Probabilistic Decision-Making Models for Multi-Agent Systems and Human-Robot Collaboration

Yang You

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Probabilistic Decision-Making Models for Multi-Agent Systems and Human-Robot Collaboration

THÈSE

présentée et soutenue publiquement le February 22, 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

YOU YANG

Composition du jury

Président :	Armelle Brun	Professeur, Université de Lorraine
Rapporteurs :	Abdel-Illah Mouaddib	Professeur, Université de Caen Normandie
	Adriana Tapus	Professeur, ENSTA Paris
Examinateurs :	Olivier Buffet	Chargé de recherche, INRIA (directeur)
	Caroline Chanel	Maître de conférences, ISAE-Supaéro
	Vincent Thomas	Maître de conférences, Université de Lorraine (co-directeur)
Invité :	Rachid Alami	Directeur de recherche, LAAS-CNRS

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Résumé

Dans cette thèse, nous nous intéressons à la prise de décision haut niveau (planification de tâches) pour la robotique à l'aide de modèles de prise de décision markoviens et sous deux aspects : la collaboration robot-robot et la collaboration homme-robot.

Dans le cadre de la collaboration robot-robot (RRC), nous étudions les problèmes de décision de plusieurs robots devant atteindre un objectif commun de manière collaborative, et nous utilisons le cadre des processus de décision markoviens partiellement observables et décentralisés (Dec-POMDP) pour modéliser de tels problèmes. Nous proposons deux nouveaux algorithmes pour résoudre les Dec-POMDP. Le premier algorithme (Inf-JESP) trouve des équilibres de Nash en construisant itérativement pour chaque agent la politique meilleure réponse aux autres agents jusqu'à ce qu'aucune amélioration ne soit possible. Pour traiter les Dec-POMDP à horizon infini, nous représentons la politique de chaque agent à l'aide d'un contrôleur à états finis. Le deuxième algorithme (MC-JESP) étend Inf-JESP avec des modèles génératifs, ce qui nous permet de passer à l'échelle pour des grands problèmes. Nous démontrons expérimentalement que nos méthodes sont compétitives par rapport aux solveurs Dec-POMDP existants.

Dans le cadre de la collaboration homme-robot (HRC), nous ne pouvons contrôler que le comportement du robot, lequel doit faire face à des objectifs humains et à des comportements induits incertains. Nous cherchons ainsi à dériver des politiques de robot qui sont robustes aux incertitudes sur les comportements humains. Pour cela, nous discutons des modèles mentaux qui peuvent être utilisés pour modéliser le comportement humain dans une telle tâche collaborative. Nous décrivons ensuite une approche générale pour dériver, automatiquement et sans connaissance préalable, un modèle de comportements humains basé sur l'hypothèse que l'humain contrôle aussi le robot. À partir de là, nous proposons deux algorithmes pour calculer des politiques robustes pour le robot en se basant sur la résolution d'un POMDP dont l'état contient l'état interne de l'humain. Le premier algorithme fonctionne hors ligne et fournit une politique complète qui peut être utilisée par le robot pendant son exécution. Le deuxième algorithme est une méthode en ligne, c'est-à-dire qu'il décide de l'action du robot à chaque pas de temps au cours de l'exécution. Par rapport à l'approche hors ligne, la méthode en ligne ne nécessite qu'un modèle génératif et peut donc s'adapter à de grands problèmes. Des expériences avec des humains synthétiques et réels sont menées dans un environnement simulé pour évaluer ces algorithmes. Nous observons que nos méthodes peuvent fournir des décisions robustes pour des robots collaboratifs malgré les incertitudes sur les objectifs et les comportements humains.

Dans cette thèse, notre recherche sur la collaboration robot-robot fournit une base pour construire des politiques meilleure réponse dans un cadre partiellement observable et multi-agent, base qui sert d'étape intermédiaire importante pour aborder les problèmes de collaboration homme-robot. De plus, pour chaque contribution, nous fournissons des algorithmes plus flexibles utilisant des modèles génératifs dont nous pensons qu'ils faciliteront la mise en œuvre de nos contributions à des applications du monde réel.

Abstract

In this thesis, using Markov decision models, we investigate high-level decision-making (task-level planning) for robotics in two aspects: robot-robot collaboration and human-robot collaboration.

In robot-robot collaboration (RRC), we study the decision problems of multiple robots involved to achieve a shared goal collaboratively, and we use the decentralized partially observable Markov decision process (Dec-POMDP) framework to model such RRC problems. Then, we propose two novel algorithms for solving Dec-POMDPs. The first algorithm (Inf-JESP) finds Nash equilibrium solutions by iteratively building the best-response policy for each agent until no improvement can be made. To handle infinite-horizon Dec-POMDPs, we represent each agent's policy using a finite-state controller. The second algorithm (MC-JESP) extends Inf-JESP with generative models, which enables us to scale up to large problems. Through experiments, we demonstrate our methods are competitive with existing Dec-POMDP solvers.

In human-robot collaboration (HRC), we can only control the robot, and the robot faces uncertain human objectives and induced behaviors. Therefore, we attempt to address the challenge of deriving robot policies in HRC, which are robust to the uncertainties about human behaviors. In this direction, we discuss possible mental models that can be used to model humans in an HRC task. We propose a general approach to derive, automatically and without prior knowledge, a model of human behaviors based on the assumption that the human could also control the robot. From here, we then design two algorithms for computing robust robot policies relying on solving a robot POMDP, whose state contains the human's internal state. The first algorithm operates offline and gives a complete robot policy that can be used during the robot's execution. The second algorithm is an online method, *i.e.*, it plans the robot's action at each time step during execution. Compared with the offline approach, the online method only requires a generative model and thus can scale up to large problems. Experiments with synthetic and real humans are conducted in a simulated environment to evaluate these algorithms. We observe that our methods can provide robust robot decisions despite the uncertainties over human objectives and behaviors.

In this thesis, our research for RRC provides a foundation for building best-response policies in a partially observable and multi-agent setting, which serves as an important intermediate step for addressing HRC problems. Moreover, we provide more flexible algorithms using generative models in each contribution, and we believe this will facilitate applying our contributions to real-world applications.

Acknowlegements

I would like first to thank my supervisors, Olivier Buffet, Vincent Thomas, and Francis Colas. We have worked together since my master's internship and then my Ph.D. at INRIA Nancy. Without their help and guidance, this thesis would not have existed.

I want to thank all my committee members for coming to my defense and sharing insightful perspectives. In particular, I want to thank Rachid Alami for providing the necessary ideas for human-robot collaboration experiments at the beginning of my Ph.D.

I want also to thank all my friends who supported me during my thesis, especially Danlei Zhang, Abir Bouaouda, Aya Yaacoub and Nima Mehdi.

Finally, I want to sincerely thank my parents, especially my mother, who cares about me every day when I live and study abroad.

Contents

List of Figures

1

Introduction

1.1 Motivation and Research Questions

Collaborative robots are widely used in modern society in various domains. For example, in agriculture, a robot can work as a sprayer to help farmers get rid of harmful insects [Berenstein and Edan, 2012]; in daily life, robots provide services in hospitals for therapeutic programs and guide passengers in airports [Tapus et al., 2007, Triebel et al., 2016]; in industry, robots are used to lift heavy objects, perform high-precision manipulations, or collaborate with human workers [Vysocky and Novak, 2016]. We can mainly categorize these robot collaboration applications into:

- Robot-Robot Collaboration (RRC): multiple robots are involved together to collaboratively achieve a task or a specified goal;
- Human-Robot Collaboration (HRC): Humans and robots work together cooperatively; usually, a goal is specified by human users or linked to humans' objectives.

This thesis is funded by the ANR project *Flying Co-Worker*, an aerial manipulator robot that acts as a teammate of a human worker to realize complex tasks. As shown in Figure 1.1, one possible scenario is that two workers are in a working space, and a flying drone is designed to deliver tools between workers. In this thesis, we are interested in the robot's decision-making system in a broader view in both RRC and HRC. In other words, we investigate how the robot should decide on its actions in order to achieve a task, no matter whether the partners are humans or other robots. We focus on sequential decision-making as the collaboration tasks take multiple time steps to accomplish, so we search for policies to maximize the team's global performance. Those policies indicate what actions the agents should perform for each condition they encounter. Moreover, one can decompose the decision-making process hierarchically into low-level and high-level decisions [Cacace et al., 2015], here (in robotics) with low-level decisions corresponding to low-level control, i.e., motor commands, and high-level decision [1] corresponding to task-planning, i.e., deciding to "pick an object", "repair a device". Our thesis focuses on the high-level decision (hence task-planning) for robotics in both RRC and HRC.

However, the research questions raised by task planning in RRC and HRC are different due to their different features. In RRC, we assume that we can assign policies for all robots, and then they behave independently in the environment. Thus, we investigate how to design algorithms that compute optimal policies for each agent to maximize the overall performance. On the other

[1] In robotics, a high-level decision will eventually compile to low-level controls. Thus, the high-level decision making is (in a sense) about decomposing the original problem into sub-problems.

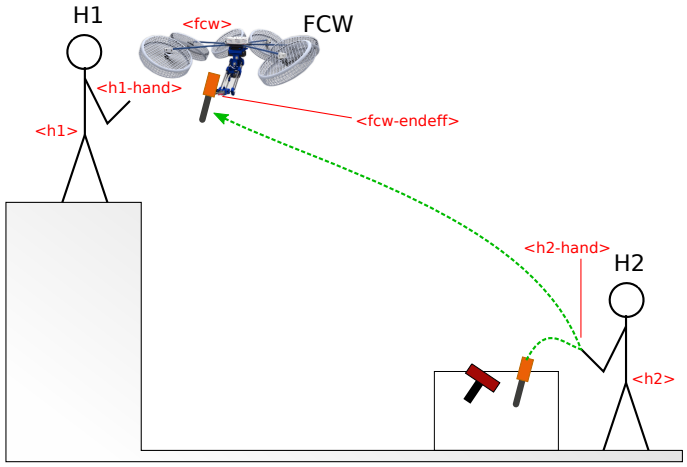

Figure 1.1: An illustration of the Flying Co-Worker project scenario: a flying drone equipped with an end effector delivers tools for the workers in a working space.

hand, in HRC, only the robot can be controlled but not the human (in this thesis, we assume there is one human and one robot in an HRC team). Therefore, the challenge is to design a decision-making system only for the robot, and it should adapt to the human behaviors regarding his objectives in order to achieve the collaboration task.

1.2 Overview of Robot-Robot Collaboration

A multi-agent system (MAS) defines a set of individual agents that work collaboratively to solve problems such as electric grids, sensor networks, or intelligent transportation. In this sense, the robots in a robot-robot collaboration (RRC) task make up a MAS since they work together to achieve a shared goal. One can use Markov decision models to formalize sequential decision-making problems in the MAS setting. The key feature of Markov decision models is the Markov property, i.e., the system's next state only depends on the current state and the executed action [Bellman, 1957]. Among all the Markov decision models, the Markov decision process (MDP) is the most basic one for single-agent problems. It addresses a sequential decision-making problem by formalizing a controllable agent in its environment with discrete time, stochastic dynamics, and fully observable states. The *partially observable Markov decision process* (POMDP) is a generalization of the MDP. It models one agent's decision process in the same way as an MDP. However, the agent in a POMDP cannot directly observe the true world state but only partial or noisy information. The *decentralized partially observable Markov decision process* (Dec-POMDP) [Bernstein et al., 2002] extends POMDPs to the multi-agent setting, The objective of solving a Dec-POMDP is to derive strategies for all agents to achieve a common task. Overall, the Dec-POMDP is the most general framework among all Markov decision models, and it fits the multi-agent and collaboration setting for RRC problems in this thesis.

In terms of solving techniques, there are mainly two approaches: reinforcement learning (RL) and planning. RL is about learning the optimal behavior to maximize a numerical reward signal by trial and error through interactions with the environment [Sutton and Barto, 2018]. One can distinguish "offline" RL (with 2 phases: training and execution) and "online" RL (with a single phase where one learns directly in the real world). In both cases, RL methods do not need a complete problem model but only a black-box simulator (typically offline RL) or even just the

real environment (typically online RL). On the other hand, planning methods usually rely on a complete problem model. Unlike RL approaches, where the agent interacts with the environment to gather information, planning methods can often directly use the model information to compute optimal policies. In this thesis, we would like to have a generic method taking advantage of the model of the task, we thus investigate the planning approaches to design the robot decision-making system.

To reduce ambiguities, we need to emphasize that we focus on the high-level, sequential decision-making problems, and the planning methods in our context are related to the Markov models.

1.2.1 Planning for Single-Agent Settings

Before introducing planning methods for multi-agent decision-making, we first look at planning in a single-agent case. Generally, single-agent problems are often modeled using MDPs if the environment state is fully observable or POMDPs if the underlying state cannot be directly observed. In our thesis, we won't directly use MDPs or POMDPs for modeling the RRC problems since they are designed for single-agent decisions. However, in HRC, the robot's decision could be modeled using POMDPs because of its limited observations, and because we cannot control the human but only the robot.

The planning methods for solving such single-agent problems can be classified into two types: model-based or simulator-based. Model-based planning methods require an explicit model of the problem, which means that the dynamics of the environment should be known, including how the environment state evolves, and how the environment is perceived if it is not fully observable. Classic model-based planning methods such as *value iteration* and *policy iteration* for MDPs are also the foundations of RL algorithms [Bellman, 1957, Howard, 1960], and state-of-the-art model-based planning algorithms such as HSVI [Smith and Simmons, 2004] and SARSOP [Kurniawati et al., 2008] can give ϵ-optimal solutions for POMDPs. However, for some large-scale problems, such as planning in an urban environment [Delamer et al., 2019], building explicit models is difficult since it will cost a lot of memory to represent the full dynamics. Moreover, exact computations may be too costly (at least regarding CPU time), so sampling-based methods can be a good solution in large-scale problems. Sampling-based planning approaches [Metropolis and Ulam, 1949, Silver and Veness, 2010] only require a black-box simulator which takes inputs of the current state and executed action, then gives the next state, agent's observation and instant reward as outputs.

One can also distinguish single-agent planning algorithms into two types: offline and online planning. Offline planning contains two steps: policy construction and policy execution. In the policy construction phase, an optimal policy is computed which contains the best actions for all possible situations. This optimal policy will then be executed in the execution phase. Unlike offline planning, online planning methods [Ross et al., 2008, Metropolis and Ulam, 1949, Silver and Veness, 2010] interleave the planning and execution processes. In the planning phase, online algorithms try to find the best action given the current situation within a bounded memory or time budget. Then, when the planning phase terminates, the agent performs this action in the execution phase and updates its information state in order to launch the next plan. Compared with offline planning, online planning can better fit applications with fixed time constraints and non-stationary environments [Chatzis and Kosmopoulos, 2014, Lecarpentier and Rachelson, 2019]. In those cases, using offline methods to enumerate all possible situations is computationally infeasible. However, if enough resources and time are given, offline methods [Smith and Simmons, 2004, Kurniawati et al., 2008] will converge to near-optimal solutions.

1.2.2 Planning for Multi-Agent Settings

For multi-agent planning, the purpose is to compute optimal policies for each agent in order to achieve the common goal. As in the single-agent setting, we divide current multi-agent planning algorithms into mainly two categories:

- Explicit model-based algorithms: This approach requires an explicit model of the multi-agent problem, including the environment dynamics and reward function. Then, using the model knowledge, optimal or suboptimal policies for each agent are computed by using heuristic search [Szer et al., 2005, Dibangoye et al., 2016], parameters optimizations [Bernstein et al., 2009] or finding Nash-equilibrium solutions [Nair et al., 2003, You et al., 2021a];

- Simulator-based (sampling-based) algorithms: in this case, algorithms [Wu et al., 2013, Liu et al., 2015, Dibangoye and Buffet, 2018] do not need an explicit model, but a black-box simulator, with which agents can interact and receive observations and rewards.

Compared with the single-agent setting, multi-agent decision-making brings more challenges in several aspects. First, agents usually have limited access to the environment and cannot observe the exact status of other agents. Thus it is difficult for the agents to coordinate their behaviors only based on local observations. Second, the complexity of computation grows exponentially with the number of agents. Therefore, scalability is also an issue for multi-agent decision-making. In this thesis, we will give more details regarding the models and theories in Chapter 2 and related algorithms in Chapter 3. Our contributions in both explicit model-based and simulator-based planners for solving multi-agent problems modeled with Dec-POMDPs are presented in Chapter 4.

1.3 Overview of Human-Robot Collaboration

In the previous section, we briefly introduced Robot-Robot Collaboration (RRC) and related decision-making methods. In this section, we introduce *Human-Robot Collaboration* (HRC), which has emerging applications in recent years, such as in manufacturing [Zanchettin et al., 2016, Guerin et al., 2015, Wang et al., 2017, Zanchettin and Rocco, 2013] and healthcare [Tapus et al., 2009, Jacob et al., 2013]. In HRC, a collaborative team of humans and robots work together to achieve a shared goal. HRC is also part of the broader field of human-robot interaction (HRI), where the various partners do not necessarily benefit from the interaction [Bauer et al., 2008]. Here, let us summarize some dimensions and challenges in HRC:

- Robot design: To enable a successful collaboration between the human and robot, the first step is to study how to design the robot, which involves the following aspects:

  - Robot appearance: The human's acceptability to the robot partner varies because of the robot appearance [Walters et al., 2008, Mori, 2012], which will affect the human's attention, effectiveness during a collaboration task Goetz et al. [2003], Walters et al. [2009], Kwak [2014]. Thus, one important topic is how to design the robot's appearance.
  - Hardware and software: HRC tasks require the robot to provide different kinds of service or assistance to the human partner. In response to those requirements, the robot should be equipped with appropriate hardware [Wang et al., 2019, Khalid et al., 2016] such as smart sensors [Schmidt and Wang, 2014, Wang, 2015] to detect human

behaviors, and software such as tools to facilitate programming of collaborative robots [Mokaram et al., 2017].

– Interface and communication: In HRC, the human user should feel easy to interact and communicate with the robot [Villani et al., 2018]. This requires properly designed user interfaces such as providing augmented reality [Tian and Paulos, 2021] or virtual reality [Wang et al., 2019] to illustrate the robot status, and enabling remote operation with dedicated tools [Lv et al., 2020, Wang et al., 2019].

Safety: In HRC, human safety is a fundamental prerequisite, especially for industrial applications. To ensure safe interactions between humans and robots in a shared workspace, one particular research question is collision avoidance. To do so, it is important to detect human movements and identify potential risks, such as a human approaching within a danger range. There are multiple ways to detect human movements, such as using depth cameras [Nascimento et al., 2020, Mohammed et al., 2017] and laser scanners [Safeea and Neto, 2019]. Moreover, to increase the detection precision, methods such as data fusion have been applied in order to efficiently use multiple sensors' data [Lin et al., 2020]. After potential risks have been detected, the robot should take appropriate reactions such as emergency stops or changing its moving direction [Nascimento et al., 2020] to prevent collisions.

The outcome of such research on safety may lead to new industrial standards. Up to now, standards have been established to ensure the safety of collaborative robots mainly in the industry, such as ISO 13850, 13851, 10218-1/2 [Villani et al., 2018]. Those safety standards define the requirements and design guidelines for developing robot systems in HRC. For example, ISO 10218-1/2 [ISO 10218-1:2011, ISO 10218-2:2011] identify four collaborative modes including *safety-rate monitored stop, hand guiding, speed and separation monitoring, power, and force limiting*, and list required robot functionalities to ensure safety. In the future, more standards will be created with the increased usage of collaborative robots in domains like home service [Taipalus and Kosuge, 2005] and healthcare [Tapus et al., 2009, Jacob et al., 2013].

- Robot autonomy: In HRC, the robot autonomy can be roughly divided into two parts, *perception system* and *decision system*, as follows:

  – Perception system: In HRC, robot sensors receive signals from the environment and the human partner. Thus, one issue is how to let the robot understand those signals and make use of them. To answer this question, multiple techniques have been applied in HRC, such as using histogram of oriented gradients (HOG), support vector machines (SVM), or convolutional neural networks (CNN) for image recognition [Feng and Yuan, 2013, Zhao et al., 2019]; processing IMU's data though hidden Markov model (HMM) [Roitberg et al., 2014, Wang et al., 2018]; and using data fusion techniques to analyze multi-sensor data [Lin et al., 2020]. Recently, deep learning has made great progress, especially in the machine vision domain, with deep learning methods outperforming human experts' accuracy for classification tasks. Therefore, state-of-the-art deep learning methods such as YOLOv5 [Jocher, 2020] have been introduced to HRC applications [Liau and Ryu, 2021] as powerful tools for motion and gesture detection.

  – Decision system: The core of the robot's autonomy is its decision system. It reflects how the robot adapts to different situations and assists the human partner with appro-

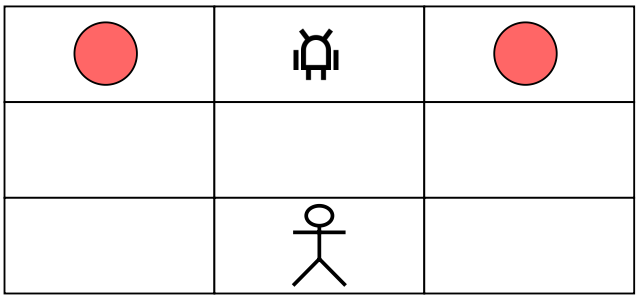

Figure 1.2: An illustration of the coordination issue for the robot in HRC. The human and robot need to repair all the broken devices (top left and top right ones). They can repair one device if and only if they are in the same cell and performing repairing actions simultaneously.

priate decisions. In classical planning, the robot plans its actions considering temporal and domain constraints and controls executions in real time [Alami et al., 1998]. To handle tasks with uncertain environments, many works [Zheng et al., 2018, Görür et al., 2018, Unhelkar et al., 2020, Nikolaidis et al., 2017, Chen et al., 2018] also use Markov decision models to represent the robot decision framework in HRC. For example, Nikolaidis et al. proposed a robot decision framework for a collaborative grasping task based on a POMDP [Nikolaidis et al., 2017]. Since human has different statuses during the task and evolves over time, Nikolaidis et al. define the human's status as a part of the state and integrate human status dynamics inside the POMDP's transition function. Therefore, the robot could update its belief on the human status from observed human actions. Then, solving this POMDP would result in a robot decision solution (robot policy) considering human behaviors. Moreover, other methods such as imitation learning [Lee et al., 2011] and reinforcement learning [Liu et al., 2021] can be also used in HRC.

Among those research directions, in this thesis, we focus on robot decision-making for HRC.

1.3.1 Robot Decision-Making in HRC

HRC brings more challenges for the robot's decision-making compared with RRC. Unlike RRC, which specifies all agents' behaviors, the human partner in HRC cannot be controlled, and his intention and future behaviors are often hidden from the robot. Thus, a real challenge for the robot is how to make decisions on its behavior to coordinate with the human. Here we use a simple example to explain this challenge, as shown in Figure 1.2, where two broken devices are located on the top left and top right corners of the map. The goal is to repair all the broken devices. In RRC, this is an easy problem since we can directly control two agents and move them to the broken devices' locations together and do the repair simultaneously. However, in HRC, the robot does not know which device the human wants to repair first. Thus the robot needs to observe the human behavior to infer his intention. Moreover, humans are not fully rational, and their behaviors can be uncertain due to multiple factors such as emotions and fatigue. Therefore, in this simple example, humans could have different paths to broken devices, some of which may be erratic. For example, a human may go left first, wait for a while, return to his original position, and then head to the top-left broken device. From the robot's point of view, to make correct decisions, all those uncertainties must be considered.

As we previously mentioned, many researchers [Zheng et al., 2018, Görür et al., 2018, Unhelkar

et al., 2020, Nikolaidis et al., 2017, Chen et al., 2018] rely on Markov decision models, especially using the POMDP framework to represent the robot decision-making for HRC, and a human model [Nikolaidis et al., 2017] is usually integrated as parts of the environment dynamics. By doing so, the robot could reason about its actions considering possible human behaviors and following consequences. However, to have such human models, expert knowledge or collected data in specific domains are usually required [Nikolaidis et al., 2017, Chen et al., 2018].

In this thesis, we also rely on Markov decision models to design the robot decision for HRC. We investigate and make contributions to the following challenges:

- How should the robot model the human partner in order to estimate his intention and behavior, this modeling process is automatically done by the robot, and it should be generic enough so that no expert knowledge and data are needed;
- How should the robot decide its own actions to coordinate with the human partner, the robot's decisions should be robust enough regarding the uncertainties of human intention and behavior.

We will give more details regarding the related works of robot decisions for HRC in Chapter 5 and our contributions for HRC are written in Chapter 6.

1.4 Contribution

We summarized the relations of main contributions in this thesis as in Table 1. The contribution in this thesis can be divided into two main parts (RRC and HRC).

For RRC, this thesis provides novel techniques for solving muti-agent decision problems modeled with Dec-POMDPs in which agents only have partial observations of the environment. More specifically, inspired by JESP [Nair et al., 2003], we would like to solve Dec-POMDPs by finding Nash equilibrum solutions. Thus, we iteratively optimize each agent's policy until no improvements can be made. Moreover, we address JESP's drawback that it can only solve finite-horizon problems because it uses policy trees. In our thesis, we use finite-state controllers to represent each policy, and we propose two novel algorithms (Inf-JESP and MC-JESP) for solving infinite-horizon Dec-POMDPs. These two contributions differ in that one uses an explicit model while the other uses a generative model. Through experiments on benchmark problems, we demonstrate the validity of our methods, which perform quite well compared to other existing Dec-POMDP solvers.

On the other hand, for human-robot collaboration, we focus on the robot's decision-making and aim to compute robot policies that are robust to uncertain human objectives and behaviors. Relying on Markov decision models, we provide a robust robot planning algorithm that relies on solving a POMDP where one hidden state variable corresponds to human behavior. However, human behaviors are usually uncertain and unknown to the robot. Unlike other approaches [Nikolaidis et al., 2017, Chen et al., 2018], which need expert knowledge or collected data to provide such a human model, we provide an approach to automatically generate an uncertain human behavior (a policy) for each human objective. Moreover, in our approach, the generated human model is not myopic, which can account for possible robot behaviors and consider long-term consequences. We distinguish two variants of this robust robot planning method. The first algorithm is based on knowing the explicit model of the collaboration task and computing a robust robot policy by estimating possible human behaviors. This method operates offline, so we first compute a robust robot policy and assign it to the robot before the actual execution. The second algorithm works online, and only a black box simulator is sufficient. To evaluate our

Table 1.1: Table of the relations about contributions

Task Type	RRC	HRC
Explicit Model	Inf-JESP	Robust Robot Planning - Offline
Generative Model	MC-JESP	Robust Robot Planning - Online
Research Interest	Optimality	Robustness

methods, a co-working scenario is designed, which allows conducting experiments and presenting qualitative and quantitative results to evaluate our approach.

1.5 Outline

The structure of this thesis is organized as follows:

- Chapter 1 gives an overview of this thesis, the motivation, research questions, contributions and outlines;
- Chapter 2 is the background part, where we introduce different Markov decision models related to our research and discuss their relations and variations;
- Chapter 3 discusses the related works of multi-agent decision making applicable to robot-robot collaboration;
- Chapter 4 presents our contributions on how to solve multi-agent decision problems with partial observations;
- Chapter 5 describes the related works of human-robot collaboration, which are categorized based on how the robot models its human partner;
- Chapter 6 presents our contributions to solving human-robot collaboration problems by computing robust robot policies;
- Chapter 7 concludes the thesis and discusses future directions.

2

Background

2.1 Frameworks for Sequential Decision Making

Decision making is an important part of daily life. We, humans, make decisions and follow plans to achieve our goals. It is the same for automated software, robotics, and all kinds of machines to decide what to do or how to react in different conditions.

One big branch in this context is classical planning, where the aim is to find a sequence of actions to end up in a goal state. Classical planning has a wide range of real-world applications, such as in logistics [Helmert, 2009, García et al., 2013], robotics [Alami et al., 1998, González et al., 2017] and web services [Yang and Qin, 2010]. However, in classical planning, the environment and actions are usually deterministic, and a practical approach is to "re-plan" whenever the state encountered is not the one anticipated, but this can lead to catastrophes. To that end, *contingent planning* and *conformant planning* [Cimatti and Roveri, 2000, Hoffmann and Brafman, 2005, 2006, Albore et al., 2009] are meant to handle planning problems with uncertainty about the initial state and action outcomes under partial or no observability. A key issue with these approaches is that they ignore probabilities. Thus, they can only rely on criteria such as the "worst-case outcome" of the (conditional) plan, which can be overly pessimistic

Another classic framework for sequential decision making is the Markov decision process, which is the main tool we used in this thesis. In Markov Decision models, the agent(s) will take actions and receive observations come from the environment state. The environment dynamic is modeled by the probabilities of state transition and observation. The reward defines the agents' goal, where the agents need to maximize the expected accumulated reward during the task. The key feature in Markov decision models, which is called *Markov property*, is that the next state only depends on the current state and the action performed, not the history. In this chapter, we introduce Markov decision processes in detail, including MDPs, POMDPs for single-agent settings and Dec-POMDPs for multi-agent settings.

2.2 Markov Decision Process (MDP)

In this section, we present the Markov decision process (MDP) model, which is one of the most basic decision-making formalisms for single-agent settings. Despite its simplicity, it can be used to formalize tasks where an agent is in a stochastic environment with full observability of the environment state.

As illustrated on Figure 2.1, an MDP is formally defined by a tuple $\langle \mathcal{S}, \mathcal{A}, T, r \rangle$ where

- $\mathcal{S}$ is the state space;

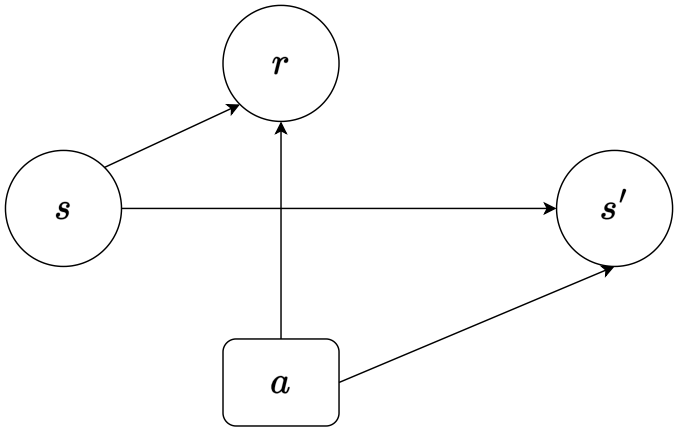

Figure 2.1: Markov decision process illustration: the agent chooses an action a at state s, will have a reward r and reach next state s'.

- $\mathcal{A}$ is the action space;
- $T(s, a, s') = Pr(s'|s, a)$ is the transition function, which encodes the dynamic of the environment (the probability of reaching the next state s' given the current state s and an executed action a);
- $r(s, a)$ is the reward function, which gives the instant reward when performing action a at state s. Note that one can also specify $r(s, a, s')$, but then define $r(s, a) = \sum_{s'} T(s, a, s') \cdot r(s, a, s')$.

2.2.1 Performance Measurement

As mentioned above, after choosing an action a at state s, a reward r will be given. However, the instant reward cannot evaluate a policy since one can easily be misled by a high reward in the current state and lose the opportunity to gain more in the future.

In MDPs, a policy π is a mapping from states to actions $\pi : \mathcal{S} \to \mathcal{A}$, and the goal is to find the optimal policy π^* which brings the highest accumulated rewards. We call $V^\pi(s)$ the *value function*, and it evaluates policy π when executed from state s:

$$V^\pi(s) = E_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s \right]. \tag{2.1}$$

The discounted factor γ indicates the importance of future rewards compared with the current reward. Especially if $\gamma = 0$, it means the agent will behave greedily and only considers the current reward; if $\gamma = 1$, the future reward and current reward are equally important, but this will cause a problem in the infinite-horizon setting since the agent sees no difference from picking any action in terms of expected value at infinite steps. Thus, in infinite-horizon settings, γ is in range $[0, 1)$ but in finite-horizon settings, γ could be set to 1. Given a policy π, the agent's action selection is denoted as $a = \pi(s)$, then Equation (2.1) can be rewritten as

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s'). \tag{2.2}$$

This value function V^π is also called *state-value function*. On the other hand, if we also consider the action a, one can define the *action-value function* as follows:

$$Q^\pi(s,a) = E\left[\sum_{i=0}^{\infty}\gamma^i r_i | s_0 = s, a_0 = a\right] \tag{2.3}$$

$$= \sum_{s'\in S} T\left(s,a,s'\right)\left[r\left(s,a\right) + \gamma V^\pi\left(s'\right)\right] \tag{2.4}$$

$$= r(s,a) + \gamma \sum_{s'\in S} T(s,a,s')V^\pi\left(s'\right). \tag{2.5}$$

According to the definition of *state-value function* and *action-value function*, the state value is the expected value when performing action $\pi(s)$; Similarly, the *action-value function* is defined by an instant reward plus the discounted future expected state value.

In MDPs, the goal is to find an optimal policy,*i.e.*, one that maximizes the expected value as defined in Equation (2.1). This optimal policy is denoted π^*, and it should maximize the value function for any state s as shown in Equation (2.6).

$$\pi^* = \arg\max_\pi V^\pi(s). \tag{2.6}$$

Note that there may be multiple optimal policies in an MDP, but all of them share the same value function. Here we define the optimal *state-value function* V^* as in Equation (2.7). This mathematical expression is also called *Bellman optimality equation*.

$$V^*(s) = \max_a [r(s,a) + \gamma \sum_{s'\in S} T(s',a,s)V^*(s')]. \tag{2.7}$$

Similarly, the optimal *action-value function* Q^* is defined in Equation (2.8). It is the sum of the current instant reward plus the discounted expectation of optimal future values.

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'\in S} T(s,a,s')V^*(s'). \tag{2.8}$$

From those two definitions, we know that the optimal value in a given state equals the expected return of its best action, knowing that an optimal policy is followed afterward.

2.2.2 Optimization Algorithms

In the previous section, we have defined the MDP model and value functions. In this section, we introduce algorithms for solving MDPs that give optimal policies.

Dynamic Programming Methods

Dynamic programming (DP) is a classic approach that breaks down a complex problem into simpler sub-problems in a recursive manner [Howard, 1960]. For MDPs, there are two main DP algorithms, *value iteration* and *policy iteration* respectively. Before introducing those two detailed algorithms, we first present the *Contraction Mapping Theorem* (also called *Banach fixed-point theorem*).

