Scalable Individual Planning
In
Open and Typed Agent Systems

by

Maulik Shah

(Under the Direction of Prashant Doshi)

Abstract

Individual planning in open multiagent systems, which involve a large number of agents and no communication medium among them, is a particularly difficult problem to solve due to severe uncertainty and exponential computational complexity with each added agent. Furthermore, the agent openness requires an agent to predict the presence of other agents as each agent can leave or rejoin the environment during the operation, and wrongly predicting an agent's presence can add to a non-optimal behavior. For example, autonomous firefighting robots having no communication medium among them, tasked with fighting wildfires, may run out of suppressants and be temporarily unavailable to assist their peers. That requires agents to predict not only the actions of all the firefighting agents yet their suppressants levels as well, which becomes computationally intractable with the increase in the number of agents. To solve such a complex problem, this research proposes a novel method in this context that enables an agent to scalably and individually reason about others' presence as well as their actions in its shared environment. With effective use of well-established sampling approaches in statistics, the method uses a principled approach to sample only a subset of neighboring agents and extrapolate their models to the overall population and combines it with an extension of Monte Carlo tree search to individual agent reasoning in multiagent

environments. Simulations of multiagent wildfire suppression experiments demonstrate the potency of the new framework compared with alternative baseline methods and manifest the new framework as a reliable methodology for individual planning in multiagent systems with a large number of agents. The results exhibit the utility of the sampling methods in reducing the effects of the agents facing the Volunteer's dilemma and directionless mental models.

INDEX WORDS: Individual Planning, Planning under Uncertainty, Multiagent Systems, Monte Carlo Tree Search, $I-POMDP$, Agent Openness, Planning under anonymity, $I-POMCP$

Scalable Individual Planning
In
Open and Typed Agent Systems

by

Maulik Shah

B.E., Gujarat Technological University, India, 2012.

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2019

Scalable Individual Planning
In
Open and Typed Agent Systems

by

Maulik Shah

Major Professor: Prashant Doshi

Committee: Khaled Rasheed
Fredrick Maier

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
May 2019

Dedication

Dedicated to my beloved parents and grandmother, whose constant unconditional support has made this possible.

Acknowledgments

I want to express my deep gratitude towards Dr. Prashant Doshi for his constant guidance and encouragement throughout this research project. Dr. Adam Eck from Oberlin College has pioneered this project, and Dr. Leenkiat Soh from the University of Nebraska has provided his deep insights for advancement at each stage. Working with this expert team, I feel to have grown significantly in the area of decision planning.

I am thankful to the Institute for Artificial Intelligence for partially funding my research and reducing my financial stress. Dr. Fredrick Maier and Dr. Khaled Rasheed have served graciously on my committee and have provided steady support to remove hurdles during this research. I would also like to thank Ms. Sonya Tino for her support with administrative work.

I would like to extend my thanks to Muthu for spending his precious time to help me begin this project. Adithya, Anuja, Keyang, Shibo, Saurabh, and Nihal from THINC lab have helped me brainstorm several issues and find a way forward. In the end, I would like to thank my family members and friends for their unconditional support.

Table of Contents

List of Figures

Chapter 1

Introduction

Intelligent agents in real-world applications can be modeled cooperative or competitive according to the requirements of their systems. Although an agent in a multiagent system (MAS) incorporating many agents can choose its actions just by observing its surroundings, it can be considerably be benefited by reasoning how the other agents would behave to achieve its cooperative or self-interested goals.[1] For example, a wildfire fighting robot needs to know where the peers will spend their limited suppressant resources to determine how to contribute to the shared goal of quickly putting out all the fires. Likewise, an autonomous ride-sharing car needs to predict both where potential customers are located and the locations of its competitors, so that it can appropriately position itself to maximize its daily revenue. Such a level of reasoning reflects the natural human behavior characteristics, specifically when humans try to approximate the behavior of their peers while making a decision in an environment involving many individuals. For example, a quarterback in a college football game would throw the ball to a receiver, only if he observes the ground positions of the fellow team members in a favorable condition to advance the ball, the receiver in a suitable position to receive the ball, and his prediction of possible actions of the opposing team members do not hinder the throw.

In individual planning frameworks, predicting other agents' actions is an established way of interaction amongst agents in the systems where a direct communication channel is not a feasible way to interact. [2, 3] The assumption of not having a communication channel

[1] In this document, the 'other agents' and the 'neighboring agents' terms are used interchangeably. A neighboring agent can be defined as an agent connected with the subjective agent via a shared action region in an interaction subgraph. [1]

is realistic and can be applied to many real-world applications. For example, the agents fighting the wildfire in the remote areas might be made up of collaborative teams who do not share the communication protocol. Similarly, autonomous surveillance drones from different organization might not share the same communication protocol or even do not have any communication channel to collaborate in a rescue operation.

Modeling other agents for an autonomous agent is a quite challenging task in a system with many agents and these challenges are contributed to primarily by two factors: (1) agent openness, and (2) agents with different types. The agent openness occurs whenever any agent either leaves the system temporarily or permanently. [4] In this research, the focus in agent openness is limited by an added assumption that no new agent joins the system during operation. For example, a firefighter fighting a wildfire might leave the environment due to empty suppressants and rejoins after refilling them, though the number of active firefighters in the system remains constant from the beginning to the end. Effectively, agent openness requires an agent to model other agents' presence in the system, in addition to their actions in the given circumstances which largely increases the computational complexity. Inaccurately predicting the presence of other agents can lead to suboptimal decisions and substantial degradation in performance. For example, a ride-sharing car may leave the system due to empty fuel or for maintenance reasons; if an autonomous car agent in such a system predicts the presence of its competitors incorrectly, then it might lose customers and revenue. While the agent openness remains a challenge, the diversity of the agents in an environment can make its effects worse. For example, a firefighting agent fighting wildfires can be anyone from a ground firefighting unit, a helicopter, or an airplane. All such firefighters have their own effectiveness in operation and their individual properties, for which an agent needs to predict the number of firefighters of each type in the system. With each newly added agent, even approximating the presence of agents with different types becomes computationally intractable which makes scaling the planning methods in such systems extremely challenging.

1.1 Related Work

Decision planning in partially observable MAS has interested many artificial intelligence (AI) researchers over several decades, while formalization of a single agent planning using the partially observable Markov decision process ($POMDP$) has paved a way for advancement in this field. [5] Several frameworks have been introduced to solve MAS by using individual or decentralized planning based on $POMDP$ framework. $Dec-POMDP$, a popular framework for decentralized planning, assumes that the agents are fully cooperative, and it has a common reward function to represent this idea. [6] Furthermore, it considers communication as an integral part of the framework and performs planning centrally for all the agents except some local state factors. On the other hand, Interactive - POMDP ($I-POMDP$), a decision-making framework for individual planning, assumes no communication channel between the agents and provides each individual autonomy in planning. [2] In the $I-POMDP$ framework, each agent performs planning using an interactive state, which incorporates the mental models of other agents and planning agent's state factors.

The $I-POMDP$ framework does not scale well with the number of agents due to an exponential increase in the state-space with each added agent. The $I-POMDP^{Lite}$ framework makes an improvement on this problem by making an assumption that while planning individually, the other agents have full observability of the system, and it introduces the $Nested-MDP$ framework for assessing other agents' policies prior to performing the actual planning. [3] The inclusion of agent openness in individual planning has helped to address the planning requirement of open agent systems [7]; to achieve this, Bayes adaptive $POMDP$ is used to convert the learning problem of predicting agents' behavior of leaving and reentering the system to a planning problem by creating an empirical distribution over this behavior using Dirichlet process. [8, 9] All mentioned individual planning frameworks focus on planning with a small number of agents hardly stretching to two digits and do not work well with the systems involving many agents. The introduction of the idea of planning under agent anonymity can be viewed as substantial progress in individual planning for the

systems involving a large number of agents. [10] The agent anonymity framework generalizes individual agents of the same type and doing the same action under frame-action configurations and drastically reduces the possible number of joint actions. The experiments have demonstrated that planning under agent anonymity can scale well in the systems involving hundreds of agents. In the $Dec-POMDP$ literature, several attempts have been made to plan for a small number of agents as well as for a large number of agents, though none of the investigations incorporates agent openness. [11, 12, 13]

In this investigation, to cope up with the computational complexity incurred by planning for many agents, a bandit-based Monte Carlo (MC) approximating technique is used. Previously, several frameworks for single-agent planning in Markov decision process (MDP), known as MC tree search ($MCTS$) [14] and in $POMDP$, known as partially observable MC planning ($POMCP$) [15], have been introduced for online planning. Furthermore, several research studies have revealed some interesting improvements and use cases of such planning in practical applications. [16, 17, 18] MC frameworks employ a branch and bound methodology to create AND-OR trees and apply a bandit-based heuristic for tree exploration. One previous investigation for incorporating the $POMCP$ with the $I-POMDP$ was focused on just two agents, yet presented an interesting use case of individual planning in social interaction. [19] Though it was a novel approach for MAS in individual planning, it did not focus on scaling the solutions to many agents. Another such investigation in multiagent $POMDP$s ($MPOMDP$) for many agent MAS suggested using factored statistics and factored trees to obtain an optimal solution, however, the approach did not incorporate agent openness. [20] An $MPOMDP$ is a multi-agent planning model that unfolds over a number of steps and receives individual observations at each step. However, in an $MPOMDP$, all individual observations are shared via communication, allowing the team of agents to act in a centralized manner. [21]

1.2 Contributions

This research makes the following contributions. First, the research combines the prior methods for modeling a few other agents in open agent systems, such as wildfire suppression, and many agents in general settings using agent anonymity to account for both properties simultaneously. Second, the research incorporates agent anonymity in nested modeling over the random other agents and introduces a filtering approach to restrict the most unlikely combinations of the other agents' actions to reduce the planning time. Third, to further improve the scalability of agent reasoning, a sampling-based solution where agents intelligently select a subset of peers to model and subsequently extrapolate their expected behaviors to the larger set of all agents is introduced, which increases the efficiency of reasoning, made difficult by agent openness. Fourth, the presented method utilizes a generalization of $MCTS$ in multiagent planning, accounting for an agent's predictions of others actions.

Finally, the research expands previous benchmark simulations of wildfire suppression to include a larger number of agents and empirically demonstrate the benefits of the new method over the standard baseline methods which include a no-operation baseline, a heuristic baseline, and a nested-reasoning baseline. The results demonstrate that the new framework fares better than the standard baseline methods in the majority of experimental setups while produces statistically equivalent results in the remaining ones. Several interesting phenomenons such as Volunteer's dilemma has been observed in the agents' behavior, and more such effects have been reduced effective use of sampling errors. Overall, the new methodology demonstrates itself as a reliable framework for individual decision planning in many agent MAS.

1.3 Document Structure

This document is organized in a standard thesis document format as follows. Chapter 2, Background, introduces major frameworks and methodologies employed in this research and changes incorporated in them to produce the novel framework. Chapter 3, Methodology, provides details about the primary algorithm of this research and describes the methods of sampling and extrapolation. Chapter 4, Experiments, reports the results and provides analysis of the experiments conducted on the wildfire domain to demonstrate the potency of the new framework over the standard baseline methods. Chapter 5, Conclusion and Future Works, explains the takeaways from this research and the possible enhancements in the current methodology to improve performance and usability. The Appendix sections provide some quantitative analyses to enhance the reader's understanding of the concepts.

Chapter 2

Background

This chapter describes several fundamental frameworks and concepts used in this research, while it also establishes the base for the primary algorithm of this research, Interactive - partially observable Monte Carlo planning ($I - POMCP$). First, the chapter provides an overview of $POMDP$, one of the core decision-theoretic frameworks; the $POMDP$ section also introduces the concepts of policy and value function. Next, the $I - POMDP^{Lite}$ section explains the traits of the $I-POMDP^{Lite}$ framework and how the concept of agent anonymity and configuration filtering is incorporated into it. In the end, the prerequisite for the $I - POMCP$ framework, the 2AND-OR tree and unique aspects of its design are established, after a brief explanation of its single agent predecessors.

2.1 Partially Observable Markov Decision Process ($POMDP$)

Decision-theoretic planning in a single agent setting under partial observability is formalized by the partially observable Markov decision process ($POMDP$). $POMDP$ is a well-established framework based on Bayesian reasoning which provides a theoretical assurance for finding optimal planning solutions under uncertain conditions. [5]

Mathematically, a $POMDP$ can be defined as the following tuple:

$$POMDP = \langle S, A, T, \Omega, O, R, OC, \gamma \rangle \tag{2.1}$$

where,

- S is the physical state space of the environment. Extending this definition for state factors, a physical state might be factored into multiple variables called state-factors (F), which

can be represented as $S = F_1 \times F_2 \times \ldots \times F_k$. For example, a state in a wildfire domain can be made up of k state-factors representing the intensities of k fires in the environment;

- A is the set of Actions the agent can perform;
- $T : S \times A \times S' \rightarrow [0, 1]$, is a stochastic transition function which provides a distribution over the next state (S') after performing an action (A) on the current state (S);
- Ω is the set of observations the agent receives;
- $O : S' \times A \times \Omega : [0, 1]$, is a stochastic observation function which provides the probability with which the agent receives an observation (o) given the resulting state (S') and the performed action (A);
- $R : S \times A \rightarrow \mathbb{R}$, is a reward function that provides the reward for the agent given its state and the action from the state;
- OC is the optimality criteria to maximize a sum of the discounted rewards obtained over a fixed number of steps H (horizon) or over an infinite number of steps;
- $\gamma \in (0, 1]$ is a discounting factor used for weighting the reward values over a range of future time steps.

