b_{j,L-1}(\cdot, w_L^t.h_j^t)$. This recursion continues until level l, at which point MCTS is used to run a single expansion starting from the belief node corresponding to the agent k history in the sampled history-state, $w_{l+1}^t.h_k^t$. Each expansion, starts by sampling a history-state from the root of $\mathcal{T}_{k,l}$ and then traversing the tree $\mathcal{T}_{k,l}$ in two stages using Monte Carlo simulation and the generative model $\mathcal{G}$.

The first stage involves traversing $\mathcal{T}_{k,l}$ based on the UCT algorithm until a leaf node is reached. Suppose the simulation starts from the sampled history-state $w^t \in \mathcal{W}^t$. Similar to UCT, I-NTMCP needs to select an action to progress the Monte Carlo simulation. However, unlike a typical UCT algorithm, I-NTMCP needs to use a joint action $a \in \mathcal{A}$ to perform this simulation, which requires it to infer the action of the other agent. Therefore, we separate action selection into two types. For the agent represented by the tree being expanded, agent k in this case, actions are selected using UCB1 as is typical with UCT, $a_k \leftarrow \arg\max_{a \in \mathcal{A}_k} V_{k,l}(h_k^t a) + c\sqrt{\frac{logN_{k,l}(h_k^t)}{N_{k,l}(h_k^t a)}}$. For the other agent, $-k$, if $l > 0$ actions are sampled using the level $l-1$ belief tree, $\mathcal{T}_{-k,l-1}$, while if $l = 0$ the fixed level 0 policy, $\pi_{-k,0}$ is used. Actions are selected using the history for agent $-k$ contained in the sampled history-state, $w^t.h_{-k}^t$. When selecting actions from $\mathcal{T}_{-k,l-1}$ actions are sampled for the given $h_{\bar{k}}^t$ using a softmax function, $Pr(a_{-k}|h_{-k}^t) = \eta \exp \frac{N_{-k,l-1}(h_{-k}^t a_{-k})}{\sqrt{N_{-k,l-1}(h_{-k}^t)}}$, where η is a normalizing constant. The softmax policy accounts for $\mathcal{T}_{-k,l-1}$ being an approximation of the true level $l-1$ policy and it's accuracy being dependent on the visit count of the node. Once both actions have been selected, the joint action, $a^{t+1} = \langle a_k^{t+1}, a_{-k}^{t+1} \rangle$, is used along with the state in the sampled history-state, $w^t.s^t$, to generate the next state, joint observation, and joint reward, $\langle s^{t+1}, o^{t+1}, r^{t+1} \rangle \sim \mathcal{G}(w^t.s^t, a^{t+1})$. All this information together provides the next history-state, $w^{t+1} = \langle s^{t+1}, w^t.h^t a^{t+1} o^{t+1} \rangle$, which is added as a particle to the next belief $b_{k,l}(h_k^t a_k^{t+1} o_k^{t+1})$ as part of the Monte Carlo belief update process. The first stage continues until a leaf node in the search tree $\mathcal{T}_{k,l}$ is reached.

In the second stage, a rollout is used to estimate the value of the leaf node. Rollouts are performed in a similar manner

Algorithm 1 POSGMCP Search

Require: Generative model $\mathcal{G}$, Discount γ
Require: Nested Trees $\mathcal{T}_{1,L}, \mathcal{T}_{2,L-1}, ..., \mathcal{T}_{k,0}$
Require: Number of simulations per level M

procedure SEARCH(h_1)
  ▷ h_1 here is the real history for agent-1 at level L
  for $d \leftarrow 0, ..., L$ **do**
    for $1, ..., M$ **do**
      NESTEDSIMULATION(h_1, L, d)
    end for
  end for
  return $\arg\max_{a_1 \in \mathcal{A}_1} V_{1,L}(h_1 a_1)$
end procedure

procedure NESTEDSIMULATION(h_k, l, $depth$)
  $w \sim b_{k,l}(\cdot, h_k)$ ▷ $w = \langle s, \langle h_k, h_{-k} \rangle \rangle$
  if $l = depth$ **then**
    SIMULATE(w, h_k, l, 0)
  else
    NESTEDSIMULATION($w.h_{-k}$, $l-1$, $depth$)
  end if
end procedure