Definition 1 *An operator $\mathcal{F}$ on a vector space $\mathcal{X}$ is a γ-contraction for all $x, y \in \mathcal{X}$ if $0 < \gamma < 1$:*

$$\|F(x) - F(y)\| \leqslant \gamma\|x - y\| \tag{2.9}$$

Theorem 1 *For a given γ-contraction $\mathcal{F}$ in $\mathcal{X}$, it has two properties (proofs can be found in [Denardo, 1967]):*

- *the iteration of $\mathcal{F}$ will converge to a fixed point in $\mathcal{X}$ in any start point;*
- *the convergence rate is determined linearly by γ.*

Value Iteration: We first introduce the *value iteration* algorithm as shown in Algorithm 1. In

Algorithm 1: Value Iteration

```
1  [Initilization:] V(s) = 0, for all s ∈ S
2  repeat
3  |   △← 0
4  |   for s ∈ S do
5  |   |   v ← V(s)
6  |   |   V(s) ← max_a[r(s,a) + γ Σ_{s'∈S} T(s,a,s')V(s')]
7  |   |   △← max(△, |v − V(s)|)
8  until △< ε (a small positive number)
9  π(s) = arg max_a[r(s,a) + γ Σ_{s'∈S} T(s,a,s')V(s')]
10 return π
```

the value iteration algorithm, all states' values are initialized to 0 (line 1). Then, it recursively updates values based on the best next state (line 6). In line 8, this process is stopped when the change is smaller than ϵ. Therefore, we obtain the optimal value function V^*, which stores the optimal value for each state. An optimal policy π^* is easily computed by finding the best action at each state (line 9). The key to this method is to recursively refine the estimator V^*:

$$V^*_{i+1}(s) \leftarrow \max_a \sum_{s' \in S} [r(s,a) + \gamma T(s,a,s') V^*_i(s')], \tag{2.10}$$

which is a contraction mapping between vector spaces $V_{i+1} = H_{max} V_i$ (see Definition 1), and H_{max} is a γ-contraction operator. Therefore, applying this operator repeatedly is guaranteed to converge to a fixed optimal value function V^*

Policy Iteration: On the other hand, *Policy iteration* starts with a given policy π and improves it iteratively until an optimal policy is obtained. As shown in Algorithm 2, it contains two steps, policy evaluation and policy improvement. In policy evaluation, the aim is to evaluate $V^\pi(s)$ for the current policy π. This is a linear system since V^π contains $|S|$ linear equations with $|S|$ unknowns. This linear system $V = H^\pi V$ relies on a contraction mapping operator H^π (line 6), so repeatedly applying this operator leads to the only solution V^π. In the policy improvement step, with the evaluated values, the aim is to find the best action for each state s (line 14) in order to improve the current policy. Given π_k and π_{k+1} the two policies before and after improvement, policy iteration iterates this process until $\pi_{k+1} = \pi_k$, meaning that an optimal policy has been reached.

Sampling-Based Methods

In the previous section, we briefly introduced two DP methods: *value iteration* and *policy iteration*. Both used knowledge of the explicit model, including the state transition probabilities

Algorithm 2: Policy Iteration

[Initilization:] initialize $V(s)$ and $\pi(s)$ arbitrarily, for all $s \in \mathcal{S}$
Fct Policy Evaluation
  repeat
    $\triangle \leftarrow 0$
    for $s \in \mathcal{S}$ **do**
      $v \leftarrow V(s)$
      $V(s) \leftarrow r(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V(s')$
      $\triangle \leftarrow \max(\triangle, |v - V(s)|)$
  until $\triangle < \epsilon$ *(a small positive number)*
Fct Policy Improvement
  $policy - stable \leftarrow true$
  for $s \in \mathcal{S}$ **do**
    $old - action \leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_a [r(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') V(s')]$
    if $old - action \neq \pi(s)$ **then**
      $policy - stable \leftarrow false$
  if $policy - stable$ **then**
    return π
  else
    go to **Policy Evaluation**

and expected rewards, to compute an optimal policy π^*. However, for large problems, there is a "curse of dimensionality" since the cost of computing the expectation over all possible future states grows exponentially with the number of states [Rust, 1997]. Moreover, for such large problems, representing explicitly the full model could even be unfeasible. Unlike the DP methods we mentioned in the last section, sampling-based planning methods only require a generative model (also called a simulator). Such an MDP generative model G takes a state-action pair of state and action $\langle s, a \rangle$ as input, then returns a next state s' and an instant reward r.

In this section, we mainly introduced the *Monte-carlo tree search* algorithm (MCTS) [Browne et al., 2012], which is one of the most popular sampling-based planning methods. In MCTS, the curse of dimensionality is avoided by only sampling states from a generative model instead of calculating all possible state transitions using an explicit MDP model. Globally, MCTS searches a tree policy as shown in figure 2.2, going through 4 main steps at each iteration:

1. *Selection*: Starting from the root node, recursively select child nodes until reaching a leaf node L;

2. *Expansion*: Randomly create an unvisited child node C of node L if L is not a terminal node;

3. *Simulation*: Starting from node C, perform simulations using rollout-policy $\pi_{Rollout}$ (also called default policy which is usually a random policy) till a terminal state;

4. *Back-propagation*: Use the result obtained from the simulation step to update the values of all parent nodes in the selected path.

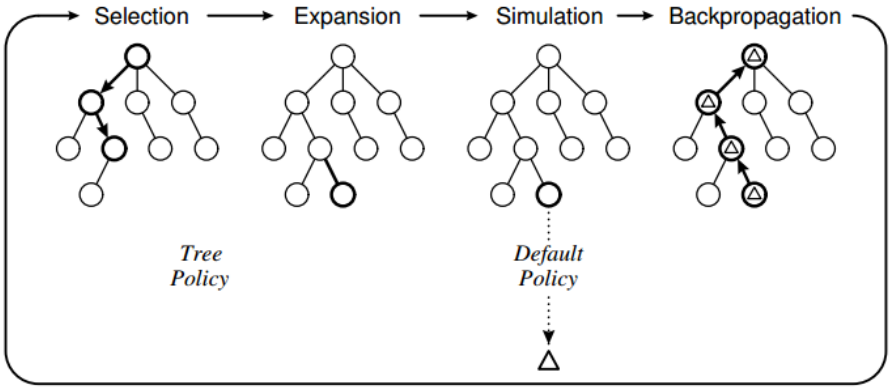

Figure 2.2: The four steps of an iteration of Monte-Carlo Tree Search [Browne et al., 2012]: Selection, Expansion, Simulation, and Back-propagation.

For MCTS, an important question is how to balance exploration and exploitation in the search process. To that end, a practical method is picking actions *optimistically* by adding a bonus to action-value estimates that are less reliable (because they are less visited). For example, the UCT algorithm [Kocsis and Szepesvári, 2006] uses the upper confidence bound (UCB) for the tree search/exploration, where the action value is modified by adding an exploration bonus:

$$Q^{+}(s,a) = Q(s,a) + c\sqrt{\frac{logN(s)}{N(s,a)}}. \tag{2.11}$$

The visit number for each state action pair $\langle s, a\rangle$ is denoted as $N(s,a)$, and $N(s)$ is the overall visit number in state s, so that $N(s) = \sum_a N(s,a)$. The term $c\sqrt{\frac{\log N(s)}{N(s,a)}}$ allows adding a larger bonus to the rarely selected actions rather than frequently visited ones. The parameter c controls the ratio of exploration and exploitation, the algorithm tries more exploration actions when c increases and acts as a greedy policy that only exploits estimates if $c = 0$. Note that one can remove the exploration bonus to pick actions at execution time. Another common approach is to pick the most often selected action (the one maximizing $N(s,a)$), which can be more reliable.

2.2.3 Categorization of the Offline and Online Planning

For single-agent planning approches, an important categorization is to distinguish offline and online algorithms. As shown in Figure 2.3, the offline planning method first computes a complete policy for the agent (a mapping from states to actions in MDPs), and the agent executes the computed policy afterward. On the other hand, the online planning method interleaves the planning and execution phases. In the planning phase, the online approach tries to find the best immediate action given the current situation within a given time and memory budget. Then, when the planning phase terminates, the agent executes an estimated best action in the execution phase and updates its estimate of the environment state to launch the next online planning phase.

We take a simple example of a path planning problem for a robot to illustrate the difference between the offline and online methods. In offline planning, the robot has complete knowledge of the environment, including the positions of obstacles and the goal on a map, then a complete path is computed using this complete environment information before the robot behaves. In the online planning case, the planning is carried out continuously when the robot receives observations

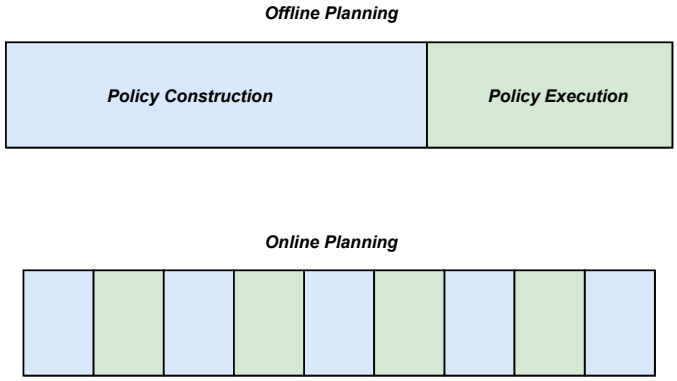

Figure 2.3: An illustration of the offline and online approaches [Ross et al., 2008].

(complete or partial) from the environment; it may also find the changes in the environment, such as the goal position or the obstacles being moved. This being said, the offline planning method is generally used for the static environment but is usually able to find optimal solutions; the online algorithms, on the other hand, are more suitable for unknown or dynamic changing environments [Ross et al., 2008, Chatzis and Kosmopoulos, 2014, Lecarpentier and Rachelson, 2019]. The dynamic programming algorithms we describe in Section 2.2.2 are meant to be used offline and guarantee to converge to optimal policies no matter the start state; and the sampling-based method, such as MCTS, belongs to online planning, which may not need the explicit model and tolerate the changes in the environment but gives approximate solutions in terms of realistic computation resources.

2.3 Partially Observable Markov Decision Process (POMDP)

In some cases, the complete environment's information is not directly accessible, the agent only has partial information about the current state [Åström, 1965]. The Partially Observable Markov Decision Process (POMDP) model is designed for such decision problems. A POMDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, T, O, r, b_0 \rangle$ where $\langle \mathcal{S}, \mathcal{A}, T, r \rangle$ is an underlying MDP, and:

- Ω is the observation space;
- $O(o, a, s') = Pr(o|s', a)$ is the observation function; it indicates the probability of receiving an observation o given the next state s' reached while performing action a;
- b_0 is an initial distribution over states.

As illustrated in Figure 2.4, at each time step t, performing an action a when in state s induces a reward $r(s, a)$ and leads to state s' with probability $T(s, a, s')$. However, the agent can only receive an observation o, which contains partial or noisy information on the next state.

In MDPs, the state is a sufficient statistics to reason about the current situation and make optimal decisions since the agent does not need any additional information. However, in POMDPs, since the real state is not accessible by the agent anymore, all the information the agent has

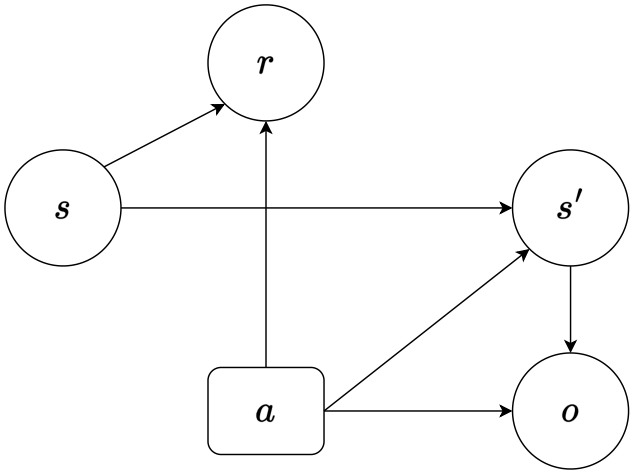

Figure 2.4: Partially observable Markov decision process

at execution time is the action-observation history (AOH), so its decisions will depend on that. In POMDPs, we call AOH a *information state*. However, in long-horizon problems, storing such histories is costly. Therefore, another popular method is using a probability distribution over possible states to represent the current information state, and this distribution is called a belief state (or belief) b. Moreover, the agent needs to update its belief after an action a is executed and an observation o is received. This belief update process is detailed as follows, where $b^{a,o}$ is a new belief and $b^{a,o}(s')$ is the probability of being in the state s' in that belief state:

$$b^{a,o}\left(s'\right) = Pr\left(s'|o,a,b\right) = \frac{Pr(s',o,a,b)}{Pr(o,a,b)}, \tag{2.12}$$

$$= \frac{Pr\left(o|s',a,b\right) Pr\left(s'|a,b\right) Pr(a,b)}{\sum_{s'}\sum_{s} Pr(o,s,s',a,b)}, \tag{2.13}$$

$$= \frac{Pr\left(o|s',a\right) \sum_{s} Pr\left(s'|a,b,s\right) Pr(s|a,b)}{\sum_{s'} Pr\left(o|s',a\right) \sum_{s} Pr\left(s'|a,b,s\right) Pr(s|a,b)}, \tag{2.14}$$

$$= \frac{O\left(s',a,o\right) \sum_{s} T\left(s,a,s'\right) b(s)}{\sum_{s'} O\left(s',a,o\right) \sum_{s} T\left(s,a,s'\right) b(s)}. \tag{2.15}$$

As for MDPs, the objective of POMDPs is to generate a policy whose execution maximizes the agent's expected accumulated rewards. The optimal value function of POMDPs is defined in Equation (5),

$$V^*(b) = \max_{a\in\mathcal{A}} \left[r(b,a) + \gamma \sum_{o\in\Omega} Pr(o|b,a) V^*(b') \right] \tag{2.16}$$

and its optimal policy is defined as follows:

$$\pi^*(b) = \arg\max_{a\in\mathcal{A}} \left[r(b,a) + \gamma \sum_{o\in\Omega} Pr(o|b,a) V^*(b') \right]. \tag{2.17}$$

Here the function $r(b,a)$ specifies the expected immediate reward of executed action a under the belief b:

$$r(b,a) = \sum_{s\in\mathcal{S}} b(s) r(s,a),$$

and the function $Pr(o|b,a)$ is the probability of recieving an observation o in belief b with performed action a:

$$Pr(o|b,a) = \sum_{s' \in \mathcal{S}} O\left(s', a, o\right) \sum_{s \in \mathcal{S}} T\left(s, a, s'\right) b(s).$$

2.3.1 History-Based Policy

Policy Trees

For the finite-horizon setting, a POMDP policy can be represented in a tree structure as shown in Figure 2.5 (a). Each tree node contains an action a to be executed by the agent, then steps to a lower branch depending on which observation o the agent received. For a finite horizon H, a policy tree contains $\sum_{t=0}^{t=H-1} |\Omega|^t = \frac{|\Omega|^H - 1}{|\Omega| - 1}$ nodes. In our example Figure 2.5 (a), the horizon $H = 3$ and the size of observation space $|\Omega| = 2$, thus we have $2^0 + 2^1 + 2^2 = 7$ nodes. Since each node has $|A|$ choices of actions, the number of all policy trees in horizon H is $|A|^{\frac{|\Omega|^H - 1}{|\Omega| - 1}}$.

Moreover, using policy trees, one can approximate the value of a discounted infinite-horizon POMDP as follows:

$$V^*(b_0) - V_H^*(b_0) \leqslant \frac{\gamma^H}{1-\gamma}[R_{max} - R_{min}], \tag{2.18}$$

where $V^*(b_0)$ is the optimal value of the true POMDP and $V_H^*(b_0)$ is the optimal value given a truncated horizon H.

Finite State Controllers

An issue of the policy tree structure is its complexity in the long horizon problem and its impossibility of storing all the decisions in the infinite horizon problems. Therefore, finite state controllers (FSCs) are usually used to represent POMDP policies for the infinite-horizon setting. Compared with the policy tree, an FSC is defined by a directed graph that contains cycles to handle infinite horizons. Here we first introduce the most generic form, the stochastic FSC. For some POMDP sets $\mathcal{A}$ and Ω, a stochastic FSC is defined by a tuple $fsc \equiv \langle N, \beta, \eta, \psi \rangle$, where:

- N is a finite set of (internal) nodes;
- β is a probability distribution from which to sample the initial node;
- $\eta : N \times \mathcal{A} \times \Omega \times N \to \mathbb{R}$, the transition function, gives the probability $\eta(n, \langle a, o \rangle, n')$ of transiting from node n to n' if a is performed and o is observed; a deterministic transition can be noted $n' = \eta(n, \langle a, o \rangle)$;
- $\psi : N \times \mathcal{A} \to \mathbb{R}$, the action selection function, gives the probability $\psi(n, a)$ of choosing $a \in \mathcal{A}$ when in n.

But in most cases, deterministic FSCs are used to present POMDP policies in policy search algorithms [Hansen, 1997, 1998] where the action selection function ψ gives a deterministic action a at each node n. An illustration of such a deterministic FSC is presented in Figure 2.5 (b).

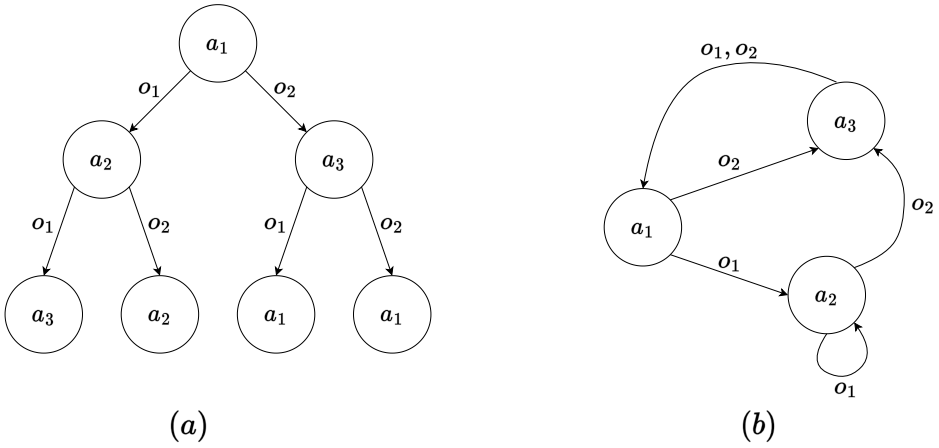

Figure 2.5: Two example representations of POMDP policies: (a) policy tree; (b) finite state controller [Amato, 2010].

2.3.2 Belief-Based Policy

α-vectors

Another type of policy representation is using α-vectors, which is a belief-based policy. For a given POMDP, although its belief space B is infinite, it is proved that one can approximate the POMDP's value function as an upper envelope of a finite set of linear functions $\Gamma = \{\alpha_1, \dots \alpha_n\}$ [Sondik, 1978],which are also known as α-vectors. Each α-vector is attached to an action a and contains $|S|$ values for each corresponding state, and in this case, for a specific belief b, the representation of the value function is defined as:

$$V(b) = \max_{\alpha \in \Gamma} b \cdot \alpha, \tag{2.19}$$

$$\text{where} \quad b \cdot \alpha = \sum_{s \in S} b(s)\alpha(s). \tag{2.20}$$

This representation provides a lower-bound to approximate V^*, and to derive a policy from α-vectors, one can find out which α-vectors is attached to the current belief (see Equation (2.19)), and the action attached to the α-vector is the one to perform. This approach guarantees an expected return above the value of this lower bound. Modern POMDP solvers such as SARSOP [Kurniawati et al., 2008] and HSVI [Smith and Simmons, 2004] usually give solutions in the form of α-vectors.

Moreover, one can transform a set of α-vectors to a corresponding FSC. Grześ et al. propose such a method called *Alpha2FSC* as presented in Algorithm 3 [Grześ et al., 2015]. In this method, it is assumed that a set of α-vectors in Γ is given, and each $\alpha_i \in \Gamma$ is attached with a belief b_i and action a_i (line 1). Then, an FSC node n_i is created for each α_i and labed by action a_i (lines 4-6). In lines 9-13, the transitions between nodes are determined by finding which α-vector has the highest value of $b_i^{a_i,o}$ (which is the updated belief for each node n_i with attached action a_i and received observation o), and a self-loop is created for any impossible observation given the current node n_i. Moreover, if a set of α-vectors in Γ is the optimal value function, one can transform Γ to an optimal corresponding FSC [Grześ et al., 2015]. However, this transformation guarantees that an optimal corresponding FSC is derived only if the set of α-vectors in Γ is the optimal value function. Otherwise, if the set of α-vectors is suboptimal (which is the case for

Algorithm 3: Alpha2FSC

```
[inputs:] Γ: a set of ⟨α_i, b_i, a_i⟩-tuples
[outputs:] an FSC ⟨N, ψ, η⟩
N ← ∅
for i = 1 to |Γ| do
    N ← N ∪ n_i
    ψ(n_i) ← a_i
for i = 1 to |Γ| do
    for o ∈ Ω do
        if Pr(o|b_i, a_i) > 0 then
            best ← arg max_j b_i^{a_i,o} · α_j
            η(n_i, o) ← n_best
        else
            η(n_i, o) ← n_i
return ⟨N, φ, ψ⟩
```

modern POMDP solvers), the transformation is an approximation, and its quality may be better or worse. In this thesis, we also provide other approximate transformation methods in Chapter 4.

Last but not least, a deterministic FSC's value function (*i.e.*, η and ψ being both deterministic) is the solution of the following system of linear equations, with one α-vector per node n [Hansen, 1997]:

$$\alpha_s^n = R(s, a_n) + \gamma \sum_{s',o} T(s, a_n, s') O(a_n, s', o) \alpha_{s'}^{\eta(n,o)}, \tag{2.21}$$

where $a_n \stackrel{\text{def}}{=} \psi(n)$. Using the fixed point theorem, a solution can be found using an iterative process typically stopped when the Bellman residual (the largest change in value) is less than a threshold ϵ, so that the estimation error is less than $\frac{\epsilon}{1-\gamma}$.

2.3.3 Optimization Algorithms

Many algorithms have been proposed to solve POMDPs. For example, in the finite-horizon case, a POMDP can be turned into a finite-horizon belief-space MDP, thus with finitely many belief states [Åström, 1965]. Therefore, exact value iteration and policy iteration methods for MDPs can be applied. But the real breakthroughs in the past decades are the point-based algorithms, including PBVI [Pineau et al., 2003], HSVI [Smith and Simmons, 2004] and SARSOP [Kurniawati et al., 2008]; and sampling-based methods such as POMCP [Silver and Veness, 2010] and DESPOT [Somani et al., 2013]. Both types of methods can solve complex POMDP problems with many thousands of states.

Point-Based Method

The Point-Based Value Iteration algorithm (PBVI) [Pineau et al., 2003] is the foundation of modern point-based solvers [Shani et al., 2012]. As we know in POMDPs, belief space is continuous, meaning there are infinitely many belief points. However, in practice:

- some belief points are unreachable with any given action-observation history;

- we want to focus on points that are reachable under an optimal policy;

Therefore, the problem is quickly figuring out which set of belief points to use (and at which to perform Bellman updates). To that end, PBVI focuses on solving POMDPs within a set of reachable belief points, and the value function is approximated as an upper envelope of α-vectors (which acts as a lower bound) denoted as Γ.

Algorithm 4: Point-Based Value Iteration Algorithm

```
1 [inputs: ] B_0: initial belief set | Γ_0: initial value | N: expansion number | H: iteration
    number
2 B = B_0
3 Γ = Γ_0
4 for N expansions do
5 |   for H iterations do
6 |   |   Γ = BACKUP(Γ, B)
7 |   B' = EXPAND(Γ, B)
8 |   B = B ∪ B'
9 return Γ
```

As shown in Algorithm 4, a belief set B is first initialized with starting belief b_0. Then in lines 5–6, in each iteration, PBVI updates the value function V by performing a Bellman backup at each belief point b stored in B, in which:

$$\Gamma_B^{i+1} = backup(\Gamma_B^i, b), b \in B; \tag{2.22}$$

$$backup(\Gamma, b) = \arg\max_{a \in A, \alpha \in \Gamma} b \cdot \alpha_a^b; \tag{2.23}$$

$$\alpha_a^b = r_a + \gamma \sum_{o \in \Omega} \arg\max_{\alpha \in \Gamma} b \cdot \alpha^{a,o}. \tag{2.24}$$

This process is repeated H times, and usually, the H is selected so that $\gamma^H[R_{max} - R_{min}] < \epsilon$ where ϵ is the error bound. After H backups are finished, PBVI expands its belief set B by selecting for belief $b \in B$ a reachable successor belief b' (lines 7–8), which has the longest distance from the current belief set B. In the end, PBVI returns the value function Γ, a set of α vectors, and each $\alpha \in \Gamma$ comes with an action to perform.

Based on PBVI, researchers made improvements to further increase the scalability of the point-based methods. For example, in HSVI [Smith and Simmons, 2004] and SARSOP [Kurniawati et al., 2008], heuristics are introduced to focus on the points which are most relevant to the value function, and an upper and lower bound are maintained over the value function to guide local updates and bound the error.

Sampling-Based Method

As in an MDP, sampling-based methods for a POMDP do not need an explicit model but only a generative model G (a black-box simulator). Such a model takes a state-action pair and returns a tuple of successor state, observation, and instant reward as follows:

$$\langle s', o', r \rangle \sim G(s, a). \tag{2.25}$$

In Section 2.2.2 (page 12), we have introduced the Monte-Carlo tree search (MCTS) algorithm for solving MDPs. Partially Observable Monte-Carlo Planning (POMCP) [Silver and Veness, 2010], one of the most successful sampling-based methods, extends MCTS to POMDPs. The main algorithm is present in Algorithm 5. Compared with MCTS, POMCP has two improvements to

Algorithm 5: POMCP

Fct *Search(h)*
  repeat
    if $h = empty$ **then**
      $s \sim b_0$
    else
      $s \sim B(h)$ // $B(h)$: set of state particles for the history h
    Simulate$(s, h, 0)$
  until *Timout()*
  return $\arg\max_a Q(h, a)$
Fct *Simulate(s, h, depth)*
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  if $h \notin T$ **then**
    for $a \in \mathcal{A}$ **do**
      $T(h, a) \leftarrow (N_{init}(h, a), Q_{init}(h, a), \varnothing)$
    return *Rollout(s, h, depth)*
  $a \leftarrow \arg\max_a [Q(h, a) + c\sqrt{\frac{\log N(h)}{N(h,a)}}]$
  $(s', o, r) \sim G(s, a)$
  $R \leftarrow r + \gamma Simulate(s', hao, depth + 1)$ // *hao*: new action-observation history
  $B(h) \leftarrow B(h) \cup \{s\}$
  $N(h) \leftarrow N(h) + 1$
  $N(h, a) \leftarrow N(h, a) + 1$
  $Q(h, a) \leftarrow Q(h, a) + \frac{R - Q(b,a)}{N(h,a)}$
  return R
Fct *Rollout(s, h, depth)*
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  $a \leftarrow \pi_{rollout}(h)$ // $\pi_{rollout}$: default policy (random policy)
  $(s', o, r) \sim G(s, a)$
  return $r + \gamma Rollout(s', hao, depth + 1)$

solve the POMDP problem. First, POMCP extends the UCT algorithm to a partially observable setting called PO-UCT. In PO-UCT, a search tree is built using histories instead of states. Each tree node is labeled with an action-observation history h, and in the search process, actions are selected using the same UCB method:

$$Q^+(h, a) = Q(h, a) + c\sqrt{\frac{\log N(h)}{N(h, a)}}. \tag{2.26}$$

Second, POMCP approximates the agent's belief with particles and uses a particle filter to update the current belief state after each actual transition. In the beginning, the approximate belief $\tilde{b}$ is represented with K particles of states sampled from the initial belief. Then, after an action a is executed in the real environment with an observation o received, we randomly select a state s from the current $\tilde{b}$ and pass s with executed action a to the generative model G which will return a tuple $\langle s', o', r \rangle$. If the sample observation is the same as the real observation $o' = o$, we keep the successor state s' and add it to a new approximate belief $\tilde{b}'$. This process repeats until we have k particles in $\tilde{b}'$.

There exists also other powerful sampling-based methods such as DESPOT (Determinized Sparse Partially Observable Tree) [Somani et al., 2013]. However, in this thesis, POMCP is used in our contributions because of its simplicity of implementation and efficiency in solving POMDPs.

2.3.4 Multi-Agent POMDP

For the multi-agent setting, if there is a centralized controller that controls all agents' actions and receives all agents' observations in at each time step, then we can use multi-agent POMDP (MPOMDP) [Pynadath and Tambe, 2002] to formulate such a problem. An MPOMDP is essentially a POMDP since its action space is the joint action space of all agents and the same for the observation space. All the properties and algorithms built for POMDPs can be directly applied to MPOMDPs.

2.4 Decentralized Partially Observable Markov Decision Process (Dec-POMDP)

The Dec-POMDP formalism is an extension of the POMDP formalism where several agents interact in the same environment and share the same reward function. Compared with a POMDP, a Dec-POMDP factors:

- the (joint) action into one action per agent,
- the (joint) observation into one observation per agent, and
- the (joint) policy into one individual policy per agent.

Formally, a Dec-POMDP with $|\mathcal{I}|$ agents is represented as a tuple $M \equiv \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, T, O, R, b_0, H, \gamma \rangle$ [Bernstein et al., 2002], where:

- $\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$ is a finite set of agents;
- $\mathcal{S}$ is a finite set of states;
- $\mathcal{A}^i$ is the finite set of agent i's actions; $\mathcal{A} = \times_i \mathcal{A}^i$ is the finite set of joint actions;
- Ω^i is the finite set of agent i's observations; $\Omega = \times_i \Omega^i$ is the finite set of joint observations;
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ denotes the transition function; $T(s, a, s')$ is the probability of transiting to the next state s' from state s if joint action a is performed;
- $O : \mathcal{A} \times \mathcal{S} \times \Omega \to \mathbb{R}$ is the observation function; $O(a, s', o)$ is the probability of observing o if joint action a is performed and the resulting state is s';

- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; $R(s, a)$ gives the immediate reward for executing joint action a in state s;
- b_0 is the initial probability distribution over states;
- $H \in \mathbb{N} \cup \{\infty\}$ is the (possibly infinite) time horizon;
- $\gamma \in [0, 1)$ is the discount factor, which defines the discount applied to future rewards.

Each agent i can be equipped with a policy π^i that maps its possible action-observation histories to actions. The objective is then to find a joint policy $\pi = \langle \pi^1, \ldots, \pi^{|\mathcal{S}|} \rangle$ that maximizes a performance criterion, here the expected discounted return from b_0:

$$V_H^\pi(b_0) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(S_t, A_t) \mid S_0 \sim b_0, \pi\right].$$

Dec-POMDPs are difficult to solve and are proved to be NEXP-complete [Bernstein et al., 2002, Oliehoek and Amato, 2016]. In the finite-horizon setting, the number of possible deterministic joint policies (set of all agents' individual policies) is:

$$O(|\mathcal{A}^\dagger|^{\frac{|\mathcal{I}|\{|\Omega^\dagger|^{H-1}\}}{|\Omega^\dagger|-1}}),$$

where $|\mathcal{A}^\dagger|$ and $|\Omega^\dagger|$ are the largest individual action and observation sets.

2.5 Solving Dec-POMDPs by finding Nash Equilibria

In this thesis, one of our main contributions is solving Dec-POMDPs (see Chapter 4). Specifically, we search Nash equilibrium solutions based on the JESP scheme (Joint Equilibrium based search for policies) [Nair et al., 2003]. Therefore, in this section, we detail the JESP method for solving Dec-POMDPs, and we present other Dec-POMDP algorithms in Chapter 3.

In a Nash equilibrium solution, each agent's policy is the best response to the other's policies in that equilibrium, and one has nothing to gain by only modifying its own policy [Kreps, 1989]. For a Dec-POMDP, the optimal solutions are a subset of these Nash equilibrium solutions (the ones with the highest values).

The JESP algorithm is designed for finite-horizon Dec-POMDPs, and aims to find Nash equilibrium solutions. In JESP, each agent's policy is represented using a policy tree. JESP iterates over each agent and tries to improve agent i's policy, considering that other agents' policies are fixed. More specifically, when knowing other agents' fixed policies $\pi_{\neq i}$, an extended state POMDP is built for agent i. In this POMDP, the extended state is $e^t = \langle s^t, \vec{w}^t_{\neq i} \rangle$, where s^t is the Dec-POMDP state and $\vec{w}^t_{\neq i}$ is the observation histories of other agents $\neq i$. Here, from agent i's point of view, it is sufficient to consider observation histories rather than action-observation histories because the other agents' policies $\pi_{\neq i}$ are deterministic policy trees. Therefore, other agents' actions are directly derived with the observation histories as $a^t_{\neq i} \leftarrow \pi_{\neq i}(\vec{w}^t_{\neq i})$ at each time step t. The dynamics of this extended POMDP for agent i can be defined with Dec-POMDP's transition function T, observation function O, and reward function R as:

$$\begin{aligned}
T_e(e^t, a_i^t, e^{t+1}) &= Pr(e^{t+1} \mid e^t, a_i^t) \\
&= [\Pi_{j \neq i} O_{dec,j}(s^{t+1}, \langle a_i^t, \pi_{\neq i}(\vec{w}^t_{\neq i}) \rangle, \omega_j^{t+1})][T_{dec}(s^t, \langle a_i^t, \pi_{\neq i}(\vec{w}^t_{\neq i}) \rangle, s^{t+1})], \\
O_e(a_i^t, e_i^{t+1}, o_i^{t+1}) &= Pr(o_i^{t+1} | a_i^t, e_i^{t+1})
\end{aligned}$$

$$= O_{dec,i}(s^{t+1}, \langle a_i^t, \pi_{\neq i}(\vec{w}_{\neq i}^t)\rangle, \omega_i^{t+1});$$
$$R_e(e_i^t, a_i^t) = R(s^t, \langle a_i^t, \pi_{\neq i}(\vec{w}_{\neq i}^t)\rangle).$$

Then agent i maintains an *extended belief* $\beta_i^t = Pr(e_i^t|\vec{w}_i^t, \vec{a}_i^{t-1}, b_0)$ and its value function is defined as:

$$V_i^t(\beta_i^t) = R(\beta_i^t, a_i^t) + \sum_{o^{t+1} \in \Omega_i} Pr(o^{t+1}|\beta_i^t, a_i^t) V_i^{t+1}(\beta_i^{t+1}). \tag{2.27}$$

The improvement process for agent i consists in finding the optimal policy to optimize its value function. Nair et al. propose two choices to achieve this operation, either by an exhaustive search (Exhaustive-JESP) or dynamic programming (DP-JESP) to compute a new policy tree for agent i. But, of course, any other exact finite-horizon POMDP solver could also be used for the same purpose. JESP stops when no more improvements can be made for any agent. In each iteration, the value monotonically increases and guarantees to converge to Nash equilibrium solutions, thus, local optima.

For JESP, the main drawback is that it cannot guarantee to find the global optimal solution, and the Nash equilibrium solution computed by JESP may be a very poor local optimum. Therefore, JESP often runs using multiple restarts with different random initial policies in order to find one good local optimum solution. Another disadvantage is that JESP works only for finite-horizon Dec-POMDPs since policies are represented using policy trees.

2.6 Conclusion

In this chapter, we mainly introduced three general types of Markov models, MDPs, POMDPs, and Dec-POMDPs, respectively. To sum up, we have:

- MDP: the most basic Markov decision-making formalism for single-agent settings. In an MDP, the agent faces an uncertain environment but can directly observe the environment state;
- POMDP: the POMDP extends MDP in which the agent only observes partial or noisy information about the environment;
- Dec-POMDP: an extension of the POMDP to multi-agent settings where each agent has its own actions and observations;

Moreover, in the multi-agent setting, if we can control all agents' actions and access their observations, then we can model this problem as an MPOMDP, which is essentially a POMDP with joint action and joint observation spaces.

Regarding the solving algorithms, one can solve MDPs optimally with dynamic programming methods, including value iteration and policy iteration algorithms. Then, using generative models (simulators), a sampling-based method such as MCTS can be applied to find approximate solutions. On the other hand, for solving POMDPs, the first type of modern solvers are based on point-based value iteration, including SARSOP and HSVI; the second type is the sampling-based method such as POMCP, which is an extension of MCTS for POMDPs. We also introduced the JESP algorithm for solving Dec-POMDPs by finding Nash equilibrium solutions, which guarantees to converge to a local optimum.

In this thesis, the POMDP and Dec-POMDP formalisms will be used in our contributions to RRC and HRC (see Chapters 4 and 6).