Due to partial observability, the agent does not know its current state exactly, therefore it maintains a probability distribution over possible current states, termed as an agent's belief or a belief state (b). The agent updates its belief at each time step using the Bayes filter, and the belief update takes into account the previous belief over the possible states, b, the agent's action at the previous time step, a, and the received observation after the transition, o. The new belief, b' for a current possible state (s') can be given as:

$$b'(s') = \beta O(o, s', a) \sum_{s \in S} b(s) T(s, a, s') \tag{2.2}$$

where $\beta = 1/O(o, b, a)$ is a normalization constant. Above equation can be summarized for an entire belief as $b' = SE(b, a, o)$.

The solution of a $POMDP$ is a policy (π), which is a function that maps the agent's beliefs to a distribution over actions, $\pi : B \times A \rightarrow [0, 1]$. Modestly, the policy is nothing other

than a strategy for an agent to perform a possible best action in the given environmental state. To obtain the utility of an action over a given belief, a value function in a form of a Bellman equation is defined to provide the expected long-term reward as:

$$V^h(b) = \max_{a \in A} \sum_{s \in S} b(s) \left(R(s,a) + \gamma \sum_{s'} T(s,a,s') \sum_{o \in \Omega} O(o,a,s') V^{h-1}(SE(b,o,a)) \right) \quad (2.3)$$

where h is the horizon.

This value function can be optimized by choosing the actions which can provide the maximum utility in a given state, that in turn leads to an optimal policy (π^*). There are many algorithms available in the $POMDP$ literature which can be employed to needed optimization of this function, and one of the most popular of these algorithms is Value Iteration, which iteratively optimizes the value function to obtain an optimal policy. [22] The policy can be a stochastic policy ($a \sim \pi(b)$) which provides a distribution over action against a belief state or a deterministic policy ($a = \pi(a|b)$) which provides a deterministic action against a belief state.

When the agent has full observability in the environment, the $POMDP$ structure becomes an MDP, and since the agent precisely knows its current state, there remains no need to maintain a belief over possible current states. In such conditions, the stochastic policy ($a \sim \pi(s)$) would return a distribution over action given a physical state (s) and a deterministic action in case of a deterministic policy ($a = \pi(s)$).

2.2 $I - POMDP^{Lite}$ FRAMEWORK

The $I - POMDP^{Lite}$ framework should be considered as a state-of-the-art approach in individual planning in MAS. [3] The framework improves over its predecessor $I-POMDP$ by introducing a nested reasoning approach to model the other agents having full observability in the environment. Though the introduced approach is an approximation of the actual behavior of the other agents, the framework performs as effective as its predecessor with reduced computational costs.

$I-POMDP^{Lite}$ for a planning agent i at a reasoning level l can be mathematically be represented as follows:

$$I-POMDP^{Lite}_{i,l} \triangleq \langle Ag, S, A, T_i, \Omega_i, O_i, R_i, \gamma, b_{i,0}, \{\mathcal{M}_{j,l-1}, \mathcal{M}_{k,l-1}, \ldots, \mathcal{M}_{z,l-1}\}\rangle \quad (2.4)$$

where,

- Ag is the set of agents, inclusive of the subject agent i and its neighboring agents $j, k, .., z$. Here, an agent can be of any frame from the set of discrete frames in the environment, $\Theta = \{\theta_1, .., \theta_{|\Theta|}\}$ that represents its capabilities and type of reasoning process. For example, protesters at a site might have different frames of peaceful and violent protesters based on their behavior, while violent protesters can cause far more damage to properties than peaceful protesters.
- S is the physical state space of the environment, which can be represented as $S = F_1 \times F_2 \times \ldots \times F_k$ with the state factors;
- A: $A_i \times A_j \times \ldots \times A_z$ is the set of possible joint actions performed by all the agents (Ag) in an environment. Frequently, notation A_{-i} is used to indicate the joint action performed by the other agents' for the subject agent i;
- T_i: $S \times A \times S' : [0, 1]$ is the transition function for agent i which provides a distribution over the next state (S') given the joint action of all the agents (A) and the current state (S);
- Ω_i is the set of observations received by the agent i;
- $O_i : S \times A \times \Omega_i : [0, 1]$ is a stochastic observation function for agent i, which provides the probability with which the agent receives an observation (o) given the resulting state (S') and the joint action (A);
- $R : S \times a_i \times a_{-i} \rightarrow \mathbb{R}$ is the reward function of agent i that rewards (or penalizes) the agent for its various actions dependent on the current state (S), the current agent's action (a_i) and other agents' joint action (a_{-i});

- $\gamma \in (0, 1]$ is a discounting factor used for weighting the reward values over a range of future time steps;
- $b_{i,0}$ represents the initial belief state of subject agent i;
- $\{\mathcal{M}_{j,l-1}, \mathcal{M}_{k,l-1}, \ldots, \mathcal{M}_{z,l-1}\}$ is the set of mental models for the other agents $(Ag - \{i\})$ at the reasoning level $l-1$. Here, the planning agent's reasoning level (l) represents that all the neighboring agents reason at level $l-1$ or lower. In a special case, the planning agent reasoning at level 1 implies that all the neighbors choose actions randomly (or are at level 0) or from a chosen distribution. Each mental model $\mathcal{M}_{j,l-1}$ is a $Nested - MDP$ model represented as:

$$\mathcal{M}_{i,l} \triangleq \langle S, A, T_i, R_i, \{\pi_{j,d}, \pi_{k,d}, \ldots, \pi_{z,d}\}_{d=0}^{l-1}, \gamma \rangle$$

where $\{\pi_{j,d}, \pi_{k,d}, \ldots, \pi_{z,d}\}_{d=0}^{l-1}$ is the set of policies of the other agents, obtained by solving their reasoning processes as $Nested - MDP$s.

The belief update in $I - POMDP^{Lite}$ for a next state (s') in the next belief state (b') can be represented as:

$$b_i'(s') = \beta O_i(o_i, s', a_i) \sum_{s \in S} b_i(s) T_i(s, a_i, a_{-i}, s') \tag{2.5}$$

where $\beta = 1/O(o_i, b_i, a_i, a_{-i})$ is a normalization constant. Above equation can be summarized for an entire belief as $b_i' = SE(b_i, a_i, a_{-i}, o)$.

To solve an $I - POMDP^{Lite}$, an agent chooses actions that maximize the cumulative, discounted reward function over a finite horizon H: $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^{H-1} r_{h-1}$, by considering a set of Value functions for each belief:

$$V_{i,l}^h(b_i) = \max_{a_i \in A_i} \left(\rho_i(b_i, a_i) + \gamma \sum_{s' \in S, o_i \in \Omega_i} \mathcal{T}_i^{a_i, o_i}(s', o_i | b_i, a_i) \times V_{i,l}^{h-1}(SE(b_i, a_i, a_{-i}, o)) \right) \tag{2.6}$$

where, $\rho_i(b_i, a_i) = \sum_{s \in S} \sum_{a_{-i} \in A_{-i}} \prod_{-i \in \{j,k,\ldots,z\}} \pi_{-i,l-1}(s, a_{-i}) R_i(s, a_i, a_{-i}) \times b_i(s)$, and policies $\pi_{-i,l-1}(s, a_{-i})$, $-i \in \{j, k, \ldots, z\}$ are solutions of the other agents' mental models $\mathcal{M}_{-i,l-1}$.

The $I-POMDP^{Lite}$ provides a theoretical assurance of near optimal solution for individual planning in MAS, which makes it an attractive choice for individual planning applications.

2.3 Introducing Agent Anonymity in $I-POMDP^{Lite}$

As described in the previous section, the $I-POMDP^{Lite}$ provides a theoretical way for individual planning in MAS in the light of the actual state and the prediction of the reasoning of the other agents, however as the agent population grows larger, the planning becomes computationally infeasible in such an identity-preserving framework. In many agent settings, instead of focusing on individual agents, focusing on a group of agents sharing similar traits, would provide huge computational benefits, and this property is termed as frame-action anonymity. [10] The individual planning under anonymity provides an identity-independent framework to group the agents by frames and the actions performed by them under these frames in an environment involving many agents having different types or frames (θ).

Theoretically, the frame-action anonymity property in a many-agent environment is:

$$C = \left\langle c_{a_1,\theta_1}, \ldots, c_{a_{|A|},\theta_1}, c_{a_1,\theta_2}, \ldots, c_{a_{|A|},\theta_{|\Theta|}} \right\rangle \tag{2.7}$$

where,

- C is a frame-action configuration or modestly a configuration;
- c_{a_i,θ_j} represents the count of agents having the type θ_j and performing the action a_i.

Note that, the dynamic parameters of the environment T, O, and R depends only on these counts c in an identity independent manner, such that two different joint actions (A) producing the same frame-action configuration (C) would also produce the same transitions and rewards. For example, in the wildfire environment, agents of a different type (e.g., ground firefighters, helicopters, and airplanes) extinguish fires at different rates, while their location would allow them to fight only certain surrounding fires. Such type and action properties can create different frames for firefighting agents, yet each firefighting agent in the same

frame contributes the same amount of fire suppression when fighting a particular fire. This implies that the representation of T, O, and R can be greatly compacted since the number of possible configurations $\binom{|N|+\nu+1}{\nu+1}$ is much fewer than the number of possible joint actions $\nu^{|N|}$, where $\nu = \max\{|A_i|, |A_j|, \ldots, |A_z|\}$ and N is the number of agents in the environment.

To incorporate agent anonymity in the $I - POMDP^{Lite}$, the joint actions of the other agents can be converted into configurations of the other agent's actions ($\mathcal{C}_{-i}$). Therefore, while obtaining the other agents' policies in the $Nested - MDP$, the framework can use $\mathcal{C}_{-i}$ to consider other agents' actions and reduce the planning time significantly. For notational convinience, $\mathcal{C}_{-i}$ can be replaced with just $\mathcal{C}$. After incorporating the configuration changes, the value function for the $Nested - MDP$ can be defined as:

$$V(s) = \max_{a_i \in A_i} \sum_{C(a_i) \in \mathcal{C}(a_i)} \left(P(C(a_i)) \left(R(s, C(a_i)) + \gamma \sum_{s' \in S} T(s, C(a_i), s') V(s') \right) \right) \tag{2.8}$$

where,

- $\mathcal{C}(a_i)$ is the set of all possible configurations where agent i chooses action a_i;
- $C(a_i)$ is an arbitrary configuration where agent i chooses action a_i.
- $P(C(a_i))$ is the probability of a configuration $C(a_i)$, which can be calculated using multinomial distribution, where the counts of each frame-action pair can serve as its parameters.

Exploiting the idea of the frame-action anonymity further in many agent settings, the agents with the same frame, sharing the same action-set and having the same environmental constraints could be modeled as agents having the same reasoning process. Creating just one $Nested - MDP$ model for all such agents could reduce the number of $Nested - MDP$ models and computational cost drastically. For example, 30 ground firefighters in the wildfire domain at the same location, surrounded by the same fires could follow the similar fire fighting policy if modeled by the $Nested - MDP$ framework. Therefore, creating just one $Nested - MDP$

model for a ground firefighter from this group would be sufficient to represent all the ground firefighters. Interestingly, even though the agents would follow the same $Nested - MDP$ policies in such a group, they would have a unique internal state at each time-step, and therefore, they are still separate agents. For example, each ground firefighter in this group of 30 has a unique suppressant level, and each would follow its own trajectory over the time based on its individual suppressant transition.

2.4 Filtering Configurations in $Nested - MDP$

This section focuses on filtering the number of configurations in the $Nested - MDP$ framework after incorporating the agent anonymity as illustrated in equation (2.8). The part in the inner paranthesis in equation (2.8) is a geometric series [23], and given the maximum reward (R_{max}),the reward upper bound ($\bar{R}$) can be defined as:

$$\bar{R} = R_{max}/(1-\gamma) \tag{2.9}$$

The most that an arbitrary configuration $C(a_i)$ can contribute to V in equation (2.8) is $P(C(a_i)) * \bar{R}$. Since $P(C(a_i))$ is close to 0 for many configurations, the maximum possible contribution $P(C(a_i)) * \bar{R}$ will be trivial for them. Leaving out such $C(a_i)$'s will have minimal impact on the overall estimate of V.

Let $\epsilon \in \mathbb{R}^+$ be a small constant value, then the set of filtered configuration $\hat{\mathcal{C}}(a_i)$ can be defined as:

$$\hat{\mathcal{C}}(a_i) = \left\{ C(a_i) \in \mathcal{C}(a_i) | P(C(a_i)) * \bar{R} >= \epsilon \right\} \tag{2.10}$$

The $\hat{\mathcal{C}}(a_i)$ is the subset of possible configurations where agent i chooses action a_i such that each configuration contributes at least ϵ to the estimate of V.Changing the equation (2.8) to use $\hat{\mathcal{C}}(a_i)$ will be a close approximation of the true V with a significant reduction in the computation time. Choosing a smaller ϵ will increase the size of $\hat{\mathcal{C}}(a_i)$, yielding a closer approximation to V in equation (2.8), however with a drawback of higher computation time.

Appendix A provides a chart that plots the number of configurations in $\hat{\mathcal{C}}(a_i)$ after filtering the configurations against different ϵ values using wildfire domain setups.