procedure SIMULATE(w, h_k, l, t)
  if $\gamma^t < \epsilon$ **then**
    return 0
  end if
  if $h_k \notin \mathcal{T}_{k,l}$ **then**
    for all $a_k \in \mathcal{A}_k$ **do**
      $\mathcal{T}_{k,l}(h_k a_k) \leftarrow \langle N_{init}(h_k a_k), V_{init}(h_k a_k), \emptyset \rangle$
    end for
    return ROLLOUT($w.s$, h_k, t)
  end if
  $a \leftarrow$ SELECTACTION(w, h_k, l)
  $\langle s', o, r \rangle \sim \mathcal{G}(w.s, a)$
  $w' \leftarrow \langle s', w.hao \rangle$
  $R_k \leftarrow r_k + \gamma$ SIMULATE($w', h_k a_k o_k, t+1$)
  $b_{k,l}(h_k a_k o_k) \leftarrow b_{k,l}(h_k a_k o_k) \cup \{w'\}$
  $N_{k,l}(h_k a_k) \leftarrow N_{k,l}(h_k a_k) + 1$
  $N_{k,l}(h_k a_k o_k) \leftarrow N_{k,l}(h_k a_k o_k) + 1$
  $V_{k,l}(h_k a_k) \leftarrow V_{k,l}(h_k a_k) + \frac{R_k - V_{k,l}(h_k a_k)}{N_{k,l}(h_k a_k)}$
  return R_k
end procedure

procedure SELECTACTION(w, h_k, l)
  $a_k \leftarrow \arg\max_{a \in \mathcal{A}_k} V_{k,l}(h_k a) + c\sqrt{\frac{logN_{k,l}(h_k)}{N_{k,l}(h_k a)}}$
  if $l = 0$ **then**
    $a_{-k} \sim \bar{\pi}_{-k,0}(\cdot | w.h_{-k})$
  else
    $a_{-k} \sim \mathcal{T}_{-k,l-1}(w.h_{-k})$
  end if
  return $\langle a_k, a_{-k} \rangle$
end procedure

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.

to Monte Carlo planning for POMDPs, with the exception that actions are selected for both agents. Starting from the sampled history-state at the leaf node w, a sequence of history-states is generated using rollout policies for each agent, until a terminal state or discount horizon is reached. The value of the leaf node is estimated by the return R_k from the rollout. During the rollout, actions for agent i and agent j are selected using history-based rollout policies, $\pi_{i,rollout}(a_i|h_i)$ and $\pi_{j,rollout}(a_j|h_j)$ - e.g. uniform random action selection or using domain knowledge. Following the rollout, the generated return R_k is used to update the value estimates of all the parents of the new leaf node. After each simulation, exactly one new node is added to the tree $\mathcal{T}_{k,l}$ which corresponds to the first new history encountered during that simulation.

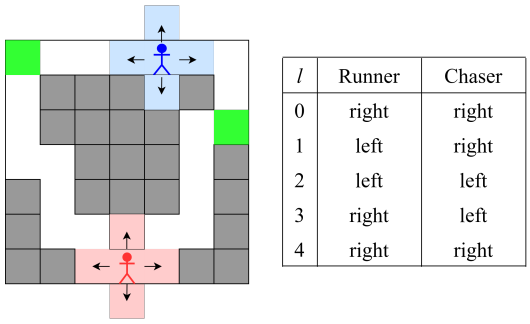
B. Belief Update

The I-NTMCP tree is updated top-down after each real step. After agent i at nesting-level L performs action $a_{i,L}^{t-1}$ and receives observation $o_{i,L}^t$, the root belief nodes of each search tree are updated to incorporate the new information. The tree $\mathcal{T}_{k,l}$ at each level is updated using a distribution over possible histories for the agent at that level. Let $X_{k,l}^t$ denote a discrete distribution over $\mathcal{H}_k^t$. At the top level L, the exact history for the agent is known since it is the actual history for the acting agent. Hence, $X_{i,L}^t$ is the discrete distribution with probability 1 for the true history of agent i and probability 0 for all other time t histories. Using $X_{i,L}^t$ the distribution over possible histories of the other agent, $X_{j,L-1}^t$, is calculated. $X_{j,L-1}^t$ specifies which time t belief nodes to keep and update in the tree $\mathcal{T}_{j,L-1}$. In general, given the history distribution $X_{i,l}^t$ for agent i at level $l > 0$, the probability of a given history $h_j^t \in \mathcal{H}_j^t$ for agent j at level $l-1$, denoted as $X_{j,l-1}^t(h_j^t)$ is given by:

$$X_{j,l-1}^t(h_j^t) = \sum_{h_i^t \in \mathcal{H}_i^t} \frac{X_{i,l}^t(h_i^t)}{|b_{i,l}^t(h_i^t)|} \sum_{w^t \in b_{i,l}^t(h_i^t)} \delta_K(w^t.h_j^t, h_j^t) \quad (1)$$

Where δ_K is the Kronecker delta function which is 1 when $w^t.h_j^t = h_j^t$ and 0 otherwise. $|b_{i,l}^t(h_i^t)|$ denotes the number of particles in the belief node of the level l tree for history h_i^t. Note that in eq. 1, $X_{j,l-1}^t(h_j^t)$ is calculated by iterating over the set of all time t histories for both agents. In practice the number of histories used is not directly dependent on the size of the time t history sets, but instead depends on the number of particles stored in the trees which in turn is a function of the number of simulations being used for planning and the observation branching factor of the environment. This means that even as the number of possible histories grows as the episode horizon grows, the update time remains relatively constant. Of course this speed comes at the cost of accuracy but one of the big advantages of MCTS based methods is that the compute resources will naturally be used to explore the most likely histories in expectation.

The I-NTMCP update function also employs a pruning and reinvigoration step. Similar to MCTS based POMDP planners, the pruning step removes unreachable branches from each tree. In I-NTMCP this corresponds to removing any time t histories in the tree that have a zero probability in the update history distribution $X_{i,l}^t$. The reinvigoration function adds additional particles to each remaining time t belief node in the tree in order to combat particle depletion. For our experiments we added a fixed number of particles K across the time t belief nodes in each tree. The number of additional particles added to a given belief node was weighted by the probability of that belief node's history. For a given history h_i^t the number of additional particles was equal to $KX_{i,l}^t(h_i^t)$. In this way only a constant number of particles were added each update and the particles were distributed based on belief node probability.



l	Runner	Chaser
0	right	right
1	left	right
2	left	left
3	right	left
4	right	right

Fig. 2. The 7x7 *Runner-Chaser* problem (left) and the optimal finite-nested reasoning path choice for each agent based on nesting level l (right).

V. Experiments

We evaluate I-NTMCP on two competitive environments. The aim of these experiments is two-fold. First, to evaluate the effectiveness of I-NTMCP for planning with nested-reasoning in the multi-agent, partially observable setting. Second, to evaluate the scalability of I-NTMCP in terms of nesting level and problem complexity.

A. Experimental Setup

As an online solver, I-NTMCP interleaves planning and execution. The planning time reported for I-NTMCP was based on the number of simulations used per execution step, which was set to be constant for a given experiment and agent across nesting levels. For example, an experiment using M simulations for I-NTMCP with nesting level L, would use M simulations at all nesting levels $l \in [0, L]$, resulting in a total of $(L+1)M$ simulations for each execution step. $K = M/16$ additional particles were added during belief reinvigoration after each execution step.

We set the UCT exploration constant c to be $c = R_{hi} - R_{lo}$, where R_{hi} was the highest return achieved during sample runs of I-NTMCP with nesting level 0 and $c = 0$ against a random opponent, and R_{lo} was the lowest return achieved by a random agent against a random opponent [20]. We used $\epsilon = 0.1$ with a discount $\gamma = 0.95$ giving a discount horizon of 45 steps. For all experiments, we used a uniform random policy for the other agent level 0 policy, $\pi_{-k,0}$.

Our implementation is open-source and available at `https://github.com/RDLLab/i-ntmcp`.

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.

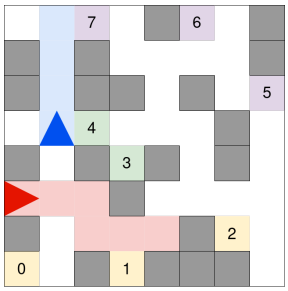


Fig. 3. The 8x8 *Pursuit-Evasion* problem. Cells 3 and 4 are possible start locations of the *Pursuer* (blue). Cells 0-2 & 5-7 are possible start and goal locations for the *Evader* (red). The evader's start and goal locations are always on opposite sides of the map, shown by the different colors. The field of vision for both agents expands in a cone in the direction the agent is facing as shown by the blue and red colored cells.