3

Related Work on Multi-Agent Decision Making

3.1 Dec-POMDPs' overview

One primary interest in this thesis is multi-agent decision making, and Dec-POMDP is the most general framework among all Markov decision models, where multiple agents optimize a shared performance criterion under uncertainties due to the environment's dynamics and the agents' partial and noisy observations. However, as explained in Section 2.4, solving a Dec-POMDP is very difficult. Even for a finite-horizon Dec-POMDP, the solving process has been proven to be NEXP in the worst case [Bernstein et al., 2002] and solving an infinite-horizon Dec-POMDP is undecidable [Oliehoek and Amato, 2016, Madani et al., 2003]. A key difference compared with POMDPs is that the utility of one agent's policy is not just linked to the dynamics of the environment but also depends on the interactions with other agents and their policies.

There are various techniques for solving Dec-POMDPs, and one can classify algorithms according to several dimensions:

- finite vs infinite temporal horizon;
- explicit vs generative model (simulator);
- reasoning about all agents simultaneously vs one agent at a time;
- types of approach such value-based, policy search, Bayesian inference, and finding Nash equilibria.

In this chapter, we present a selection of representative techniques for solving Dec-POMDPs organized as follows:

- policy search algorithms that directly search in the joint policy space are presented in Section 3.2;
- methods based on transforming the Dec-POMDP into a (PO)MDP are presented in Section 3.3;
- Section 3.4 gives the methods for searching Nash equilibrium solutions in Dec-POMDPs;
- Section 3.5 presents the sampling-based methods for Dec-POMDPs when explicit models are unavailable.

Then, we conclude in Section 3.6 by positioning our contributions in this domain. But of course, this chapter does not contain all related algorithms, a larger overview can be found in [Oliehoek and Amato, 2016].

3.2 Policy Search in Dec-POMDPs

In this section, we introduce approaches that explore the joint policy space by using heuristic search (MAA* [Szer et al., 2005]) or by optimizing the parameterized policies of all the agents (EM methods [Kumar and Zilberstein, 2010, Kumar et al., 2011, Pajarinen and Peltonen, 2011]). More precisely, MAA* solves finite-horizon Dec-POMDPs by transforming the Dec-POMDP into a deterministic shortest-path problem, and EM methods focus on the optimization conducted on fixed-size finite-state controllers (FSCs) for each agent for infinite-horizon setting.

3.2.1 Multi-Agent A*

Multi-Agent A* (MAA*) is an optimal heuristic search algorithm for solving finite-horizon Dec-POMDPs. It turns the finite-horizon Dec-POMDP planning into a deterministic shortest-path problem before solving it using an A* search. In MAA*, a joint policy δ is presented as a vector of individual policy trees. Given a horizon t, a joint policy is defined as $\delta^t = (q_1^t, \ldots, q_n^t)$ and q_i^t is the policy tree with depth t for agent i. Assuming the goal is to solve a Dec-POMDP with horizon T, then a completion Δ^{T-t} is defined as a set of possible policy trees with depth $T-t$ that can be attached to the leaf node of an executed policy tree δ^t. As A*, MAA* defines a *value estimator* F as:

$$F^T\left(s_0, \delta^t\right) \stackrel{\text{def}}{=} V\left(s_0, \delta^t\right) + H^{T-t}\left(s_0, \delta^t\right), \tag{3.1}$$

where $V\left(s_0, \delta^t\right)$ is the value from start state s_0 with a given joint policy δ^t, and H is the heuristic function which gives an overestimation of the actual value of δ^t's completion Δ^{T-t}.

To fit MAA* in a standard A* scheme, each joint policy δ^t is treated as a node. Then, in each iteration, MAA* computes F^T and selects the node with the highest estimated value to expand. The expansion process consists in adding all possible child nodes δ^{t+1} (which correspond to all possible combinations of all possible expansions of each individual policy tree) to an open list D and removing the visited parent node δ^t from that open list. The search is finished when its most promising element has a F-value smaller (in the maximizing case) than the best complete solution found up to now:

$$\exists \delta^T \in D, \forall \delta \in D, \quad F^T(s_0, \delta) \leqslant F^T(s_0, \delta^T) = V(s_0, \delta^T). \tag{3.2}$$

Thus, an optimal joint policy has been found.

In MAA*, the heuristic function H, which overestimates the future expected return (or underestimates the cost-to-go, as for any A* search) is a key ingredient. Three methods are provided for computing H in MAA*:

- H_{MDP}: the easiest way is to overestimate the actual value function by relaxing a Dec-POMDP to an MDP. In this case, it assumes that there is only one agent that controls the joint action and has access to the full state of the system.
- H_{POMDP}: a tighter estimation consists in relaxing a Dec-POMDP to a POMDP. In this case, all agents are seen as a single one who observes the joint observation and decides the joint action.

- H_{MAA*}: this is a heuristic method where another MAA* is used to estimate the δ^t's completion Δ^{T-t}, thus, leading to a recursive computation. Compared with H_{MDP} and H_{POMDP}, no relaxation of Dec-POMDPs has been made by using H_{MAA*}. Therefore, it is assumed that this heuristic method can give the tightest estimation, close to the actual value, with enough resources.

Experiments confirm that the H_{MAA*} gives the tightest estimation, and a recursive MAA* search performs better than other heuristics, however, it suffers from heavy computations. A critical issue of MAA* lies in its expansion process, which induces a combinatorial explosion with the increase of T. Thus, MAA* is only able to solve small and finite-horizon Dec-POMDPs.

3.2.2 Expectation-Maximization (EM) for Dec-POMDPs

Before introducing the Expectation-Maximization (EM) algorithm, we need to know that a single-agent planning problem can be transformed into a probabilistic inference problem.

Attias first proposes such a transformation for some "special" MDPs and POMDPs [Attias, 2003]. Here, we use an MDP case to illustrate how it works. In an MDP, a transition function is denoted as $Pr(s^{t+1}|s^t, a^t)$, and the agent needs to select an action sequence $a_{1:N} = (a_1, \ldots, a_T)$ that ensures being in the goal state $s^{N+1} = g$ after N time steps. Now, let's define a a probability distribution over all sequences of states and actions as follow:

$$Pr\left(s_{2:N+1}, a_{1:N}\right) = \prod_{n=2}^{N} Pr\left(s_n \mid s_{n-1}, a_{n-1}\right) Pr\left(a_n \mid a_{n-1}\right) \cdot Pr\left(a_1\right) Pr\left(s_{N+1} \mid s_N, a_N\right),$$

where we have:

$$Pr\left(s_n = s' \mid s_{n-1} = s, a_{n-1} = a\right) = \lambda_{s'sa};$$
$$Pr\left(a_n = a' \mid a_{n-1} = a\right) = \eta_{a'a};$$

for $n = 2 : N + 1$, and $Pr\left(a_1 = a\right) = \eta_a$ for $n = 1$. Therefore, such an inference model has a parameter set $\theta = \{\lambda_{s'sa}, \eta_{a'a}, \eta_a\}$. Then, a posterior distribution over an action sequence is defined as:

$$Pr\left(a_{1:N} \mid s_1 = s_{init}, s_{N+1} = g\right), \tag{3.3}$$

where s_{init} is the initial state. The planning problem is thus turned into maximizing the posterior over actions:

$$\hat{a}_{1:N} = \arg\max_{a_{1:N}} p\left(a_{1:N} \mid s_1 = i, s_{N+1} = g\right). \tag{3.4}$$

However, one should note that Attias's formulation is not built for standard MDPs or POMDPs, because it assumes:

1. the solution is constrained to be a fixed sequence of actions; and
2. the objective is to maximize the probability of reaching a goal state (no reward function) after a fixed number of time steps.

A more generic approach for transforming standard MDPs and POMDPs into probabilistic inference can be found in Toussaint et al.'s work [Toussaint et al., 2006].

The Expectation-Maximization (EM) algorithm is a well-known method in machine learning, especially for data mining applications [Jung et al., 2014]. EM is usually used to estimate parameters in statistical models, *i.e.*, a probability distribution, where the model depends on unobserved

variables. Given the model's parameters θ, observed data y, and latent data z (unobserved data), the expectation step (E-step) is defined as

$$Q(\theta, \theta^i) = \int_z \log(Pr(\theta|z, y))Pr(z|\theta^i, y)dz.$$

This is the expectation of the log-likelihood function of θ with respect to the conditional distribution of z given the observed data y and current estimates of the parameter θ^i. Then, in the maximization step (M-step), one maximizes the Q function concerning θ to update θ^i:

$$\theta^{i+1} = \arg\max_{\theta} Q(\theta, \theta^i).$$

A diagram of EM is given in Figure 3.1. First, EM initializes parameter values and assumes the current observed data comes from a specific model (a probability distribution). Then, EM seeks to find the maximum likelihood estimate (MLE) by iteratively applying the E and M steps. EM checks if the assumed model's parameters θ^i have converged (by testing if the change of parameter value is smaller than a threshold); the algorithm stops if there is a convergence, otherwise the process repeats. The convergence in the EM approach is guaranteed since the data likelihood function is monotonically increased in each iteration. However, EM only guarantees that the converging point has zero gradients with respect to the parameters. Therefore, it may stuck into local optima or saddle points.

With those two necessary steps described above, a new class of Dec-POMDPs algorithms has been proposed [Kumar and Zilberstein, 2010, Kumar et al., 2011, Pajarinen and Peltonen, 2011] that relies on transforming the Dec-POMDP problem into an equivalent mixture of dynamic Bayes networks (DBNs) [Toussaint et al., 2006], where each agent's policy is a fixed-size FSC. Its parameters are optimized iteratively through the EM approach, which gives local optimal solutions.

3.2.3 Conclusion

This section introduced various policy searching methods, including MAA* and a series of EM algorithms for solving Dec-POMDPs. For MAA*, the main advantage is its standard A* scheme, where an optimal solution is guaranteed. However, the main issue is its scalability. Since each node in MAA*'s decision tree is a joint policy up to the current depth t, the complexity grows exponentially with horizons. Therefore, MAA* is only suitable for small finite-horizon Dec-POMDPs.

On the other hand, by recasting the planning problem into a probabilistic inference problem, the EM approach provides a new class of Dec-POMDP solvers where inference tools could be applied. In general, both finite-horizon and infinite-horizon Dec-POMDPs can be solved within the EM approach. One drawback is that the EM approach is only guaranteed to converge to local optima.

3.3 Transforming Dec-POMDPs into (PO)MDPs

In this section, we introduce a type of approach for solving Dec-POMDPs consisting of transforming the Dec-POMDP problem into a Markov decision process whose state is a sufficient statistics of the process, as in FB-HSVI [Dibangoye et al., 2016] and PBVI-BB [MacDermed and Isbell, 2013].

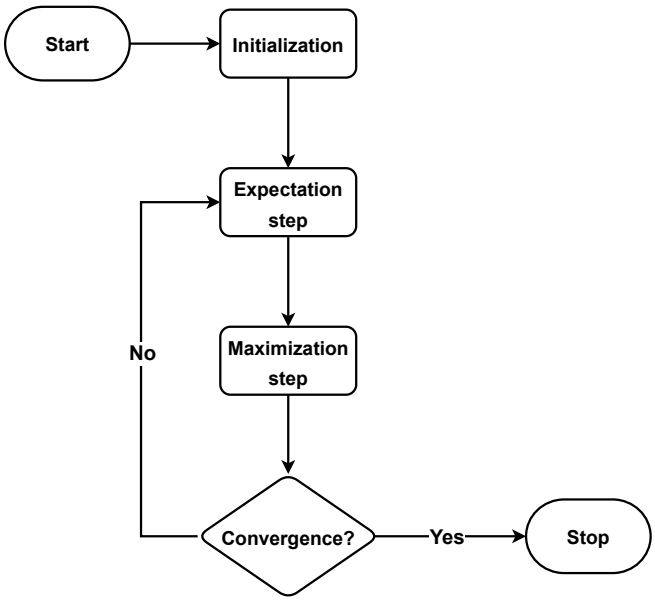

Figure 3.1: An illustration of the Expectation-Maximization (EM) algorithm [Do and Batzoglou, 2008].

3.3.1 Feature Based HSVI

Dibangoye et al. propose a method called *Feature based HSVI* (FB-HSVI) for solving finite-horizon Dec-POMDPs. A starting point of FB-HSVI is to transform the Dec-POMDP problem into a continuous MDP using sufficient statistics for planning as a state. This *occupancy state* η^t is a probability distribution over all Dec-POMDP states and joint action-observation histories of all agents. The action space of this MDP is a set of joint decision rules, each joint decision rule is denoted as d^t, which is an N-tuple of individual decision rules $(d_1^t, \ldots, d_N^t)$, one individual decision rule for each agent. Each individual decision rule d_i^t is a mapping from t-step individual histories to individual actions of agent i. Since the occupancy-state is a distribution over states and histories, the next occupancy-state η^{t+1} is deterministically controlled by the current occupancy-state η^t and joint decision rule d^t where $\eta^{t+1} = F(\eta^t, d^t)$. This being said, this MDP has deterministic dynamics. Thus, solving the Dec-POMDP at hand is equivalent to solving this occupancy-state MDP, where the optimality equation is defined as follows:

$$V_t^*(\eta^t) = \max_{d^t} \left[R(\eta^t, d^t) + \gamma V_{t+1}^* F(\eta^t, d^t) \right]. \tag{3.5}$$

Note that this transformation also amounts to deriving a non-observable MDP (NOMDP) [Boutilier et al., 1999], which is a special case of POMDP where the observation space is empty. This NOMDP has the following features:

1. the state comprises the Dec-POMDP state plus each agent's action-observation history (AOH);

2. the horizon is finite, and the state space depends on the time step;

3. the dynamics are deterministic;

4. actions are selected by solving a constraint satisfaction problem (CSP);

5. compression techniques are used that merge equivalent AOHs.

Therefore, a POMDP solver (HSVI) can be applied to solve this continuous MDP, which uses heuristic initializations to initialize a lower and upper bound to direct the search. The lower and upper bounds will converge in the limit, but in practice, the algorithm stops when a precision (a gap) ϵ is reached.

3.3.2 PBVI-BB

Point Based Value Iteration with Optimal Belief Compression for Dec-POMDPs (PBVI-BB) [MacDermed and Isbell, 2013] can be seen as a variant of FB-HSVI in the infinite-horizon setting. Like FB-HSVI, one crucial process is transforming the Dec-POMDP into an (PO)MDP with appropriate state and action spaces. To do so, PBVI-BB assumes that each agent has a fixed maximum number of beliefs at each time step. With this assumption, PBVI-BB transforms a Dec-POMDP to a NOMDP where:

1. an agent's individual belief is akin to an internal state;

2. an agent's individual policy thus amounts to an FSC (with bounded size), as the agent alternatively picks an action and the next internal state;

3. the NOMDP's belief is over the current system state and the agents' internal states;

4. in this POMDP, V^* is still PWLC, which allows applying modern POMDP solvers.

In their case, this NOMDP (they call it *Belief-POMDP*) is solved using a POMDP solver Perseus [Spaan and Vlassis, 2005] (a variant of PBVI, see Algorithm 4 on page 20).

Conclusion

To summarize, both FB-HSVI and PBVI-BB transform a Dec-POMDP problem into a (PO)MDP problem. This transformation preserves optimal solutions and allows adapting point-based POMDP solvers (HSVI and Persus). In practice, both methods can be used for infinite-horizon Dec-POMDPs. To that end, FB-HSVI approximate infinite-horizon using finite horizons within an error bound [Dibangoye et al., 2014], and ϵ-optimal solutions. Compared with FB-HSVI, PBVI-BB has weaker properties due to (1) the choice of FSCs, and (2) using Perseus instead of HSVI (a variant of HSVI-BB would be ϵ-optimal for a choice of FSC size). But one potential issue is their scalability when facing large problems because of the combinatorial explosion. In the next section, we present algorithms to avoid this issue by iteratively only optimizing one agent's policy considering other agents' policies are fixed.

3.4 Finding Nash Equilibria in Dec-POMDPs

This section reviews related works for solving Dec-POMDPs by finding Nash equilibrium solutions where each agent's policy is the best response to others, and no one could improve the global performance by modifying only its own policy.

3.4.1 JESP

As we described in Section 2.5, the Joint Equilibrium-based search for policies (JESP) algorithm was first proposed to search Nash equilibria for finite-horizon Dec-POMDPs. In JESP, all agents' policies are randomly initialized using policy trees. Then, JESP iterates over each agent i to improve its policy π_i while fixing other agents' policies $\pi_{\neq i}$. This improvement is made by solving an extended state POMDP built for agent i, where an extended state $e^t = \langle s^t, \vec{w}^t_{\neq i} \rangle$ consists of the Dec-POMDP state and other agents' observation histories. By solving this POMDP, the agent i's policy $\pi_i^{'}$ is now a best-response to $\pi_{\neq i}$, and the value of the joint policy is at least as good as the previous one (if no improvement, $\pi_i^{'} = \pi_i$). Therefore, in each iteration of JESP, its value monotonically increases and converges to a Nash equilibrium.

However, JESP uses policy trees and thus is limited to the finite-horizon setting. Another drawback is that the JESP scheme has no guarantees of globally optimal solutions, and some random initial policies may converge to very poor local optimum. Therefore, JESP usually needs multiple restarts to find good local optima.

3.4.2 Bounded Policy Iteration for Dec-POMDPs (Dec-BPI)

For the infinite-horizon setting, Dec-BPI [Bernstein et al., 2005] is a Dec-POMDP extension of bounded policy iteration (BPI) for POMDPs [Poupart and Boutilier, 2003] and provides a locally optimal solution. In Dec-BPI, agent policies are represented as FSCs. Dec-BPI iterates over the nodes of each agent's FSC and tries to improve one agent's FSC assuming the other agents' policies are fixed. In each node of local policies, linear programming is used to search a new probability distribution over actions and transitions if the value will increase for any initial state and any initial node of other agents' FSCs. Each agent improves its FSC in turn and ends when no improvement can be made on any agent. An improvement of Dec-BPI is optimizing each agent FSC using a non-linear program in each iteration [Amato et al., 2012].

In Dec-BPI, with each iteration's improvement, the value of the joint policy is at least as high as the last iteration. This monotonic improvement ensures that Dec-BPI converges, but with no optimality guarantee. The reason is that, in each iteration of Dec-BPI, only improving one agent's FSC may lead to a sub-optimal Nash equilibrium solution in which each agent's policy is the best response to others' fixed policies (which is similar to JESP [Nair et al., 2003] approach described in Section 3.4.1). However, one should also note that this best-response policy during the improvement can only be obtained in the best case, which is not true in practice. Because in each iteration of Dec-BPI, the optimization of an FSC does not find the optimal parameters. Thus, even the Nash equilibrium solution is not guaranteed by Dec-BPI.

Conclusion

In this section, we introduce two algorithms, JESP and Dec-BPI, for solving Dec-POMDPs, and both methods try to find Nash equilibrium solutions. JESP guarantees to converge to Nash equilibria but can only be applied to finite-horizon Dec-POMDPs since it uses policy trees. On the other hand, Dec-BPI uses FSCs to represent agents' policies and solves infinite-horizon Dec-POMDPs, but without a guarantee of finding Nash-equilibrium solutions. Last but not least, in Dec-BPI, the improvement in each iteration is made for all initial state distributions. This being said, Dec-BPI finds a local optima solution for any starting belief b_0.

3.5 Solving Dec-POMDPs with a generative model

Previous sections introduced various methods for solving Dec-POMDPs when explicit models are available. In other words, this requires the exact dynamics of the Dec-POMDP problem we are addressing. However, we often do not have such an explicit model for large problems or real applications where we do not know the transition and observation probabilities. Usually, in this case, only a black-box simulator (also called a generative model) G is available:

$$s', \vec{o}, r \leftarrow G(s, \vec{a}). \tag{3.6}$$

The inputs of G are the current state s and joint action $\vec{a}$ of all agents. Then, it outputs the next state s', the observations $\vec{o}$ for all agents, and the instant reward r. This section introduces several sampling-based methods for Dec-POMDPs requiring only a generative model G.

3.5.1 Monte-Carlo Expectation Maximization (MCEM) based algorithms for solving Dec-POMDPs

MCEM for Dec-POMDPs [Wu et al., 2013] is a model-free version of the EM approach we introduced in Section 3.2.2. Like the EM methods for solving Dec-POMDPs, MCEM based approach optimizes θ, which is the parameter of each agent's FSC with a fixed size. For the EM algorithm, one relies on the explicit model to compute Equation (3.3). In the MCEM approach, this computation is approximated by a Monte-Carlo integration as follows:

$$Q_{i+1}(\theta, \theta^{i+1}) = \frac{1}{m} \sum_{j=1}^{m} \log(Pr(\theta | z^j, y)). \tag{3.7}$$

The samples $\{z^1, \ldots, z^m\}$ are generated from the current approximation to the conditional distribution $Pr(z|\theta^i, y)$. Then, the M-step consists in maximizing Equation (3.7), and the conditional distribution is updated with the new maximizer. The algorithm iterates until convergence to a local optimal or a saddle point.

One improvement is constructing agents' FSCs with variable sizes as in Dec-SBPR (Stick-Breaking Policy learning for Dec-POMDPs) [Liu et al., 2015] to eliminate two side effects of fixed-size FSCs: an overly small number won't be able to represent good policies, and an overly large number will cause an over-fitting problem and slow convergence.

3.5.2 Occupancy-State SARSA (oSARSA):

Similar to FB-HSVI introduced in Section 3.3.1, oSARSA [Dibangoye and Buffet, 2018] is also interested in recasting Dec-POMDPs into occupancy-state MDPs. Instead of using policies, oSARSA computes plans which are linear functions over occupancy states and joint decision rules since it is simple to store and update while having an optimal performance for Dec-POMDPs. Compared with FB-HSVI, greedy or soft maximization is replaced by mixed-integer linear programming to save resources and scale up better. Given enough resources and an accurate estimation of the occupancy states, oSARSA can converge to a near-optimal plan for finite-horizon Dec-POMDPs.

3.6 Conclusion

To sum up, in this chapter, we introduced various algorithms for solving Dec-POMDPs, which can be categorized as:

- algorithms that optimize the policies of all the agents simultaneously, including MAA*, FB-HSVI, and PBVI-BB. For MAA*, it finds an optimal solution in a standard A* scheme for finite-horizon Dec-POMDPs. On the other hand, FB-HSVI and PBVI-BB transform a Dec-POMDP to a (PO)MDP and solve it with POMDPs techniques. Both FB-HSVI and PBVI-BB can be used for infinite-horizon Dec-POMDPs, and FB-HSVI gives ϵ-optimal solutions by approximating infinite-horizon using finite horizons within an error bound. Moreover, oSARSA extends the FB-HSVI to the generative model setting where an explicit Dec-POMDP is unavailable. However, all those methods may face a combinatorial explosion issue when facing large problems.
- algorithms that alternate between agents, including JESP and Dec-BPI, which avoid this combinatorial explosion of all agents' policies by optimizing each agent's policy in turn while considering others' policies are fixed. But its associated drawback is that such methods can only find Nash equilibria at best. In this category, JESP is limited to finite horizons, and Dec-BPI can be used for infinite-horizon Dec-POMDPs with any starting state distribution.
- algorithms based on the EM approach transform the Dec-POMDP planning problem into a probabilistic inference. Each agent's policy is represented as an FSC, and its parameters are optimized using the EM algorithm. EM-based methods can be used for infinite-horizon settings, and an adaption of Monte-Carlo integration extends their usage with generative models. However, EM methods only guarantee to converge to local optima or even worse saddle points.

Moreover, we conclude those representative Dec-POMDP algorithms in Figure 3.2, to our knowledge, the JESP scheme has been introduced to neither infinite-horizon nor simulator-based settings. Therefore, in this thesis, two of our main contributions (see Chapter 4) consist of: 1. a first algorithm that extends the JESP approach to the infinite-horizon setting called Inf-JESP; 2. an extension of Inf-JESP that works with generative models (MC-JESP).

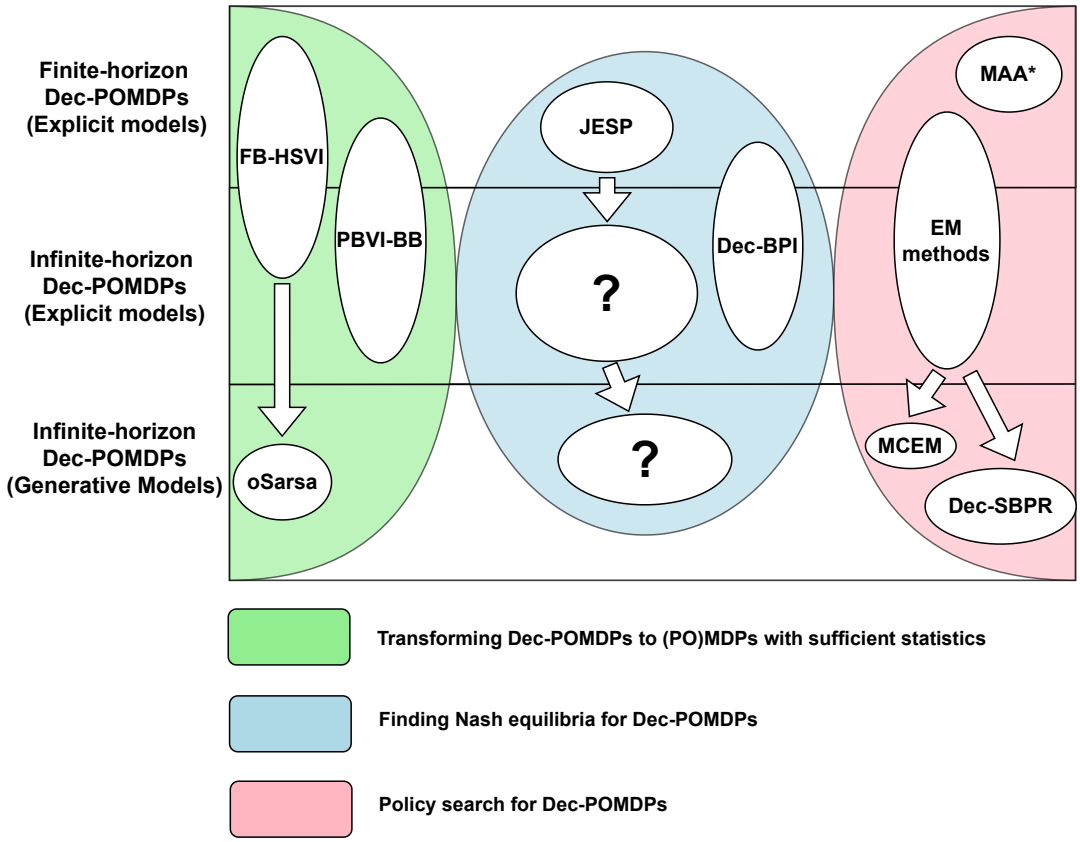

Figure 3.2: An overview of representative algorithms for Dec-POMDPs: The area colored with light green gives the methods based on transforming Dec-POMDPs to (PO)MDPs with sufficient statistics; The light pink area contains algorithms that directly search in the joint policy space; Algorithms that find Nash equilibrium solutions are presented inside the light blue area. All those methods are also categorized with the temporal horizons (finite or infinite) and model types (explicit model or generative model).

4

Searching for an Equilibrium in an Infinite-Horizon Multi-agent Problem

As we discussed in Chapter 2, a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) represents a multi-agent sequential decision-making problem where the objective is to derive the policies of all agents so that their decentralized execution maximizes the average cumulative reward. Each agent's policy is a mapping from his individual history (the sequence of his executed actions and received observations up to now) to individual actions. However, solving a Dec-POMDP is difficult and is proved to be NEXP complete in the worst case [Bernstein et al., 2002]. The main difficulties when solving a Dec-POMDP lie in two aspects: first, the state of the environment evolves according to all agents' actions; and, second, each agent picks his actions based on his own action-observation history, making it difficult for agents to coordinate. Therefore, all policies are interdependent, which means each agent's optimal decision depends on others' possible histories and future policies.

In this thesis, most of our contributions rely on computing the best response to other agents. For instance, in our contributions to compute robust robot policies for HRC in Chapter 6, the robot policy is the best response to possible human behaviors. For solving Dec-POMDPs, we are interested in finding a Nash equilibrium solution consisting of each agent's policy being a best response to the other agents' policies. To that end, we propose to solve Dec-POMDPs through a JESP approach (Joint Equilibrium-Based Search for Policies) [Nair et al., 2003], which searches for Nash equilibrium solutions by optimizing each agent's policy one after the other while fixing the other's policies, until convergence (see Figure 4.1). However, JESP has two weaknesses:

1. in JESP, policies are represented by trees; therefore, this algorithm addresses only finite-horizon Dec-POMDPs;

2. the Nash equilibria solutions obtained by JESP are local but not global optima.

This thesis addresses JESP's first drawback by extending it to the infinite-horizon setting. Our starting point is to rely not on policy trees but on finite-state controllers (FSCs). Then, we demonstrate how to derive the (best-response) POMDP faced by an agent while other agents' policies are fixed. From there, we solve this POMDP to find individual best-response policies (in FSCs), and integrate this step into a JESP scheme.

We propose two novel algorithms in this chapter based on the above ideas. The first algorithm (Inf-JESP), presented in Section 4.1, solves Dec-POMDPs with explicit models. And the second method (MC-JESP), shown in Section 4.2, assumes that generative models of Dec-POMDPs (*i.e.*, simulators) are available. Experiments on state-of-the-art benchmark problems have been

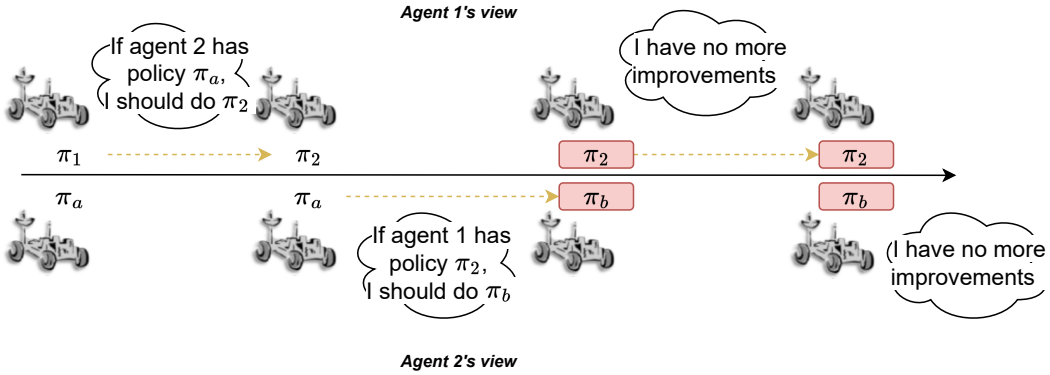

Figure 4.1: JESP's iterative optimization process for a 2-agent problem, starting with policies π_1 and π_a

conducted to evaluate our algorithms in Section 4.3. Finally, we conclude this chapter and discuss perspectives in Section 4.4.

4.1 Infinite-Horizon JESP

The JESP algorithm finds Nash equilibrium solutions by iteratively building one agent's best response to other agents' fixed policies. JESP (with its policy trees) is meant for a finite-horizon setting. Although one can approximate an infinite-horizon Dec-POMDP as a finite-horizon one, this approximation usually needs a long horizon (see Equation (2.18)). Thus, it requires overly large policy trees and extended state spaces for the POMDP built in each iteration of JESP (see Section 2.5). Also, no action is prescribed when going beyond this approximated long horizon (because any action can be applied), which the end-user may not appreciate. Therefore, those drawbacks make JESP unsuitable for solving infinite-horizon problems. To address this issue, we propose solving infinite-horizon Dec-POMDPs using finite state controller (FSC) policy representations in a JESP scheme. FSCs are meant to represent infinite-horizon policies, and we try to control their size in our work. We call this new algorithm *Inf-JESP*.

Inf-JESP's main algorithm is introduced in Section 4.1.1, and we explain how to combine FSCs with a Dec-POMDP to generate the POMDP required at each iteration of Inf-JESP in Section 4.1.2. Then, in order to use state-of-the-art point-based POMDP solvers, Section 4.1.3 proposes an algorithm to generate an agent's FSC from a set of α-vectors. Moreover, heuristic initialization methods are provided in Section 4.1.4.

4.1.1 Inf-JESP Main Algorithm

Inf-JESP relies on a local search procedure, which is typically restarted multiple times with different random initializations to converge to different local optima. This local search, presented in this section, relies iteratively on (i) defining the best-response POMDP for each agent based on the policies of other agents (see Section 4.1.2), and (ii) solving it to extract and evaluate the associated FSC (see Section 4.1.3). (Note that we also provide experiments with a non-random initialization of Inf-JESP, which is described in Section 4.1.4)

As described in Algorithm 6, each agent's policy is represented as an FSC. To control the computational cost at each iteration, the size of solution FSCs is bounded by a parameter $K \in \mathbb{N}^*$.

The local search thus starts with $|\mathcal{I}|$ randomly generated K-FSCs in vector *fsc* (line 2). Then, it loops over the agents, each iteration attempting to improve an agent i's policy by finding (line 7) a K-FSC fsc'_i that is a best response to the current (fixed) FSCs $fsc_{\neq i}$ of the $|\mathcal{I}|-1$ other agents (denoted $\neq i$). Line 8 relies on Equation (2.21) (page 19) to evaluate $\langle fsc'_i, fsc_{\neq i}\rangle$ at b_0, unless the POMDP solver at line 7 provides this information. Then, in line 10, if an improved solution has been found, fsc'_i replaces fsc_i in *fsc*. The process stops if the number of consecutive iterations without improvement, $\#ni$, reaches $|\mathcal{I}|$.

Algorithm 6: ∞-horizon JESP

```
1  [Input:] K: FSC size | fsc: initial FSCs
2  Fct LocalSearch(K, fsc ≝ ⟨fsc_1, ..., fsc_|I|⟩)
3      v_bestL ← eval(fsc)
4      #ni ← 0                                   // #(iterations w/o improvement)
5      i ← 1                                     // Id of current agent
6      repeat
7          fsc'_i ← Solve2FSC(fsc_≠i, K)
8          v ← eval(fsc'_i, fsc_≠i)
9          if v > v_bestL then
10             fsc_i ← fsc'_i                    // replace the current agent i's policy with an improved one
11             v_bestL ← v
12             #ni ← 0
13         else
14             #ni ← #ni + 1
15         i ← (i mod |I|) + 1
16     until #ni = |I|                           // looped over agents w/o improving
17     return ⟨fsc, v_bestL⟩
```

Properties For agent i, each of its K nodes is attached to an action in $\mathcal{A}_i$, and each of its $K \times |\Omega|$ edges is attached to a node, so that the number of *deterministic* FSCs $|\mathcal{FSC}_i|$ is at most $|\mathcal{A}_i|^K \cdot K^{K\times|\Omega|}$.[2] Thus, assuming an optimal POMDP solver leads to the following properties.

Proposition 1 *Inf-JESP's local search converges in finitely many iterations to a Nash equilibrium.*

Proof 1 *The search only accepts increasingly better solutions so that the number of iterations (over all agents) is, at most, the number of possible solutions:* $|\mathcal{FSC}| = \prod_i |\mathcal{FSC}_i|$.

The search stops when all agents have no more improvements by only modifying their own policies. This being said, each agent's FSC is a best response to the other agents' FSCs, i.e.*, in a Nash equilibrium.*

While these equilibria are only local optima, allowing for infinitely many random restarts guarantees to converge to a global optimum with probability 1. Of course, the set of Nash

[2]The exact number is smaller due to symmetries and because, in some FSCs, not all internal nodes are reachable.

equilibria depends on the set of policies at hand, thus on the parameter K in our setting. Increasing K just allows for more policies, and thus possibly better Nash equilibria.