2.5 Monte Carlo tree for many agents

Monte Carlo tree search ($MCTS$) is a popular online decision planning framework for solving MASs. The framework uses the MC sampling to approximate the actual solution and saves valuable time compared to computationally expensive offline planning methods, while it scales well with state-action-reward trajectories. In the single-agent $MCTS$ planning, the planning agent iteratively constructs an AND-OR tree representing the state - action trajectories up to a predetermined maximum level and computes expected discounted utility for each action. Of note, in the $MCTS$ tree, the AND nodes represent the selection or environmental output, and the OR nodes represent the sampling process.

In the case of the fully observable $MCTS$ planning, the root OR nodes represent a physical state, while action AND nodes represent possible actions an agent can take. The edge between the state nodes and the action nodes are annotated by the utility of the action and the number of visits to those nodes. The action nodes further lead to the next possible states after the transition. With each pass through the tree, the environment is simulated until the maximum horizon, and the utility of each action taken at each horizon is calculated in a bottom-up manner, such that the action nodes connected to the root node receives a cumulative discounted utility of all the action nodes in the sampled trajectory. It is important for an $MCTS$ tree to have enough explored trajectories for choosing the possible best action and to ensure a near-optimal solution.

On the other hand, in $MCTS$ planning for partially observable environments, known as partially observable MC planning ($POMCP$), the root OR node is a representation of a belief state (b_ϕ) in the form of a set of state particles. Similar to the root node, all the other OR nodes are belief nodes (b_h) in the form of a set of state particles and represent the updated belief state at each level in the tree after the transition. An edge between the belief

OR node and the action AND node in $POMCP$ is represented by the estimated Q-value ($Q(b_h, a_h)$) and the number of visits to that action node ($N(b_h, a_h)$) in the context of belief. For sampling an action from a belief node, the POMCP uses UCT, a popular bandit-based heuristic, which allows fair exploration of all the action nodes with a fair selection of the bandit constant (c). From an action AND node, an observation (o_{h+1}) leads to the next belief OR node (b_{h+1}), and the edge between an action AND node and a belief OR node is annotated with the probability of receiving this observation (o_{h+1}) given by the observation function $O(b_{h+1}, a_h, o_{h+1})$. After each pass, the maximum of one belief node is added to the tree and the annotated Q-values and N-values of the visited trajectories are revised. The algorithm continues execution until the time budget runs out and subsequently it selects the action having the optimal Q-value from the root node.

In this research, the AND-OR tree used in $POMCP$ is augmented to suit the requirements of many agents settings in the context of $I - POMDP^{Lite}$ framework updated with agent anonymity changes. The newly created many-agent tree adds a layer of frame-action configurations between the layers of belief nodes and action nodes, representing incorporation of other agents' actions. Contrary to the $POMCP$ tree, belief nodes in the many-agent tree are AND nodes instead of OR nodes, as the configurations are results of action-selection from the other agents' policies. In general, the many-agent tree contains the belief AND nodes, configuration OR nodes and the action AND nodes, and as the tree trajectories contain two consecutive AND nodes, action and belief nodes, followed by a configuration OR node, it can be said a 2AND-OR tree.

Figure 2.1 illustrates the many-agent 2AND-OR tree with a representation of trajectory paths (p) in the left; for notational convenience, the visit counts at action nodes (N) use the trajectory paths. In the many-agent MCTS tree, all the configuration nodes are fully connected to their immediate action nodes, representing the planning agent's action selection over all possible actions of the other agents, and this context is represented in all notations with a generalization of the configuration nodes in a layer as C. The double line in the

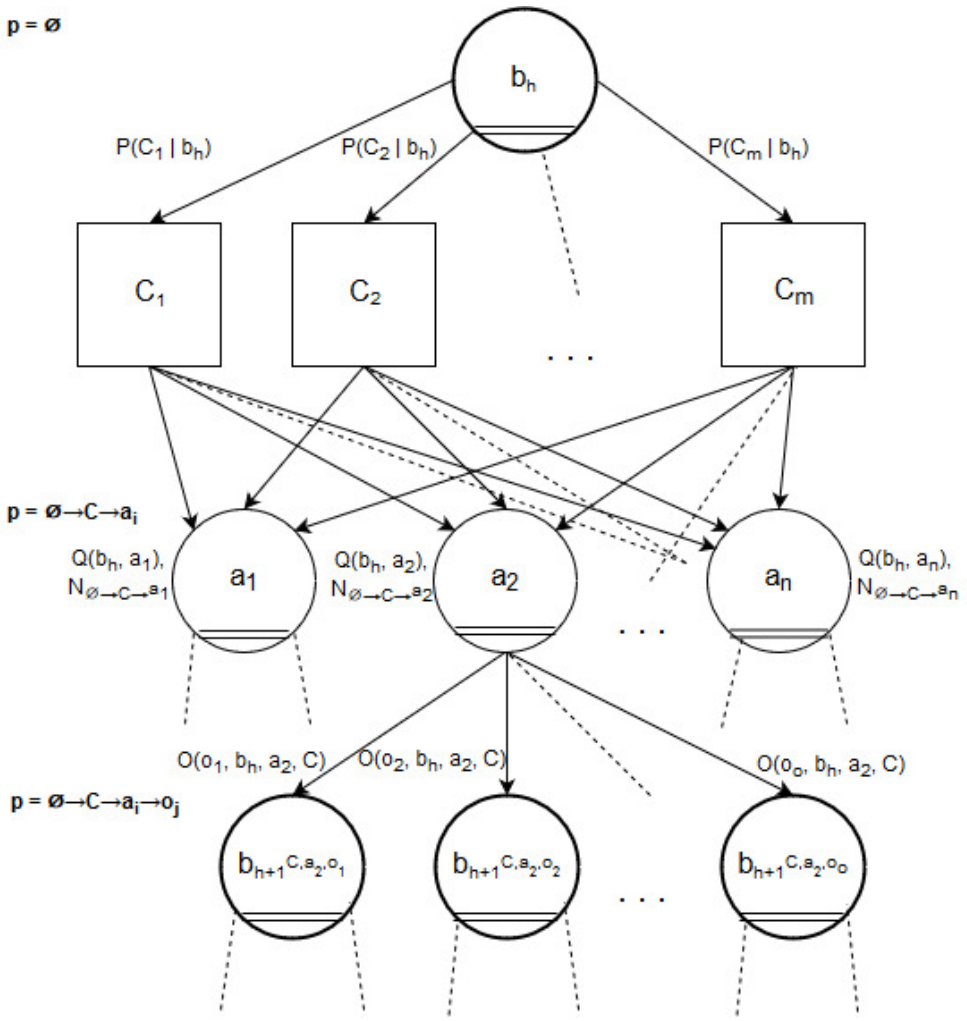


Figure 2.1: The Monte Carlo tree for many agents setting consists of an additional configuration layer compared to the traditional MC tree. All the configuration nodes are fully connected to the action nodes in the tree.

belief nodes and the action nodes represents the probability distribution over the respective configurations and observations.

In case, to augment the $POMCP$ tree for the $I - POMDP$ framework, the configuration nodes would be replaced by the other agents' joint actions nodes. Due to the enormous number of joint actions compared to configurations for the same number of agents, the resulting tree would produce sparse trajectories and due to that, it would result into

worse performance than the tree with configurations in the same time bound in online planning. Next chapter, Methodology, explains the algorithm to create this many-agent tree and improve its scalability using sampling and extrapolation strategies.

Chapter 3

Methodology

This chapter drafts the central methodology of this research, created from the components mentioned in the Background chapter. The chapter begins with explaining the interactive - partially observable MC planning ($I - POMCP$) algorithm, the primary algorithm of this research to build 2AND-OR trees, and subsequently, it advances to explain the handling of agent openness in the context of the algorithm. Later in the chapter, the sampling process of other agents and extrapolation of the behavior of sampled agents to the entire population are described.

3.1 $I - POMCP$ Algorithm

The Algorithm-1 exhibits the primary $I - POMCP$ algorithm of this research for many-agent MAS. In the algorithm, the IPOMCP procedure iteratively construct the 2AND-OR tree (T) using the UpdateTree procedure until the time budget (τ) runs out. In each of these iterations, a particle is sampled from the initial belief (b_ϕ) as illustrated in line 4 and is propagated in the tree up to the maximum horizon (H). As already mentioned, with each iteration, the maximum of one node is added to the tree (T). After the time budget (τ) runs out, the algorithm returns the action having the optimal Q-value at the root node as illustrated in line 7.

Each time a particle passes through the UpdateTree procedure, it is being added to the belief node b_p at particle's trajectory p as illustrated in line 11. If a trajectory (p) is already a part of the tree (T), then first, a configuration is sampled using the mental models and $Nested - MDP$ policies of the other agents, and an action is being chosen using UCT

Algorithm 1 $I-POMCP$ algorithm

Note: T is the tree (initially empty), p is a path from the root of the tree (with $p=\emptyset$ is the root of the tree), b_p is the set of particles representing a belief at the node at p in the tree and $b_\emptyset$ indicates initial belief, N is the number of visits to each node initialized to some constant $\nu \geq 0$, Q is the Q function initialized to R value, c is the bandit constant, and τ is the time budget.

procedure $IPOMCP(b_\emptyset, \tau)$
  $time \leftarrow 0$
  while $time < \tau$ **do**
    $s^0, M^0 \leftarrow SampleParticle\,(b_\emptyset)$ ▷ Sample a particle from the initial belief
    $UpdateTree\left(s^0, M^0, 0, \emptyset\right)$
    Increment $time$
  return $\underset{a \in A_i}{\text{argmax}}\, Q(\emptyset, a)$
procedure UPDATETREE(s^h, M^h, h, p)
  if $h >= H$ **then**
    return 0
  $b_p \leftarrow b_p \cup \{\left(s^h, M^h\right)\}$
  if $p \notin T$ **then**
    $T \leftarrow T + \text{leafnode}(p)$ ▷ Add explored trajectory at p to the tree
    return $Rollout\left(s^h, M^h, h\right)$
  else
    $C^h \leftarrow SampleConfiguration\left(s^h, M^h\right)$
    $a_i^h \leftarrow ChooseAction\left(p, C^h\right)$
    $\left(s^{h+1}, M^{h+1}, o_i^h, r_i^h\right) \leftarrow Simulate\left(s^h, M^h, a_i^h, C^h\right)$
    $N_p \leftarrow 1 + N_p, N_{p \mapsto C^h \mapsto a_i^h} \leftarrow 1 + N_{p \mapsto C^h \mapsto a_i^h}$
    $p' \leftarrow p + \left(a_i^h, C^h, o_i^h\right)$
    $R \leftarrow r_i^h + \gamma \cdot UpdateTree\left(s^{h+1}, M^{h+1}, h+1, p'\right)$
    $Q(p, a_i^h) \leftarrow Q(p, a_i^h) + \frac{R - Q(p, a_i^h)}{N_{p \mapsto C^h \mapsto a_i^h}}$
    return R
procedure ROLLOUT(s^h, M^h, h)
  $R \leftarrow 0, h' \leftarrow h$
  while $h < H$ **do**
    $C^h \leftarrow SampleConfiguration(s^h, M^h)$
    $a_i^h \leftarrow SampleAction(A_i)$ ▷ Sample an action from the chosen distribution
    $(s^{h+1}, M^{h+1}, o_i^h, r_i^h) \leftarrow Simulate(s^h, M^h, a_i^h, C^h)$
    $R \leftarrow R + \gamma^{h-h'} \cdot r_i^h,\ h \leftarrow h+1$
  return R
procedure CHOOSEACTION(p, C^h)
  return $\underset{a \in A_i}{\text{argmax}}\, Q\,(p, a) + c\sqrt{\frac{\log N_p}{N_{p \mapsto C^h \mapsto a_i^h}}}$ ▷ UCT Heuristic

heuristic as illustrated in lines 16 and 17. The ChooseAction procedure provides the UCT heuristic for the action selection. Second, the N-values are being updated after the simulation as illustrated in line 19. Finally, the Q-values are updated in a recursive manner representing a bottom-up update as illustrated in lines 20-22. On the other hand, if the trajectory p is not part of the tree (T), then the particle is rolled out using the Rollout procedure up to the maximum horizon (H). The Rollout procedure simulates the environment using a sampled configuration as well as a sampled action, and same as the UpdateTree procedure, it calculates R values in a bottom-up manner, as illustrated in lines 26-30.

Compared to its single agent predecessor $POMCP$, the $I-POMCP$ algorithm illustrates some fundamental differences in characteristics to incorporate other agents' reasoning. First, a particle in $I - POMCP$ includes the mental models (M^h) of the other agents, apart from the environmental state (s^h). Effectively, while sampling, the $I - POMCP$ algorithm not only samples s^h, however M^h of the other agents as well. Here, a mental model (M^h) is an approximation of the hidden factor of a neighboring agent's state. For example, the suppressant level of the neighboring agent in the wildfire domain, which is hidden from the planning agent. Tracking mental models of the other agents helps the subject agent to predict the other agents' actions accurately and provides a correct estimate of the likelihood of sampling configurations (C^h) before the agent chooses its own action (a^h).