B. Scenarios

1) Runner-Chaser: Figure 2 illustrates the 7x7 *Runner-Chaser* scenario. The runner (red) must reach one of the green goal squares while avoiding capture by the chaser (blue). Each agent knows the environment exactly, including the start and goal locations of both agents. Both agents have four deterministic actions corresponding to moving one cell in each of the four cardinal directions. They observe perfectly the four adjacent cells, specifically whether each cell is empty, has a wall, or contains the opponent. The runner is considered caught if the chaser observes the runner. For both agents each action has a small cost of -1. The runner gets a reward of $+100$ if it reaches the goal and -100 if it gets caught, while the chaser receives the opposite terminal rewards. An episode ends if the runner reaches a goal, is caught, or has moved over 20 steps (which is considered a draw). Three problem sizes were used: 3x3, 4x4, and 7x7.

This environment requires agents to reason about the sequential strategy of the opponent. The map is designed so that the chaser cannot guard both goals and therefore must reason about which goal the runner will target. Meanwhile, the runner must reason about which goal the chaser will guard. Predicting the strategy of the opponent requires modelling the opponent's sequential decision making problem.

2) Pursuit-Evasion: Inspired by [25], the pursuit-evasion problem adds significant additional complexities to the runner-chaser problem. Similarly, it involves an *evader* (red) trying to reach a goal location before it is caught by a *pursuer* (blue). However, unlike the runner-chaser problem the goal location of the evader is not known to the chaser. This is analogous to the chaser having to reason about different models for the runner. Furthermore, the map is larger and more complex with more path options. Each agent is aware of the start location of their opponent, while only the evader knows it's goal location. The pursuer only knows the possible goal locations for the evader. Lastly, the observation space of both agents is expanded. Each agent receives six bits per step. Four bits indicate whether there is a wall or not in each of the cardinal directions, one bit indicates whether the opponent can be seen in the agent's field of vision, and the final bit indicates whether the opponent can be heard within Manhattan distance 2 of the agent. Due to the lack of precision of these observations, the pursuer never knows the exact position of the evader and vice versa. Knowledge of the shortest path to the goal was incorporated into the evader using preferred actions and rollout policy. The action space, rewards and terminal conditions are the same as *runner-chaser* except the step limit is increased to 40. The transition and state space are also the same with the addition of the direction each agent is facing and the goal location for the evader. The *Pursuit-Evasion* problem has 88,752 states, 4 actions and 64 observations per agent, more than two orders of magnitude larger than existing I-POMDP benchmarks.

C. Results

1) Scalability: To evaluate the scaling performance of I-NTMCP, we compare it with I-POMDP Lite [10] in the *Runner-Chaser* scenarios. For I-POMDP Lite we used the original program as provided by the authors. I-POMDP Lite is considered one of the fastest I-POMDP solvers, albeit an offline solver. We ran both I-POMDP Lite and I-NTMCP using nesting level 1 to compute the policy for the Runner, against random and POMDP Chasers. The POMDP Chaser models the runner's actions as a uniform random policy.

Table I indicate that as the problem size increases, I-NTMCP scales significantly better than I-POMDP Lite. Even if we aggregate the planning time for I-NTMCP over the entire planning horizon (20 steps), the total planning time of I-NTMCP is over $50\times$ less than I-POMDP Lite for the 7x7 scenario, while generating a substantially better policy.

Moreover, in all but one scenario, I-NTMCP computes equal or better policy than I-POMDP Lite. The only result where I-NTMCP performs worse than I-POMDP Lite is on the 4x4 problem against a Random Runner. The reason is that unlike I-NTMCP, during execution, I-POMDP Lite has access to the opponents performed action for its belief update. This information allows I-POMDP Lite to know the exact location of the opponent and overcome errors in it's opponent modelling introduced by the random opponent. Despite this additional advantage, the results in Table I indicate that when the problem size becomes larger, I-POMDP Lite's ability to generate a good policy suffers much more than I-NTMCP.