In practice (see Section 4.1.2), we use a sub-optimal POMDP solver in **Solve2FSC** (line 7) in which FSC sizes are implicitly bounded (K is not actually used) because the solving time is bounded. If this POMDP solver returns a solution fsc_i' worse than fsc_i, it will be ignored, so that monotonic improvements are preserved, and the search still necessarily terminates in finite time. Final solutions may be close to Nash equilibria if the POMDP solver returns ϵ-optimal solutions.

4.1.2 Building Best-Response POMDP for Agent i

As we introduced in Section 2.5, in each iteration, an extended state POMDP is built for agent i while considering that other agents' policies $\pi_{\neq i}$ are known and fixed. Solving this POMDP will derive a policy for agent i, which is the best response to other agents' fixed policies $\pi_{\neq i}$. In this chapter, we call this POMDP for agent i a *best-response POMDP* (BR-POMDP). In JESP's finite horizon setting, the state in agent i's best-response POMDP' is denoted $e^t \stackrel{\text{def}}{=} \langle s^t, \vec{\omega}^t_{\neq i} \rangle$, with s^t the current state and $\vec{\omega}^t_{\neq i}$ the observation histories of other agents. Agent i maintains a belief b^t_{JESP} over e^t, which is a sufficient statistic for planning as it allows predicting the system's evolution (including other agents) as well as future expected rewards. However, as we pointed out above, in infinite-horizon problems, the number of observation histories $\vec{\omega}^t_{\neq i}$ grows exponentially with time, making for an infinite state space. We thus consider using FSCs to represent other agents' policies so that agent i can reason about other agents' internal nodes instead of their histories. But then, an important question is raised: how to derive a valid BR-POMDP for agent i given the Dec-POMDP and other agents' FSCs?

In Inf-JESP, to construct a BR-POMDP for agent i in each iteration, we first need to decide which variables are contained in its extended state e^t. Then, the action and observation sets of this BR-POMDP should correspond to the action an observation sets of this agent i as defined in the Dec-POMDP model. Moreover, the choice we made for the extended state, along with agent i's actions and observations, has to guarantee that we get a proper POMDP that contains valid transition, observation, and reward functions. In this section, we present one possible way to construct this BR-POMDP, but we also provide other possible formalizations in appendix (see Section A.1). Here, we choose to represent the extended state $e^t \in \mathcal{E}$ as shown in Figure 4.2, *i.e.*, containing:

- s^t, the current state of the Dec-POMDP problem;
- $n^t_{\neq i} \equiv \langle n^t_1, \ldots, n^t_{i-1}, n^t_{i+1}, \ldots, n^t_n \rangle$, the nodes of the $|\mathcal{I}| - 1$ other agents' (agents $\neq i$) FSCs at the current time step, and;
- o^t_i, agent i's current observation.

We thus have $\mathcal{E} \stackrel{\text{def}}{=} \mathcal{S} \times N_{\neq i} \times \Omega_i$. Given an action a^t_i, the extended state $e^t \equiv \langle s^t, n^t_{\neq i}, o^t_i \rangle$ evolves according to the following steps:

1. each agent $j \neq i$ selects its action a_j based on its current node;
2. s^t transitions to s^{t+1} according to T under joint action $a^t \equiv \langle a^t_i, a^t_{\neq i} \rangle$; and
3. the FSC nodes of agents j (including i) evolve jointly according to their observations o^{t+1}_j, which can be inferred from s^{t+1} and the joint action a^t; these observations may not be independent from each other, so that the FSC nodes may not evolve independently from each other.

This design choice for e^t induces a POMDP because of the following properties:

- it induces a Markov process controlled by the action (see dynamics below);
- the observation o_i^t depends on a_i^t and e^{t+1}; and
- the reward depends on e^t and a_i^t.

Indeed, deriving the transition, observation, and reward functions for this POMDP (as detailed in Appendix Sec. A.1) leads to:

$$\begin{aligned}
T_e(e^t, a_i^t, e^{t+1}) &= Pr(e^{t+1}|e^t, a_i^t) \\
&= \sum_{o_{\neq i}^{t+1}} T(s^t, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}) \cdot \eta_{\neq i}(n_{\neq i}^t, o_{\neq i}^{t+1}, n_{\neq i}^{t+1}) \\
&\quad \cdot O(s^{t+1}, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, \langle o_{\neq i}^{t+1}, o_i^{t+1} \rangle), \\
O_e(a_i^t, e_i^{t+1}, o_i^{t+1}) &= Pr(o_i^{t+1}|a_i^t, e_i^{t+1}) \\
&= Pr(o_i^{t+1}|a_i^t, \langle s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}_i^{t+1} \rangle) \\
&= \mathbf{1}_{o_i^{t+1} = \tilde{o}_i^{t+1}}, \\
r_e(e^t, a_i^t) &= r(s^t, a_i^t, \psi_{\neq i}(n_{\neq i}^t)),
\end{aligned}$$

where $\eta_{\neq i}(n_{\neq i}^t, o_{\neq i}^{t+1}, n_{\neq i}^{t+1}) = \prod_{j \neq i} \eta(n_j^t, \tilde{o}_j^{t+1}, n_j^{t+1})$ and $\psi_{\neq i}(n_{\neq i}^t) = a_{\neq i}^t$.

Note that, if some of the state variables are fully observable, then the problem is more precisely a *Mixed Observability Markov Decision Processes* (MOMDP), and this mixed observability can be exploited to accelerate the solving process [Ong et al., 2009, Araya-López et al., 2010]. But, since this is not the priority in this thesis, we only consider standard POMDP solvers.

Moreover, we observe that some extended states will never be reached starting from the initial belief. We thus make $\mathcal{E}$ smaller through a *state elimination* process. More specifically, we first initialize $\mathcal{E}$ by computing all possible starting extended states given the initial belief b_0, and an open list is initialized with the same elements. Then, an extended state e^t is popped out from this open list and processed with all possible agent i's actions to gather possible s^{t+1}, $n_{\neq i}^{t+1}$, and o_i^{t+1}, which result in a list of next possible extended states $\{e^{t+1}\}$. If an extended state in $\{e^{t+1}\}$ does not exist in $\mathcal{E}$, we insert it to $\mathcal{E}$ and the open list. We repeat this process until the open list is empty. At the end of the process, $\mathcal{E}$ contains only the extended states that are reachable from the initial belief.

4.1.3 Building Agent i's FSC with bounded size

In Section 4.1.2, we presented how to build a best-response POMDP for agent i considering others' fixed policies. Then, once the BR-POMDP for agent i is built, we need a POMDP solver to obtain agent i's policy. As we introduced in Section 2.3.3, there are different types of POMDP solvers. For Inf-JESP, we take advantage of modern point-based solvers and choose SARSOP [Kurniawati et al., 2008] for solving the built POMDP, but other point-based approaches such as PBVI [Pineau et al., 2003] or HSVI [Smith and Simmons, 2004] are also applicable. On the other hand, the policy search methods [Hansen, 1997, Spaan and Vlassis, 2005] can directly give an FSC that can be used to build the next BR-POMDP, but they are usually less efficient than state-of-the-art point-based solvers.

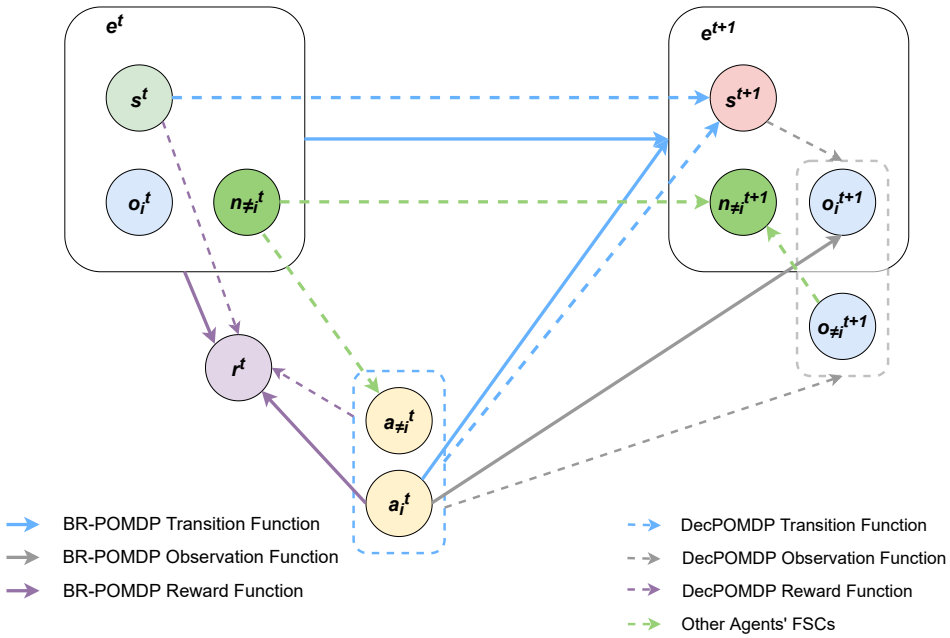

Figure 4.2: An illustration of the best-response POMDP for agent i based on the Dec-POMDP model and other agents' fixed FSCs.

Algorithm Description In Section 2.3.2 (page 19), we mentioned that one could transform a set of α-vectors Γ to an FSC, and Grześ et al. proposed an approximate method *Alpha2FSC*. However, Alpha2FSC assumes each α-vector in Γ (the POMDP solution) comes with an associated belief. This assumption is not true in our work (with the POMDP solver SARSOP), and we need to compute a representative belief for each α-vector. Therefore, we propose a new transformation method as presented in Algorithm 7, to turn an α-vector set Γ into an FSC. In our method, each FSC node $n \stackrel{\text{def}}{=} \langle \alpha, a, b, w \rangle$ contains an α-vector α with its (sometimes omitted) action a, and a representative belief b (weighted average of the encountered beliefs mapped to this node) with its positive weight w. This weight tries to capture the importance of the attached node by approximating its visit frequency and is computed incrementally when building the FSC.

More specifically, Algorithm 7 begins by a start node n_0 from b_0 and Γ with $w = 1$ (line 2), and adds it to both the new FSC (N) and an open list (G). Then, each n in G is processed (in fifo order) with each observation o_i as follows (0-probability observations induce a self-loop (line 22)). Lines 10–12 compute the updated belief $b_{a_i}^{o_i}$, and the associated w' and α_i. The node $n' \in N$ that contains α_i is extracted if it exists and $b_{a_i}^{o_i}$ is merged into its representative belief (lines 18–20), otherwise a new node $n' \stackrel{\text{def}}{=} \langle \alpha_i, b_{a_i}^{o_i}, w' \rangle$ is created (line 14) and added to both N and G. An edge $n \to n'$, labeled with o_i, is then added (line 23).

Note that our method (as Alpha2FSC) returns an FSC that approximates the policy induced by the α-vectors set Γ (which might need a possibly infinite FSC to represent). This approximation leads to an FSC result that may be better or worse compared with Γ.

About Self-Loops As in Grześ et al.'s work, self-loops are added when an impossible observation $Pr(o_i|b, a_i) = 0$ is received. This may happen because, when building the FSC, each node n_i is associated to a tuple $\langle \alpha_i, b_i \rangle$, and outgoing edges from n_i will be created only for observa-

tions that have non-zero probability in b_i. Yet, during execution, n_i may be reached while in a different belief b' from which a_i (the action attached to α_i, thus n_i) may induce "unexpected" observations.[3] Adding self-loops is a way to equip the agent with a default strategy when such an unexpected observation occurs.

Differences with Alpha2FSC There are three differences in our method compared with Grześ et al.'s work: First, in [Grześ et al., 2015], each input α-vector comes with an associated belief, while we instead try to compute a belief representative of the reachable beliefs "attached" to the vector. Second, in [Grześ et al., 2015], each α-vector induces an FSC node ($|N| = |\Gamma|$), while we only consider α-vectors reachable by the algorithm from b_0. Third, in our JESP setting, we have another reason for adding self-loops. Indeed, each agent i's FSC is obtained considering fixed agent $\neq i$'s FSCs; yet, changing other agents' FSCs will induce a new POMDP from i's point of view, potentially with new possible observations when in certain nodes of fsc_i.

Algorithm 7: Extract FSC $\langle N, \eta, \psi \rangle$ from set Γ

```
[Input:] Γ: α-vector set
Start node n_0 ← node(⟨arg max_{α∈Γ}(α · b_0), b_0, 1⟩)
N ← {n_0}
G.pushback(n_0)
while |G| > 0 do
    n ← G.popfront()
    ⟨b, a_i, w⟩ ← ⟨n.b, ψ(n), n.w⟩
    forall o_i ∈ Ω_i do
        if Pr(o_i|b, a_i) > 0 then
            b^{o_i}_{a_i} ← beliefUpdate(b, a_i, o_i)
            w' ← w · Pr(o_i|b, a_i)
            α_i ← argmax_{α∈Γ}(α · b^{o_i}_{a_i})
            if α_i ∉ N then
                n' ← node(⟨α_i, b^{o_i}_{a_i}, w'⟩)
                N ← N ∪ {n'}
                G.pushback(n')
            else
                n' ← N(α_i)
                n'.b ← n'.w/(n'.w+w') · n'.b + w'/(n'.w+w') · b^{o_i}_{a_i}
                n'.w ← n'.w + w'
        else
            n' ← n
        η(n, o_i) ← n'
return ⟨N, η, ψ⟩
```

Note that this algorithm does not bound the resulting number of internal nodes explicitly. Instead, we rely on (i) SARSOP returning finitely many α-vectors, and (ii) the FSC extraction

[3] This will happen when some state s has zero-probability in b_i but not in b' and can induce this observation when performing a_i.

producing (significantly) less than $|\Gamma|$ internal nodes. But one can easily limit the FSC size with a given bound as presented in our second main contribution (see Algorithm 11, page 48).

Once fsc_i is obtained, the next step is to evaluate all agents' policies in the Dec-POMDP. To that end, it is sufficient to evaluate fsc_i in the corresponding best-response POMDP, since the latter combines in a single model the Dec-POMDP and the FSCs for agents $\neq i$. Here we used the FSC evaluation technique described in Section 2.3 on page 19, which computes an FSC's value function by solving a system of linear equations.

4.1.4 Heuristic Initialization

As previously mentioned, the initialization of Inf-JESP is important. Even if Inf-JESP improves the value of the joint FSC at each iteration, random initializations may often lead to poor local optima. Thus, we want to investigate if some non-random heuristic initializations allow to find good solutions quickly and reliably.

Our method relies on assuming public observations in the Dec-POMDP, so that we are facing a Multi-agent POMDP (MPOMDP) [Pynadath and Tambe, 2002], *i.e.*, a problem that is formally equivalent to a POMDP and thus solved using a POMDP solver. We extract individual FSCs from the resulting MPOMDP policy as detailed below, and use them to initialize Inf-JESP.

MPOMDP-based Stochastic Initial FSC (M-S) – Algorithm 14 extracts a policy fsc_i for agent i from an MPOMDP policy. This approach is similar to Algorithm 7, the main difference being that agents $\neq i$'s observations and actions should also be considered to compute a next belief. Yet, this is not actually feasible since agent i is not aware of them. To address this issue, given agent i, let us consider a current node $n = \langle \alpha, b \rangle$ and an individual observation o_i (*i.e.*, with non-zero occurrence probability). The MPOMDP solution specifies a joint action $a = \langle a_i, a_{\neq i} \rangle$ at b, a being the action attached to α. We arbitrarily assume that (i) each possible stochastic transition of the FSC (from n and coming with observation o_i) corresponds to a possible joint observation $o_{\neq i}$ of the other agents, (ii) such a transition occurs with probability $Pr(o_{\neq i} \mid b, a, o_i)$, and (iii) it "leads" to a new belief $b_a^{\langle o_i, o_{\neq i} \rangle}$ and thus a node n' attached to the dominating MPOMDP α-vector (α_i, *cf.* line 12) at this point. As only one FSC node should correspond to a given α-vector, a new n' attached to α_i is created only if necessary (lines 13, 14 and 18). As multiple joint observations $o_{\neq i}$ may lead to the same n', the corresponding transition probabilities are cumulated in $\eta(n, o_i, n')$ (line 19). 0-probability joint observations $o_{\neq i}$ given b, a and o_i are ignored. 0-probability individual observations o_i given b and a induce the creation of a self-loop (line 21). This self-loop allows the FSC to still be a valid policy against any possible policies of other agents, which may change in the next iteration.

Note that the resulting FSCs are stochastic, and some of them may never be improved on, so that, in this case, the solution returned by Inf-JESP may contain stochastic FSCs.

MPOMDP-based Deterministic Initial FSC (M-D) – A very simple variant of this method is to assume that the only possible transition from n under o_i corresponds to the most probable joint observation $o_{\neq i}$ of the other agents. In this case, lines 10 and 11 are modified as follows:

$$o_{\neq i} \leftarrow \arg\max_{o_{\neq i} \in \Omega_{\neq i}} Pr(o_{\neq i} | o_i, b, a). \tag{4.1}$$

The transition between nodes is then deterministic ($\eta(n, o_i, n') = 1$).

Algorithm 8: Extracting FSC_i for agent i from MPOMDP α-vector set Γ

```
1  [Input:] i: agent | Γ: MPOMDP α-vector set
2  Start node n_0 ← node(argmax_{α∈Γ} α · b_0, b_0)
3  N ← {n_0}
4  G.pushback(n_0)
5  while |G| > 0 do
6      n ← G.popfront()
7      (b, a) ← (n.b, ψ(n))
8      forall o_i ∈ Ω_i do
9          if Pr(o_i|b, a) > 0 then
10             forall o_{≠i} ∈ Ω_{≠i} do
11                 if Pr(o_{≠i}|o_i, b, a) > 0 then
12                     α_i ← argmax_α(b_a^{⟨o_i,o_{≠i}⟩}, Γ)
13                     if α_i ∉ N then
14                         n' ← node(α_i, b_a^{⟨o_i,o_{≠i}⟩})
15                         N ← N ∪ {n'}
16                         G.pushback(n')
17                     else
18                         n' ← N(α_i)
19                     η(n, o_i, n') ← η(n, o_i, n') + Pr(o_{≠i}|o_i, b, a)
20         else
21             η(n, o_i, n) ← 1
22 return ⟨N, η, ψ⟩
```

MPOMDP-based Average-Belief Initial FSC (M-A) – Algorithm 9 provides another way to extract a policy fsc_i for agent i from an MPOMDP policy. In this heuristic initialization, we assume that all agents share the same belief b and act according to the same joint action a. Then, the belief is updated by agent i by marginalizing agents $\neq i$'s possible observations to ignore them. This *one-sided-observation belief update* (line 10) is obtained as follows:

$$Pr(s'|o_i, \langle a_i, a_{\neq i}\rangle, b) = \frac{Pr(s', o_i, \langle a_i, a_{\neq i}\rangle, b)}{Pr(o_i, \langle a_i, a_{\neq i}\rangle, b)} \tag{4.2}$$

$$= \frac{\sum_s \sum_{o_{\neq i}} Pr(s', s, \langle o_i, o_{\neq i}\rangle, \langle a_i, a_{\neq i}\rangle, b)}{\sum_{s'} \sum_s \sum_{o_{\neq i}} Pr(s', s, \langle o_i, o_{\neq i}\rangle, \langle a_i, a_{\neq i}\rangle, b)} \tag{4.3}$$

$$= \frac{\sum_{o_{\neq i}} Pr(\langle o_i, o_{\neq i}\rangle|s', \langle a_i, a_{\neq i}\rangle) \sum_s Pr(s'|s, \langle a_i, a_{\neq i}\rangle) b(s)}{\sum_{s'} \sum_s \sum_{o_{\neq i}} Pr(\langle o_i, o_{\neq i}\rangle|s', \langle a_i, a_{\neq i}\rangle) Pr(s'|s, \langle a_i, a_{\neq i}\rangle) b(s)} \tag{4.4}$$

$$= \frac{\sum_{o_{\neq i}} O(\langle o_i, o_{\neq i}\rangle|\langle a_i, a_{\neq i}\rangle, s') \sum_s T(s, \langle a_i, a_{\neq i}\rangle, s') b(s)}{\sum_{s'} \sum_s \sum_{o_{\neq i}} O(\langle o_i, o_{\neq i}\rangle|\langle a_i, a_{\neq i}\rangle, s') T(s, \langle a_i, a_{\neq i}\rangle, s') b(s)}. \tag{4.5}$$

Notes:

1. As all (i) these heuristic initialization methods and (ii) Inf-JESP's local search are deterministic, using a deterministic procedure to derive best-response FSCs induces a deterministic algorithm for which restarts are useless.

2. These heuristic initialization methods can be also adapted to the finite-horizon setting for JESP by changing the policy representation to finite-horizon trees.

Algorithm 9: Extracting FSC_i for agent i from MPOMDP α-vector set Γ

```
[Input:] i: agent | Γ: MPOMDP α-vector set
Start node n_0 ← node(argmax_{α∈Γ} α · b_0, b_0)
N ← {n_0}
G.pushback(n_0)
while |G| > 0 do
    n ← G.popfront()
    (b, a) ← (n.b, ψ(n))
    forall o_i ∈ Ω_i do
        if Pr(o_i|b, a) > 0 then
            α ← argmax_α(b_a^{o_i}, Γ)
            if α ∉ N then
                b_a^{o_i} ← BeliefUpdateOneSide(b, a, o_i)
                n' ← node(α, b_a^{o_i})
                N ← N ∪ {n'}
                G.pushback(n')
            else
                n' ← N(α)
            η(n, o_i, n') ← 1
        else
            η(n, o_i, n) ← 1
return ⟨N, η, ψ⟩
```

For Inf-JESP, we tested M-S and M-D heuristic initialization methods. Since the results are already good, implementing M-A for Inf-JESP is not our main priority and will be considered as a future perspective.

4.2 Monte-Carlo JESP

In Section 4.1, we presented the Inf-JESP algorithm, which solves Dec-POMDPs by finding Nash equilibria solutions with explicit models. However, explicit models are not always available, especially for large problems. Building an explicit model may cost a lot of computational resources and memory. In this section, we present another algorithm based on the Monte-Carlo search in a JESP scheme, which only requires a generative model, *i.e.*, a simulator. We name this algorithm *MC-JESP*. Also, as Inf-JESP, we will use FSCs to represent agent policies. In Section 4.2.1, we present how to use the generative Dec-POMDP model and other agents' policies to build a *best-response generative model* for agent i, and we introduce its relation with the best-response POMDP detailed in Section 4.1.2. Then, Section 4.2.2 describes how to build agent i's FSC policy with its best-response generative model. MC-JESP's main algorithm and an initialization method are presented in Section 4.2.3.

4.2.1 Building Best Response Generative Model for Agent i

Similar to Inf-JESP, we aim to find Nash equilibrium solutions by iteratively building each agent's best response to other agents' fixed policies. We thus stick to representing policies as FSCs since we still address infinite-horizon problems, and we rely on the same formalism for best-response POMDPs as in Inf-JESP, *i.e.*, in particular with the same representations of extended states, actions, and observations. However, the method described in Section 4.1.2 for building the best response POMDP for agent i is only possible if we have an explicit Dec-POMDP model. Then, a question for MC-JESP is how to build such a best response model if only a generative model of the Dec-POMDP is available? To address this issue, in MC-JESP, we propose an alternative approach that builds a *best-response generative model* relying on a Dec-POMDP simulator and other agents' FSCs. In the following content, we call this model "G_{BR}" for simplificity.

Algorithm Description: The problem of building a G_{BR} for agent i is to sample the next extended state e^{t+1}, observation o_i^{t+1}, and reward r^{t+1}, given a current extended state e^t and action a_i^t, and as shown in Figure 4.3, we can only rely on the available functions of the Dec-POMDP simulator and other agents' FSCs. Algorithm 10 details the process of building such an extended generative model for agent i. It first decomposes the extended state, and gets other agents' actions $a_{\neq i}^t$ according to action selection functions $\psi_{\neq i} \equiv \langle \psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots, \psi_{|\mathcal{I}|} \rangle$ (lines 3 and 4). Then, in line 5, the joint action $\langle a_i^t, a_{\neq i}^t \rangle$ is passed to the Dec-POMDP simulator G, which outputs the next state s^{t+1}, joint observation $\langle o_i^{t+1}, o_{\neq i}^{t+1} \rangle$ and instant reward r^{t+1}. With the obtained others' observation $o_{\neq i}^{t+1}$, line 6 computes their next nodes $n_{\neq i}^{t+1} \equiv \langle n_1^{t+1}, \ldots, n_{i-1}^{t+1}, n_{i+1}^{t+1}, \ldots, n_{|\mathcal{I}|}^{t+1} \rangle$. In the end, we build the next extended state s_e^{t+1} and return the results (lines 7 and 8). In this algorithm, stochasticity exists only in the Dec-POMDP simulator G, the FSC functions ψ and η are deterministic.

Algorithm 10: Agent i's Best Response Generative Model $G_{BR-POMDP}$

1 [Input:] s_e^t: extended state | a_i^t: agent i's action
2 [Paramater:] G: Dec-POMDP simulator | $\langle N_{\neq i}, \psi_{\neq i}, \eta_{\neq i} \rangle$: other agents' FSCs
3 $\langle s^t, n_{\neq i}^t, o_i^t \rangle \leftarrow s_e^t$
4 $a_{\neq i}^t \leftarrow \psi_{\neq i}(n_{\neq i}^t)$
5 $s^{t+1}, \langle o_i^{t+1}, o_{\neq i}^{t+1} \rangle, r^{t+1} \leftarrow G(s^t, \langle a_i^t, a_{\neq i}^t \rangle)$
6 $n_{\neq i}^{t+1} \leftarrow \eta(n_{\neq i}^t, o_{\neq i}^{t+1})$
7 $s_e^{t+1} \leftarrow \langle s^{t+1}, n_{\neq i}^{t+1}, o_i^{t+1} \rangle$
8 **return** $s_e^{t+1}, o_i^{t+1}, r^{t+1}$ ▷ return step results

4.2.2 Computing Agent i's FSC using Monte Carlo Methods

In the previous section, we saw how to build the best-response generative model $G_{BR-POMDP}$ for agent i considering others' fixed FSCs. However, unlike in Inf-JESP, modern point-based POMDP solvers can not be applied to solve such generative models in MC-JESP. We thus need to rely on a simulation-based solver, *i.e.*, POMCP [Silver and Veness, 2010], which is an online algorithm that gives only the best action for the current belief. Therefore, the question is how to use a simulator (G_{BR}) and a simulation-based solver to obtain agent i's FSC. To address this issue, we propose an algorithm that

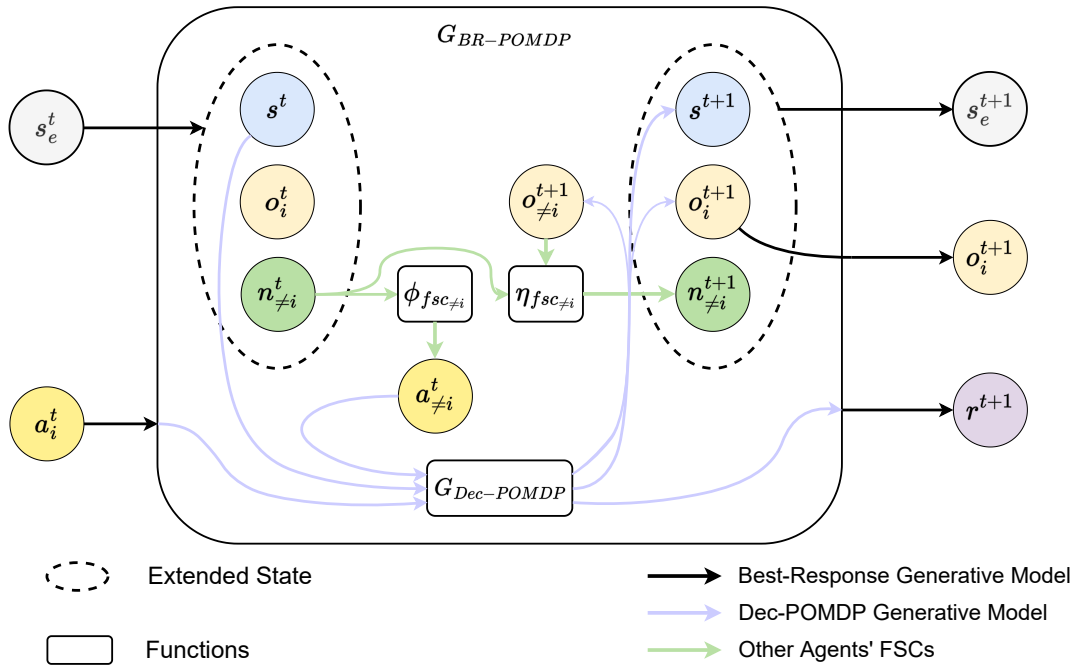

Figure 4.3: Structure of the best-response generative model G_{BR}: The black arrows are the inputs and outputs of the POMDP generative model; The blue arrows show the inputs and outputs of a Dec-POMDP simulator G; The green arrows show the evolution of agents $\neq i$' FSCs.

- uses the Monte-Carlo approach (POMCP) to compute the best action for a given node, which is labeled by a unique belief;
- expands reachable beliefs and creates new nodes using computed actions to gradually build an FSC.

Moreover, to avoid building an infinite-size FSC (with infinite-size beliefs), we explicitly bound the FSC size with a given parameter $N_{max\text{-}fsc}$.

Algorithm Description: We use Algorithm 11 to build a policy fsc_i for agent i with bounded size $N_{max\text{-}fsc}$ given its best-response generative model G_{BR}. First, POMCP is used to compute agent i's best action a_i^0 at the start belief in line 4. Then, before creating a new node, we process the computed action for the given belief (line 5). This *ProcessAction* function samples many transitions until each feasible observation (according to the samples) is attached to a sufficient number of particles $N_{min\text{-}particles}$ (line 31), and returns the set of feasible observations Ω_i^{t+1} and the set of induced beliefs B_i^{t+1} (each belief is a collection of particles). Then, in line 6, a start node n_0 is created with b_{BR}^0, a_i^0, B_i^1, Ω_i^1, and a weight $w = 1$. This start node is added to the FSC under construction (N) and an open list (L). While L is not empty, L is sorted according to the node weights w (from higher weights to lower weights), and the first node n stored in the sorted L is popped out (lines 10–11). Node n's compatibility with each individual observation o_i is then checked, *i.e.*, $o_i \in n.\Omega_i$, and self-loops are introduced for impossible observations (line 27). Line 14 gets the new belief of receiving observation o_i, and a new associated weight w' is computed in line 15. Then, line 16 computes agent i's best action a_i' given its new belief b_{BR}' using POMCP. Before creating a new node n', in line 17, the algorithm verifies (i) the non-existence of b_{BR}',

i.e., measuring the distance between (estimated) belief states in Norm-1 and comparing it to ϵ: $\|b'_{BR} - b\|_1 \leqslant \epsilon$; and (ii) if the current FSC size is smaller than $N_{max\text{-}fsc}$. If those conditions are not satisfied, we update the weight of the closest existing node (lines 23–24); otherwise, n' is added to both N and L (lines 19–21). In line 25, an edge $n \to n'$ is created with a label o_i.

4.2.3 Main Algorithm and Initialization

Main Algorithm: The main algorithm of MC-JESP is presented in Algorithm 12. Compared with Inf-JESP's main algorithm, the differences are threefold:

- in line 7, MC-JESP needs to construct a best-response generative model G_{BR} for agent i in each iteration, while Inf-JESP constructs an explicit best-response POMDP model;
- MC-JESP uses a Monte-Carlo method to construct agent i's FSC (line 8) which differs from Inf-JESP's extraction process (see Algorithm 7);
- simulations are conducted in line 9 to estimate the value obtained using agent i's policy fsc'_i under initial distribution b^0_{BR} with simulator G_{BR}, while in Inf-JESP, the value can be directly computed following a system of linear equations (see Equation (2.21)).

As in Inf-JESP, the FSC size is bounded in MC-JESP, and in each iteration of MC-JESP, the value improves monotonically (assuming perfect evaluation of FSCs). Therefore, the search will necessarily terminate in finite time. In the end, the final solution may be close to a Nash equilibrium, given that POMCP is an approximate online solver.

Heuristic Initialization: Although MC-JESP monotonically improves the value of the joint policy at each iteration, some random initializations may lead to poor locally optimal solutions. Therefore, based on Algorithm 11, we develop a heuristic initialization method for MC-JESP similar to the one used in Inf-JESP (we use the same MPOMDP extraction process and belief update method as described for Algorithm 9). Algorithm 13 shows the process to build a heuristic FSC for agent i. As shown in red, it differs from Algorithm 11 in two aspects:

- the Dec-POMDP simulator G is used as an MPOMDP simulator for POMCP to get the best joint action with a given belief (lines 3 and 16);
- instead of a standard belief update, we use a one-sided-belief-update method (lines 4 and 18) which is first presented in Equation (4.2) when using an exact model. The *ProcessAction* function repeatedly samples transitions using the computed joint action and collects particles for each encountered agent i's observation.

4.3 Experiments

This section evaluates our contributions with other state-of-the-art solvers on various Dec-POMDP benchmarks (*cf.* `http://masplan.org/problem_domains`):

- Decentralized Tiger (Dec-Tiger) [Nair et al., 2003],
- Recycling Robots (Recycling) [Amato et al., 2012],
- Meeting in a 3×3 grid (Grid 3×3) [Amato et al., 2009],

Algorithm 11: Compute agent i's FSC

[Input:] b^0_{BR}: initial (extended) state distribution of G_{BR}
[Parameters:] G_{BR}: best response generative model for agent i | $N_{max\text{-}fsc}$: max FSC size for agent i | $N_{min\text{-}particles}$: minimum particle size for each belief
Fct BuildFSC(b^0_{BR})
  $a^0_i \leftarrow POMCP(G_{BR}, b^0_{BR})$
  $B^1_i, \Omega^1_i \leftarrow ProcessAction(b^0_{BR}, a^0_i)$
  $n_0 \leftarrow node(b^0_{BR}, a^0_i, B^1_i, \Omega^1_i, w = 1)$
  $N \leftarrow \{n_0\}$
  $L[w] \leftarrow n_0$
  while $|L| > 0$ **do**
    $L.sort()$
    $n \leftarrow L.popfront()$
    for $o_i \in \Omega$ **do**
      if $o_i \in n.\Omega_i$ **then**
        $b'_{BR} \leftarrow n.B'_i(o_i)$
        $w' \leftarrow n.w * \frac{|B'_i(o_i)|}{|B'_i|}$
        $a'_i \leftarrow POMCP(G_{BR}, b'_{BR})$
        if $(b'_{BR} \notin N(\epsilon)) \wedge (N.size < N_{max})$ **then**
          $B''_i, \Omega''_i \leftarrow ProcessAction(b'_{BR}, a'_i)$
          $n' \leftarrow node(b'_{BR}, a'_i, B''_i, \Omega''_i, w')$
          $N \leftarrow N \cup \{n'\}$
          $L[w] \leftarrow n'$
        else
          $n' \leftarrow N.find(b'_{BR})$
          $n'.w \leftarrow n'.w + w'$
        $\eta(n, o_i, n') \leftarrow 1$
      else
        $\eta(n, o_i, n) \leftarrow 1$
Fct ProcessAction(b^t_{BR}, a^t_i)
  $B^{t+1}_i \leftarrow \varnothing$
  $\Omega^{t+1}_i \leftarrow \varnothing$
  repeat
    $e^t \sim b^t_{BR}$
    $\langle e^{t+1}, o^{t+1}_i, o^{t+1}_{\neq i}, r^{t+1} \rangle \sim G_{BR}(e^t, a^t_i)$
    if $o^{t+1}_i \notin \Omega^{t+1}_i$ **then**
      $\Omega^{t+1}_i \leftarrow \Omega^{t+1}_i \cup \{o^{t+1}_i\}$
    $B^{t+1}[o^{t+1}_i] \leftarrow B^{t+1}[o^{t+1}_i] \cup \{e^{t+1}\}$
  until $Timeout() \vee (MinBeliefParticles(B^{t+1}) > N_{min\text{-}particles})$
  return B^{t+1}_i, Ω^{t+1}_i

Algorithm 12: Monte-Carlo JESP

[Input:] K: FSC size | fsc: initial FSCs | b^0: initial belief | G: Dec-POMDP simulator
Fct LocalSearch$(K, fsc \stackrel{\text{def}}{=} \langle fsc_1, \ldots, fsc_{|\mathcal{I}|} \rangle, G)$
  $v_{bestL} \leftarrow eval(fsc)$
  $\#ni \leftarrow 0$ `// #(iterations w/o improvement)`
  $i \leftarrow 1$ `// Id of current agent`
  repeat
    $G_{BR}, b^0_{BR} \leftarrow$ **BuildBestResponseGenerativeModel**$(G, b^0, fsc_{\neq i})$
    $fsc'_i \leftarrow$**ComputeFSC**$(b^0_{BR} \mid G_{BR},\ K)$
    $v \leftarrow$ **EvalSim**$(fsc'_i, b^0_{BR} \mid G_{BR})$
    if $v > v_{bestL}$ **then**
      $fsc_i \leftarrow fsc'_i$
      $v_{bestL} \leftarrow v$
      $\#ni \leftarrow 0$
    else
      $\#ni \leftarrow \#ni + 1$
    $i \leftarrow (i \mod |\mathcal{I}|) + 1$
  until $\#ni = |\mathcal{I}|$
  return $\langle fsc, v_{bestL} \rangle$

- Cooperative Box Pushing (Box-Pushing) [Seuken and Zilberstein, 2007],
- Mars Rover [Amato and Zilberstein, 2009].