Second, the Q-value at each action node is an approximate result of the planning agent's action over possible frame-action configurations of the other agent's actions (C^h), which are sampled from their precalculated $Nested - MDP$ policies. In an ideal case, the Q-value at an action node representing an action (a_i^h) under a belief node (b_p) is a weighted sum of all Q-values for the action (a_i^h) and an arbitrary configuration (C) given b_p. Mathematically, this idea can be represented as:

$$Q(b_p, a_i^h) = \sum_{C^h \in C_{b_p}^h} P(C^h|b_p) * Q(b_p, C^h, a_i^h) \tag{3.1}$$

Finally, the belief update in $I-POMCP$ algorithm involves the mental-model particles, in addition to the state particles to create the new belief state as illustrated in Algorithm-2. After performing the actual action a_r, the planning agent receives an actual observation o_r from the environment. To update the belief, the agent collects the previous tree particles from the actual next belief node at the $\emptyset \mapsto C \mapsto a_r \mapsto o_r$ trajectory, as illustrated in line 2 of BeliefUpdate procedure. The new belief state (b_{new}) for the next time-step is created by sampling maximum allowed (max_p) particles from these collected particles (b_{next}), as illustrated in lines 3-6. In this research, the wildfire domain is designed to have a Mixed Observability MDP ($MOMDP$) [24] state, where the actual state particle consists of fire-intensity and planning agent's suppressant level is fully-observable and is provided by the environment at each time-step, while the mental models of the other agents are needed to be maintained as a belief. For this specific case, the belief update method receives the next state (s') as a parameter, and line 5 in the Algorithm-2 can be updated to represent these changes as follows: $s_{new}, M_{new} \leftarrow s', SampleParticle(b_{next}) \mapsto M$.

Algorithm 2 Belief Update in $I-POMCP$

Note: a_r is the real action performed by the agent, o_r is the real observation received from the environment, T is the current $I-POMCP$ tree for the agent, and max_p is the maximum number of particle allowed in the root belief node.

1: **procedure** BELIEFUPDATE(a_r, o_r, T, max_p)
2: $b_{next} \leftarrow getParticles(T, \emptyset \mapsto C \mapsto a_r \mapsto o_r)$ ▷ Receive particles from actual next belief node
3: $b_{new} \leftarrow \emptyset$
4: **while** $sizeOf(b_{new}) < max_p$ **do** ▷ Create a belief state with maximum particles
5: $s_{new}, M_{new} \leftarrow SampleParticle(b_{next})$
6: $b_{new} \leftarrow b_{new} + (s_{new}, M_{new})$
7: return b_{new}

3.2 Handling Agent Openness

As already mentioned in the Introduction chapter, in this research, the definition of agent openness is being restricted to the environments, where agents may leave or re-enter anytime, yet no new agent joins during the operation. Keeping track of the agents in a multiagent

system is important to predict the actions of the other agents. For example, in the wildfire domain keeping track of the suppressants of the neighboring agents in the system would guide the agent in predicting the number of agents who are out of the environment refilling, which would, in turn, improve subjective agent's decision-making accuracy.

A naive way of tracking the presence of neighboring agents in open environments involves each agent i maintaining an additional state variable P_j for each other agent $j \in N(i)$, where $P_j \in \{\text{present}, \text{absent}\}$ and $N(i)$ is the set of neighboring agents of the planning agent i. The resulting state space increases by at least a factor of $2^{|N(i)|}$ due to the addition of P_j, which is an exponential increase in the size of the agent's state space. For example, modeling 40 neighbors' presence increases the state space by a factor of at least $2^{40} \gg 100$ billion states. To mitigate the intractability, the P_j variable can be placed as an internal state variable within a mental model $M_{j,l-1}$ attributed by agent i to a neighboring agent j. Consequently, the overall increase in the size of the problem model is restricted to being linear in the number of agents, which promotes scalability. For example, in the wildfire domain, instead of keeping track of each neighboring agent's presence as a state factor, an agent's suppressant level 'Empty' in the mental model would represent its absence, and its presence otherwise.

However, even a linear increase in the size of the problem still poses an important challenge for $I-POMCP$. In both the UpdateTree and Rollout procedures of Algorithm 1, the Simulate step must simulate transitions for each P_j variable as part of calculating the next set of mental models M^{h+1} based on M^h, s^h, and a^h. An increase in the time complexity of sampling a path through the tree directly reduces the number of paths that can be sampled within a fixed amount of time budget τ. Increasing the number of visits to a sampled configuration would increase its accuracy and would provide assurance of reliable Q-values for each action.

3.3 Selectively Modeling Neighbors

To resolve the issue of maintaining the transition of many mental models, this research proposes a sampling-based solution, which would approximate the behavior of all neighboring agents in the system by tracking only some of them. In the field of surveying, the measurers choose a small set of people to represent the opinion of the entire population. For example, in the election exit polls, the surveyors choose several thousand voters randomly to conduct their surveys, while they maintain that the sample contains representation from each section of the society. Ordinarily, this sample-based analysis is extrapolated to the entire population, which approximately represents the opinion of the population. By following the standard statistical procedures, some reliable theoretical bounds can be achieved. [25, 26, 27]

Adapting this sampling approach to the $I-POMCP$ makes the algorithm run faster, as the agent maintains a small subset of mental models of its neighbors in the particle filter instead of maintaining all of them. For example, in the wildfire domain, each firefighter maintains only a subset of its neighboring agents' suppressant levels and track them over time, in-turn extrapolating the behavior of this small set of agents to the entire neighborhood. This kind of design also provides the solution to agent openness by extrapolation of the recharging agents from the sampled agents to the entire population. Also, the sampling action is not computationally costly, as it is performed just once at the beginning of each experimental trial.

3.3.1 Sampling Configurations

Refering back the equation 2.7, the configuration (C) can be considered as a random variable composed of counts of agents performing the same action under the same frame, and which implies that the distribution over C is a multinomial distribution . Mathematically, the distribution over a configuration random variable C can be parameterized as:

$$P(C|s^h, M^h) = Multi(|N(i)|, \{p_{a_1,\theta_1,N(i)}, \ldots, p_{a_{|A|},\theta_{|\Theta|},N(i)}\}) \tag{3.2}$$

where,

- $N(i)$ is agent i's neighbouring agents;
- $p_{a,\theta,N(i)}$ is the likelihood that neighbors in $N(i)$ of type θ will perform action a.

The configuration (C) can be considered as a concatenation of several multinomial variables of each type ($\theta \in \Theta$), and a configuration for a type (C_θ) can be defined as a tuple of the counts of agents of the same type performing a range of actions. Mathematically, the C_θ is:

$$C_\theta = \left\langle n_{a_1,\theta}, n_{a_2,\theta}, \ldots, n_{a_{|A|},\theta} \right\rangle \tag{3.3}$$

Using the equations 3.2 and 3.3, the probability of a configuration of a type (C_θ) can be defined using the multinomial distribution as:

$$P(C_\theta | s^h, M^h) = Multi(|N_\theta(i)|, \{p_{a_1,\theta,N_\theta(i)}, \ldots, p_{a_{|A|},\theta,N_\theta(i)}\}) \tag{3.4}$$

where

- $N_\theta(i) \subseteq N(i)$ is the subset of agent i's neighbors each with type θ.

If an agent chooses to model a subset of its neighbors, $\hat{N}(i) \subset N(i)$, it can still estimate $p_{a,\theta,N_\theta(i)}$ for various actions that parameterize the multinomial distribution $P(C_\theta | s^h, M^h)$ for each type. Let $\hat{N}_\theta(i)$ be the subset of $\hat{N}(i)$ with type θ and $\hat{n}_{a,\theta,\hat{N}_\theta(i)}$ be the number of agents in $\hat{N}_\theta(i)$ predicted to perform action a based on the current state s^h, their presence $P_j \in m_{j,l-1}$ and their policies derived from their $Nested - MDP$ models. Subsequently the likelihood that an arbitrary modeled neighbor of type θ will perform action a can be given as:

$$\hat{p}_{a,\theta,\hat{N}_\theta(i)} = \frac{\hat{n}_{a,\theta,\hat{N}_\theta(i)}}{|\hat{N}_\theta(i)|} \tag{3.5}$$

Using the estimated proportion in the equation 3.5, the SampleConfiguration function is established as given in the Algorithm 3. The SampleConfiguration function is called from two different procedures UpdateTree and Rollout in the Algorithm 1 to sample the configurations. This procedure samples a full configuration C for all agents in a neighborhood using

Algorithm 3 Sampling Configurations from Modeling $\hat{N}(i)$

1: **function** SAMPLECONFIGURATION(s^h, M^h)
2: $C(a, \theta) \leftarrow 0 \; \forall a, \theta$
3: $\hat{n}_{a,\theta,\hat{N}_\theta(i)} \leftarrow 0 \; \forall a, \theta$
4: **for** $\mathcal{M}_{j,l-1} \in M^h$ **do**
5: $a \sim \pi_{j,l-1}(s^h)$
6: $\hat{n}_{a,\theta_j,\hat{N}_{\theta_j}(i)} \leftarrow \hat{n}_{a,\theta_j,\hat{N}_{\theta_j}(i)} + 1$
7: **for** $\theta \in \Theta$ **do**
8: **for** $a \in A$ **do**
9: $\hat{p}_{a,\theta,\hat{N}_\theta(i)} \leftarrow \frac{\hat{n}_{a,\theta,\hat{N}_\theta(i)}}{\sum_{a' \in A} \hat{n}_{a',\theta,\hat{N}_\theta(i)}}$
10: **for** $j \in N_\theta(i)$ **do**
11: $a \sim Cat(\hat{p}_{a_1,\theta,\hat{N}_\theta}, \hat{p}_{a_2,\theta,\hat{N}_\theta}, \ldots, \hat{p}_{a_{|A|},\theta,\hat{N}_\theta})$
12: $C(a, \theta) \leftarrow C(a, \theta) + 1$
13: return C

only the mental models maintained for a subset of the neighborhood. First, all the values of $\hat{n}_{a,\theta,N_\theta(i)}$ from the modeled neighbor's mental models $M_{j,l-1} \in M^h$, including their policies $\pi_{j,l-1}$ are calculated, on lines 3-6. Subsequently these counts are used to calculate the likelihoods $\hat{p}_{a,\theta,N_\theta(i)}$ using the equation 3.5, on lines 8-9. Finally, the calculated likelihoods are used to sample the actions for every neighbor (modeled or otherwise) and add it to the configuration counts, on lines 10-12. Note that here $Cat(\hat{p}_{a_1,\theta,\hat{N}_\theta}, \hat{p}_{a_2,\theta,\hat{N}_\theta}, \ldots, \hat{p}_{a_{|A|},\theta,\hat{N}_\theta})$ is a categorical distribution, where a single action is chosen from each of their likelihoods (just as actions can be chosen from stochastic policies, which are also categorical distributions).

3.3.2 Number of Agents to Sample

The error of the approximation of $P(C|s^h, M^h)$ using the likelihoods $\hat{p}_{a,\theta,\hat{N}_\theta(i)}$ is calculated utilizing the equation 3.5, decreases as the number of neighbors that the agent models (i.e., $|\hat{N}(i)|$) increases, giving an agent control in balancing the trade-off between the quality of

planning and the size of its model by selecting the number of agents it chooses to model in $\hat{N}(i) \subset N(i)$.

Specifically, the approximation error of each $\hat{p}_{a,\theta,\hat{N}_\theta(i)}$ can be bounded as follows. For notational convenience and without loss of generality, let $\hat{p} = \hat{p}_{a,\theta,\hat{N}_\theta(i)}$ and $n = |\hat{N}_\theta(i)|$. With $1-\alpha$ confidence, it can be said that the true proportion of neighbors of type θ who will choose action a will lie within the range:

$$\hat{p} \pm t_{n-1,\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{3.6}$$

Hence with $1-\alpha$ confidence, the error $e_{\hat{p}}$ of $\hat{p}$ is:

$$e_{\hat{p}} \leq t_{n-1,\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \leq t_{n-1,\frac{\alpha}{2}} \sqrt{\frac{0.5^2}{n}} \tag{3.7}$$

because the range in equation 3.6 has maximal width whenever $\hat{p} = 0.5$. For reorganize this inequality, an appropriate number of neighbors is selected to model such that the agent's estimations of $\hat{p}$ are within some desired error bound:

$$n \geq \left(\frac{t_{n-1,\frac{\alpha}{2}}}{2e_{\hat{p}}} \right)^2 \tag{3.8}$$

However, the range given in equation 3.6 is rather loose as it makes the assumption of infinitely sized neighborhood $N_\theta(i)$. For smaller neighborhoods, this error bound can be tightened using the finite population correction. [26] Let $N = |N_\theta(i)|$ be the size of the finite neighborhood. Subsequently the error bound is given by:

$$\hat{p} \pm t_{n-1,\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \sqrt{\frac{N-n}{N-1}} \tag{3.9}$$

so that

$$e_{\hat{p}} \leq t_{n-1,\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \sqrt{\frac{N-n}{N-1}} \leq t_{n-1,\frac{\alpha}{2}} \sqrt{\frac{0.5^2}{n}} \sqrt{\frac{N-n}{N-1}} \tag{3.10}$$

and that provides

$$n \geq \frac{N \left(\frac{t_{n-1,\frac{\alpha}{2}}}{2e_{\hat{p}}} \right)^2}{N - 1 + \left(\frac{t_{n-1,\frac{\alpha}{2}}}{2e_{\hat{p}}} \right)^2} \tag{3.11}$$

.

For instance, consider an example environment where $N = 50$. To make sure the agents estimation $\hat{p}_{a,\theta,n}$ is within $e_{\hat{p}} = 0.1$ of the true proportion for the entire neighborhood (with 95% confidence), the number of agents needed to model are $n \geq 34$, whereas only $n \geq 18$ agents are needed to be modeled if $e_{\hat{p}} = 0.2$ is acceptable instead. Figure B.1 in Appendix - B illustrates the change in the value of n for different values of N with $e_{\hat{p}}$ and α.