The performance of I-NTMCP in *Pursuit-Evasion* is shown in Table II. Since the problem is too big for I-POMDP Lite, in *Pursuit-Evasion* we compared I-NTMCP with uniform random and shortest path baselines. The shortest path policy selects actions that follow the shortest path to the goal (evader policy) or the evader's start position (pursuer policy). I-NTMCP outperformed the two baseline policies against each opponent policy. Furthermore, as the nesting level of the opponent increases, deeper nesting levels are required to perform well, as shown by the drop in performance of I-NTMCP $l \in [0, 1]$ between the different opponents. These result indicate I-NTMCP's ability to plan using finite-nested reasoning on a very large I-POMDP.

2) Convergence: We used the *Runner-Chaser* scenario to evaluate the convergence of I-NTMCP. In this scenario, the optimal finite-nested reasoning policy (FNR) for both the

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.

Problem, Chaser	$\|\mathcal{S}\|$	I-POMDP Lite $l=1$			I-NTMCP $l=1$		
		H	Time (s)	Mean Disc. Return	Simulations	Time (s)	Mean Disc. Return
3x3, Random	128	2	101	94.00 ± 0.00	1024	0.41 ± 0.00	94.00 ± 0.00
3x3, POMDP	128	2	101	94.00 ± 0.00	1024	0.45 ± 0.00	94.00 ± 0.00
4x4, Random	288	4	2524	**82.90 ± 0.79**	1024	0.67 ± 0.00	52.56 ± 3.70
4x4, POMDP	288	4	2524	63.80 ± 0.00	1024	0.69 ± 0.00	**77.73 ± 0.01**
7x7, Random	1,152	1	5791	-12.91 ± 0.11	4096	5.71 ± 0.17	**54.94 ± 1.55**
7x7, POMDP	1,152	1	5791	NA	4096	5.77 ± 0.32	**56.23 ± 1.31**

TABLE I

COMPARISON OF I-NTMCP AND I-POMDP LITE USING NESTING LEVEL 1 IN *Runner-Chaser*. FOR I-POMDP LITE, WE USE THE SHORTEST BEST PERFORMING PLANNING HORIZON (H) COMPLETED IN 1 HOUR FOR THE 3X3 AND 4X4 PROBLEMS AND 2 HOURS FOR THE 7X7 PROBLEM. FOR I-POMDP LITE THE TIME COLUMN SHOWS TOTAL PLANNING TIME, FOR I-NTMCP IT SHOWS MEAN PLANNING TIME PER STEP ($\pm$ SD). THE MEAN DISCOUNTED RETURNS ($\pm$ 95% CI) USING $\gamma = 0.95$ ARE SHOWN. THE POMDP OPPONENT WAS GENERATED USING PBVI [26] AND POMCP [20] FOR IPOMDP LITE AND I-NTMCP, RESPECTIVELY. PBVI COULD NOT BE RUN FOR THE 7X7 PROBLEM DUE TO PROBLEM SIZE. THE RESULTS ABOVE ARE BASED ON 1000 SIMULATION RUNS FOR EACH PROBLEM AND CHASER.

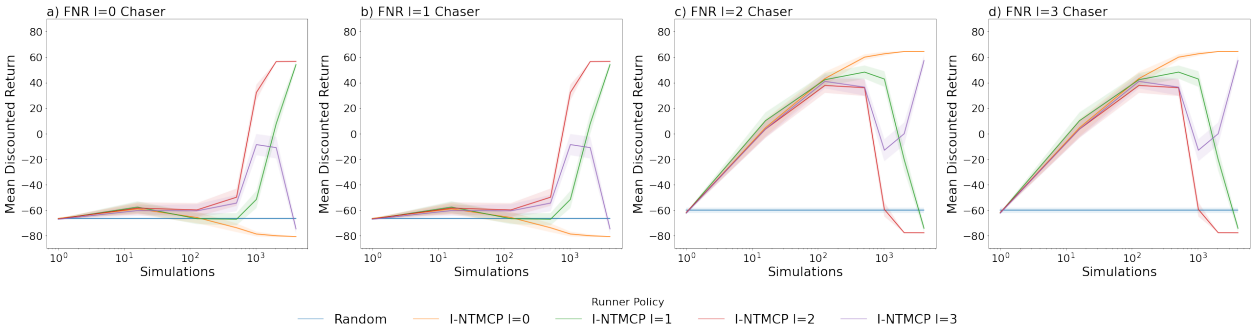


Fig. 4. Performance of I-NTMCP runner against Finite-Nested Reasoning (FNR) chaser in the *Runner-Chaser* problem. Each line shows the mean discounted return ($\pm$95% CI) for a random policy and I-NTMCP with different nesting levels l as the number of planning simulations increases. Each figure shows results against a FNR chaser with a different nesting level l.