The evaluation of Inf-JESP is presented in Section 4.3.1 and compared with other Dec-POMDP solvers which rely on explicit models. Section 4.3.2 evaluates MC-JESP, which only requires a Dec-POMDP simulator, and we compare MC-JESP's results with other existing Dec-POMDP solvers (including algorithms relying on explicit models and simulators). Moreover, all the evaluations are conducted in an infinite-horizon setting.

4.3.1 Evaluation of Inf-JESP

Algorithm settings

We compare the different variants of Inf-JESP with state-of-the-art Dec-POMDP solvers, namely: FB-HSVI [Dibangoye et al., 2016], Peri [Pajarinen and Peltonen, 2011], PeriEM [Pajarinen and Peltonen, 2011], PBVI-BB [MacDermed and Isbell, 2013] and MealyNLP [Amato et al., 2010]. We ignore JESP as it only handles finite horizons with policy trees, and it will cost a large number of memories and time to approximate infinite horizons with long horizons. Dec-BPI is compared separately because we can only estimate values from empirical curves on some benchmark problems [Bernstein et al., 2009].

For Inf-JESP, SARSOP [Kurniawati et al., 2008] is used as our POMDP solver, with a 0.001 Bellman residual (also for FSC evaluation) and a 5 s timeout. Moreover, we have tested 4 variants of Inf-JESP:

- IJ(M-D) and IJ(M-S) are Inf-JESP initialized with the M-D and M-S heuristics (Section 4.1.4) and without restarts (because they behave deterministically).

Algorithm 13: Build a heuristic FSC for agent i

[Input:] b^0: initial state distribution of G | i: heuristic agent index
[Parameter:] G: Dec-POMDP simulator | N_{max}: max FSC size for agent i
$\langle a_i^0, a_{\neq i}^0 \rangle \leftarrow POMCP(G, b^0)$
$B_i^1, \Omega_i^1 \leftarrow ProcessAction(b^0, \langle a_i^0, a_{\neq i}^0 \rangle)$
$n_0 \leftarrow node(b^0, a_i^0, B_i^1, \Omega_i^1, w = 1)$
$N \leftarrow \{n_0\}$
$G[w] \leftarrow n_0$
while $|G| > 0$ **do**
  $G.sort()$
  $n \leftarrow G.popfront()$
  $\langle b, a \rangle \leftarrow n$
  for $o_i \in \Omega$ **do**
    if $o_i \in n.\Omega_i$ **then**
      $b' \leftarrow n.B_i'(o_i)$
      $w' \leftarrow n.w * \frac{|B_i'(o_i)|}{|B_i'|}$
      $\langle a_i', a_{\neq i}' \rangle \leftarrow POMCP(G, b')$
      if $(b' \notin N(\epsilon)) \wedge (N.size < N_{max})$ **then**
        $B_i'', \Omega_i'' \leftarrow ProcessAction(b', \langle a_i', a_{\neq i}' \rangle)$
        $n' \leftarrow node(b', a_i', B_i'', \Omega_i'', w')$
        $N \leftarrow N \cup \{n'\}$
        $G[w] \leftarrow n'$
      else
        $n' \leftarrow N.find(b')$
        $n'.w \leftarrow n'.w + w'$
      $\eta(n, o_i, n') \leftarrow 1$
    else
      $\eta(n, o_i, n) \leftarrow 1$

- IJ(R-1) and IJ(R-100) are Inf-JESP with random initializations (at most 5 nodes per FSC) and, respectively, 1 (re)start and 100 restarts.

Each restart has a timeout of 7200 s (2 h), and IJ(R-x_y) denotes y runs of IJ(R-x) used to compute statistics. Due to a large number of iterations which often lead to a timeout and cost memories, we did not perform full random-restart tests on the Box-Pushing and Mars Rover domains.

The experiments with Inf-JESP were conducted on a laptop with an i5-1.6 GHz CPU and 8 GB RAM. The source code is available at `https://gitlab.inria.fr/ANR-FCW/InfJESP` .

Comparison with state-of-the-art algorithms

Table 4.1 presents the results for the 5 problems. For IJ(R-x_y), we kept the highest value among the restarts in each run, and then computed the average of this value over the various runs. The columns provide:

- (*Alg.*) the different algorithms at hand;

Table 4.1: Comparison of different algorithms in terms of final FSC size, number of iterations, time, and value, on five infinite-horizon benchmark problems with $\gamma = 0.9$ for all domains. R-R_N = N runs with R random restarts each. M-S and M-D = deterministic initializations with stochastic FSCs and deterministic FSCs extracted from an MPOMDP solution. Solvers are arranged in descending order by their values.

Alg.	FSC size	#Iterations	Time (s)	Value
DecTiger ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 2, \lvert\mathcal{A}^i\rvert = 3, \lvert\mathcal{Z}^i\rvert = 2$)				
FB-HSVI			154	13.45
PBVI-BB				13.45
Peri			220	13.45
IJ(R-100_5**)**	$9_{\pm 3} \times 8_{\pm 4}$	$33_{\pm 1}$	12 418	13.44
IJ(M-D)	6×6	36	201	13.44
IJ(M-S)	6×6	27	213	13.44
PeriEM			6450	9.42
MealyNLP			29	−1.49
IJ(R-1_{500}**)**	$29_{\pm 55} \times 32_{\pm 54}$	$21_{\pm 14}$	124	−35.47
Recycling ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 4, \lvert\mathcal{A}^i\rvert = 3, \lvert\mathcal{Z}^i\rvert = 2$)				
FB-HSVI			3	31.93
PBVI-BB				31.93
MealyNLP			0	31.93
Peri			77	31.84
PeriEM			272	31.80
IJ(R-100_{10}**)**	$2_{\pm 0} \times 2_{\pm 0}$	$3_{\pm 0}$	3	31.62
IJ(R-1_{1000}**)**	$3_{\pm 2} \times 3_{\pm 2}$	$3_{\pm 0}$	0	30.60
IJ(M-S)	8×8	3	0	26.57
IJ(M-D)	4×4	3	0	25.65
Grid3*3 ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 81, \lvert\mathcal{A}^i\rvert = 5, \lvert\mathcal{Z}^i\rvert = 9$)				
IJ(M-D)	8×10	3	2	5.81
IJ(M-S)	9×17	3	9	5.81
IJ(R-100_5**)**	$13_{\pm 6} \times 9_{\pm 0}$	$4_{\pm 0}$	414	5.81
FB-HSVI			67	5.80
IJ(R-1_{500}**)**	$11_{\pm 18} \times 11_{\pm 18}$	$3_{\pm 1}$	4	5.79
Peri			9714	4.64
Box-pushing ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 100, \lvert\mathcal{A}^i\rvert = 4, \lvert\mathcal{Z}^i\rvert = 5$)				
FB-HSVI			1715	224.43
PBVI-BB				224.12
IJ(M-S)	250×408	6	963	220.25
IJ(M-D)	274×342	8	1436	203.41
Peri			5675	148.65
MealyNLP			774	143.14
PeriEM			7164	106.65
Mars Rover ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 256, \lvert\mathcal{A}^i\rvert = 6, \lvert\mathcal{Z}^i\rvert = 8$)				
FB-HSVI			74	26.94
IJ(M-D)	125×183	6	122	26.91
IJ(M-S)	84×64	5	66	24.17
Peri			6088	24.13
MealyNLP			396	19.67
PeriEM			7132	18.13

- (*FSC size*) the final FSC sizes obtained for Inf-JESPs;
- (*Iterations*) the number of iterations required to converge (for Inf-JESPs);
- (*Time*) the running time, recorded in seconds;
- (*Value*) the final value (lower bounds for Inf-JESPs; the true value is at most 0.01 higher).

In terms of final value achieved, Inf-JESP variants find good to very good solutions in most cases, often very close to FB-HSVI's near-optimal solutions. Comparing with estimated values obtained by Dec-BPI on the Dec-Tiger, Grid, and Box-Pushing problems (using the figures in [Bernstein et al., 2009, p. 123]), Inf-JESP outperforms Dec-BPI in these three problems. Note also that Dec-BPI's results highly depend on the size of the considered FSCs and that Dec-BPI faces difficulties with large FSCs, *e.g.*, in the Dec-Tiger problem [Bernstein et al., 2009].

Regarding the solving time, Inf-JESP variants have different behaviors depending on the problem at hand. For example, in Dec-Tiger, IJ(M-S) and IJ(M-D) have almost the same solving time, and both of them require more time than IJ(R-1_{500}). However, in other problems, IJ(M-D) and IJ(M-S) may require very different solving times. For instance, IJ(M-D) converges faster than IJ(M-S) in Grid but behaves contrary in Box-pushing and Mars Rover domains. As mentioned above, we give a timeout of 5 s for SARSOP in each iteration to solve one agent's POMDP. In Dec-Tiger, most of the time is spent calling SARSOP for multiple iterations, but in Box-Pushing and Mars Rovers, most of the time is used for building the explicit POMDP model and evaluating the obtained FSC in each iteration. Overall, IJ(M-D) and IJ(M-S) can give good solutions within an acceptable time.

Last but not least, we observe that the number of iterations before converging behaves independently of the solving time. For example, the shortest solving time in Dec-Tiger does not correspond to the smallest number of iterations.

A Closer Look at Inf-JESP's behavior

Figure 4.4 (right) presents the distribution over final values of IJ(R-1). Most runs end with (near) globally optimal values in the three problems at hand, despite initial FSCs limited to 5 nodes. Other local optima turn out to be rarely obtained, except in Dec-Tiger. These distributions show that few restarts are needed to reach a global optimum with high probability.

Figure 4.4 (left) presents the evolution of the FSC values during a JESP run as a function of the iteration number for IJ(R-1) (average + 10th and 90th percentiles, in blue), IJ(M-D) (in green) and IJ(M-S) (in red). For random initialization (IJ(R-1)), at each iteration, the average is computed over all runs, even if they have already converged. This figure first shows that (i) Inf-JESP's solution quality monotonically increases during a run, and (ii) most runs converge to local optima in a few iterations. It also illustrates that MPOMDP initializations are rather good heuristics compared to random ones but, as expected, do not always lead to a global optimum. For example, in the Recycling problem, the MPOMDP heuristic leads to a sub-optimal solution, while random initializations often lead to near-globally-optimal (thus better) solutions.

Complementary results

Regarding *state elimination* (as explained in Section 4.1.2), it turned out that, for the "MOMDP" best-response POMDP, the ratio of the initial number of (extended) states over its number of states after state elimination depends on the problem at hand, but does not depend on the Inf-JESP run (see Figure 4.5). This ratio was 1 for the Dec-Tiger problem, 50 for the Grid problem,

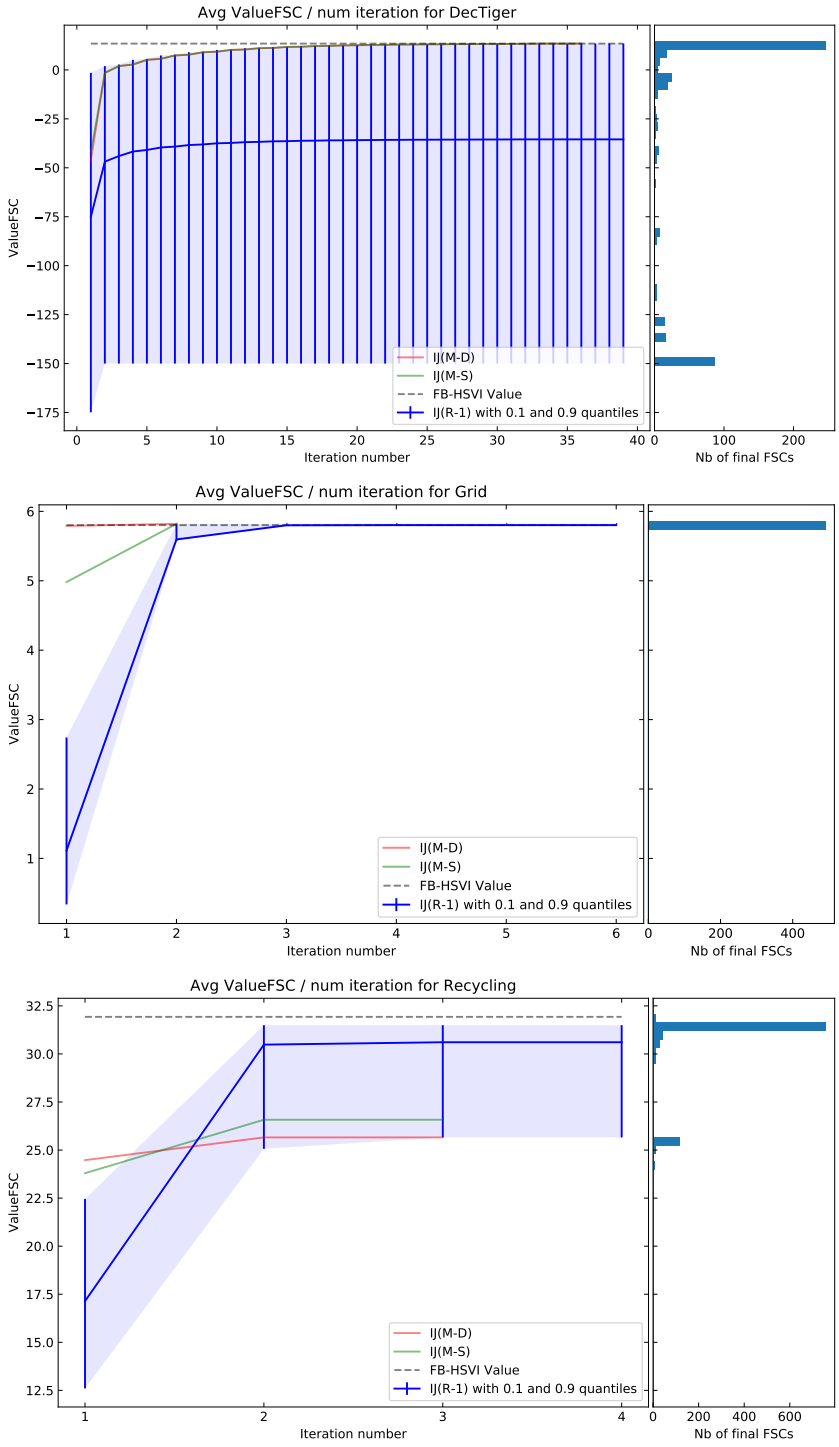

Figure 4.4: Values of the joint policy for the Dec-Tiger, Grid and Recycling problems (from top to bottom). The left part of each figure presents the evolution (during a run) of the value of the joint policy at each iteration of: IJ(R-1) (avg + 10th and 90th percentiles in blue), and the deterministic IJ(M-D) (in red) and IJ(M-S) (in green). The dashed line represents FB-HSVI's final value. The right part presents the value distribution after convergence of IJ(R-1).

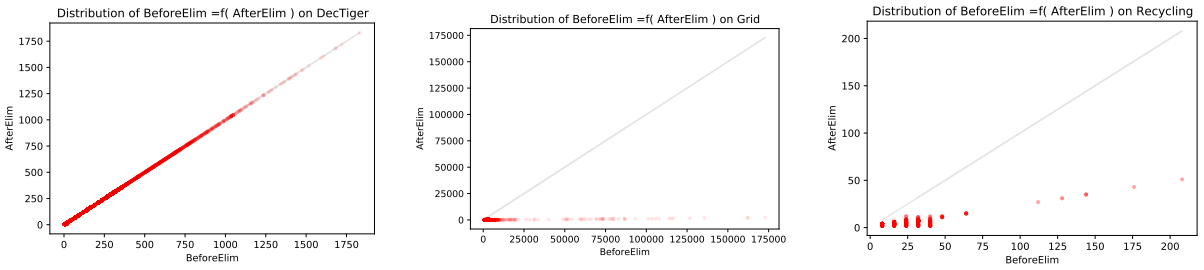

Figure 4.5: Effect of the state elimination for the Dec-Tiger, Grid and Recycling problems (from left to right). Each figure plots, for each Best-Response POMDP obtained while executing Inf-JESP(R-1_1), the number of states of that POMDP after state elimination as a function of that number before state elimination.

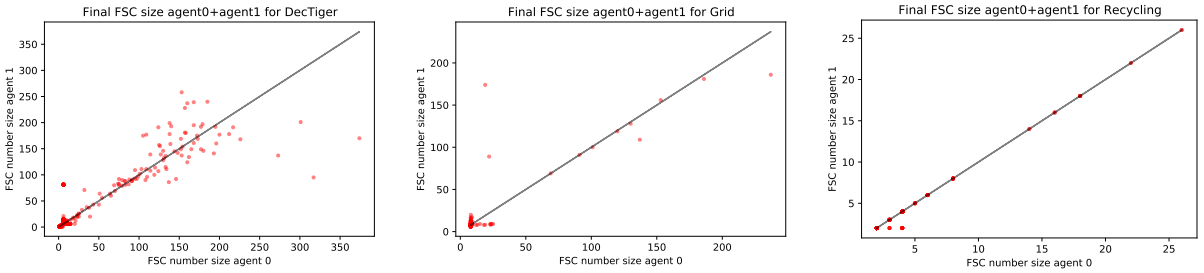

Figure 4.6: Sizes of the final FSCs obtained with Inf-JESP(R-1_1) after convergence for the Dec-Tiger, Grid and Recycling problems (from left to right). Each dot corresponds to a pair containing the sizes of both agents' FSCs.

and 5 for the Recycling problem. These differences are due to the stochasticity of the observation process, which limits or even prevents state elimination. As it is currently done, state elimination is based on the probability of generating a specific observation from a state considering possible actions. However, for the Dec-Tiger problem, every observation can be generated whatever the considered action, leading to no state elimination (contrary to the Grid and the Recycling problem).

Regarding *the size of the final FSCs* obtained after convergence of an Inf-JESP run with a random initialization (see Figure 4.6), we also observed different behaviors. Applied to the Dec-Tiger problem, Inf-JESP leads to a very large distribution of the sizes of the final FSCs. Applied to the Recycling problem, Inf-JESP also leads to a large distribution of the sizes of the final FSC, but the sizes of the FSCs of both agents are symmetrical. Applied to the Grid problem, Inf-JESP generates a large number of FSC pairs of size 10 with sometimes huge variation and with a tendency to be asymmetrical, the first optimized agent having a tendency to have more FSC nodes. These results need to be further investigated in order to understand them clearly and see if it is possible to link them to the nature of the addressed domain.

It must also be noted that *the sizes of the FSCs* do not monotonically increase during one Inf-JESP iteration (data not presented here). Sometimes the size of the FSC computed during one Inf-JESP improvement decreases, meaning that the solution to the best-response POMDP is an FSC with a smaller size but a higher value (as commonly observed in the Dec-Tiger problem). When looking at the FSCs obtained at the end of one Inf-JESP run on Figure 4.7, another observed phenomenon is that the more iterations required to reach the equilibrium, the smaller

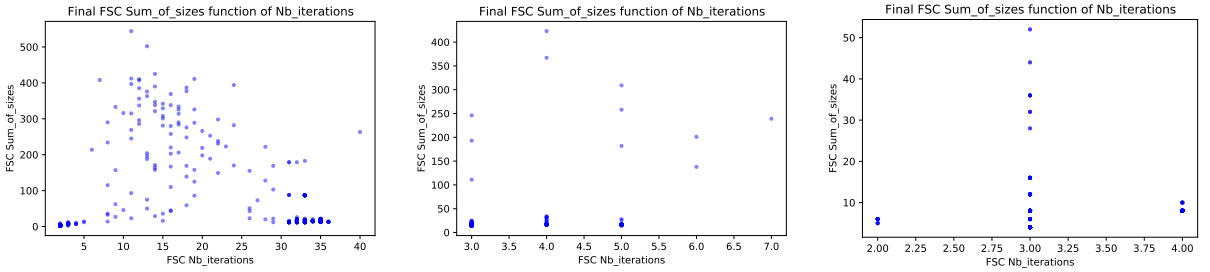

Figure 4.7: Sum of the sizes of the final FSCs obtained with Inf-JESP(R-1_1) after convergence as a function of the required number of iterations for the Dec-Tiger, Grid and Recycling problems (from left to right).

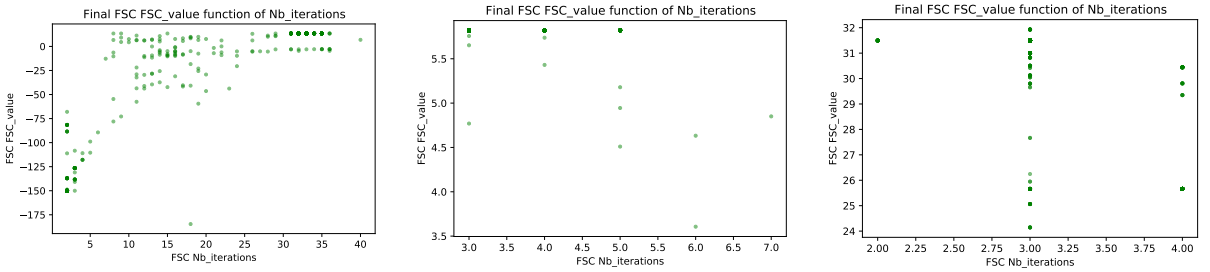

Figure 4.8: Value of the final FSCs obtained with Inf-JESP(R-1_1) after convergence as a function of the required number of iterations for the Dec-Tiger, Grid and Recycling problems (from left to right).

the sizes of the final FSCs. But this does not mean that the associated value is higher (see Figure 4.8).

Finally, Figure 4.9 presents the values of the obtained equilibria as a function of the sum of sizes of the FSCs obtained by Inf-JESP(R-1_1) after convergence. It can be observed that small FSCs seem to be sufficient to generate a high value, opening new directions on combining Inf-JESP with FSC compression.

Summary Those experiments showed that Inf-JESP exhibits different behaviors depending on the problem at hand. For instance, removing unreachable extended states (Section 4.1.2) divides

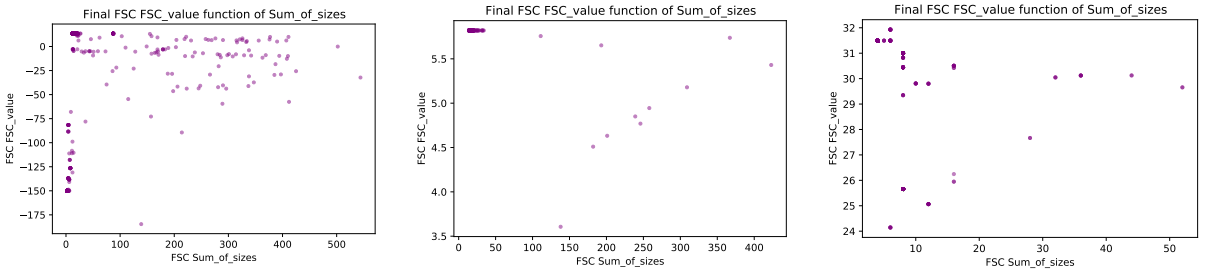

Figure 4.9: Value of the final FSCs obtained with Inf-JESP(R-1_1) after convergence as a function of the sum of the sizes of final FSCs for the Dec-Tiger, Grid and Recycling problems (from left to right).

their number on average by

- 1 in Dec-Tiger;
- 50 in Grid3×3, and;
- 5 in Recycling.

We also observed that final FSCs with the highest values are not obtained with more iterations and do not contain more nodes. Small FSCs seem sufficient to obtain good solutions, opening an interesting research direction to combine Inf-JESP with FSC compression, *e.g.*, one may want to force restarts to use small FSCs at the beginning, and then only progressively increase the maximum FSC size.

4.3.2 Evaluation of MC-JESP

Algorithm settings

The experiments with MC-JESP were conducted on a laptop with a 2.3 GHz i9 CPU. We compare MC-JESP with two types of state-of-the-art Dec-POMDP solvers:

- algorithms which rely on explicit models: FB-HSVI [Dibangoye et al., 2016], Peri [Pajarinen and Peltonen, 2011], PeriEM [Pajarinen and Peltonen, 2011], PBVI-BB [MacDermed and Isbell, 2013], MealyNLP [Amato et al., 2010] and Inf-JESP; and
- algorithms which rely on generative models: MCEM [Wu et al., 2013], Dec-SBPR [Liu et al., 2015] and oSARSA [Dibangoye and Buffet, 2018].

Although MC-JESP is a sampling-based algorithm, we still compared it with the methods which rely on explicit models to show MC-JESP's power. For MC-JESP, POMCP is used as our sampling-based POMDP planner with a timeout of 5 s. Moreover, we experiment with MC-JESP using three different bounds on the FSC sizes: 10, 30, and 50, respectively, when building each agent's best-response policy.

Comparison with state-of-the-art algorithms

Table 4.2 presents the results for the 5 problems in which the solvers are ordered from best to worse value. For MC-JESP(M-x), we kept the highest value among x restarts and then computed the average of this value over the various runs in MC-JESP(M-1_x). The columns provide:

- (*Alg.*) the different algorithms at hand, with a $*$ exponent for those who rely on an explicit model;
- (*FSC size*) the final FSC size (for Inf-JESP and MC-JESP respectively);
- (*Iterations*) the number of iterations required to converge (for Inf-JESPs and MC-JESP);
- (*Time*) the running time;
- (*Value*) the final value (lower bounds for Inf-JESPs and MC-JESP, the true value being at most 0.01 higher).

Table 4.2: Comparison of different algorithms in terms of final FSC size, number of iterations, time, and value, on five infinite-horizon benchmark problems with $\gamma = 0.9$ for all domains. The solvers are listed in a decreasing order of value.

Alg.	FSC size	Iterations	Time (s)	Value
DecTiger ($\|\mathcal{I}\| = 2, \|\mathcal{S}\| = 2, \|\mathcal{A}^i\| = 3, \|\mathcal{Z}^i\| = 2$)				
FB-HSVI*			154	13.45
Peri*			220	13.45
INF-JESP*	6×6	27	213	13.44
MC-JESP(M-20)	10×10	5		13.44
PeriEM*			6450	9.42
oSARSA				-0.20
MC-JESP(M-1_{20})	10×10	3	235	-5.86
Dec-SBPR			96	-18.63
MCEM			20	-32.31
Recycling ($\|\mathcal{I}\| = 2, \|\mathcal{S}\| = 4, \|\mathcal{A}^i\| = 3, \|\mathcal{Z}^i\| = 2$)				
FB-HSVI*			3	31.93
Peri*			77	31.84
PeriEM*			272	31.80
INF-JESP*	2×2	3	3	31.62
Dec-SBPR			147	31.26
MC-JESP(M-20)	50×50	3		31.12
MC-JESP(M-1_{20})	50×50	3	606	30.51
Grid3*3 ($\|\mathcal{I}\| = 2, \|\mathcal{S}\| = 81, \|\mathcal{A}^i\| = 5, \|\mathcal{Z}^i\| = 9$)				
INF-JESP*	8×10	3	2	5.81
MC-JESP(M-20)	50×50	5		5.81
FB-HSVI*			67	5.80
MC-JESP(M-1_{20})	50×50	5	1495	5.78
Peri*			9714	4.64
Box-pushing ($\|\mathcal{I}\| = 2, \|\mathcal{S}\| = 100, \|\mathcal{A}^i\| = 4, \|\mathcal{Z}^i\| = 5$)				
FB-HSVI*			1715	224.43
MC-JESP(M-20)	50×50	4		223.26
INF-JESP*	250×408	6	963	220.25
MC-JESP(M-1_{20})	50×50	4	1245	220.00
Peri*			5675	148.65
oSARSA				144.57
PeriEM*			7164	106.65
Dec-SBPR			290	77.65
Mars Rover ($\|\mathcal{I}\| = 2, \|\mathcal{S}\| = 256, \|\mathcal{A}^i\| = 6, \|\mathcal{Z}^i\| = 8$)				
FB-HSVI*			74	26.94
INF-JESP*	125×183	6	122	26.91
MC-JESP(M-20)	50×50	4		26.78
MC-JESP(M-1_{20})	50×50	4	1543	25.36
Peri*			6088	24.13
Dec-SBPR			1286	20.62
PeriEM*			7132	18.13

In terms of final value achieved, MC-JESP finds good solutions in most cases, often very close to FB-HSVI's near-optimal solutions, which rely on an explicit Dec-POMDP model. Compared with other explicit model-based algorithms, MC-JESP can achieve even better solutions. On the other hand, compared with other sampling-based methods, MC-JESP shows its dominance in large problems (Grid, Box-pushing, and Mars Rovers), achieving much better solutions, close to FB-HSVI, while other sampling-based methods fail.

However, compared with the explicit model-based algorithms, MC-JESP requires more solving time. For example, in large problems such as Mars Rovers, MC-JESP takes 1543 s on average to solve the task, while Inf-JESP takes 122 s with M-D initialization. But this is not surprising since MC-JESP only uses a black-box simulator and, with restarts, MC-JESP can give good solutions within an acceptable time.

A Closer Look at MC-JESP's behavior

We study the MC-JESP's performance with three different bounded FSC sizes (brown for 10, orange for 30, and blue for 50). Figures 4.10 and 4.11 (right parts) present the distribution over final values of MC-JESP with 20 restarts. In the five problems at hand, MC-JESP with max FSC size 50 (blue) has distributions more concentrated on good values than others, and most values are close to FB-HSVI's ones (thus, near-optimal values). These distributions show that few restarts are needed to reach good solutions with high probability if we give large enough FSC sizes for the given problem.

The left parts of Figures 4.10 and 4.11 present the evolution of the values during each iteration of MC-JESP with three different maximum FSC sizes. The average is computed over all runs, even if they have already converged. This figure first shows that MC-JESP monotonically increases during each run, as Inf-JESP, and most runs converge to good local optima in a few iterations. Second, we observe that, for large problems (Box-Pushing and Mars Rovers), there are already significant drops from MC-JESP in the first iteration with an FSC size limit decreasing from 50 to 10. This indicates that, for large problems, we must give large enough FSC size limits, while this is not necessary for small problems.

Last but not least, in Dec-Tiger, although some restarts of MC-JESP end with optimal values, we observe that the average value is still relatively low compared with FB-HSVI. Therefore, we conducted another experiment to investigate the impact of different POMCP timeouts (note that there is a fixed timeout of 5 s for the experiments illustrated in Figures 4.10 and 4.11). To that end, we limit the FSC size in each iteration to at most 50 nodes, and we test MC-JESP with five POMCP timeouts (1 s, 5 s, 10 s, 20 s, and 30 s). The distribution of final values is shown in Figure 4.12. We observe that the average value increases and the variability is shrunk when we give more time to POMCP. However, it also indicates that, when we increase the time budget, we have a lower chance of getting "lucky" good values.

4.4 Conclusion and Perspectives

4.4.1 Conclusion

Inf-JESP We proposed a new infinite-horizon Dec-POMDP solver, Inf-JESP, which is based on JESP, but using FSC representations for policies instead of trees. FSCs allow not only to handle infinite horizons but also to possibly derive compact policies. Any restart of the ideal algorithm provably converges to a Nash equilibrium, and, despite the existence of local optima, experiments show frequent convergence close to global optima in five standard benchmark problems. One

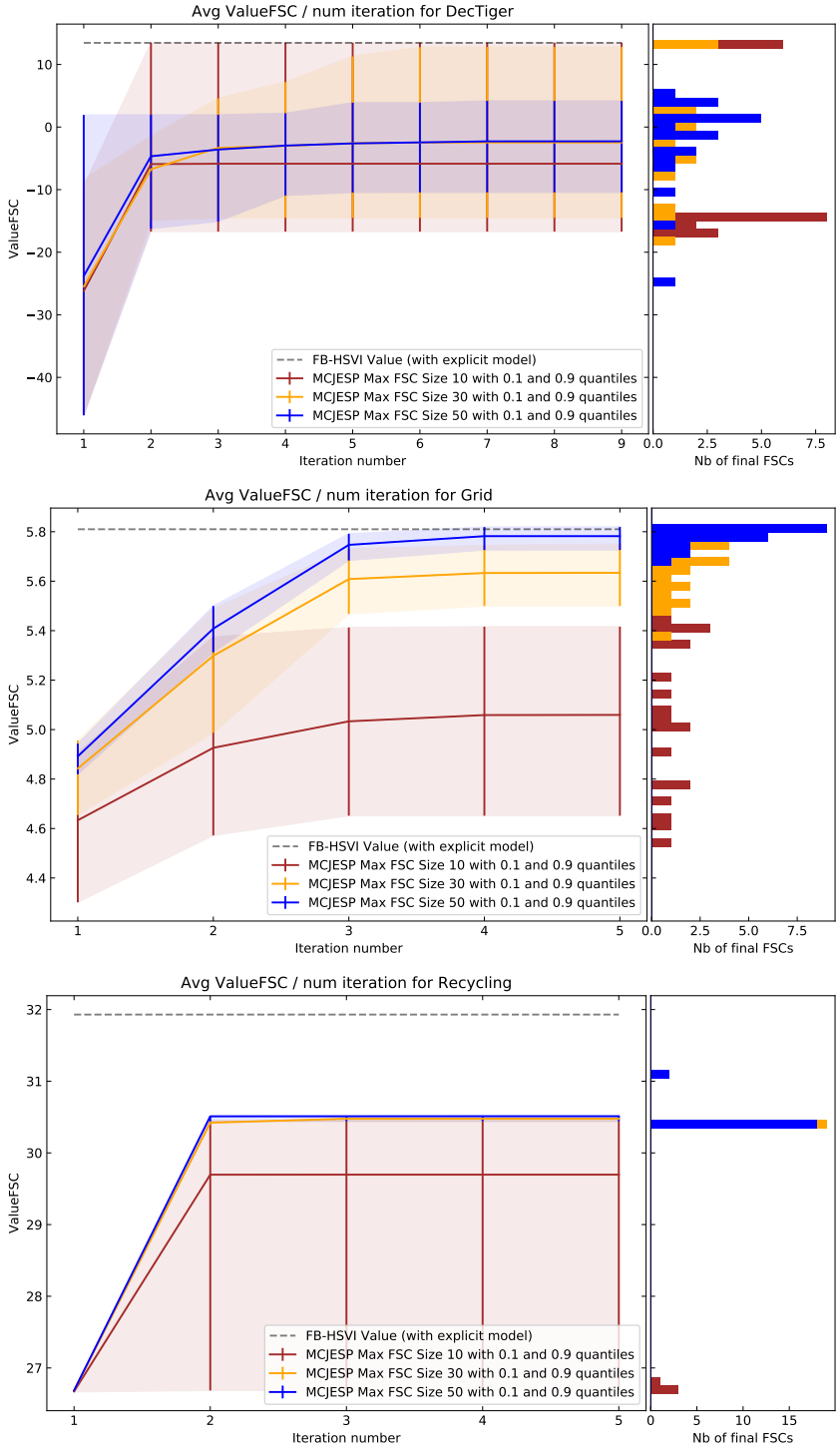

Figure 4.10: Values of the joint policy for the Dec-Tiger, Grid and Recycling problems (from top to bottom). The left part of each figure presents the evolution (during a run) of the value of the joint policy at each iteration of MC-JESP(1_{20}) (avg + 10th and 90th percentiles) with different bounded FSC sizes (10, 30, and 50, respectively). The dashed line represents FB-HSVI's final value. The right part presents the value distribution after convergence of MC-JESP(1_{20}).