3.3.3 Choosing the Neighbors to Model

Once the number of agents to model for each action a and agent type θ pair is determined, random sampling can be performed to ultimately determine which corresponding neighbors will be modeled by the agent to form $\hat{N}_\theta(i)$ for each frame. As previously mentioned, this process is executed only once when the agent joins the environment, and thus does not add much to the computational complexity of the $I - POMCP$ planning. Algorithm 4 provides a roadmap for how to sample the agents in the $I - POMCP$ algorithm, which incrementally constructs a set of neighbors *chosen*, so that the total number of modeled agents for each θ, a frame-action pair is equal to (or greater than) n, where $n = |\hat{N}_\theta(i)|$.

To understand the process of agent selection, there are several theoretical aspects needed to be understood. First, a subset of the subjective agent's neighborhood $N_\theta(i)$ based on what actions the neighbors can perform $N_{\theta,a}(i) = \{j \in N_\theta(i) | j \text{ can perform action } a\}$ provides a sample set for selecting neighbours in order to estimate $\hat{p}_{a,\theta,\hat{N}_\theta(i)}$, and construct $\hat{N}_\theta(i)$. Second, as the agents can perform more than one action, they can be part of several interaction subgraphs. [1] As it is important not to duplicate the selection of the agent, once an agent is chosen from one interaction subgraph, it would be considered as selected from all other subgraphs containing that agent. Duplicating the selection of agents would reduce the accuracy of the $I - POMCP$ as the algorithm would not be able to model enough agents to build a reliable prediction. *modeled* set in the Select function keeps track of the modeled agents, and provision on line 5 takes care of avoiding duplication.

Algorithm 4 Choose Neighbors for Modeling in $I-POMCP$

function ChooseNeighbors($I-POMDP_{i,l}^{Lite}$, $e_{\hat{p}}$, α)
  $subsets \leftarrow \{N_{\theta_1,a_1}(i), \ldots, N_{\theta_{|\Theta|},a_{|A|}}(i)\}$
  $sorted \leftarrow$ SortAscendingSize($subsets$)
  $chosen \leftarrow \emptyset$
  for $N_{\theta,a}(i) \in sorted$ **do**
    $N \leftarrow |N_{\theta,a}|$
    $n \leftarrow \lceil \frac{N \cdot \left(\frac{t_{n-1,1-\frac{\alpha}{2}}}{2*e_{\hat{p}}}\right)^2}{N-1+\left(\frac{t_{n-1,1-\frac{\alpha}{2}}}{2*e_{\hat{p}}}\right)^2} \rceil$
    $chosen \leftarrow chosen \cup$ Select($N_{\theta,a}(i), n, chosen$)
  return $chosen$

function Select($N_{\theta,a}(i), n, chosen$)
  $selected \leftarrow \emptyset$
  $modeled \leftarrow chosen \cap N_{\theta,a}(i)$
  while $|modeled| < n$ **do**
    $j \leftarrow$ RandomlyPick($N_{\theta,a}(i) \setminus modeled$)
    $modeled \leftarrow modeled \cup \{j\}$
    $selected \leftarrow selected \cup \{j\}$
  return $selected$

The agents who can perform more actions have more chances of being modeled in this algorithm because they are in the greatest number of subsets $N_{\theta,a}(i)$ of the subjective agent's neighborhood. This property is beneficial as these neighbors that can perform the most actions might be both (1) the most influential agents within the system, and (2) the most necessary to model, as their reasoning will be more complex than other agents, and thus are less predictable from the rest of the sample. Thus, the algorithm biases the selection process to pick such agents to best inform the estimates of the overall neighborhood behavior.

Above that, this algorithm also has the possibility of oversampling agents for some subsets $N_{\theta,a}(i)$. That is after n agents are selected from a particular $N_{\theta,a}(i)$, the remaining agents might later be selected from another $N_{\theta,a}(i)$ to which they belong. Thus, subsets $N_{\theta,a}(i)$ considered early in the algorithm might end up with more than the necessary n agents selected for modeling. This property is the motivation for starting with the interaction subgraphs

with the smallest numbers of agents in the algorithm because these are actually the most difficult sub-neighborhoods to model. The error in an agent's estimate is inversely proportional to n, hence the fewer neighbors that the agent models, the more inherent error in its estimates. Thus, each additional modeled neighbor provides more marginal benefit to the estimates in these small subsets than in the larger subsets. The algorithm implements this idea by sorting the subsets by the number of agents at line 3 in ChooseNeighbors.

3.4 Summary of methodology

As the methodology of this research is extensive and divided into many chunks, it is better to put it in a single chart to avoid any confusion. Figure 3.1 puts the entire methodology in a chart by providing the sequence of operations to utilize this individual planning framework. In the first step of generating $Nested-MDP$ policies, the agents who represent the entire population are selected based on frames and constraints. After selecting the ϵ value for filtering configurations, the $Nested-MDP$ policies are generated for each selected agent as mentioned in the Background chapter. The $Nested-MDP$ policies can be calculated prior to the actual online planning and could be shared with all the agents to reduce the online planning time and the use of computational resources. Next, for choosing the agents to model in $I-POMCP$ at the beginning of a trial, Algorithm 4 is employed. Subsequently, for each time-step in operation, the $I-POMCP$ framework is utilized as illustrated in Algorithm 1 given the initial belief state. On the other hand, at the end of each time-step, Algorithm 2 is used for belief update after observing the effect of joint-action of the agents on the environment.

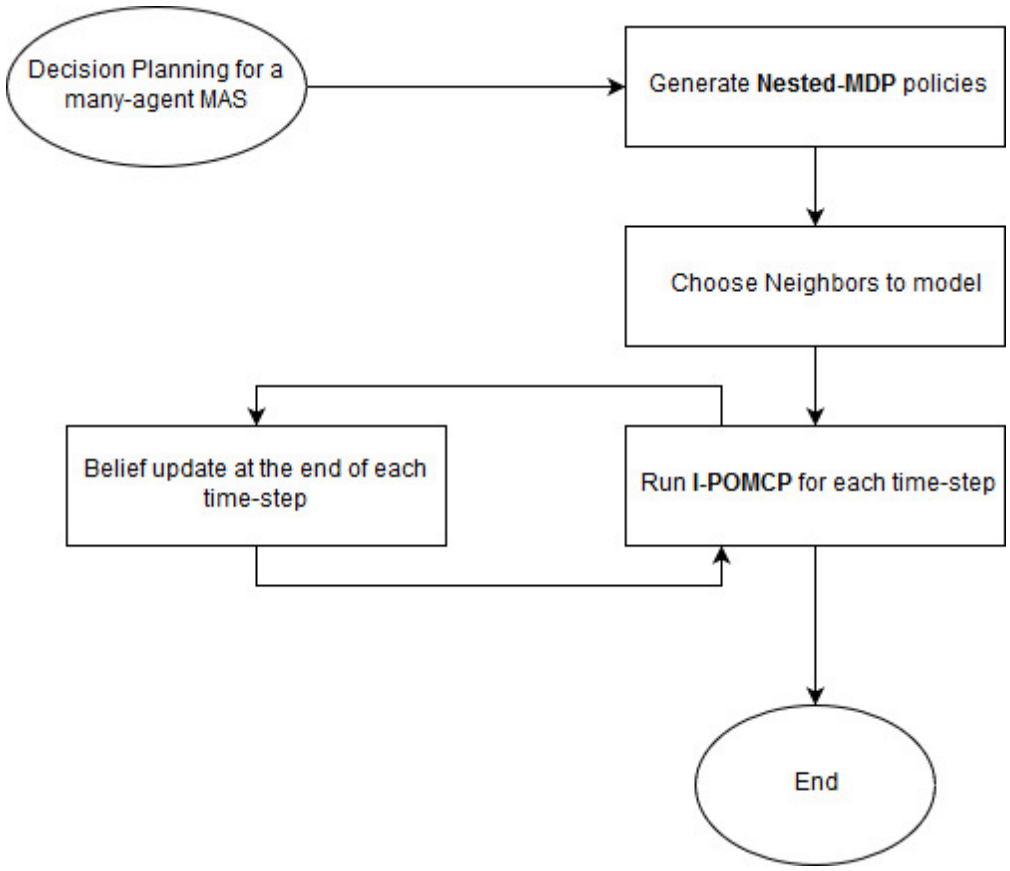


Figure 3.1: Methodology sequence in a flowchart.

Chapter 4

Experiments

Even though modeling other agents in a many-agent environment provides the acting agent a fair sense of the current state as a whole, it is also important to maintain a belief accurately over time to generate optimal actions. An empirical evaluation of the presented approach on multiple setups of different complexity levels is needed to demonstrate the algorithm's efficiency over time.

4.1 Experimental Setup

This section illustrates the empirical evaluation of the $I - POMCP$ algorithm using a realistic wildfire domain with many agents. The domain has been created after scaling the wildfire domain designed for a small group of agents in [7]. In the wildfire simulation, each agent in a setup obtains a joint reward of 40 for extinguishing a big fire and 20 for a small fire, while it obtains a penalty of 1 for each location that burns out. An agent receives an individual penalty of 100 for doing anything other than a no-operation action (NOOP) while recharging suppressants or trying to fight a nonexistent fire. Agents have three suppressant levels: empty and recharging, half full, and full with a stochastic transition between the levels. The intensity of a fire at a location ranges from 0 for an extinguished or no fire to 4 for a burnt out location, which is an irreducible intensity level; an extinguished location can catch fire if any of its neighborhood locations is on fire. Also, the extinguishing power of the agents must match a specific level to make an impact on the fire, while adding extinguishing power above that level would increase the chances of impact.

The performance of each methodology is assessed in four ways: (1) the average cumulative rewards obtained, (2) the average intensities of fires, (3) number of fires put out, and (4) the average suppressants used. The first evaluates the agent's ability to maximize rewards, the second and third evaluate the agents' approach to achieve the overall objective, and the final way evaluates the efficiency in achieving that objective. For a logical evaluation, the $I - POMCP$'s performance is compared with the performance of baseline methods that represent what would happen to the forest,

1. If no agents were present (called NOOP as this situation is equivalent to all agents always taking NOOP actions labeled as NOOP);

2. If each agent follows a heuristic (labeled as Heuristic) - fight an existing fire at random only if the agent has available suppressants, else take a NOOP, representing a scenario where agents do not plan on how or when to interact with their peers;

3. If each agent follows a level-1 $Nested - MDP$ policy, where the policy is generated by assuming that all the other agents' actions follow a uniform random distribution, representing a scenario where agents follow a calculated policy however do not maintain a belief over other agents' states.

To elicit different interaction between agents, seven setups are used, as illustrated in figure 4.1. [1] In Setup 1, 20 ground firefighters are given the responsibility of protecting the forest from four fires. In a 3 x 3 grid, A1 (a ground firefighter unit of ten agents) at one location has one big shared fire (F1) with A2 (a ground firefighter unit of ten agents) and one individual small fire (F0) to take care, while A2 has two individual small fires (F2, F3) apart from the shared fire with A1 (F1). In Setup 2, each A1 and A2 has two shared fires and one individual fire, which makes coordination more challenging. Setups 3-6 are the scalable versions of setups 1 and 2, scaling the firepower by 2 times in setups 3 and 4, and 3 times in setups 5 and 6 respectively; the setups are also scaled in terms of extinguishing power by

[1] Use of icons made by Twitter from www.flaticon.com is licensed by CC 3.0 BY.

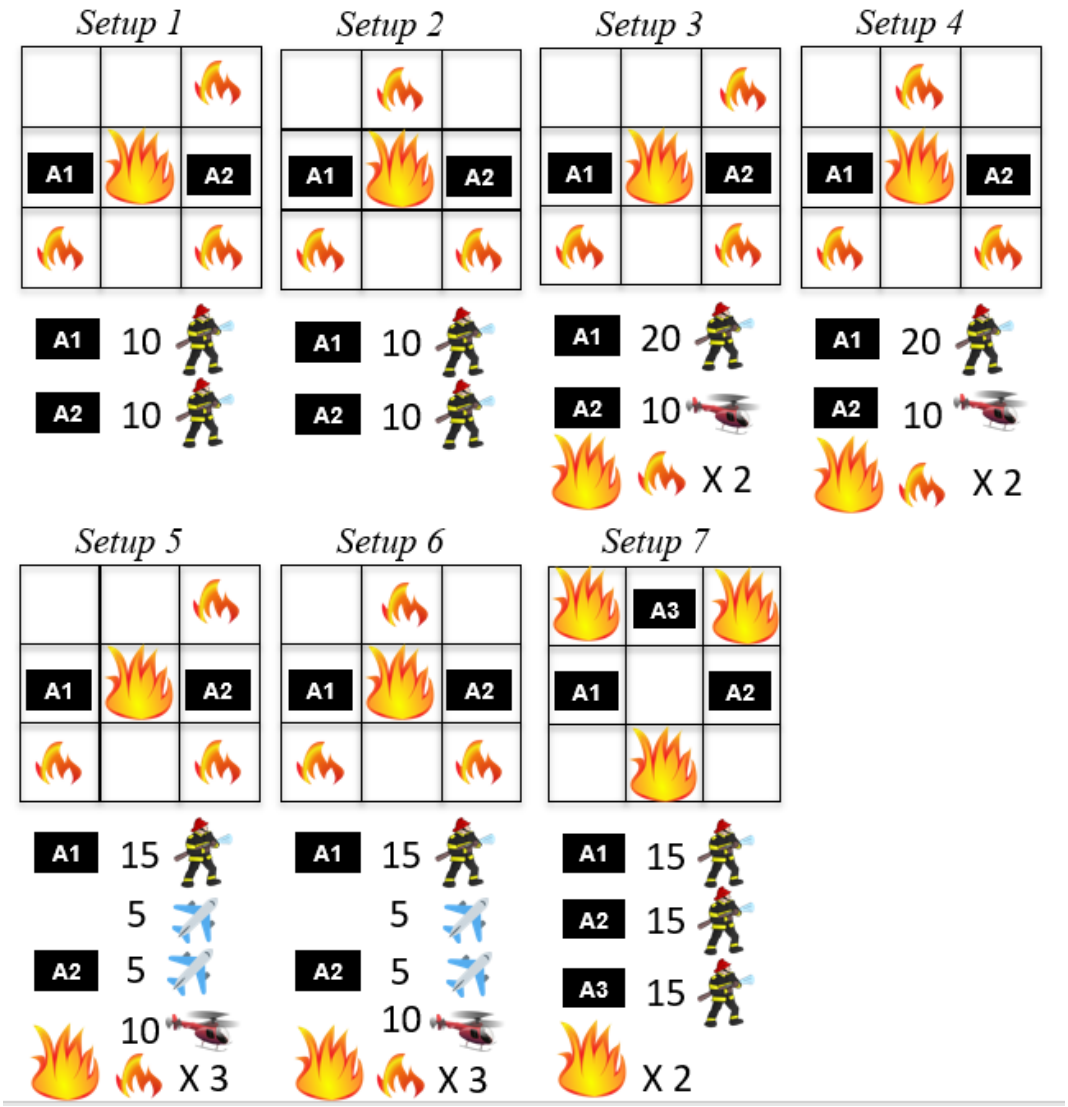


Figure 4.1: Experimental setups: The wildfire experimental setups test the capability of the new framework compared to standard baselines. Setup 1 has just one big shared fire, while the Setup 2 has one big and one small shared fires. Setups 3 and 5 are the scaled versions of Setup 1 while Setup 4 and 6 are the scaled version of Setup 2. Setup 7 is a special setup where all the agents share two fires.

having more agents of diverse types, such as helicopters and airplanes. Setup 7 illustrates the reasoning of the agents in a setup having 3 agent units, each sharing a fire with every other unit in the setup.