Pursuit-Evasion	Random	Shortest Path	I-NTMCP $l=0$	I-NTMCP $l=1$
Random	0.31	0.52	0.06	0.08
Shortest Path	0.94	0.72	0.35	0.4
I-NTMCP $l=0$	**1.00**	0.59	0.4	0.33
I-NTMCP $l=1$	0.99	0.79	0.53	0.36
I-NTMCP $l=2$	0.97	**0.80**	**0.68**	**0.53**
I-NTMCP $l=3$	0.99	0.75	0.56	**0.53**

TABLE II

WIN RATE FOR PURSUER (ROW) AGAINST EVADER (COLUMN) AGENTS IN THE *Pursuit-Evasion* PROBLEM. RESULTS ARE FROM 100 EPISODES, WITH I-NTMCP USING 2048 SIMULATIONS.

runner and chaser can be easily derived and provides a stable baseline policy for comparison. The optimal policies by nesting level are shown in Figure 2 (right). We ran I-NTMCP for the Runner policy using increasingly many simulations, against the the optimal chaser policy.

The results (shown in Figure 4) indicate that as computational resources increase, the performance of the I-NTMCP policy resembles the FNR policy. For instance, when the Chaser uses nesting level 0 (i.e., moving to the right), the optimal policy for the Runner is when the Runner's nesting level is 1 or 2, as the Runner chooses the opposite path to the Chaser. The opposite is true for the Runner at nesting levels 0 and 3. The left most plot in Figure 4 indicates that the policies generated by I-NTMCP indeed converge to the expected behaviour of the FNR policies –that is, for nesting level $l \in \{1, 2\}$, the Runner policies generated by I-NTMCP converge to winning values, while for $l = \{0, 3\}$, the policies converge to loosing values. This trend is present for all nesting levels we tested.

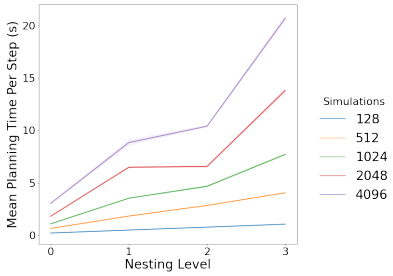


Fig. 5. I-NTMCP mean planning time per step ($\pm$95% CI) for different levels of nesting and simulation number in 7x7 *Runner-Chaser*. Deviations from linear for nesting level two are due to the runner agent's ability to reach terminal states earlier during search, leading to a small difference in mean planning time between runner and chaser agents.

It is worth pointing out that the choice of nesting level can have a significant impact on performance, as indicated by Figure 4. In this and prior work [10], [11], [12] the nesting level is treated as a hyper parameter chosen by the user. We hypothesize that for many practical problems where agents only weakly interact, the performance will be robust to the choice of nesting level given it is sufficiently high. However, ultimately it would be desirable to have the agent infer the nesting level of the other agent online as they interact.

Moreover, Figure 5 indicates that the planning time for I-NTMCP is linear with nesting level, allowing it to scale well for problems that require deeper levels of nesting.

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.

VI. Conclusion

In this paper, we have presented the I-NTMCP algorithm for online planning in multi-agent, partially observable environments. I-NTMCP combines Monte Carlo based planning for POMDPs with the finite-nested reasoning construction of I-POMDPs by constructing and maintaining a sequence of inter-related belief trees. Our experiments indicate that I-NTMCP can plan effectively in significantly larger two-agent I-POMDPs than similar existing methods. Furthermore, I-NTMCP requires only a generative model of the joint environment dynamics, permitting it's application to problems that cannot be easily modelled.