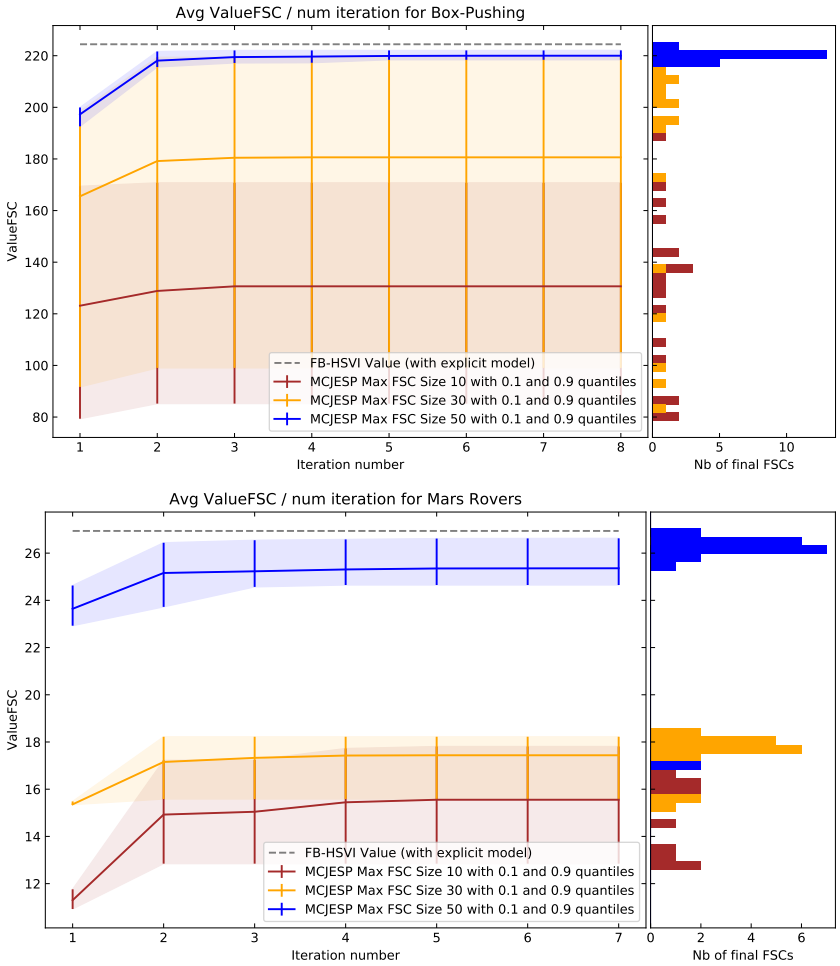

Figure 4.11: Values of the joint policy for the Box-Pushing and Mars Rovers problems (from top to bottom). The left part of each figure presents the evolution (during a run) of the value of the joint policy at each iteration of MC-JESP(1_{20}) (avg + 10th and 90th percentiles) with different bounded FSC sizes (10, 30, and 50, respectively). The dashed line represents FB-HSVI's final value. The right part presents the value distribution after convergence of MC-JESP(1_{20}).

ingredient, the derivation of a POMDP from $|\mathcal{I}| - 1$ fixed FSCs, could also be useful in other settings where other agents' behaviors are defined independently as games [Oliehoek et al., 2005] or human-robot interactions [Bestick et al., 2017], and serves as an important intermediate step in our contributions to human-robot collaborations (see Chapter 6). Moreover, Inf-JESP provides a theoretic foundation for finding Nash equilibrium solutions using explicit Dec-POMDP models, which is used and extended by MC-JESP for generative models.

We also provided a method to extract individual policies (FSCs) from an MPOMDP solution α-vector set Γ to initialize Inf-JESP. This approach can be easily adapted to JESP for the finite-horizon setting. Empirical results showed that this initialization method could, in some cases, reach good solutions with a value higher than the average answer of Inf-JESP with random initialization. However, this approach does not always work. In the Recycling problem, it is worse than the average Inf-JESP value with random initializations. How to derive (possibly randomly) better heuristic initializations from MPOMDP policies remains an open question.

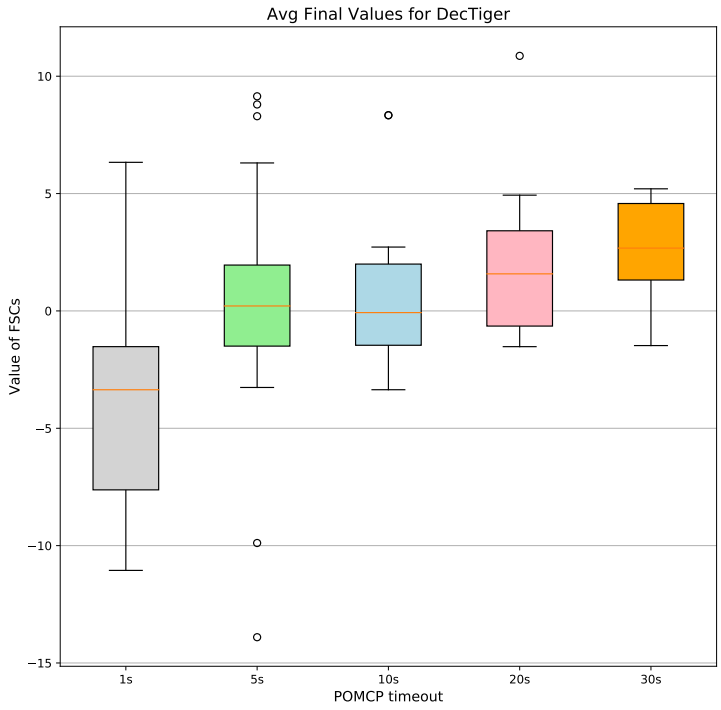

Figure 4.12: Values of the joint policy for the Dec-Tiger problem. The x-axis indicates the different POMCP timeout used when building the FSC node in the MC-JESP algorithm. The y-axis represents the final value obtained through the MC-JESP method.

MC-JESP Based on the approach followed for Inf-JESP, we propose the MC-JESP algorithm, which only requires a generative Dec-POMDP model. We describe how to build the best response generative model (the generative model of the POMDP faced by some agent i assuming known FSCs for other agents), and the process to extract an FSC for each agent. Moreover, we also provide a heuristic initialization for MC-JESP, which leads to good solutions confirmed by our experiments. To sum up, MC-JESP has the following properties:

- It does not require an explicit model, a black box simulator being sufficient. Therefore, this method can scale to very large problems where Inf-JESP cannot;

- It has good results similar to Inf-JESP, and even performs better than many explicit model-based methods. To our knowledge, MC-JESP seems to be a very good solution compared with existing sampling-based methods for solving Dec-POMDPs;

- The good performance may be due to the "best-response iteration" itself but is not linked to the completeness of the model (explicit or generative model). This would explain why MC-JESP and Inf-JESP share the same good performance.

4.4.2 Perspectives

For future works, we summarize several directions worth further investigation for Inf-JESP and MC-JESP.

Influence of different parameters and possible optimizations

First, we could do more experiments with different parameters to analyze our contributions as follows:

- different FSC size limits: in this thesis, we don't give an explicit FSC size limit for Inf-JESP, and we could develop a strategy that increases the FSC size limit through consecutive restarts;
- different best-response POMDP formalizations: in Section A.1, we also provide other possible formulations to build best-response POMDPs;
- different heuristic initialization methods: our current heuristic initialization method is based on extracting the individual policies from the MPOMDP solution, other heuristic initialization methods should be investigated.

Through experiments, we also demonstrate that parameters greatly influence our algorithms, especially for MC-JESP. Therefore, an improvement could be designing an online parameter optimization process that uses the data gathered through multiple restarts.

Second, both Inf-JESP and MC-JESP have the potential to further speed up with parallel restarts to find the best solution. Moreover, MC-JESP could greatly improve its efficiency by using the previous policy tree computed by the POMCP for the child nodes' computations. In the current setting, each node in an FSC is computed with a new POMCP from scratch.

Last but not least, in MC-JESP, we currently evaluate agents' FSCs in each iteration with a fixed number of simulations. An improvement here is adapting tools from *statistical hypothesis testing* (or developing new ones using *concentration inequalities*) to verify if there is a value increase after each iteration, *e.g.*, to check if fsc_i better or worse than fsc'_i with 0.95 probability, or if the difference is negligible.

Extending MC-JESP to continuous domains

In this thesis, MC-JESP is tested with discrete Dec-POMDP problems. However, as a simulation-based solver, it can be improved to solve continuous problems (continuous states, actions, and observations). To do so, we consider to apply *double progressive widening* when building each agent's FSC, which is already utilized to handle continuous POMDPs [Sunberg and Kochenderfer, 2018].

Evaluating the difficulty of a collaboration problem

There are multiple ways to achieve a given task in many collaboration problems. If we model such a collaboration problem with a Dec-POMDP, current solvers only give one best joint policy for all agents. However, if the agents in one team do not follow the same joint policy, a question is thus raised how to evaluate the difficulty of collaboration for a given Dec-POMDP. One way to assess the difficulty of collaboration is to look at whether several optimal joint policies exists and, if so, whether they can be "mixed" successfully or not. For a 2-agent problem, assuming one can generate N FSC pairs $\langle fsc_i^1, fsc_i^2 \rangle_{i=1}^N$ using Inf-JESP of MC-JESP, one could evaluate every

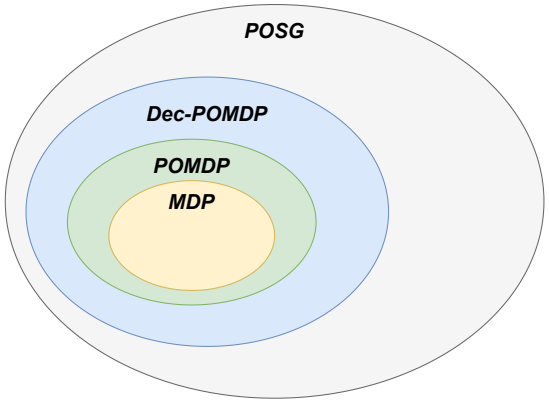

Figure 4.13: An illustration of Markov decision models and POSGs: MDP is a subset of POMDP, which is again a subset of Dec-POMDP. Dec-POMDP is a special case of POSG, in which all agents share a same reward function.

pair $\langle fsc_i^1, fsc_j^2 \rangle_{i,j=1}^N$ to see how good they perform. High overall scores would mean that no prior coordination is necessary in that problem.

We believe this approach will be useful in human-robot collaboration settings, where the human partner may often behave differently than a computed joint policy. For example, in a high-score collaboration problem, it is "safe" to equip the robot with a policy computed from a Dec-POMDP solver. However, if a low score is evaluated using this approach, it warns that the robot must be cautious and need to consider how to coordinate with the human.

Solving Some Partially Observable Stochastic Games (POSGs)

POSGs (partially observable stochastic games) provide a more general framework for multi-agent decision-making that each agent having its own reward function. A Dec-POMDP is a special case of POSG where all agents share the same reward function, thus, are willing to collaborate (see Fig. 4.13). In general POSGs, the notion of optimal solution does not exist, and one typically searches for a Nash equilibrium. Also, solution policies often need to be stochastic, particularly in competitive settings.

In this thesis, Inf-JESP or MC-JESP search for deterministic Nash equilibria for Dec-POMDPs. One question is whether they could be adapted for "collaborative POSGs", where the agents' objectives (reward functions) are close to each other, in particular, to avoid unstable behaviors (oscillations). For more general cases in POSGs, a tentative approach is to ensure that each iteration of Inf-JESP or MC-JESP will not decrease the value for any agent. However, how to modify the value function to achieve this purpose is still an open question.

5

Related Work on Task Planning for Human-Robot Collaboration

5.1 Overview of Task Planning in HRC

In Chapter 2, we describe several decision-making frameworks, including MDPs and POMDPs for the single-agent setting, and Dec-POMDPs for the multi-agent setting. For the multi-agent setting, we also provide a review of representative algorithms for solving Dec-POMDPs in Chapter 3 and give our contribution in this domain (see Chapter 4). In the HRC context, such a Dec-POMDP framework cannot be directly used since human partners may behave differently than the computed Dec-POMDP policies for several reasons:

- the computed policies are too complex for the human to learn;
- the human wants to be autonomous and is unwilling to blindly follow a pre-computed policy.

In some problems, there are unique and obvious ways of collaborating, but, in general, there may be multiple solutions, in particular, if the human also considers sub-optimal ones because he/she is not fully rational. As a consequence, for a given objective in HRC, the robot faces uncertain human behaviors, and we need to consider approaches that anticipate the human's possible behaviors. Moreover, unlike in a Dec-POMDP, where all agents optimize a shared objective function, the humans' real objectives in an HRC task may be hidden from the robots. Due to those factors, from the robot's point of view, to enable a successful collaboration in an HRC task: 1. the robot should have an idea about all the possible human objectives and behaviors and; 2. the robot needs to infer the human's current objective by considering these possibilities during execution.

However, a real question is how can we make the robot consider this uncertainty regarding human objectives and induced behaviors? In this thesis, inspired by Tabrez et al.'s work, we try to answer this question using the concept of *mental model* [Tabrez et al., 2020] to describe an explicit human model for the robot to predict human behaviors and infer his/her objective. Theoretically, in HRC, a robot could possess a mental model of a human (which may differ from actual human behaviors) among one of the following types:

- first-order mental model (1oMM): if a robot has a 1oMM of the human, then the robot assumes the human only considers himself when taking actions, *i.e.*, the robot thinks the human behaves independently without considering the robot;

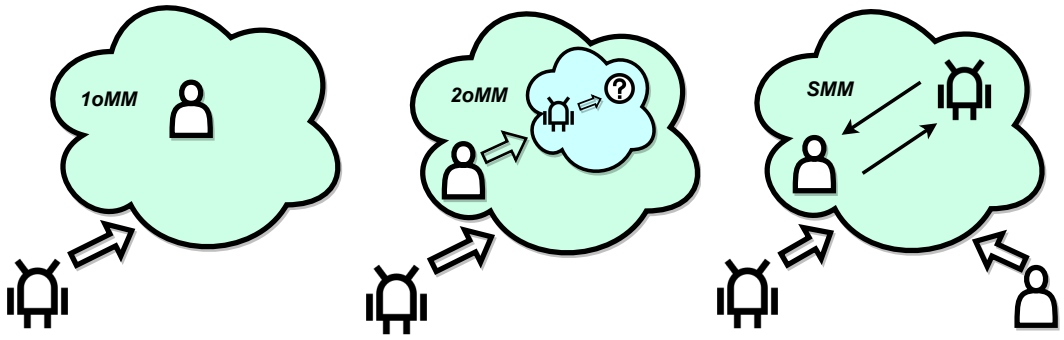

Figure 5.1: An illustration of the three types of mental models: first-order mental model (1oMM), second-order mental model (2oMM), and shared mental model (SMM).

- second-order mental model (2oMM): in this case, the robot assumes the human also takes the robot into account, *i.e.*, the robot thinks the human also has a mental model of the robot, which makes the human behaviors are not independent and will be influenced by the robot's actions;
- shared mental model (SMM): in the shared mental model setting, the human and the robot rely on common knowledge and common reasoning to make coordinated decisions.

An illustration of those mental models (for modeling a human) from the robot's point of view is presented in Figure 5.1.

One can use mental models for human-robot collaboration for several purposes. For example, when robots are equipped with proper mental models of the human, they can better fit the collaboration task with improved performance and gain the trust of the human partner [Arnold et al., 2018, Tabrez and Hayes, 2019, Tabrez et al., 2020, Chen et al., 2020]. In this thesis, besides the uncertainty about human objectives and behaviors, we would like to address another challenge regarding coordination. In many HRC tasks, it is required that the human and robot work properly to adapt to each other's behaviors. For example, a repairing task needs the human and the robot to perform correctly repairing actions simultaneously; otherwise, the reparation will fail. The human and the robot should thus coordinate their behaviors at some point in order to successfully achieve the collaboration task. Therefore, for each mental model, we are interested in the following questions:

- How is the considered agent's **uncertainty** presented in the mental model?
- Does this mental model enable the **coordination** between humans and robots?

This chapter is organized as follows. A review of related works using 1oMMs or 2oMMs is given in Sec. 5.2 and Sec. 5.3. In Sec. 5.4, we present related works linked to SMM. We conclude by discussing the pros and cons of those mental models and position our contributions in Section 5.5.

5.2 HRC with first-order mental model

The first-order mental model (1oMM) is the most basic form for modeling other agents. The robot with a 1oMM of the human will assume its human partner only considers himself when

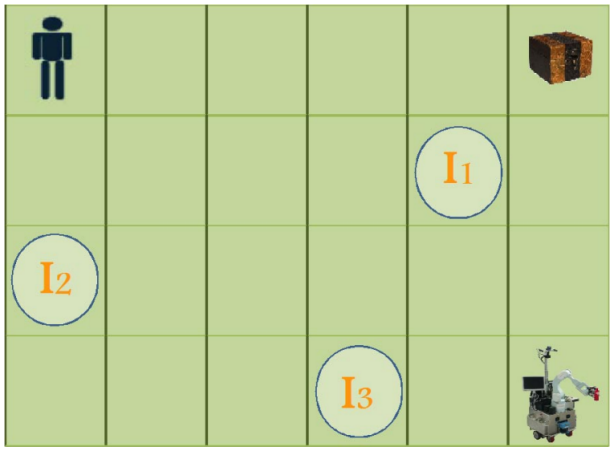

Figure 5.2: A collaborative collecting task [Karami et al., 2009]: In the grid-world environment, there are three objects I_1, I_2, and I_3. The human and the robot need to collect them to the box at the top right corner.

taking actions. This simple assumption allows computing a human model by reasoning on a single-agent problem since the human does not consider the robot's presence or the actions the robot can perform. This implies that the problem can be solved without any help from the robot.

Planned human policy inside a robot POMDP Karami et al. propose such an HRC scenario, as shown in Figure 5.2, where the robot has a 1oMM of its human partner [Karami et al., 2009, Karami, 2011]. In this work, the goal of the human-robot team is to collect all items to the target box, and an MDP is used to define the collaboration task's underlying transition and reward functions. During the task, the robot must respect the human's current intention I_h, which means the robot should avoid collecting the same object as the human. The robot could observe the executed human action at each time step. However, from the robot's point of view, the human's intention I_h is hidden. Therefore, they model the robot's decision process as a POMDP where I_h is an unobservable variable.

More specifically, Karami design the robot decision framework as shown in Figure 5.3 with following steps:

- A set of human MDPs is built for each human intention. In each human MDP, the human has full observability of the environment and does not account for robot behaviors.
- The human MDPs are solved to generate a library of Q-values (human action values), which is a mapping from pairs of environment states and human intentions to values.
- Those Q-values are used to derive a probability distribution of human actions in each situation (environment state + human intention) and are considered as parts of the dynamics in robot POMDP. One should note that, in Karami's work, the robot can observe the human action performed.

The robot policy is then computed by solving this robot POMDP. Through experiments in this specific scenario, they show that the robot can estimate the human itention and make its decisions accordingly.

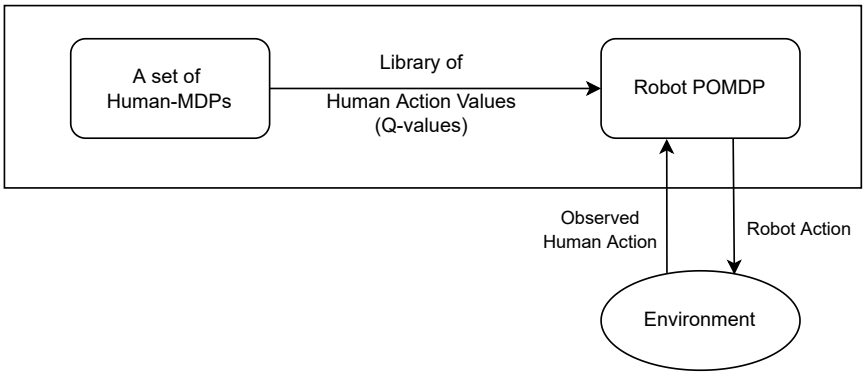

Figure 5.3: The high-level process of the robot decision system [Karami, 2011]

Other related works Similar "robot POMDPs" are solved in other existing works [Nikolaidis et al., 2016, 2017]. For example, Nikolaidis et al. introduces a bounded memory adaption model (BAM), which captures human adaptive behaviors based on an assumption of bounded memory during interactions [Nikolaidis et al., 2016]. Then, the robot integrates BAM into a POMDP decision framework to reason about its actions. But in this work, another interesting idea is the human's bounded rationality (linked to bounded memory). Their HRC experiments demonstrate that the robot can adapt to erratic human behaviors that are not optimal for a given human objective function. One can identify similar approaches to adding noise in considered human behaviors or parameterized human action selection function [Osogami and Otsuka, 2014, Otsuka and Osogami, 2016, Nikolaidis et al., 2017, Chen et al., 2018].

Limitations of 1oMMs for HRC However, coordination between the real human and robot may be difficult when the robot uses a 1oMM of the human. For example, in a cooperative repairing scenario, only both agents simultaneously repairing could fix a damaged device. But if the robot (with a 1oMM about the human) assumes that the human does not consider the robot's behaviors (which may differ from the real human), then the robot will deduce that the human will not be able to carry out the repair action in time. Consequently, the robot has no reason to attempt the repair action since it assumes this action cannot bring any benefit. Some existing works, including the one we mentioned above [Karami et al., 2009, Karami, 2011], do not require coordination between humans and robots to finish the task. The robot's role is more like a helper or assistant who accelerates or simplifies the ongoing tasks.

How to Derive a 1oMM of the human In HRC, there are several ways to build a 1oMM about the human (for the robot):

1. a hand-made human policy [Nikolaidis et al., 2016, Chen et al., 2018], where the designer designer use expert knowledge to implement a 1oMM of the human;

2. a learned human policy by observing one's behaviors [Chen et al., 2020], or human's reward function using inverse reinforcement learning (IRL) [Russell, 1998, Ng and Russell, 2000], which estimates the parameters of a function that best explains the observed behaviors;

3. a planned human policy where human behaviors are derived by solving specific problems, *i.e.*, human-MDPs [Karami et al., 2009].

To sum up, the 1oMM is easy to understand and effective in several scenarios. Moreover, to let the robot consider a variety of human behaviors, the action selection function can be parameterized with different factors such as "memory" or "trust" in the robot's 1oMM of the human or simply adding noises. However, the main drawback of 1oMMs is also obvious. It will fail in many tasks requiring explicit collaborative behaviors from the human. In other words, 1oMMs are suitable for those tasks that a human could achieve on his own, and the robot helps to accelerate or simplify the process.

5.3 HRC with Second-order Mental Model

Compared with a 1oMM, a robot with a 2oMM of the human will consider that the human behaves while taking the robot into account. But then, a question is raised: what is the human's mental model of the robot? Is it a 1oMM, a 2oMM or an SMM? As can be observed, 2oMMs can be recursively nested with an infinite depth, as illustrated in Figure 5.1 (middle), which may be interpreted as "I believe that you believe that I believe ..." [Tabrez et al., 2020]. The nesting can be finite if stopped with a final 1oMM or SMM. Such nested consideration is essential and often used in our daily lives, such as in negotiations or playing chess. In those cases, one must consider the possible actions the others might take and how their behaviors may trigger others' different strategies.

Reasoning about Others In 1980s, researchers already paid attention to this kind of nested-reasoning decision problem [Maida, 1986, Wilks and Ballim, 1987]. For example, Wilks and Ballim propose a method to generate nested beliefs heuristically for natural language understanding about "what some agents believe on others' beliefs given a specific topic". Then, with the progress of the decision-making community, especially in the Markov decision process domain, based on the POMDP framework, Doshi and Gmytrasiewicz proposed the *Interactive POMDP* (I-POMDP) formalism [Doshi and Gmytrasiewicz, 2005], which models the decision-making problem with one agent with recursive reasoning about other agents. An interactive POMDP for agent i, denoted $I\text{-}POMDP_i$, is defined as follows:

$$I\text{-}POMDP_i = \langle IS_i, A, T_i, \Omega_i, O_i, R_i \rangle,$$

where $A = A_i \times A_j$ is the joint action set of all agents, and Ω_i is agent i's observation space. T_i, O_i and R_i are the underlying dynamics in agent i's view. Here, one may wonder how to describe these dynamics with only agent i's action since we are in a multi-agent setting. This is linked to the *interactive states* is_i, which belongs to IS_i, a set of interactive states which is a combination of world states and other agent j's possible models $IS_i = S \times M_j$. An agent j's model m_j is defined as

$$m_j = \langle h_j, f_j, O_j \rangle, \quad m_j \in M_j, h_j \in H_j,$$

where:

- h_j is agent j's observation history, and H_j is the set of all the possible histories of agent j;
- f_j is agent j's action-selection function, assumed computable, which maps possible observation histories of agent j to distribution over its actions;
- O_j is the observation function of agent j.

To that end, one can construct M_j with two approaches, either the agent j models other agents again (recursive modeling) or simply uses a naive assumption such as agent j behaves with a uniform action distribution (in this case, any $m_j \in M_j$ is a 1oMM for agent i). Although this is a complex formalization, it is generic enough to represent agent i's decision, and its belief over the interactive states is a sufficient statistics. Therefore, many works for interactive applications adapt the I-POMDP formalization, such as in dialogue systems [Wang, 2013] and turn-based games [Rosello, 2016], and provide scaling-up extensions [Hoang and Low, 2013, Han and Gmytrasiewicz, 2019].

Comparison with 1oMMs Regarding coordination, 2oMMs give one possible path to let agents consider each other. This inter-consideration enables the coordination between agents. However, as already mentioned, a chicken-and-egg problem becomes unavoidable with 2oMMs. When we build a 2oMM of a human for the robot, the human behaviors are also influenced by the robot's behaviors which we are trying to derive in the first place. For example, learning a 2oMM from real human behaviors would require that a working collaborative robot already exists to demonstrate collaborative behaviors. This issue is also unavoidable when we model the robot's decision-making with 2oMM of the human as an I-POMDP described above.

Regarding the diversity of human behaviors, 2oMMs share the same properties as 1oMMs. The considered agent may have different probabilities on his action selections. But compared with 1oMMs, the reward which drives the agents' behaviors in 2oMMs can be related not only to its actions but also to the actions of all agents.

Given those pros and cons, 2oMMs are suitable for complex collaboration HRC tasks where humans and robots must consider each other's behaviors and coordinate their actions. The main disadvantage is that we need to address the chicken-and-egg problem when building a 2oMM.

5.4 HRC with Shared Mental Model

In a shared mental model, all team members assume they have the same expectation and reason in the same manner. This setting enables all agents in a team can coordinate their behaviors and lead to performance improvement regarding the joint policy [Tabrez et al., 2020]. For example, if we solve a multi-agent problem modeled using a Dec-POMDP and obtain an optimal joint policy, then it implies that each agent knows other agents will behave according to the computed joint policy, which naturally means all agents have a shared mental model.

Human-Robot Cross-Training Based on the SMM assumption, Nikolaidis and Shah propose a human-robot cross-training framework where the shared mental model is formulated as an MDP [Nikolaidis and Shah, 2013]. In their work, the robot policy is aligned with human preferences, and the robot will switch its role with the human iteratively during training. Compared with the standard reinforcement learning method, in which the human and the robot never switch their roles. They show that this cross-training approach helps the team learn a shared plan for a collaborative task and significantly improves performance in different aspects, including trust and fluency.

Cooperative Inverse Reinforcement Learning Another well-known work based on the SMM is the *cooperative inverse reinforcement learning* (CIRL) framework proposed by Hadfield-Menell et al., which is designed for a cooperative human-robot team setting [Hadfield-Menell et al., 2016]. All state variables are visible in a CIRL problem except the human's actual reward

function, represented by a generalized parameter θ, which is hidden to the robot. Both the human and the robot should maximize the human's reward function. Since only θ is hidden from the robot, CIRL reduces this two-agent cooperative decision problem to a POMDP and seeks a pair of policies (π_H, π_R) for the human and robot. Under the shared mental model assumption, CIRL assumes that the human will follow the computed policy π_H. Therefore, the computed joint policy of the human and robot is optimal, which maximizes the human's true objective θ. Moreover, they point out that such a solution can be used for active teaching (human) and active learning (robot) behaviors.

Pros and Cons Technically, the main disadvantage of using the shared mental model in human-robot collaboration is its strong assumption of sharing policies (or action selection functions) among all agents. In practice, this condition is often too difficult to satisfy or requires specially designed scenarios [Nikolaidis and Shah, 2013]. But if this assumption is satisfied, SMMs seem to be an ideal tool for enabling coordination between humans and robots (Using SMMs allows, in theory, for optimal coordination, but computing an optimal solution may be very difficult, cf. Dec-POMDP solving complexity). Given a known objective for the human-robot team, there is no need to consider others' diversities in their behaviors since the knowledge of action selection during the task is shared. On the other hand, to build an SMM between the human and robot, one way is to relax the HRC problem to a multi-agent problem, for which we compute a joint policy of all agents. Then, depending on the uncertainties and observations, standard methods for POMDPs or Dec-POMDPs can be used for such purposes. Or another approach is to use the cross-training method as presented in Nikolaidis and Shah's work so that the human and the robot can all agree to converge on a shared plan.

5.5 Conclusion

To sum up, for robots' decision-making in an HRC task, robots equipped with 1oMM can provide effective policies for simple tasks where no mandatory coordination is needed between humans and robots. 2oMMs, on the other hand, let the robot consider the possible behaviors of the human and vice-versa, thus enabling necessary coordination, but a chicken-and-egg problem will appear, and one needs to design a specific solution at a certain depth to stop this recursive nesting. SMMs rely on a strong assumption of common expectations, thus, they are suitable for teaching or being guidelines for the human partner to learn, but it may be complex given the HRC problem at hand.

We conclude by presenting Markov decision models that correspond to the three types of mental models as in Table 5.1:

- Single-agent models (MDPs and POMDPs) allow modeling robots with a 1oMM or 2oMM of the human, by incorporating the human model in the system dynamics (human is seen as part of the robot's environment).

- I-POMDPs are meant to model finitely nested 2oMMs. But one can also transform an I-POMDP into a POMDP with an extended state to express that recursive reasoning information.

- M(PO)MDPs and Dec-POMDPs are meant to model collaborative problems, thus SMMs (assuming that all agents fully collaborate and optimize the exact same performance criterion). In MPOMDPs, all agents have access to the same information at execution time, so

Table 5.1: Table of relations between mental models and Markov decision frameworks

Mental Model Type	1oMM	2oMM	SMM
Decision Making Framework	(PO)MDP	POMDP, I-POMDP	M(PO)MDP, Dec-POMDP

solving an MPOMDP amounts to solving a POMDP; In MMDPs, all agents can directly access the real environment state at each time step, which equals solving an MDP; Dec-POMDPs represent problems where each agent has its own view of the current situation.

In this thesis, we are interested in collaboration tasks where one human and one robot need to coordinate their behaviors, and the ability to consider others' behaviors is necessary. Therefore, in the robot's view, a 1oMM of the human is unfit in this case, and SMM's assumption of common exceptions is too strong and unrealistic in our setting. Thus, we have chosen a 2oMM to model the human for the robot, and propose our contributions to solve the chicken-and-egg problem in Chapter 6.

One could view our approach as improving Karami et al.'s work (see Section 5.2, page 67) on the following aspects:

- We choose to equip the robot with a 2oMM of the human, which enables necessary coordination, but is not possible in Karami et al.'s work with 1oMMs;
- We extend their work to partial observability settings where each agent only observes partial information;
- We provide a tunable method to adjust the rationality of the human considered by the robot.

Moreover, no prior human behavior is needed in our approach, therefore it can tackle various HRC problems.

6

Robust Decision Making for Human-Robot Collaboration

In this chapter, we present our contributions to generate robust robot policies for human-robot collaboration (HRC) tasks. Unlike the multi-agent decision problems modeled as Dec-POMDPs, where optimal policies are computed for all agents, we can only control the robot in HRC, and the human exhibits uncertain behaviors. This being said, from the robot's point of view, an issue is how to be robust against uncertain human objectives or uncertain human behaviors, even given a known objective. Usually, the robot relies on a mental model of the human partner to predict human actions and then make its own decisions. In Chapter 5, we have discussed three possible (robot's) human mental models as follows:

- First-order mental model (1oMM): the robot considers that the human behaves independently and does not account for the robot's possible actions;
- Second-order mental model (2oMM): the robot thinks that the human also accounts for the robot, which induces recursive modeling up to a certain depth;
- Shared mental model (SMM): all agents have common expectations and reason in the same manner, thus ensuring optimal coordination between the human and the robot.

In our work, we do not consider 1oMMs since our HRC task needs the human and the robot to coordinate their behaviors, requiring the human to consider the robot's behaviors, and the SMMs' assumption about common expectations is too strong for collaborations with real humans. Therefore, we choose to equip the robot with a 2oMM of the human partner. However, as we mentioned in Chapter 5, a 2oMM raises a chicken-and-egg problem since modeling required human behaviors implies reasoning about the robot behavior we are trying to derive in the first place, as shown in Figure 6.1 (a).

In this thesis, we propose a way to address this chicken-and-egg problem by answering the question: *"What would be the human policy if he or she could also control the robot?"* To do this, we temporarily assume that the human and the robot share their observations, which makes the human (the one in the robot's mental model, not the real human) consider a particular SMM, as presented in Figure 6.1 (b). Then, we develop a general 3-step scheme for HRC tasks as follows:

1. solving the collaboration problem under the SMM assumption;
2. extracting a human model from the SMM solution;

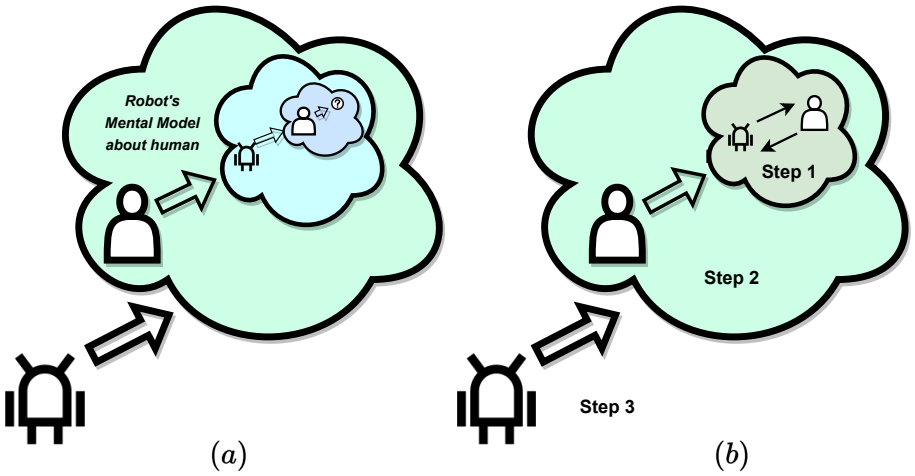

Figure 6.1: A comparison of two 2oMMs: (a) the first 2oMM has a chicken-and-egg issue with an infinite loop of considerations; (b) the robot in the second one assumes that the considered human behaves according to an SMM.