In the wildfire setups, an agent can only fight a fire that is immediately adjacent to its south, north, east, or west, or is diagonal to it. A ground firefighter has an extinguishing power of 0.1, while a helicopter has one of 0.2, and an airplane has one of 0.3. To reduce the intensity of a big fire by one with a probability of 0.75, the extinguishing power of the agents must sum up to 1.0. For example, at least ten ground firefighters are needed to reduce the intensity by one for a big fire. For the small fire, a minimum extinguishing power of 0.5 is needed to reduce its intensity. Each added extinguishing power of 0.1 would increase the chances of the fire intensity reduction with a probability value of 0.05 summing maximum up to 1.0. One more interesting constraint in intensity reduction is that a fire can not be reduced by more than one intensity level in a time-step, hence the use of excessive suppressant is wasteful. For the setups having firepower 2 times and 3 times, the number of minimum agents needed to extinguish those fires is multiplied by their fire-multipliers and the probability of the added extinguishing power is divided by the same factor, though the rewards do not change. For example, in Setup 3, at least 20 ground firefighters or 10 helicopters would be required to reduce the intensity of a big fire by 1 level, and the added extinguishing power of 0.1 would increase the fire reduction probability by 0.025.

All the setups here are designed to be over-constrained, that is the agents in any setup cannot reduce the intensity of all the fires in a single time-step. This setting helps to identify the cooperation between agents and their ability to earn rewards by keeping a balance between fighting the shared and the individual fires. As each agent makes its decision individually, and as a substantial number of agents are needed to reduce the intensity of any fire, each agent is required to perform a complex decision-making process for in-unit coordination as well as cross-unit coordination.

For each setup, 50 trials of 15 time-steps were conducted to assess the efficiency and effectiveness of each method. The methodology has been programmed in Java and BURLAP 3.0, a popular programming framework in the decision planning area. [28] Both the $I-POMCP$ and the $Nested-MDP$ were parameterized with a maximum horizon of 10. In

configuration filtering, ϵ values for generating $Nested-MDP$ policies were adjusted between 0.01 and 0.005, such that each planning agent received enough number of configurations to generate reliable policies in limited planning time. For example, ϵ value for a large setup such as Setup-6 was chosen as 0.005, which took approximately 20 hours of $Nested-MDP$ planning time to generate reliable policies on an advanced configured lab machine, on the other hand, ϵ value of 0.01 was enough for Setup-1 which took approximately 3 hours to generate policies.[2] Exploiting the anonymity assumption, the number of $Nested-MDP$ policies created for each setup was constrained by the firefighter type and location. For example, setup 5 had 4 policies, 1 policy for ground firefighters and 1 for airplanes in A1, and 1 policy for airplanes and 1 for helicopters in A2. The $I-POMCP$ experiments were conducted for different planning times (τ) of 1,5 and 10 seconds for a maximum allowed sampling error ($e_{\hat{p}}$) of 0 and 0.2. The alpha value was kept 0.05 for all the setups, and the bandit constant, c was chosen as 80 after empirical evaluation.

4.2 Results and Discussion

This section presents an empirical analysis of the experimental results conducted using the wildfire domain. Figures 4.2, 4.3, 4.4, and 4.5 provide the mean and standard errors of the values of rewards, fire intensities, the number of fires put out, and suppressants usage respectively for all the setups against used methods. Generating the $I-POMCP$ results over different values of sampling error ($e_{\hat{p}}$) and planning time (τ) has provided a deep insight into the behavior of the algorithm and has addressed the concern for how to use the algorithm in an efficient manner for a many-agent MAS.

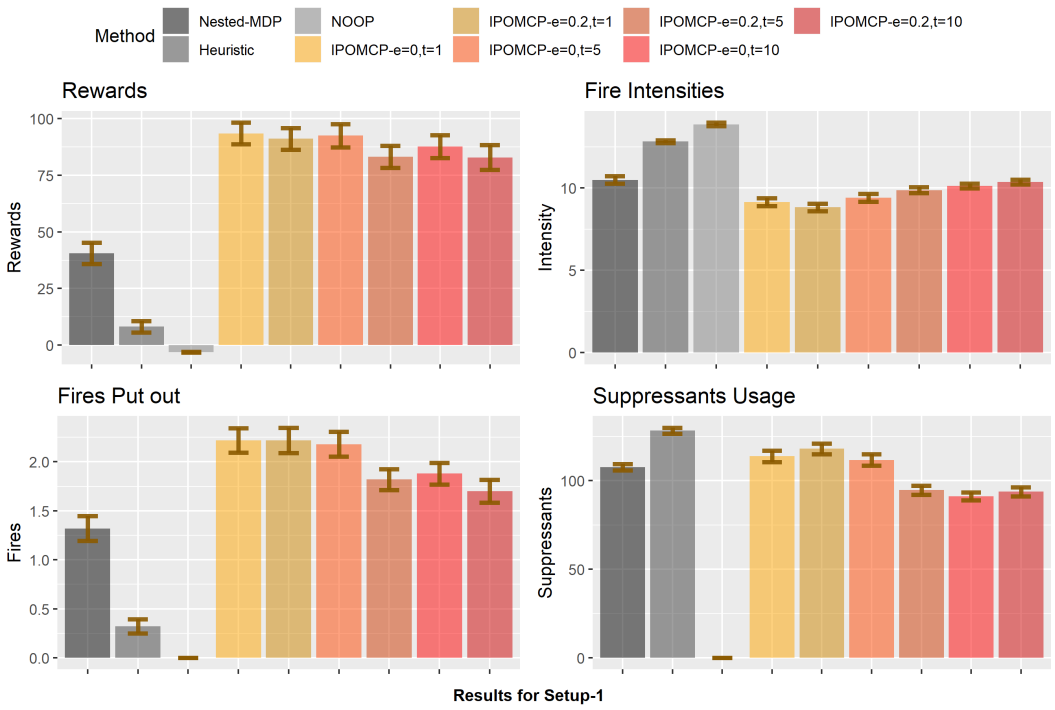
In the experiments, the NOOP baseline produced the worst performance benchmark for comparison with the other methods. The Heuristic approach can be considered as a standard way of coping with many agent MAS without any prior planning. Contrary to the human

[2]Planning time for each run may differ based on experimenting machine's computational capability.

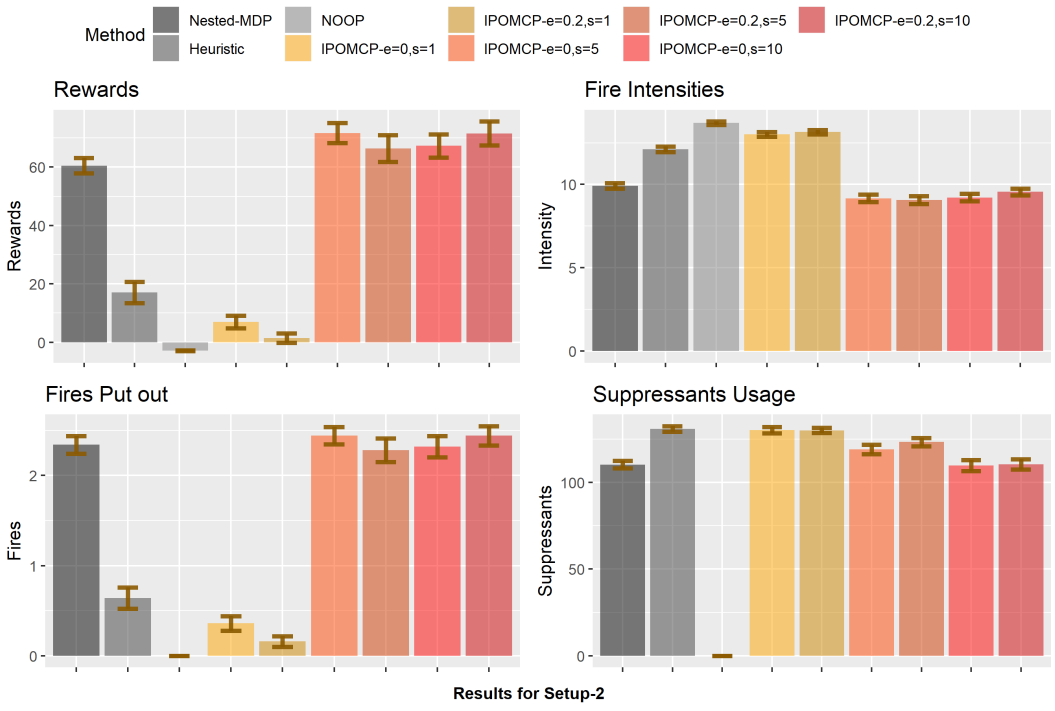
perception of the Heuristic method as an effective strategy, the method did not work well in many agent settings compared to any other planning approach. In the wildfire experiments, the Heuristic baseline manifested itself to be the least efficient strategy by spending the most number suppressant units while extinguishing only a few numbers of fires. These results provide a gist to how important decision planning can be in the real world applications.

In $Nested-MDP$ methodology, as all the agents, sharing the same frames, locations, and suppressants, followed the same level-1 policy, they were able to create virtual ad-hoc teams which followed the same instructions, though fluctuations were seen due to different internal states of the agents. Having such a formation allowed the $Nested-MDP$ agents not only concentrating on a single fire in a timestep yet eventually extinguishing multiple fires in a trial. For the $I-POMCP$, challenging such a powerful methodology was a tough task, however, it outperformed the $Nested-MDP$ in Setups 1,2,3,5, and 7, outperforming in a majority of criteria, while it produced statistically equivalent results to $Nested-MDP$ in Setups 4 and 6. The $Nested-MDP$ agents focused typically on the shared small fire in Setup 4 and 6, and as the fire reignited due to the active neighboring fires, the agents were able to extinguish it multiple times and to grow the result bars significantly. Observing the results, the use of suppressants seems directly proportional to the number of fires put out in any baseline and thus does not provide a comprehensive comparison.