Future work abounds. For instance, although I-NTMCP indicates promising results, the choice of nesting level used is manually selected by the user, similar to [10], [11], [12]). It would be desirable for the agent to infer the nesting level of the other agent online as they interact. Moreover, efficient methods for further scalability and extension to continuous spaces would also be desirable.

Last but not least, I-POMDP is a general and principled decision-making framework that accounts for various types of uncertainty and other agents' reasoning. We foresee that in scenarios that require close and substantial interactions between robots and human, such a principled framework would be required to help ensure safety and natural interactions between robots and human. This paper aims to improve the practicality of such a decision-making framework.

VII. Acknowledgements

This work is supported by an AGRTP Scholarship and the ANU Futures Scheme. The authors would like to thank Marcus Hutter for helpful discussions around the theoretical analysis and the anonymous reviewers for their feedback.

References

[1] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Operations research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[3] P. J. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings," *JAIR*, vol. 24, pp. 49–79, 2005.

[4] B. Ng, C. Meyers, K. Boakye, and J. Nitao, "Towards applying interactive POMDPs to real-world adversary modeling," in *IAAI*, 2010.

[5] F. Wang, "An I-POMDP based multi-agent architecture for dialogue tutoring," in *ICAICTE*, 2013, pp. 486–489.

[6] M. P. Woodward and R. J. Wood, "Learning from humans as an I-POMDP," *arXiv preprint arXiv:1204.0274*, 2012.

[7] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[8] P. Doshi and D. Perez, "Generalized point based value iteration for interactive POMDPs." in *AAAI*, 2008, pp. 63–68.

[9] P. Doshi and P. J. Gmytrasiewicz, "Monte-carlo sampling methods for approximating I-POMDPs," *JAIR*, vol. 34, pp. 297–337, 2009.

[10] T. N. Hoang and K. H. Low, "Interactive POMDP Lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents," in *IJCAI*, 2013, pp. 2298–2305.

[11] E. Sonu and P. Doshi, "Scalable solutions of interactive POMDPs using generalized and bounded policy iteration," *Auton Agent Multi-Agent Syst*, vol. 29, no. 3, pp. 455–494, 2015.

[12] Y. Han and P. Gmytrasiewicz, "IPOMDP-Net: A deep neural network for partially observable multi-agent planning using interactive POMDPs," *AAAI*, vol. 33, pp. 6062–6069, 2019.

[13] H. Kurniawati, "Partially Observable Markov Decision Processes and Robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, 2022.

[14] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *ICCG*, 2006, pp. 72–83.

[15] B. Rathnasabapathy, P. Doshi, and P. Gmytrasiewicz, "Exact solutions of interactive POMDPs using behavioral equivalence," in *AAMAS*, 2006, pp. 1025–1032.

[16] A. Panella and P. Gmytrasiewicz, "Interactive POMDPs with finite-state models of other agents," *Auton Agent Multi-Agent Syst*, vol. 31, no. 4, pp. 861–904, 2017.

[17] B. Ng, K. Boakye, C. Meyers, and A. Wang, "Bayes-adaptive interactive POMDPs," in *AAAI*, vol. 26, no. 1, 2012, pp. 1408–1414.

[18] Y. Han and P. Gmytrasiewicz, "Learning others' intentional models in multi-agent settings using interactive POMDPs," in *NeurIPS*, 2018, pp. 666–673.

[19] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *RSS*, 2008.

[20] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," *NeurIPS*, vol. 23, pp. 2164–2172, 2010.

[21] H. Kurniawati and V. Yadav, "An online POMDP solver for uncertainty planning in dynamic environment," in *Robotics Research*, 2016, pp. 611–629.

[22] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," *JAIR*, vol. 58, pp. 231–266, 2017.

[23] Z. N. Sunberg and M. J. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *ICAPS*, vol. 28, 2018, pp. 259–263.

[24] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," *JAIR*, vol. 13, pp. 33–94, 2000.

[25] I. R. Seaman, J.-W. van de Meent, and D. Wingate, "Nested reasoning about autonomous agents using probabilistic programs," *arXiv preprint arXiv:1812.01569*, 2018.

[26] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *JAIR*, vol. 27, pp. 335–380, 2006.

Authorized licensed use limited to: Australian National University. Downloaded on January 02,2023 at 23:18:27 UTC from IEEE Xplore. Restrictions apply.