3. computing a robot policy using the generated human model.

Based on this general scheme, two algorithms (see Section 6.1 and Section 6.2) are proposed in this thesis for deriving such human mental models and robot policies, which respectively operate in an offline and online manner. The human mental model is dedicated to deriving robot policies robust to a variety of possible actual human behaviors. This model is thus not meant to represent any human in particular. Our approach should ideally be evaluated against a variety of actual human behaviors

This chapter is organized as follows: In Section 6.1 and Section 6.2, we present two original methods: an offline and online robot robust planning algorithm for HRC tasks. Finally, Section 6.3 presents empirical results obtained with both synthetic and real humans on a high-level task in a simulated environment.

6.1 Offline Robust Planning for HRC

As previously mentioned, this chapter aims to address the uncertainty of the human's actual objective and exhibited human behaviors to derive a robust robot policy for HRC tasks. To do so, we first present the offline algorithm for building a robust robot policy, which is illustrated in Figure 6.2. First, we use Dec-POMDPs to model the collaboration problem for the various possible human objectives. These Dec-POMDPs only differ in their reward functions, each modeling a specific human objective. Then, for each human objective (each Dec-POMDP), we generate a stochastic FSC representing a variety of possible human behaviors as described in Section 6.1.1. A robust robot POMDP decision model is built in Section 6.1.2 based on the Dec-POMDP task model and the generated human stochastic FSCs. In the end, we use SARSOP [Kurniawati et al., 2008] to solve this robot POMDP and get a robust robot policy.

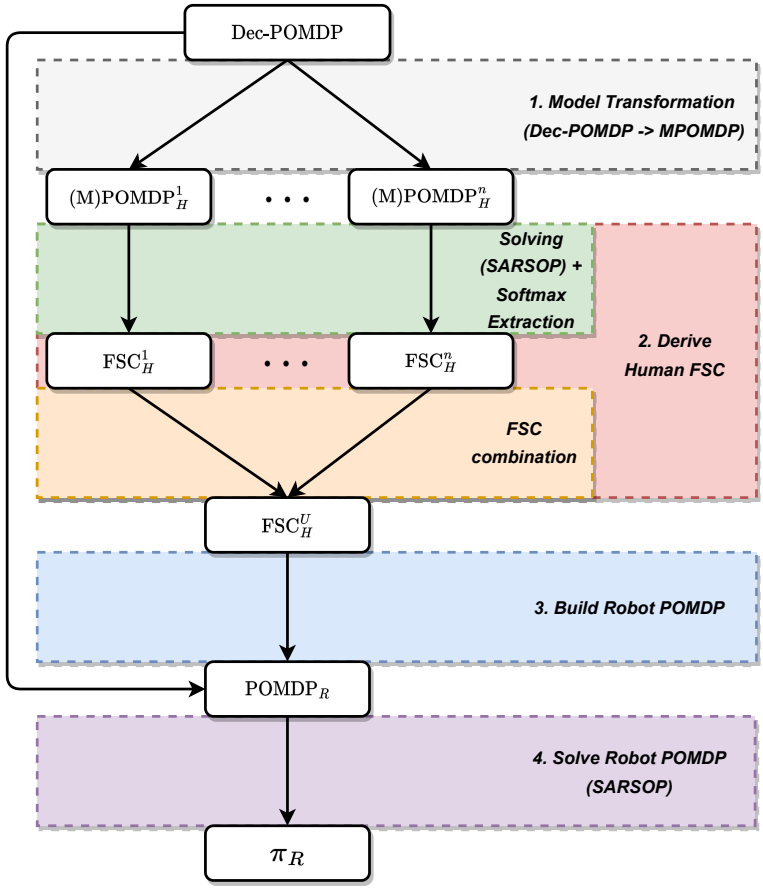

Figure 6.2: The structure of the offline robust robot planning algorithm

6.1.1 Generating Human Stochastic FSCs with a Bounded Size

As stated before, an issue when building the human mental model for the robot is that the human policies depend on the robot policies we don't have in the first place. To overcome this chicken-and-egg problem:

- We make a temporary assumption that the human in the mental model behaves according to an SMM, in which the human can control the robot with direct access to its observations (see Figure 6.1(b) - Step 1);
- Then, we extract a human policy presented by a stochastic FSC from the SMM solution, corresponding to the human model presented in Figure 6.1(b) - Step 2.

This approach ensures that the generated human FSC considers robot behaviors since this human FSC is extracted from the SMM solution, which accounts for both agents' behaviors.

To be more specific, for each human objective, the corresponding Dec-POMDP is relaxed to an MPOMDP (Multi-agent POMDP) [Pynadath and Tambe, 2002], *i.e.*, a single-agent problem that an agent has access to the joint observation and control the joint action (the first part in Figure 6.2) given the SMM assumption. This MPOMDP can be fed to a POMDP solver to compute an optimal joint policy, which maps beliefs (or joint action-observation histories) to joint actions of the human and the robot, corresponding to the Step 1 part of Figure 6.1(b).

However, this joint policy is not directly usable for the robot since it's not a proper human behavior model, *i.e.*, a mapping from human action-observation histories (alone) to human actions. To do so, we extract a human policy (Figure 6.1(b) - Step 2) from the joint policy assuming that the human:

1. does not control the robot anymore; and
2. considers that the robot still has access to the same belief b as inferred by the human.

Then, given a belief b, we can model the uncertainty over human behaviors using a softmax function over action values to obtain a distribution over multiple optimal or near-optimal actions:

$$f(a|T,b) = \frac{e^{\frac{Q^*(b,a)}{T}}}{\sum_{a'} e^{\frac{Q^*(b,a')}{T}}},$$

where $T > 0$ is a temperature parameter that makes the human more *rational* if T is low, only optimal actions being selected, and more *erratic* if T is high, with a distribution close to uniform. The robot considers that the human selects actions according to this softmax distribution. Moreover, since the human and the robot are now independent, and the human considers the robot has access to the same belief that $b_H = b_R = b$, we can define their action sampling rules σ_H^T and σ_R^T as follows:

$$\sigma_H^T(a_H|b) \stackrel{\text{def}}{=} \sum_{a_R} f(a_H, a_R|T,b), \text{ and}$$
$$\sigma_R^T(a_R|b) \stackrel{\text{def}}{=} \sum_{a_H} f(a_H, a_R|T,b).$$

Knowing these two action sampling rules, the human is able to update his belief in a Dec-POMDP given his last pair (a_H, o_H) by marginalizing the robot's private actions and observations as follows:

$$b'(s') = Pr(s'|b, a_H, o_H, \sigma_R^T(\cdot|b)) = \frac{Pr(s', o_H|b, a_H, \sigma_R^T(\cdot|b))}{\sum_{a_H} Pr(s', o_H|b, a_H, \sigma_R^T(\cdot|b))} \tag{6.1}$$

$$\propto \sum_{s,a_R} Pr(s', o_H|s, \langle a_H, a_R\rangle) \cdot \sigma_R^T(a_R|b) \cdot b(s) \tag{6.2}$$

$$= \sum_{s,a_R,o_R} Pr(s', \langle o_H, o_R\rangle|\langle a_H, a_R\rangle) \cdot Pr(s'|s, \langle a_H, a_R\rangle) \cdot \sigma_R^T(a_R|b) \cdot b(s) \tag{6.3}$$

$$= \sum_{a_R} \left(\sum_{o_R} O(\langle a_H, a_R\rangle, s', \langle o_H, o_R\rangle) \right) \cdot \left(\sum_{s} T(s, \langle a_H, a_R\rangle, s') \cdot b(s) \right) \cdot \sigma_R^T(a_R|b). \tag{6.4}$$

As expected, it relies solely on the Dec-POMDP's transition and observation functions, the previous belief b, and the robot's action sampling rule for that belief (and conditioned on the current human action).

Sampling a Stochastic Human FSC Following these ideas and processes we mentioned above to pick a human action and to update his belief, we first designed Algorithm 14, which recursively extracts a human policy (represented as a stochastic FSC) building on the method

presented in Section 4.1.3 for standard POMDPs. In the beginning, a unique start node n_0 is created (line 2) with $\sigma_H^T(\cdot|b_0)$, $\sigma_R^T(\cdot|b_0)$, b_0 and the initial weight $w = 1$. Then n_0 is added to both the new FSC (N) and an open list (G) (line 3). Getting inspiration from LAO* [Hansen and Zilberstein, 2001], we would like to expand the node with the greatest contribution to b_0's value at each iteration. Thus, while G is not empty, we select the node $n \in G$ that has the highest estimated value $V^*(n.b)$, weighted by the probability of reaching that node (estimated through $n.w$) (line 6), then process that node with possible observation-action pairs under current belief b (lines 7–9), impossible observation-action pairs inducing a self-loop (line 22). Those self-loops ensure that the resulting human FSC can be used whatever the robot's actual behavior (which may cause unexpected human observations). Lines 10 and 11 compute the updated belief b' and its weight w' (see detailed formulas below). Before creating a new node n' with those new computed components, we also need to check:

1. if the new belief b' does not already exist inside the FSC (N) considering a *Norm-1* distance threshold ϵ:

$$b' \in N(\epsilon) \text{ iff } \exists n \in N \text{ s.t. } \|n.b - b'\|_1 (\stackrel{\text{def}}{=} \sum_s |n.b(s) - b'(s)|) \leqslant \epsilon;$$

2. if the FSC does not already contain the maximum node size $N_{\max}$.

If both conditions are satisfied (line 12), we create a new node n' and add it to N. Otherwise, we find and process the node n' in N which has the closest belief to the new computed belief b'. The probability used for updating weights in line 11 is derived from

$$Pr(o_H, a_H|b, \sigma_H^T(\cdot|b), \sigma_R^T(\cdot|b)) = \sum_{a_R} Pr(o_H, \langle a_H, a_R\rangle|b, \sigma_H^T(\cdot|b), \sigma_R^T(\cdot|b)) \tag{6.5}$$

$$= \sum_{a_R, o_R, s, s'} Pr(s, s', \langle o_H, o_R\rangle, \langle a_H, a_R\rangle|b, \sigma_H^T(\cdot|b), \sigma_R^T(\cdot|b)) \tag{6.6}$$

$$= \sum_{a_R, o_R, s, s'} b(s) \cdot Pr(s', \langle o_H, o_R\rangle|s, \langle a_H, a_R\rangle) \cdot \sigma_H^T(a_H|b) \cdot \sigma_R^T(a_R|b) \tag{6.7}$$

$$= \sum_{a_R, o_R, s, s'} b(s) \cdot Pr(\langle o_H, o_R\rangle|s, \langle a_H, a_R\rangle, s') \cdot Pr(s'|s, \langle a_H, a_R\rangle) \cdot \sigma_H^T(a_H|b) \cdot \sigma_R^T(a_R|b) \tag{6.8}$$

$$= \sum_{a_R, o_R, s, s'} b(s) \cdot O(\langle a_H, a_R\rangle, s', \langle o_H, o_R\rangle) \cdot T(s, \langle a_H, a_R\rangle, s') \cdot \sigma_H^T(a_H|b) \cdot \sigma_R^T(a_R|b). \tag{6.9}$$

This approach extracts a bounded-size stochastic FSC $\langle N, \eta, \psi\rangle$, which includes $|N|$ nodes, encoding several human behaviors. For each node $n \in N$, the human's stochastic action selection function is defined as $\psi(n, a_H) = \sigma_H^T(a_H|n.b)$. Deterministic transitions between nodes are stored in η and depend on the human action-observation pair $\langle a_H, o_H\rangle$. Note that our algorithm allows trading of the FSC quality with its complexity through

1. bounding the FSC's number of nodes by $N_{\max}$;
2. merging nodes when their reference beliefs (the beliefs used when creating the nodes) are within ϵ of each other.

In our experiments, all $Q^*(b, a)$ and $V^*(b)$ values are estimated using 1. SARSOP for offline pre-computations [Kurniawati et al., 2008], and 2. POMCP to obtain good estimates online quickly, even in beliefs not visited by an optimal policy [Silver and Veness, 2010].

Algorithm 14: Extract a human stochastic FSC

[Input:] T: softmax temperature | $N_{\max}$: max # of nodes | ϵ: max belief gap
$n_0 \leftarrow node(\langle \sigma_H^T(\cdot|b_0), \sigma_R^T(\cdot|b_0), b_0, w = 1 \rangle)$
$N \leftarrow \{n_0\}$
$G.pushback(n_0)$
while $|G| > 0$ **do**
  $G.sort()$ // sort nodes according to $w \cdot V^*(b)$
  $n \equiv \langle \sigma_H^T, \sigma_R^T, b, w \rangle \leftarrow G.popfront()$
  forall $\langle o_H, a_H \rangle \in \Omega_H \times A_H$ **do**
    if $Pr(o_H, a_H | b, \sigma_H^T) > 0$ **then**
      $b' \leftarrow updateBelief(b, a_H, o_H, \sigma_R^T(\cdot|b))$
      $w' \leftarrow w \cdot Pr(o_H, a_H | b, \sigma_H^T(\cdot|b), \sigma_R^T(\cdot|b))$
      if $(b' \notin N(\epsilon)) \wedge (N.size() < N_{\max})$ **then**
        $n' \leftarrow node(\langle \psi(n', \cdot) \stackrel{\text{def}}{=} \sigma_H^T(\cdot|b'), \sigma_R^T(\cdot|b'), b', w' \rangle)$
        $N \leftarrow N \cup \{n'\}$
        $G.pushback(n')$
        $\eta(n, \langle a_H, o_H \rangle, n') \leftarrow 1$
      else
        $n' \leftarrow N.findClosest(b')$
        $n'.w \leftarrow n'.w + w'$
        $\eta(n, \langle a_H, o_H \rangle, n') \leftarrow 1$
    else
      $\eta(n, \langle a_H, o_H \rangle, n) \leftarrow 1$
return $\langle N, \eta, \psi \rangle$

Sampling a Deterministic Human FSC We also designed another method to extract a deterministic human FSC as in Algorithm 15. This algorithm will be used for generating human policies to simulate various humans in our experiments. As shown in red, it differs from Algorithm 14 only in that, in any node, a single human action is used, rather than a probability distribution over human actions. In other words, this method is achieved by replacing each node's distribution over human actions by a single action sampled from that distribution.

6.1.2 Building a robot best response policy

Here we want the robot to be robust to different possible (hidden) human objectives, each attached to different behaviors, trying to adapt to the human's actual objective. Please note that the collaboration task itself is modeled as a Dec-POMDP, except that:

- the exact reward function (among ρ candidates) is only known by the human;
- for each possible reward function r_i, we can compute a human behavior model (in the form of a stochastic FSC fsc_i); and
- the robot knows a probability distribution over the finitely many possible reward-FSC pairs: $P(\{(r_1, fsc_1), \ldots, (r_\rho, fsc_\rho)\})$.

Algorithm 15: Sample a human deterministic FSC

[Input:] T: softmax temperature | $N_{\max}$: max # of nodes | ϵ: max belief gap
$n_0 \leftarrow node(\langle a_H \sim \sigma_H^T(\cdot|b_0), \sigma_R^T(\cdot|b_0), b_0, w = 1\rangle)$
$N \leftarrow \{n_0\}$
$G.pushback(n_0)$
while $|G| > 0$ **do**
  $G.sort()$ // sort nodes according to $w \cdot V^*(b)$
  $n \equiv \langle a_H, \sigma_R^T, b, w\rangle \leftarrow G.popfront()$
  forall $o_H \in \Omega_H$ **do**
    if $Pr(o_H, a_H|b, \sigma_H^T) > 0$ **then**
      $b' \leftarrow updateBelief(b, a_H, o_H, \sigma_R^T(\cdot|b))$
      $w' \leftarrow w \cdot Pr(o_H, a_H|b, \sigma_H^T(\cdot|b), \sigma_R^T(\cdot|b))$
      if $(b' \notin N(\epsilon)) \wedge (N.size() < N_{\max})$ **then**
        $n' \leftarrow node(\langle a_H \sim \sigma_H^T(\cdot|b'), \sigma_R^T(\cdot|b'), b', w'\rangle)$
        $N \leftarrow N \cup \{n'\}$
        $G.pushback(n')$
        $\eta(n, \langle a_H, o_H\rangle, n') \leftarrow 1$
      else
        $n' \leftarrow N.findClosest(b')$
        $n'.w \leftarrow n'.w + w'$
        $\eta(n, \langle a_H, o_H\rangle, n') \leftarrow 1$
    else
      $\eta(n, \langle a_H, o_H\rangle, n) \leftarrow 1$
return $\langle N, \eta, \psi\rangle$

As detailed below, this robust robot behavior is obtained by

1. first turning this distribution over stochastic FSCs into a single FSC,
2. then using this FSC to derive a POMDP, and
3. finally solving this POMDP.

A candidate solution method to obtain one FSC for each reward function is presented in Sec. 6.1.1.

To turn this probability distribution over human FSCs into a single FSC, we take the "union" of these FSCs, the new distribution over initial nodes amounting to 1. sampling one FSC fsc_i from the distribution $P(\{fsc_1, \ldots, fsc_\rho\})$, and 2. sampling a node from β_i. More formally, noting a base FSC $fsc_i \equiv \langle N_i, \beta_i, \eta_i, \psi_i\rangle$, the *union FSC* is defined as:

$$N \stackrel{\text{def}}{=} \bigcup_{i=1}^{\rho} N_i,$$

$$\beta(n_i) \stackrel{\text{def}}{=} \beta_i(n_i) \cdot P(fsc_i),$$

$$\eta(n, \langle a, o\rangle, n') \stackrel{\text{def}}{=} \begin{cases} \eta_{i(n)}(n, \langle a, o\rangle, n') & \text{if } n' \in N_{i(n)}, \\ 0 & \text{otherwise,} \end{cases}$$

(where $i(n) \stackrel{\text{def}}{=} i$ s.t. $n \in N_i$, *i.e.*, $i(n)$ is the id of n's parent FSC), and

$$\psi(n, a) \stackrel{\text{def}}{=} \psi_{i(n)}(n, a).$$

Moreover, as detailed in Algorithm 14, each stochastic FSC node is linked to a specific human belief b_H^t, and each stochastic human FSC is linked to a human objective θ_H. Thus, using this union FSC, the pair of human's objective and internal belief $\langle \theta_H, b_H^t \rangle$ is now represented by the human union FSC node n_H. Given this union FSC, we can now formalize the robot's decision problem as a POMDP where each *extended state* $e^t \in \mathcal{E}$ contains a current world state s^t, a current robot observation o_R^t, and the current node n_H^t inside the human union FSC: $e^t = \langle s^t, n_H^t, o_R^t \rangle$. Based on the Dec-POMDP defining the task and on the associated union FSC, the dynamics and the reward function of the robot's decision problem can be written as follows:

$$\begin{aligned}
T_e(e^{t+1}, e^t, a_R^t) &\stackrel{\text{def}}{=} Pr(e^{t+1} | e^t, a_R^t) \\
&= \sum_{a_H^t} \sum_{o_H^{t+1}} T(s^t, \langle a_H^t, a_R^t \rangle, s^{t+1}) \cdot \eta(n_H^t, \langle a_H^t, o_H^{t+1} \rangle, n_H^{t+1}) \cdot \\
&\qquad O(s^{t+1}, \langle a_H^t, a_R^t \rangle, \langle o_H^{t+1}, o_R^{t+1} \rangle) \cdot \psi(n_H^t, a_H^t), \\
O_e(e^{t+1}, a_R^t, o_R^{t+1}) &\stackrel{\text{def}}{=} Pr(o_R^{t+1} | e^{t+1}, a_R^t) \\
&= \mathbf{1}_{o_R^{t+1} = \tilde{o}_R^{t+1}}, \text{ and} \\
r_e(e^t, a_R^t) &\stackrel{\text{def}}{=} \sum_{a_H^t} r_{i(n_H^t)}(s^t, \langle a_H^t, a_R^t \rangle) \cdot \psi(n_H^t, a_H^t).
\end{aligned}$$

Solving this robot POMDP gives a robust robot policy that best responds to the initial probability distribution over the given human policies. This offline algorithm is evaluated in Section 6.3.

6.2 Online Robust Planning for HRC

The last section presents an offline algorithm for building a robust robot policy. As we discussed in Section 6.1.2, if a human model (presented by a stochastic FSC) is known, we can build an extended state POMDP for the robot and derive a best-response robot policy offline. However, those computations may be expensive since we must consider all possible human behaviors (in an FSC). Moreover, even if we acquired all the elements for large problems, building an explicit model for this extended POMDP may cost a lot of time and resources (which is confirmed by our experiments in Section 6.3, see Table 6.2).

This section proposes our second contribution, an online algorithm for deriving robust robot policies, to address the heavy computation costs issue in offline settings. The core idea is that, instead of building the entire explicit extended POMDP, we develop a generative model of this extended POMDP denoted G_e (extended generative model). Then, we apply online POMDP planning to G_e to give robust robot decisions. This method has two main advantages compared with the offline approach:

1. An explicit Dec-POMDP model is not required for our method. Only a black-box simulator is needed, as many sampling-based planning methods. Thus, it has the potential to scale up to large collaboration problems.

2. Since the online algorithm only searches reachable branches based on the current situation, the robot does not need to explore and consider all possible human behaviors. Therefore, this method enables the robot to interact with humans online within a limited time.

The process for generating the robot's extended generative model G_e is presented in Section 6.2.1, and we evaluate our algorithm in Section 6.3.2.

6.2.1 Robot Extended Generative Model

A generative model (black box simulator) is usually required for online planning. For a Dec-POMDP, its generative model G is defined by

$$s', o, r \leftarrow G(s, a),$$

where G takes as input the current state s and joint action a, returning the next state s', joint observation o, and an instant reward r. However, in HRC tasks, the robot cannot directly use G to make its plans since it should reason about its actions using its own observations. Therefore, to build an online robot algorithm, the question is how to construct ~~such~~ a *robot generative model*.

In Section 6.1.2, we have presented an offline framework for building an explicit extended POMDP for the robot, which accounts for the uncertainty of human objectives and behaviors by integrating a human union FSC. We showed that this robot POMDP has an extended state space $\mathcal{S}_e$:

$$s_e^t = \langle s^t, n_H^t, o_R^t \rangle, \quad s_e^t \in \mathcal{S}_e,$$

where s^t is the real current state, n_H^t is the current internal node of the human union FSC, and o_R^t is the robot's observation. Moreover, each internal node n_H^t in the union FSC represents a pair of the human's objective θ_H and internal belief b_H^t, and stores the probability distribution of human actions given $\langle \theta_H, b_H^t \rangle$. Solving this robot POMDP gives a robust robot policy which is the best response to all possible human policies considered in the union FSC.

In the online setting, we can not use this representation directly since it requires computing a human FSC to cover all possible human objectives and induced behaviors. This being said, for online planning, the robot should infer the human's objective θ_H and internal belief b_H^t online. Thus, the robot's extended state is now defined as:

$$s_e^t = \langle s^t, \theta_H, b_H^t, o_R^t \rangle,$$

and we derive the dynamics of the robot's extended POMDP as follows:

$$\begin{aligned}
T_e(s_e^t, a_R^t, s_e^{t+1}) &= Pr(s_e^{t+1} | s_e^t, a_R^t) \\
&= \frac{Pr(s^{t+1}, b_H^{t+1}, \theta_H, o_R^{t+1}, s^t, b_H^t, o_R^t, a_R^t)}{Pr(s^t, b_H^t, \theta_H, o_R^t, a_R^t)} \\
&= \frac{\sum_{a_H^t} \sum_{o_H^{t+1}} Pr(s^{t+1}, b_H^{t+1}, \theta_H, o_R^{t+1}, o_H^{t+1}, s^t, b_H^t, o_R^t, a_H^t, a_R^t)}{Pr(s^t, b_H^t, \theta_H, o_R^t, a_R^t)} \\
&= \sum_{a_H^t} \sum_{o_H^{t+1}} Pr(b_H^{t+1} | o_H^{t+1}, a_H^t, b_H^t) \cdot Pr(s^{t+1} | s^t, \langle a_H^t, a_R^t \rangle) \cdot \\
&\quad Pr(\langle o_R^{t+1}, o_H^{t+1} \rangle | s^{t+1}, \langle a_H^t, a_R^t \rangle) \cdot Pr(a_H^t | \theta_H, b_H^t) \\
&= \sum_{a_H^t} \sum_{o_H^{t+1}} \eta(b_H^t \langle a_H^t, o_H^{t+1} \rangle, b_H^{t+1}) \cdot T(s^t, \langle a_H^t, a_R^t \rangle, s^{t+1}) \cdot
\end{aligned}$$

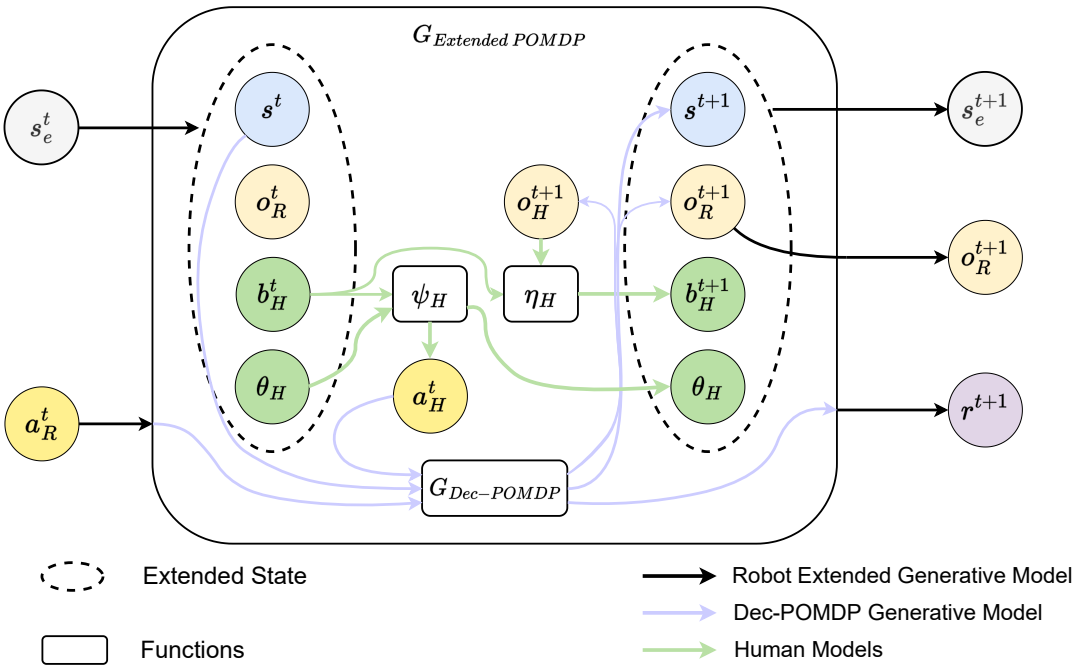

Figure 6.3: Structure of the extended generative model G_e for the robot: The black arrows are the inputs and outputs of the robot's extended generative model; The blue arrows show the inputs and outputs of a Dec-POMDP simulator G; The green arrows show the evolution of the human model.

$$O(s^{t+1}, \langle a_R^t, a_H^t \rangle, \langle o_R^{t+1}, o_H^{t+1} \rangle) \cdot \psi_H(b_H^t, \theta_H, a_H^t),$$

$$\begin{aligned} O_e(s_e^{t+1}, a_R^t, o_R^{t+1}) &= Pr(o_R^{t+1} | s_e^{t+1}, a_R^t) \\ &= Pr(o_R^{t+1} | \langle s^{t+1}, \tilde{o}_R^{t+1}, b_H^{t+1}, \theta_H \rangle, a_R^t) \\ &= \mathbf{1}_{o_R^{t+1} = \tilde{o}_R^{t+1}}, \end{aligned}$$

$$\begin{aligned} r_e(s_e^t, a_R^t) &= \sum_{a_H^t} r(s^t, \langle a_H^t, a_R^t \rangle) Pr(a_H^t | \theta_H, b_H^t) \\ &= \sum_{a_H^t} r(s^t, \langle a_H^t, a_R^t \rangle) \cdot \psi_H(b_H^t, \theta_H, a_H^t), \end{aligned}$$

where we define ψ_H is a human action selection function that $\psi_H(b_H^t, \theta_H, a_H^t) = Pr(a_H^t | \theta_H, b_H^t)$ (details are given in Algorithm 16). Following this idea, we build an extended generative model G_e as presented in Figure 6.3 with the following definitions:

- inputs: current extended state s_e^t and robot action a_R^t,
- outputs: next extended state s_e^{t+1}, robot observation o_R^{t+1}, and instant reward r.

Algorithm 16 details the internal processes of G_e. First, G_e takes as input the current extended state s_e^t and robot action a_R^t. Then, the algorithm decomposes s_e^t (line 3) and computes the Q

values for b_H^t using ~~a~~ POMCP, considering the Dec-POMDP is an MPOMDP for the human. A softmax distribution P_{A_H} of human actions is computed based on obtained Q values (line 4). Then, a human action a_H^t is sampled from P_{A_H} (line 5), and the algorithm passes the state s^t and joint action $\langle a_H^t, a_R^t \rangle$ to G (line 6). In this step, the next state s^{t+1}, all agents' observations $\langle o_H^{t+1}, o_R^{t+1} \rangle$, and an instant reward r are sampled. Then, to build the next extended state s_e^{t+1}, a particle filter is used to compute an updated human belief b_H^{t+1} using his observation o_H^{t+1} (corresponding to the function η_H in Figure 6.3). It samples a state $\tilde{s}$ from b_H^t at line 10 and passes it to G with the same joint action $\langle a_H^t, a_R^t \rangle$, if the sample observation $\tilde{o}_H$ is identical to o_H^{t+1}, the generated next state $\tilde{s}'$ is kept and added as a particle to b_H^{t+1} until the maximum particle size (lines 12–14) is reached. At this stage, all the elements for building the next extended state s_e^{t+1} are obtained (line 15). The algorithm then returns the step results.

Algorithm 16: Robot's Extended Generative Model G_e in HRC

```
1  [Input:] s_e^t: extended state | a_R^t: robot's action
2  [Parameter:] G: Dec-POMDP simulator | T: softmax parameter
3  ⟨s^t, b_H^t, θ_H, o_R^t⟩ ← s_e^t
4  P_{A_H} ← ψ_H(·|b_H, θ_H)_T        ▷ POMCP solves MPOMDP with b_H and get Q values
5  a_H^t ← Sample(P_{A_H})
6  s^{t+1}, ⟨o_H^{t+1}, o_R^{t+1}⟩, r ← G(s^t, ⟨a_H^t, a_R^t⟩, θ)
7  k ← 0
8  b_H^{t+1} ← ∅
9  while k < k_particles do
10     s̃ ← Sample(b_H^t)
11     s̃', ⟨õ_H, õ_R⟩, r̃ ← G(s̃, ⟨a_H^t, a_R^t⟩)
12     if õ_H = o_H^{t+1} then
13         b_H^{t+1} ← b_H^{t+1} ∪ {s̃'}
14         k ← k + 1
15 s_e^{t+1} ← ⟨s^{t+1}, b_H^{t+1}, θ_H, o_R^{t+1}⟩
16 return s_e^{t+1}, o_R^{t+1}, r        ▷ return step results
```

At time $t = 0$, the extended state $s_e^0 = \{s^0, b_H^0, \theta_H, \varnothing\}$ where $s^0 \in b^0$, $\theta_H \in P_\theta$, and $b_H^0 = b^0$. The robot's initial belief b_R^0 is the probability distribution over s_e^0, which is defined as follows:

$$\begin{aligned} b_R^0(s_e^0) &= Pr(s, b_H^0, \theta_H, \varnothing) \\ &= Pr(s, b_0, \theta_H, \varnothing) \\ &= Pr(s|b_0)Pr(\theta_H) \\ &= b_0(s)P_\theta(\theta_H), \end{aligned}$$

where P_θ is the probability distribution over human objectives. We thus built a valid POMDP generative model G_e for the robot. Modern online POMDP solvers such as POMCP can be used to plan the robot's action online, which accounts for possible human objectives and behaviors.

6.2.2 Online Main Algorithm

The method described in Algorithm 16 constructs an extended generative model G_e for the robot. As mentioned in the previous section, one can then use an online POMDP solver to plan robot actions using G_e. This will lead to nested online planning on two levels:

- the top-level online planning takes place for the robot to plan using G_e;
- the bottom-level planning is inside G_e, where another online planner is used to compute the probability distribution over human actions based on the SMM assumption (line 4 in Algorithm 16).

Using POMCP as an online POMDP solver, we propose an online solver for the robot in our HRC tasks, as shown in Algorithm 17. Its main ideas are the following:

- Each robot's extended state s_e contains a human belief b_H, which is used within G_e by the bottom-level POMCP to compute a policy tree.
- If in a leaf node, rather than performing rollouts using a random policy to obtain a first (rough) estimate of that node's value in the robot POMDP, we use the value of b_H (for the MPOMDP guiding the human) as a heuristic estimate as shown in red in line 17. To avoid re-computations, this is achieved by re-using the policy tree constructed in the last call to the bottom-level POMCP, which stores estimated values for all visited human beliefs.

6.3 Experiments

Our experiments were conducted on a laptop with a 2.3 GHz i9 CPU. The source code will be made available under an MIT license.

To test our approach, we have designed a scenario presented in Figure 6.4. In this scenario, a robot and a human have to repair two devices and maintain a third one located in a grid world:

- To maintain a device, the robot should be on the device's cell and perform a maintenance action. The human is not required.
- To repair a broken device, first the human needs to pick a component in a toolbox at a specific location; second the human and robot need to be at the device location and simultaneously perform repair actions.

Note also that both the human and the robot have limited observation of the environment.

This task is specified as the following Dec-POMDP:

- States (S): The state $s \in S$ of the problem is made up of: 1. the human location, 2. the robot location, 3. the status of the devices, and 4. whether the human has a component or not. The human and robot locations are represented by integer coordinates (x, y). They can be on the same cell. The state of each device is either "good", "broken", or "needs maintenance".
- Human observations (Ω_H): The human observes 1. his location; 2. whether the robot is on his cell or not; 3. the status of the device in his cell (if any).
- Robot observations (Ω_R): The robot observes 1. its location; 2. the human's location; 3. the status of the device in its cell (if any).
- Actions and Dynamics: Both the human and the robot have the following actions: *Up, Down, Left, Right* are the agents 4 move actions; *Wait*: the agent stays in his current position; *Repair*: repairs a broken device if: 1. the human holds a new component; 2. the human and the robot are in the same cell as the broken device; 3. the human and the

Algorithm 17: Online Robot Search For HRC

$N()$: counts visits to a history or history-action pair | T: policy tree | b_0: initial belief of the problem | $B(h)$: estimated belief given an action-observation history h

Fct *Search(h)*
  repeat
    if $h = empty$ **then**
      $s_e \sim b_0$
    else
      $s_e \sim B(h)$
    Simulate$(s_e, h, 0)$
  until *Timeout()*
Fct *Simulate*$(s_e, h, depth)$
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  if $h \notin T$ **then**
    for $a \in \mathcal{A}$ **do**
      $T(h, a) \leftarrow (N_{init}(h, a), Q_{init}(h, a), \varnothing)$
    $b_h \leftarrow s_e.b_h$
    $V(b_h) \leftarrow POMCP_M$ `// Get heuristic value from the built POMCP policy tree for MPOMDP`
    return $V(b_h)$
  $a \leftarrow \arg\max_a [Q(h, a) + c\sqrt{\frac{\log N(h)}{N(h,a)}}]$
  $(s_e^{'}, o, r) \sim G_e(s_e, a)$
  $R \leftarrow r + \gamma Simulate(s_e^{'}, hao, depth + 1)$
  $N(h) \leftarrow N(h) + 1$
  $N(h, a) \leftarrow N(h, a) + 1$
  $Q(h, a) \leftarrow Q(h, a) + \frac{R - Q(h,a)}{N(h,a)}$
  return $\arg\max_a Q(h, a)$

robot both perform the repair action. Upon success, the broken device turns to *good* and the human's component is consumed.