In Setup 1 and 2, the $I-POMCP$ outperformed the $Nested-MDP$ with even 1 second of planning time (τ). It implies that in the small setups, the $I-POMCP$ does not require enormous planning time to match the $Nested-MDP$ performance. With the scaling in the number of agents and types, the planning time became larger for the $I-POMCP$ to outperform $Nested-MDP$ or even to produce comparable results. One more interesting phenomenon seen in $I-POMCP$ was, in general, with the increase in planning time to 10 seconds, the suppressants use declined, and agents leaned towards a more efficient approach with a small dip in effectiveness in extinguishing fires. Though this statement may not be true for all the setups, as planning in large setups takes more time to become effective,

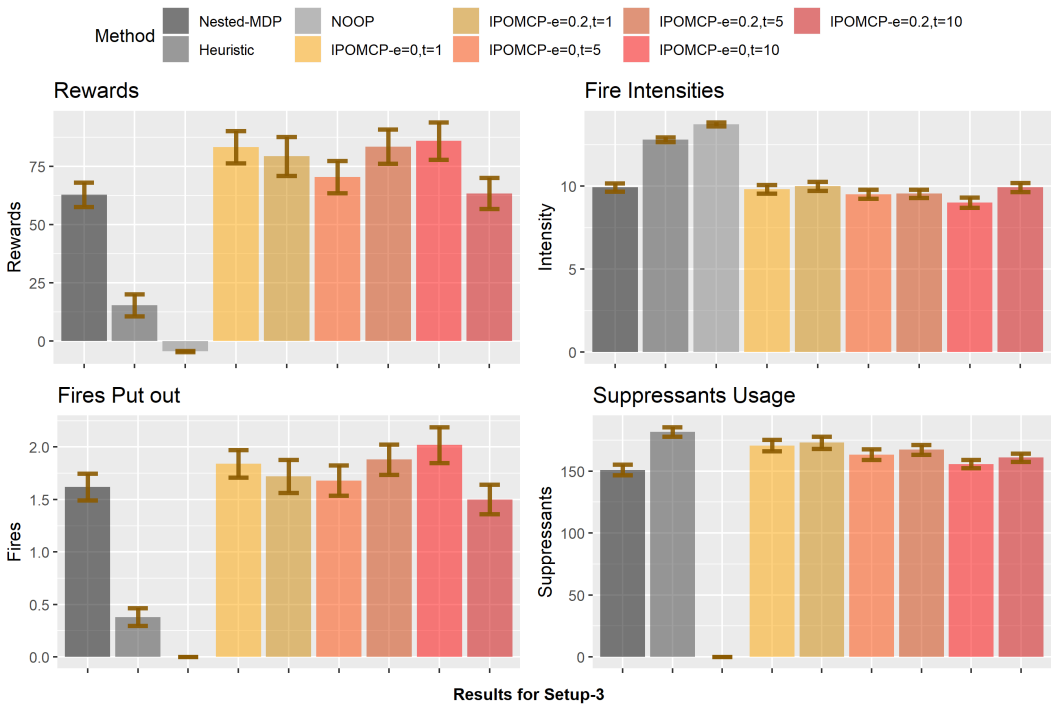


(a)

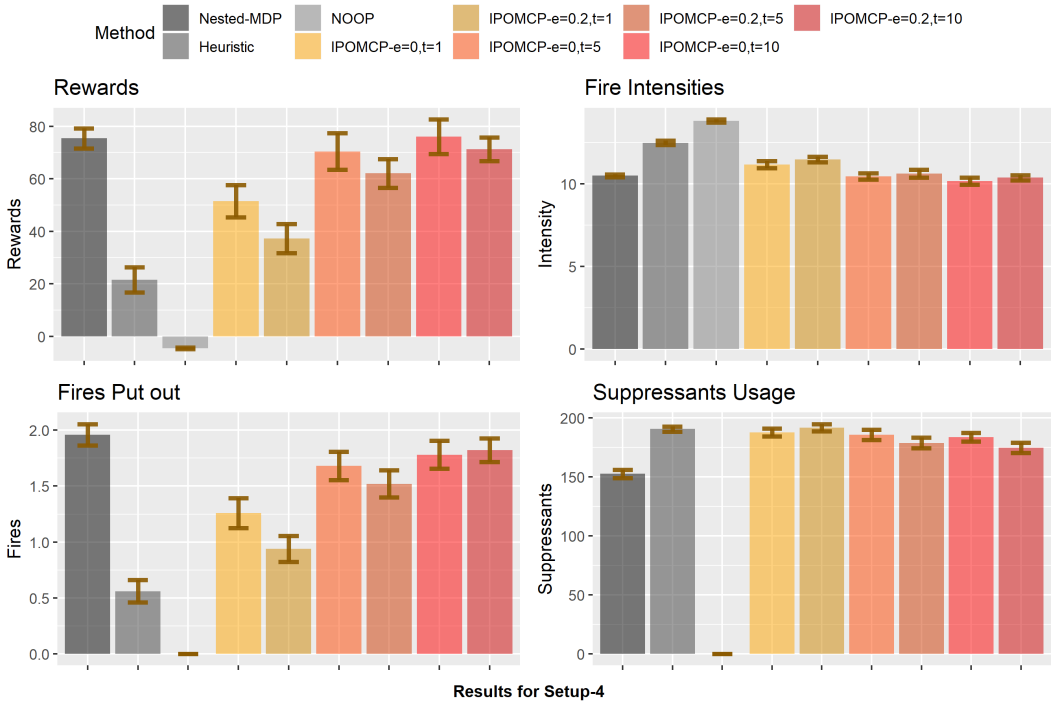


(b)

Figure 4.2: Results of Setup 1 and 2: The $I-POMCP$ perform significantly better than the other baseline methods in Setup 1, while in Setup 2, the difference is not that big. As Setup 2 is more complex than Setup 1, the $I-POMCP$ algorithm takes more time in delivering sizable results. Here, $e_{\hat{p}}$ is noted as e, and τ is noted as t for notation convenience.

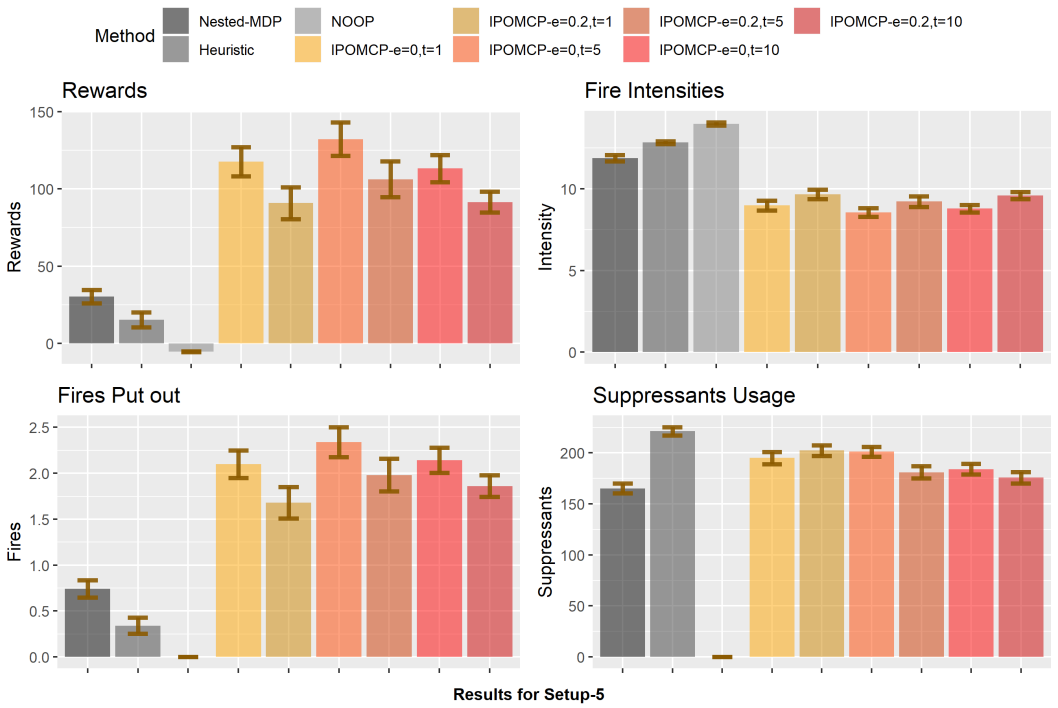


(a)

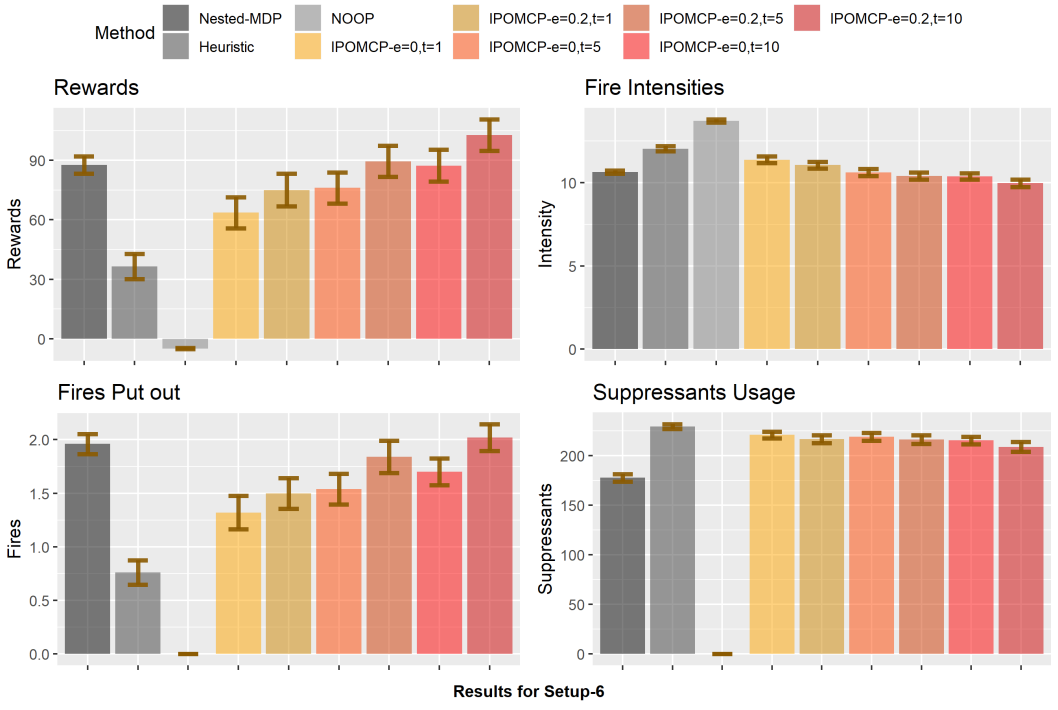


(b)

Figure 4.3: Results of Setup 3 and 4: The $I-POMCP$ perform significantly better than the other baseline methods in Setup 3, while in Setup 4, it delivers statistically equivalent results compared to $Nested-MDP$. In Setup 4, the $I-POMCP$ agent face Volunteer's dilemma as the background $Nested-MDP$ reasons to focus on just small shared fire, and sampling error helps $I-POMCP$ to overcome this issue partially. Here, $e_{\hat{p}}$ is noted as e, and τ is noted as t for notation convenience.



(a)



(b)

Figure 4.4: Results of Setup 5 and 6: The $I-POMCP$ perform significantly better than the other baseline methods in Setup 5, while in Setup 6, it delivers statistically equivalent results compared to $Nested-MDP$. In Setup 6, the $I-POMCP$ agent face Volunteer's dilemma as the background $Nested-MDP$ reasons to focus on just small shared fire, and sampling error helps $I-POMCP$ to overcome this issue partially. Here, $e_{\hat{p}}$ is noted as e, and τ is noted as t for notation convenience.

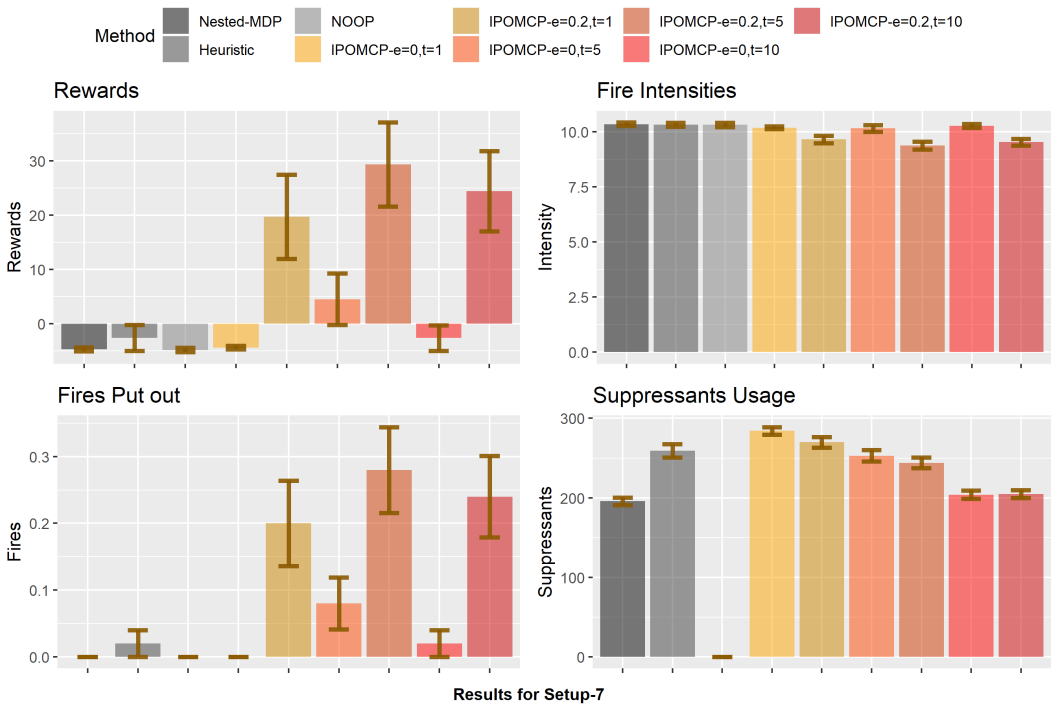


Figure 4.5: Results of Setup 7: The $Nested-MDP$ has a directionless random policy in this setup, and the $I-POMCP$ agents overcome this problem with the help of sampling error. Here, $e_{\hat{p}}$ is noted as e, and τ is noted as t for notation convenience.

allowing more time in such conditions might produce the same behavior. Conclusively, the $I-POMCP$ methodology starts exploring efficiency in operations, after it reaches to a threshold of effectiveness.

In Setups 4 and 6, the $Nested-MDP$ agents were commonly focused only on one shared small fire, and that allowed them to extinguish the fire and to earn a reward in the early stages of a trial. In such cases, as the $I-POMCP$ agents relied on the level-1 $Nested-MDP$ policies to build their own reasoning to perform actions, they faced Volunteer's dilemma in choosing actions. [29] Their reasoning about other agents at that stage, (1) all the agents would be fighting the same fire, and (2) no other agents would fight any other fire, tricked several of them in believing that their use of suppressants would be wasteful in fighting any of these fires. Such reasoning also happened in all setups at different points in a trial and

produced inefficient behavior in agents. Exploring more trajectories and larger error rates in these cases helped to reduce inefficiency, though not a complete removal of it.

The symmetrical Setup 7 produced a glitch in $Nested-MDP$ reasoning and made it produce directionless random actions. While, as discussed already, the $Nested-MDP$ policies made the agents focus on just one fire in Setup 4 and 6. In such scaled setups, the sampling error ($e_{\hat{p}}$) of 0.2 produced better results than the value 0 in the $I-POMCP$. Having a reduced number of sampled agents allowed an $I-POMCP$ agent to explore more trajectories in the tree and produced a better estimate of the effect of its actions while reducing the underlying inefficiencies. Unambiguously, with the more complex reasoning in the $Nested-MDP$, it becomes evident that sampling fewer agents within a bound produces better results for the $I-POMCP$.

Apart from the performance comparison of the $I-POMCP$ method with other baseline methods, this research tried to verify if the $I-POMCP$ framework reasons appropriately with different types of agents. To achieve such verification, experiments were conducted on Setup 1 by changing the A2 agents in the setup from ground firefighters to helicopters. The $I-POMCP$ agents behaved in accordance with such over-resourced conditions by spreading their resources to different fires at the beginning of a trial compared to just focusing on the shared fire as the $I-POMCP$ agents acted in the original Setup 1. Although the results are not included in the charts explicitly for comparison, they have illustrated that the $I-POMCP$ framework correctly reasons about the environments with different types of agents.

The wildfire domain experimental results demonstrate that the $I-POMCP$ methodology produces better coordination among the agents and justifies itself better than the other baseline approaches. Overall, the $I-POMCP$ methodology presents a reliable approach for achieving the objectives of individual planning in a limited time period.

Chapter 5

Conclusion and Future Works

In real-world applications involving many agents of different types, decision planning remains an immense challenge. Above that, many-agent systems may exhibit agent openness, where agents may leave and rejoin the environment at any point in time during the operation. The introduced novel method $I-POMCP$ generalizes the $MCTS$ algorithm for many-agent MAS by incorporating planning under agent anonymity. Extrapolation of the behavior of selectively modeled agents and tracking agent openness in the models of other agents offer an effective approach to handle scalability in a principled way. The changes proposed to modify $Nested-MDP$ by incorporating planning under agent anonymity and configuration filtering reduces the $Nested-MDP$ planning time to a realistic limit. As demonstrated in the experimental results using the wildfire domain, this new methodology improves over its competitive baselines.

Several ideas and direction for future works are as follows. First, in the introduced approach, the $Nested-MDP$ reasoning is always at level-1 while the $I-POMCP$ reasons at level-2. Even though the reasoning at a higher level is possible in $I-POMCP$, reliable planning for such would be a profoundly complex task and would not be achieved in a realistic time limit. In the future, more interesting ways could be explored to achieve planning at higher levels with the use of agent anonymity and extrapolation of agent behaviors. Second, the $I-POMCP$ approach is designed to produce deterministic actions with the underlying use of deterministic $Nested-MDP$ policies. Using stochastic policies in both frameworks may reduce the effects of Volunteer's dilemma and would allow $I-POMCP$ to explore more trajectories to make better decisions. Apparently, this change would come

with a challenge of handling a drastic rise in the fanout of the configurations at belief nodes in $I-POMCP$. Finally, both $I-POMCP$ and $Nested-MDP$ methods have an immense potential to be parallelized. Planning for several states in parallel for each loop of value iteration in $Nested-MDP$ would allow efficient planning on large distributed clusters and would significantly reduce the planning time. On the other hand, several already explored parallel planning approaches in $MCTS$ [30, 31] may be incorporated into $I-POMCP$ to explore significantly more trajectories in the same planning time.

Bibliography

[1] Yoonheui Kim, Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Exploiting locality of interaction in networked distributed pomdps. In *AAAI Spring Symposium: Distributed Plan and Schedule Management*, pages 41–48, 2006.

[2] Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[3] Trong Nghia Hoang and Kian Hsiang Low. Interactive POMDP lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. 2013.

[4] Onn Shehory. Software architecture attributes of multi-agent systems. In *International Workshop on Agent-Oriented Software Engineering*, pages 77–90. Springer, 2000.

[5] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[6] Matthijs TJ Spaan, Geoffrey J Gordon, and Nikos Vlassis. Decentralized planning under uncertainty for teams of communicating agents. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 249–256. ACM, 2006.

[7] Muthukumaran Chandrasekaran, Adam Eck, Prashant Doshi, and Leenkiat Soh. Individual planning in open and typed agent systems. pages 82–91, 2016.

[8] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In *Advances in neural information processing systems*, pages 1225–1232, 2008.

[9] Yee Whye Teh. Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. Springer, 2011.

[10] Ekhlas Sonu, Yingke Chen, and Prashant Doshi. Decision-theoretic planning under anonymity in agent populations. *Journal of Artificial Intelligence Research*, 59:725–770, 2017.

[11] Frans A Oliehoek, Matthijs TJ Spaan, Shimon Whiteson, and Nikos Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *7th International Joint conference on Autonomous agents and Multiagent Systems (AAMAS)*, pages 517–524, 2008.

[12] Prasanna Velagapudi, Pradeep Varakantham, Katia Sycara, and Paul Scerri. Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 955–962, 2011.

[13] Frans A Oliehoek, Shimon Whiteson, and Matthijs TJ Spaan. Approximate solutions for factored Dec-POMDPs with many agents. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 563–570, 2013.

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

[15] David Silver and Joel Veness. Monte-carlo planning in large pomdps. pages 2164–2172, 2010.

[16] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 1772–1780, USA, 2013. Curran Associates Inc.

[17] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[18] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

[19] A. Hula, P.R. Montague, and P. Dayan. Monte Carlo planning method estimates planning horizons during interactive social exchange. *PLOS Computational Biology*, 11(6), 2015.

[20] Christopher Amato and Frans A Oliehoek. Scalable planning and learning for multiagent POMDPs. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[21] Joao V Messias, Matthijs Spaan, and Pedro U Lima. Efficient offline communication policies for factored multiagent pomdps. In *Advances in Neural Information Processing Systems*, pages 1917–1925, 2011.

[22] Stuart Russell and Peter Norvig. *Artificial Intelligence : A modern approach*. Prentice Hall, Englewood Clifs, NJ, 2009., 3rd ed. edition.

[23] Wikipedia geometric series. `https://en.wikipedia.org/wiki/Geometric_series`. Accessed: 2018-05-01.

[24] Iadine Chades, Josie Carwardine, Tara G Martin, Samuel Nicol, Régis Sabbadin, and Olivier Buffet. Momdps: A solution for modelling adaptive management problems. 2012.

[25] J. Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.

[26] Sharon L. Lohr. *Sampling: Design and Analysis*. Brooks/Cole, Boston, MA, 2nd edition, 2010.

[27] Martin R. Frankel and Lester R. Frankel. Fifty years of survey sampling in the united states. *Public Opinion Quarterly*, 51(4):S127–S138, 1987.

[28] Burlap library. `http://burlap.cs.brown.edu/`. Accessed: 2017-09-01.

[29] Andreas Diekmann. Volunteer's dilemma. *Journal of conflict resolution*, 29(4):605–610, 1985.

[30] Guillaume MJ-B Chaslot, Mark HM Winands, and H Jaap van Den Herik. Parallel monte-carlo tree search. In *International Conference on Computers and Games*, pages 60–71. Springer, 2008.

[31] Richard B Segal. On the scalability of parallel uct. In *International Conference on Computers and Games*, pages 36–47. Springer, 2010.

Appendix A

Filtering Configurations

Figure A.1 illustrates the effect of configuration filtering using a small constant value ϵ. The reduced number of frame-action configurations ($\hat{\mathcal{C}}$) represents the set of configurations which can at least contribute ϵ value to the Q-value of a $Nested - MDP$ state.

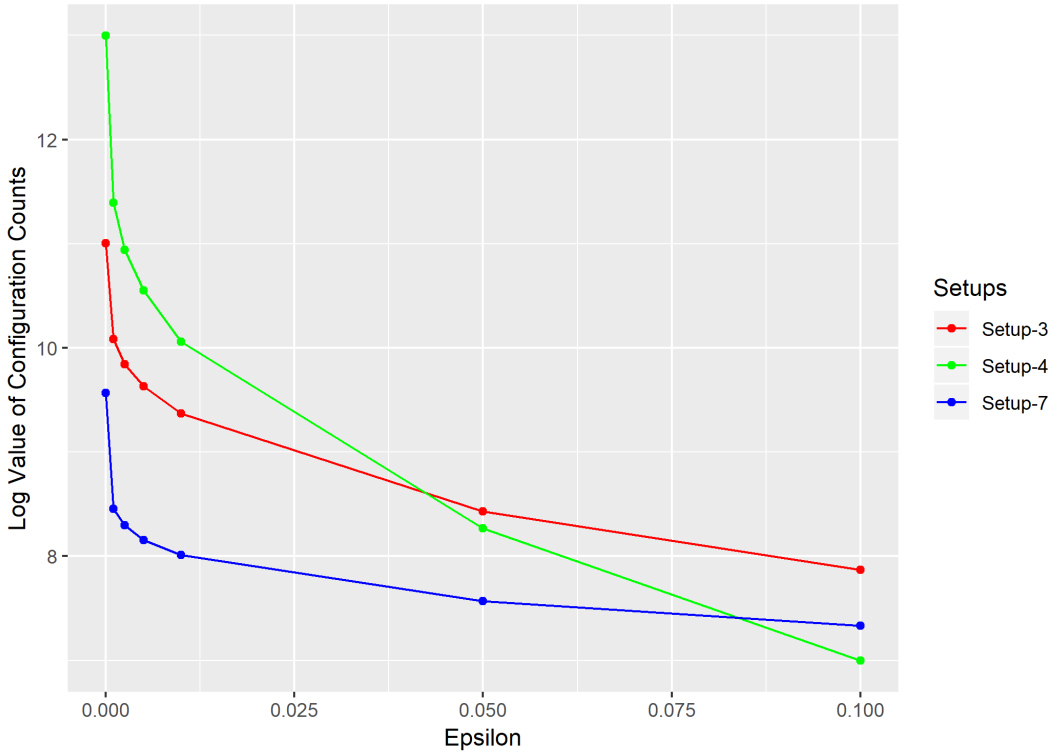


Figure A.1: Log values of filtered configurations for ground-firefighters from unit A1 in the experimental setups as illustrated in figure 4.1 against the desired values of ϵ.

Appendix B

Minimum Sample Size

Figure B.1 illustrates how the number of neighbors that an agent needs to model depends on the desired error bound of the estimation of the likelihood parameters to the multinomial distribution $P(C_\theta|s^h, M^h)$ for various neighborhood sizes N (with 95% confidence, that is α= 0.05).

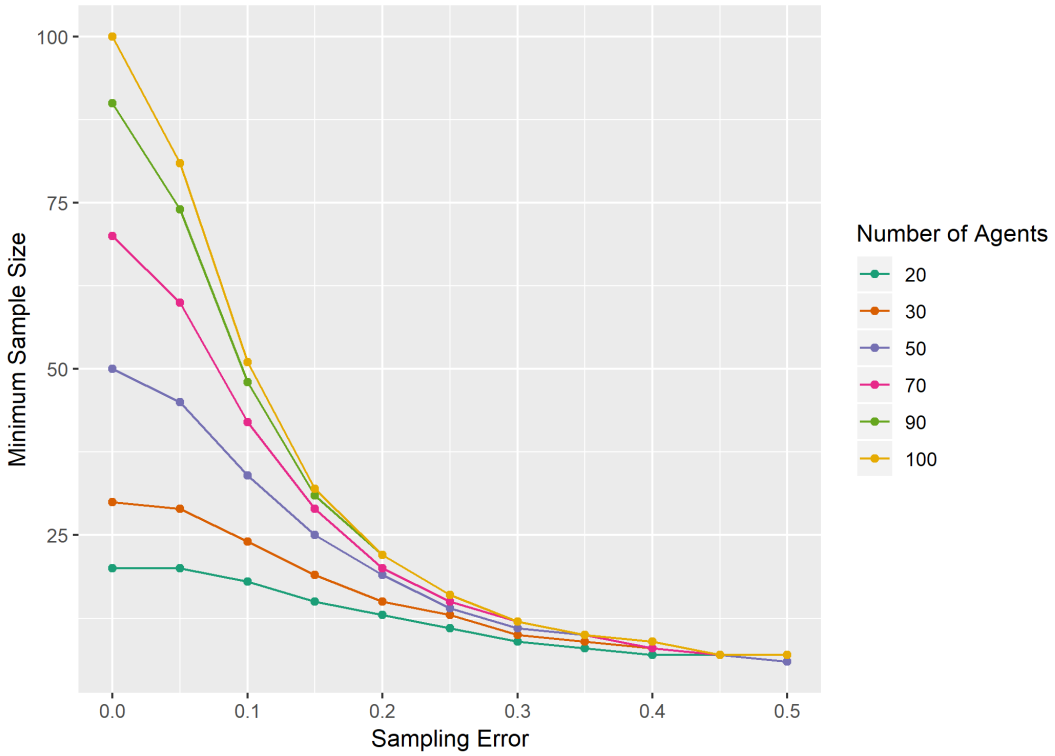


Figure B.1: Number of agents to model for desired sampling error ($e_{\hat{p}}$) and confidence ($1-\alpha$)