The human can *Pick a Component* if he is in the toolbox area and if he does not already hold one. The robot can *Maintain* a device individually if it needs to be maintained and if in the same location. Upon success, the device status turns to *good*.

- Rewards R: A reward of $+100$ is given when all devices have been repaired or maintained. A reward (penalty) of -2 is given for each action except for the human's *Wait*, and a penalty of -20 is given in case of invalid action. A penalty of -1 is given if the human waits before having repaired all the broken devices. If all devices are in "good" status, no penalty (0) is given to the human wait action. The penalty associated to the wait action encourages the robot to postpone maintenance actions if this helps the human finish repair actions early.

This large Dec-POMDP has 2 304 states, 49 joint actions (7 individual actions per agent), and 5 400 joint observations (30 individual observations for the human and 180 for the robot).

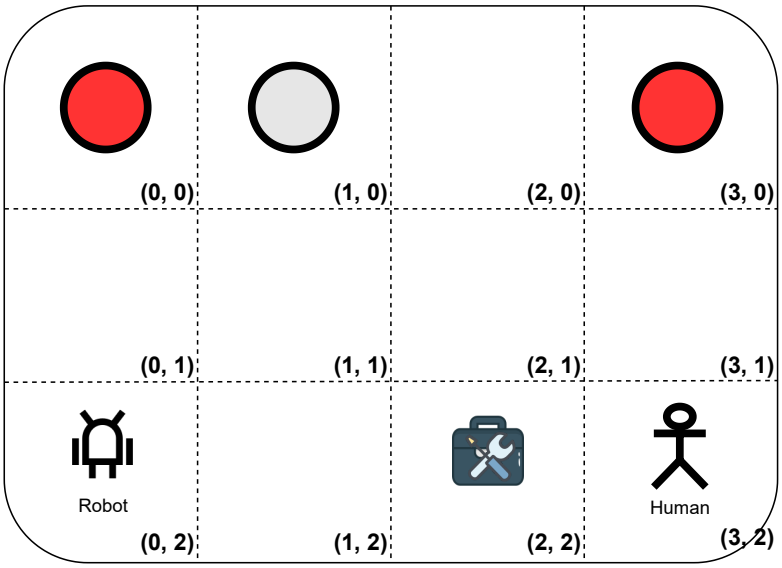

Figure 6.4: A collaboration task: A robot and a human evolve in a 4×3 grid world. The top-left cell $(0, 0)$ and the top-right cell $(3, 0)$ both contain a broken device. A device to be maintained is also located in cell $(1, 0)$. A toolbox is located in cell $(2, 2)$ where the human can pick components.

Note: This state space grows quadratically with the number of cells and exponentially with the number of devices.

To represent uncertainty about the human objectives (see Figure 6.5), we replaced the reward function described previously with two variants. In the first case, the human prefers to repair the broken device on the left first, receiving a $+10$ reward in that case. In the second case, symmetrically, the human prefers to repair the broken device on the right first. These objectives generate different human policies using Algorithm 14. Initially, the robot only has a prior over the human's rewards and associated policy (each with a probability 0.5). Due to the required coordination for repairing devices, these human policies have to account for the robot's ability to help the human when needed.

We evaluate both the offline and online robust planning algorithms described in Section 6.1 and Section 6.2. But for experiments with real humans, only the offline method is evaluated. Real human experiments using the online algorithm is kept for future work.

6.3.1 Experiments with Synthetic Humans – Offline Algorithm

Qualitative results

We used Algorithm 14 to compute the stochastic human FSCs associated with each human objective, and the method described in Section 6.1.2 to compute the robust robot policy with the POMDP solver SARSOP [Kurniawati et al., 2008]. To save resources, an *action-threshold* (here always set to 0.1) is used to prune low-probability human actions when computing the stochastic human FSCs. We conducted experiments with 2 values for the softmax parameter T (0.3 and 0.5), and 5 values for the maximum stochastic FSC size $N_{\max}$ (see Table 6.1).

We first observed that the extracted human stochastic FSC can effectively encode possible trajectories of the human to solve the task if $N_{\max}$ is high enough for the current T. For instance,

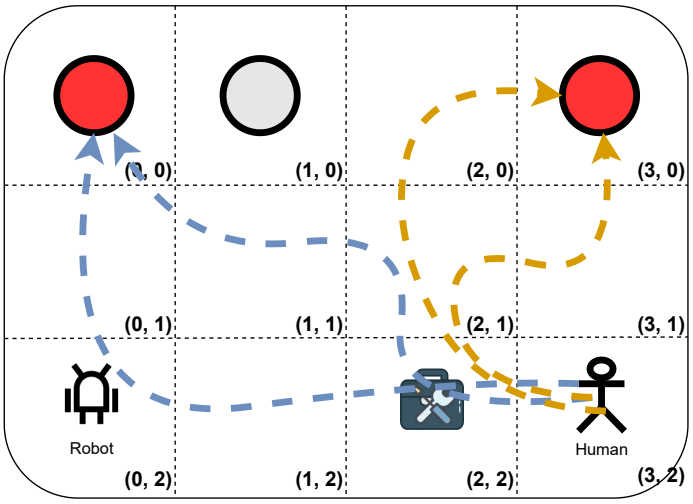

Figure 6.5: An illustration of the uncertainties about the human behavior in this HRC task: 1. Uncertainty on the human objective (presented with blue and orange colors); 2. Uncertainty on human behaviors (different trajectories)

when $T = 0.3$ and $N_{\max} = 100$, the extracted human stochastic FSCs reach depths[4] of 22 and 18 (for each objective), which are sufficient for the human to finish the task if the robot collaborates. However, when $N_{\max} = 50$, the depth covered by the FSC of the *repair-left-device-first* objective is only 13 (15 for the 2nd objective), which does not allow the human to finish the task (a depth of 15 is required). We also observed that the higher the temperature T, the higher the $N_{\max}$ value needed for the human stochastic FSC. Indeed, when T increases, more actions are considered at each node, generating more branches and preventing the expansion process from reaching the end of the task. Note also that, even if we set T to a very low value (near 0), the softmax distribution may contain several optimal actions. This allows encoding several optimal human policies in a single FSC.

When $N_{\max}$ is large enough for the current T, we observed that the computed robust robot policy can successfully solve the task, accounting for the uncertainties over human objectives and behaviors. It helps the human repair all the devices and performs the required maintenance operation. For example, when $T = 0.3$ and $N_{\max} = 100$, the robot following the generated robust policy first goes to cell $(0, 0)$ and waits for the human to pick a component at cell $(2, 2)$. Afterwards, if the robot observes that the human is approaching, it keeps waiting for him and then helps him repair the left device when the human reaches cell $(0, 0)$. But if it observes the human moving to the top-right corner, then the robot decides to go right and help the human repair the right device first. The robot then moves to cell $(1, 0)$ to perform a maintenance operation while the human is going to pick a second component. Finally, the robot helps the human repair the other broken device to finish the task.

Even if this seems like a simple pattern for the robot, it still requires the robot to reason on many possible human trajectories. For example, if there are 2 optimal actions in each state, even for 5 time steps, there are 2^5 possible optimal trajectories (and this number can increase dramatically when considering sub-optimal actions). Moreover, in this complex collaboration

[4]The depth of an FSC is the maximum distance between the (here unique) initial node and any node.

scenario, agents make decisions only based on partial observations, and, as illustrated above, the robot infers the human objective despite his uncertain behavior.

Robustness Analysis

We also wanted to quantitatively analyze the robustness of our approach by comparing the method described in Algorithm 14 with a method computing a robot policy only based on deterministic human policies. To that end, we extracted 50 pairs of deterministic human policies, one per objective, using $T = 0.5$ and $N_{\max} = 600$ (*cf.* Algorithm 15), but obtaining less than 200 nodes in all cases; then, for each pair, we computed a best-response robot policy to the equiprobable union of these two human policies (*cf.* beginning of Section 6.1.2). The resulting set of 3×50 policies is thus $\Pi_H = \{\langle \pi^1_{H,l}, \pi^1_{H,r}, \pi^1_{H,l+r} \rangle, \dots, \langle \pi^{50}_{H,l}, \pi^{50}_{H,r}, \pi^{50}_{H,l+r} \rangle\}$. For each human union policy $\pi^i_{H,l+r}$, we also compute a robot best response $\pi^i_{R,BR}$ by solving a POMDP obtained as in Section 6.1.2.

In our experiments, we compare the robust robot policies π^*_R obtained for different values of T and $N_{\max}$ against

1. the 50 best responses $\pi^i_{R,BR}$ described above (averaging over all of them), and
2. the robot policy obtained by solving (with Inf-JESP [You et al., 2021a]) the corresponding Dec-POMDP.[5]

The obtained results and standard deviations are presented in Table 6.1 (ignoring the last three rows, which correspond to the online algorithm and will be discussed later). For each robot FSC and each human preference (left or right first):

- we compute the exact value against each human FSC (through an iterative process, see Equation (2.21), page 19), then obtain the average value and standard deviation over the various humans;
- we simulate its behavior once against each human FSC (which is sufficient since both agents' FSCs are deterministic) to obtain the success rate over all considered human FSCs.

Then, the result for the third setting $\pi^{Sample}_{H,l+r}$ (uncertain preference) is obtained as the average of the first two columns (as expected from the theory, but also empirically validated).

First, as expected, the worst results are obtained with the Dec-POMDP and Best-Response solutions ($\pi_{R,Dec-POMDP}$ and $\pi_{R,BR}$), with near-zero success rates. Our approach improves a little bit in terms of success rate when using $T = 0$, *i.e.*, when assuming that the human only chooses among a small subset of actions at each time step, which leads to fairly small human FSCs (24 and 23 nodes for the two independent objectives). The average reward remains very low, though. Using $T = 0.3$ or 0.5 leads to much better results. In the *prefer-right* scenario, human FSCs with 200 nodes are even sufficient to get very good solutions (90% success rate). Moving from $T = 0.3$ to $T = 0.5$, this phenomenon is all the more important that more erratic behaviors require larger FSCs to better account for most likely trajectories. In the *prefer-left* scenario, the success rates and values drop significantly. This is due to the longer trajectories needed in this case, which in turn require large human FSCs to encode good enough policies. Then, in the "union" scenario where the human's preference is randomly sampled, the results are a simple combination of the results in the *prefer-left* and *prefer-right* scenarios.

[5]This solution assumes a strong a priori coordination since the human and robot agree on their individual policies.

Table 6.1: Average value (and success rates in parentheses) of various robot policies vs. various human behaviors. For each offline robust robot policy $\pi^*_R(T, N_{\max})$, we also indicate the sizes of both generated human FSCs as $N = (N_{left}, N_{right})$. For the online method, we present the different timeouts used in each $\pi^*_{R,Online}$.

	3×50 synthetic human FSCs sampled using $T = 0.5$			10 real humans
	$\pi^{Sample}_{H,l}$	$\pi^{Sample}_{H,r}$	$\pi^{Sample}_{H,l+r}$	π^{Real}_{H}
$\pi_{R,Dec-POMDP}$	−34.9± 6.7 (0%)	−82.5±14.4 (0%)	−58.7±10.5 (0%)	
$\pi_{R,BR}$	−172.1± 2.5 (3%)	−174.4± 2.5 (3%)	−173.3± 2.5 (3%)	
$\pi^*_R(T = 0.0, N_{max} = 100, N = (24, 23))$	−110.0±12.5 (**10**%)	−179.4±15.4 (18%)	−144.7±14.0 (14%)	−55.3± 6.5 (16%)
$\pi^*_R(T = 0.3, N_{max} = 200, N = (200, 200))$	−38.1± 9.8 (52%)	**10.0**± 2.0 (90%)	−14.0± 5.9 (71%)	
$\pi^*_R(T = —, N_{max} = 400, N = (400, 400))$	−11.9± 7.2 (68%)	**10.0**± 2.0 (90%)	−0.9± 4.6 (79%)	
$\pi^*_R(T = —, N_{max} = 600, N = (600, 443))$	−11.9± 7.2 (68%)	**10.0**± 2.0 (90%)	−0.9± 4.6 (79%)	**−20.8**± 7.4 (**81**%)
$\pi^*_R(T = 0.5, N_{max} = 200, N = (200, 200))$	−36.6± 5.6 (16%)	**10.0**± 2.0 (90%)	−13.3± 3.8 (53%)	
$\pi^*_R(T = —, N_{max} = 400, N = (400, 400))$	2.7± 2.5 (76%)	**10.0**± 2.0 (90%)	6.3± 2.3 (83%)	
$\pi^*_R(T = —, N_{max} = 600, N = (600, 600))$	**9.5**± 2.2 (84%)	**10.0**± 2.0 (90%)	**9.8**± 2.1 (**87**%)	−33.7±11.9 (75%)
$\pi^*_{R,Online}(T = 0.5, timeout = 10\,\text{s})$	5.6± 2.4 (80%)	7.3± 2.3 (84%)	6.5± 2.3 (82%)	
$\pi^*_{R,Online}(T = —, timeout = 30\,\text{s})$	7.0± 2.4 (84%)	6.7± 2.2 (86%)	6.9± 2.3 (85%)	
$\pi^*_{R,Online}(T = —, timeout = 60\,\text{s})$	**10.2**± 2.0 (90%)	**9.1**± 2.1 (90%)	**9.7**± 2.1 (**90**%)	

Table 6.2: CPU Time (in seconds) for different experiments with our offline method

	$\pi^*_R(T, N_{\max})$		
Step	$(0.0, 100)$	$(0.3, 600)$	$(0.5, 600)$
Dec-POMDP→MPOMDP	13	13	13
Get Human Stoc. FSCs	83	2032	6144
Build Robot POMDP	3	178	203
Solve Robot POMDP	2	57	105
Total time	101	2280	6465

Note that, in these experiments, the synthetic humans used for evaluation are rather erratic ($T = 0.5$), which explains that the best success rates are obtained with robot policies derived for that same temperature, while robots derived for $T = 0.3$ are not robust enough.

Table 6.2 presents the recorded computational time used in each step of our offline method. The 1st step consists in converting each Dec-POMDP task model to an MPOMDP (one by human objective); the 2nd step in generating stochastic human policies (FSCs) as presented in Sec. 6.1.1 (the most time-consuming process); the 3rd step in building the robot POMDP using the task models (Dec-POMDPs) and the human FSCs; and, the final step in solving the robot POMDP using SARSOP to get a robust robot policy. Here, most of the time is spent calling POMCP in the FSC extraction.

6.3.2 Experiments with Synthetic Humans – Online Algorithm

We use Algorithm 17 to plan the robot's actions online with a softmax temperature $T = 0.5$, and three different POMCP timeouts (10 s, 30 s and 60 s) are tested in the experiments. We evaluate our online approach using the same synthetic human policies as our offline method. The results of our online approach are presented at the bottom of Table 6.1 (the last three rows). We first observe that:

- our online method gives relatively good solutions even with a short timeout, *i.e.*, with a

timeout of 10 seconds, our online approach achieves 82% success rate on both objectives (evaluation with $\pi^{Sample}_{H,l+r}$);

- the performance of our online method increases with the timeout; from the timeout of 10 s to 60 s, the online method improves its success rate from 82% to 90% on sets $\pi^{Sample}_{H,l+r}$, and the average value is increased from 6.5 to 9.7.

Second, we observe that in the *prefer-right* scenario, our online method can achieve the same success rate as our offline method (90%), but with a lower average value (9.1 compared with 10.0). However, in the *prefer-left* scenario, the online approach can reach a success rate of 90% with a value of 10.2, which outperforms the offline method $\pi^*_R(0.5, 600)$. An explanation for this phenomenon is that, in the *prefer-left* case, the offline method $\pi^*_R(0.5, 600)$ cannot cover all possible human policies with a max FSC size of 600, and thus may perform poorly in some situations, while the online method keeps on replanning both for the human and the robot, which provides a human behavioral model making good decisions in more situations. In the *prefer-right* scenario, although both methods can achieve the same success rate, the offline approach is equipped with an explicit POMDP model that properly covers the human behavior and, therefore, can compute a better policy thanks to SARSOP with a higher value compared with the online algorithm.

6.3.3 Real Human Experiments

To further evaluate our approach, we implemented this collaboration task as a computer game played in a terminal where the human is controlled through the keyboard. To make a friendly user interface, the human player can directly observe the state of the environment, as shown in Fig. 6.4, but the robot still faces partial observability as defined in the Dec-POMDP problem, and behaves accordingly. A screenshot of this terminal game interface is presented in Figure 6.6.

Then, we selected three offline robot policies, $\pi^*_R(0.0, 100)$, $\pi^*_R(0.3, 600)$ and $\pi^*_R(0.5, 600)$, from the previous experiment, and we invited 10 real human subjects to collaborate with those robot policies. Each subject was informed that the goal of this collaboration task was to turn each device status to "good" within at most 30 time steps, but did not know or observe the instant reward. Each human subject needed to play 8 consecutive rounds of this collaboration game per robot policy (for a total of 24 rounds), *i.e.*, 8 times a first robot, then 8 times a second robot, then 8 times a third robot. The order of robot policies was randomly selected so as to remove the side effects of starting with the same robot policy:

1. the first few games may be more difficult since the human subject needs some rounds to get familiar with the task, and

2. the ordering may influence how a player acts.

Moreover, to make this HRC game more fun, we introduce vertical and horizontal flipping to the environment for every 2 rounds, in which the locations of the human, robot, and devices are different in the beginning. By doing so, we expect that our subjects will not feel bored during the experiment, which may have a negative effect on their decisions. Figures 6.7 to 6.9 present the results of the different sessions and the answers the human subjects gave to our questions after each series of 8 rounds.

Qualitatively, the robot behaviors of the online planning method are not very different from the best robot policy computed offline, so we decided to keep human experiments with the online planning method for future work.

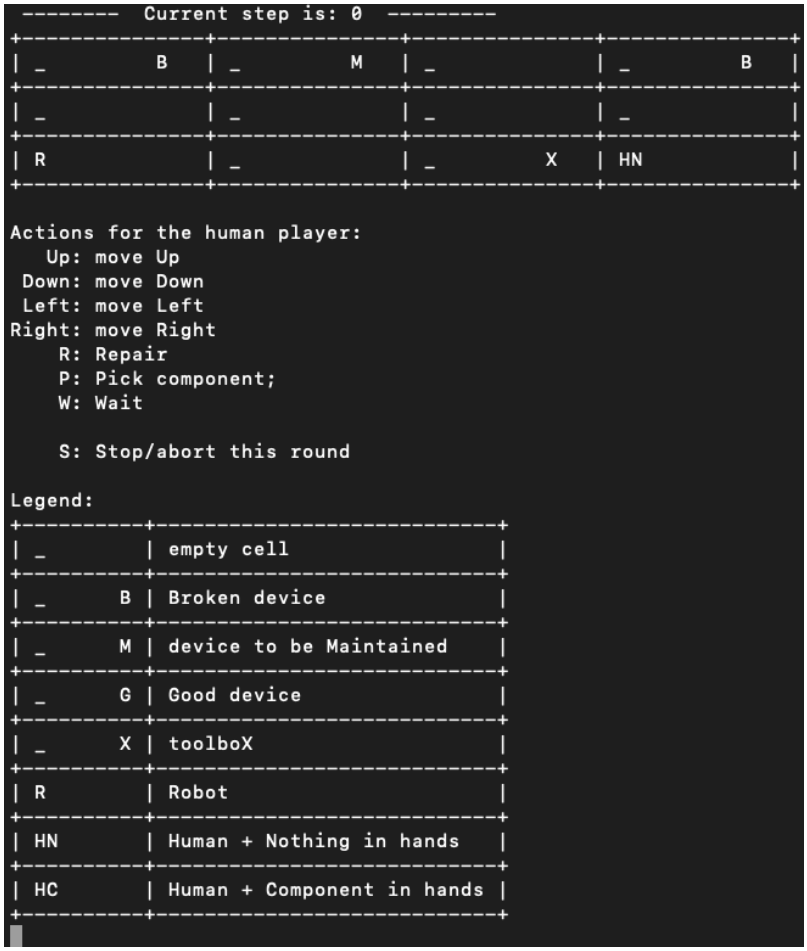

Figure 6.6: A screenshot of the human-robot collaboration game: The top part illustrates the current environment state for the human player; The middle part provides information on available human actions; The bottom part is a legend explaining how to read the map. In each time step, the top part will refresh according to the executed human and robot actions.

Qualitative results

We first observed that the robot policy $\pi_R^*(0.0, 100)$ was unable to help the human subjects since it only accounts for a few high-quality human actions. For example, a common mistake of human subjects was to forget to pick a component necessary to repair a broken device. To fix that mistake, the human subject had to go back to the toolbox, thus generating an unexpected human behavior for robot policy $\pi_R^*(0.0, 100)$, and leading to a failure due to the robot policy getting stuck in a self-loop. On the contrary, while no specific information regarding human behaviors was provided, robot policies $\pi_R^*(0.3, 600)$ and $\pi_R^*(0.5, 600)$ managed to recover and help the human to solve the task because they considered more possible human actions in each situation. In most cases, those policies adapted to different human behaviors and were able to tolerate human mistakes such as the one mentioned above.

Moreover, after the experiments, we asked our subjects how they felt about the 3 tested robot policies. Figure 6.7 presents the detailed subjects' evaluation of both players' adaptation during the experiments. After each series of 8 rounds, we asked the human subjects to assess the interaction with the robot. Each human subject had to pick one choice representing his or her

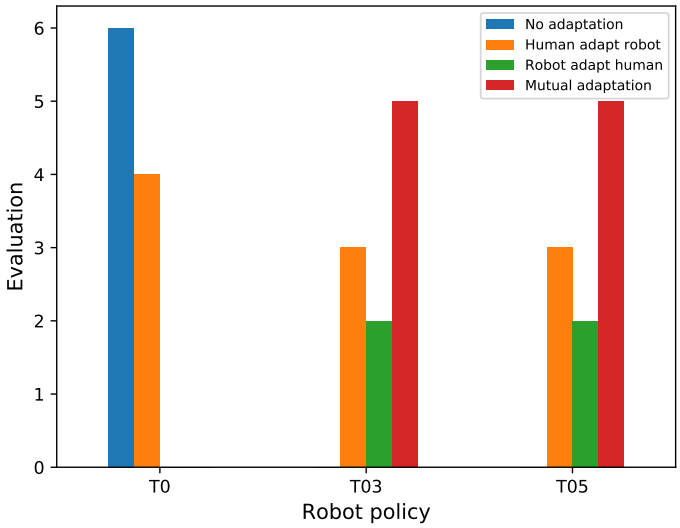

Figure 6.7: All subjects' evaluations regarding both players' adaptation during the collaboration task for each robot policy.

impression among the following four choices: "*no adaptation*", "*human adapts to robot*", "*robot adapts to human*", and "*mutual adaptation*". The results of our questionnaire are as follows:

- When playing with $\pi_R^*(0.0, 100)$:
  - 60% of our subjects reported that there was no adaptation at all; and
  - the other 40% felt that they needed to adapt to the current policy to finish the task.
- Same distribution of choice for the robot policies $\pi_R^*(0.3, 600)$ and $\pi_R^*(0.5, 600)$:
  - 50% of the subjects reported mutual adaptation;
  - 30% felt they needed to adapt to the robot; and
  - 20% reported that $\pi_R^*(0.3, 600)$ and $\pi_R^*(0.5, 600)$ were adapting to their behaviors.

This confirms that it makes little difference for the humans to play with these last two robots, while their results were significantly different in experiments with (erratic) synthetic humans (designed with $T = 0.5$).

Last but not least, we asked our subjects to choose one robot policy that makes them feel comfortable during the collaboration task. The results are shown in Figure 6.8 . As expected by the previous results, no one picked robot policy $\pi_R^*(0.0, 100)$ since it was hard to collaborate with the robot, and the interactions often led to a failure of the task, as shown in Figure 6.9. Then, 60% of our subjects picked $\pi_R^*(0.3, 600)$ and 40% picked $\pi_R^*(0.5, 600)$. The fact that $\pi_R^*(0.3, 600)$ was the most selected policy could imply that computing the robot policy based on overly erratic human models might be counterproductive (since the robot might be seen as being too indecisive), as also highlighted by the success rate presented in Table 6.1. However, more subjects and more investigations are needed to verify that assumption.

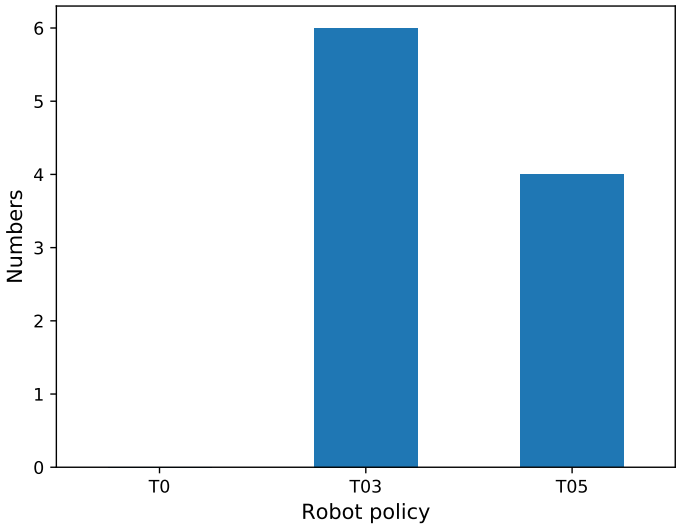

Figure 6.8: All subjects' choices when asked to pick the robot policy they preferred at the end of the experiments.

Quantitative results

The average cumulative rewards and success rates of human subjects are shown in the right part of Table 6.1. For the same three robot policies, the cumulative rewards are lower than with synthetic human experiments (-55.3, -20.8, and -33.7 respectively). There are multiple reasons causing this drop in values: first, the human subjects needed some rounds to get familiar with the game, therefore, in early rounds, humans usually make more mistakes and received penalties; moreover, since human subjects were not informed of the exact reward function, they were not aware of penalties (negative rewards) obtained when waiting or when going outside of the grid world.

Figure 6.9 presents each subject's number of successes for the collaboration task with the different robot policies. Since each subject encountered each robot policy 8 times, a value of 8 means that this human subject managed to solve the task each round, while a value of 0 means this human subject never managed to solve the task while collaborating with this robot policy. When human subjects tried to collaborate with $\pi^*_R(0.0, 100)$, 30% of the subjects (3 out of 10) never succeeded in solving the collaboration task, another 30% succeeded only once, and 40% succeeded 2 or 3 times. This shows that this robot policy does not adapt to spontaneous human behavior and that the human subjects have difficulties finding behaviors that would trigger appropriate robot reactions. However, when human subjects collaborated with robot policies $\pi^*_R(0.3, 600)$ and $\pi^*_R(0.5, 600)$, they all succeeded in more than 4 rounds, and some of them even successfully accomplished all 8 rounds with one of those two robot policies. These similar results suggest that increasing temperature T from 0.3 to 0.5 does not improve the robot's robustness to actual human behaviors.

Overall, the observed success rates of human subjects with robot policies $\pi^*_R(0.3, 600)$ and $\pi^*_R(0.5, 600)$ are 81% and 75% respectively. This shows that, even if the cumulative rewards are low, or if the human performs mistakes, $\pi^*_R(0.3, 600)$ and $\pi^*_R(0.5, 600)$ could still help the human to accomplish the task most of the time. The higher success rate (and cumulative reward) with

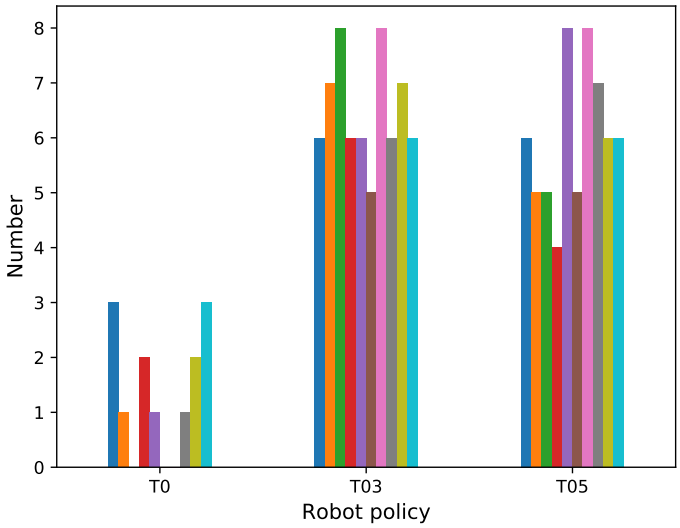

Figure 6.9: Each subject's number of successes when he or she collaborates with different robot policies.

$\pi^*_R(0.3, 600)$ than with $\pi^*_R(0.5, 600)$ (contrary to the results with synthetic humans obtained with $T = 0.5$) suggests that the robot's mental model of humans is better when it builds human FSCs using $T = 0.3$ than $T = 0.5$.

6.4 Conclusion

In this chapter, we discuss and address the problem of uncertainty over human objectives and induced behaviors in human-robot collaboration with partial observabilities. The contribution can be mainly divided into three parts:

- We discuss the pros and cons of various mental models in human-robot collaboration applications. In this thesis, we choose to equip the robot with a second-order mental model (2oMM) for modeling the human partner. We discuss the chicken-and-egg problem raised by using nested 2oMMs and provide two methods to overcome this obstacle. The first method automatically generates a human FSC, aggregating various possible behaviors for a given human objective. The second method gives a probability distribution of human actions online based on a sampling-based planner given a human objective. For both methods, parameters can be tuned to adjust the diversity of the generated human behaviors;

- We propose an offline robust robot planning algorithm that relies on an explicit POMDP model with uncertainties on the human's actual objective and behavior. We formally detail how to build this robot POMDP based on the task models (Dec-POMDPs) and a distribution over human stochastic policies (FSCs), one per possible objective. An offline robust robot policy is obtained by solving this robot POMDP;

- We also propose an online robust robot planning algorithm for the HRC tasks. Compared with the offline method, which relies on an explicit POMDP, our online approach only requires a generative model and, therefore, can scale up to large problems. We demonstrate

the process of constructing a valid POMDP generative model for the robot with an extended state space that includes the human's internal state. We also provide an online robot planning algorithm based on POMCP. Using this POMDP generative model, the robot plans its actions at each time step by estimating the human's actual objective and his/her actions online.

Through experiments, we demonstrate that our approaches are robust to uncertain human behaviors with different objectives. While, in our scenario, the two possible human objectives correspond to the same task under different preferences, different objectives could also correspond to different tasks. We believe this work is important for collaboration settings where the robot and the human need to reason about each other's possible actions, and where considering myopic or deterministic human policies is insufficient to generate robust robot policies. Moreover, our approach only requires a Dec-POMDP describing the human-robot collaboration task, with no need for prior human behavior. This makes our method generic to tackle various human-robot collaboration problems.

6.5 Perspectives

In this chapter, we contribute to deriving robust robot policies for human-robot collaboration with offline and online approaches. Let us now describe various directions for future works.

6.5.1 Experiments with more problems

Through experiments with synthetic or real humans in a simulated environment, we demonstrate that our methods are robust to adapt to different human objectives and behaviors. Therefore, the next step is conducting further experiments on more benchmark problems. In this direction, we could possibly adapt classical Dec-POMDP problems and provide graphic interfaces to real human subjects. Moreover, as a long-term goal, we would like to validate our approach in a real-world scenario where a drone has to help a human to repair devices as part of the Flying Co-Worker project.

6.5.2 Improving the robot planning by learning the human data

One could design a robust robot for HRC tasks using our approaches (either offline or online) without prior knowledge of human behaviors. This means that our methods provide an important starting step for the HRC problems where the robot and human need to coordinate to achieve the goal since we address the chicken-and-egg issue for deriving robot policies. Therefore, we project that a second step would be to use learning techniques to use the human data collected during interactions with the robot. More precisely, one could maybe learn:

1. the probability of each reward function (and thus the human preferences);
2. how erratic the human seems to be (temperature T that best models his/her behavior); or
3. more generally human's reward function (by inverse reinforcement learning) or his/her policy.

Improvements on those directions could provide a more accurate human model for the robot and generate better robust robot policies.

6.5.3 Extension to large and continuous problems

Our online robust robot planning algorithm currently relies on a generative model to plan robot actions, which can handle large state-space problems. One promising direction is further scaling up to continuous actions and observations. To that end, we can combine the double progressive widening technique used in POMCPOW [Sunberg and Kochenderfer, 2018], which extends POMCP to solve continuous problems (continuous states, actions, and observations). This adaptation seems interesting and not difficult since our online robust robot planning algorithm is also built based on POMCP.

Moreover, currently, the robot considers that the human behaves according to a probability distribution computed by a softmax function with a given temperature T. One issue is that, to handle erratic behaviors, we need to reason about many possible human trajectories. Maybe we could let the robot algorithm start, by default, a not-so-erratic human (low T value) and gradually increase the value T until matching realistic human behaviors. This will create an "adaptive" robot policy that can work with humans with different levels of rationality. A potential issue that remains is how to behave when facing unexpected observations.

7

Conclusion and Perspectives

7.1 Conclusion

In this thesis, we investigate the robot decision-making questions in two research fields, robot-robot collaboration (RRC) and human-robot collaboration (HRC).

In RRC, multiple robots are meant to work collaboratively to achieve a task. Moreover, this thesis focuses on the RRC problems where each robot has only partial observations in an uncertain environment. Relying on Markov decision models, we use the Dec-POMDP framework to model such RRC problems and propose two novel algorithms for solving Dec-POMDPs:

- First, based on the JESP scheme, we propose an algorithm called *Infinite-horizon JESP* (Inf-JESP) for solving infinite-horizon Dec-POMDPs. Inf-JESP searches Nash equilibrium solutions by iteratively building each agent's best-response policies to other agents' policies. Compared with JESP, instead of using policy trees for finite-horizon settings, in Inf-JESP each agent policy is represented by a finite-state controller (FSC), which overcomes the curse of dimensionality and can encode infinite-horizon policies. This work has been published in JFPDA [You et al., 2021b] and ICTAI [You et al., 2021a] (an extended version can be found in arXiv [You et al., 2021c]).

- The second algorithm, Monte-Carlo JESP (MC-JESP), extends Inf-JESP to the generative model setting. Like Inf-JESP, MC-JESP searches Nash equilibrium solutions through an iterative process. In each iteration, MC-JESP builds each agent's FSC node by node using a Monte-Carlo approach (POMCP). Unlike in Inf-JESP, where an explicit Dec-POMDP model is needed, MC-JESP only requires a black-box simulator, enabling us to scale up to large Dec-POMDP problems.

We observe through experiments with state-of-the-art benchmark problems that both Inf-JESP and MC-JESP are competitive with existing Dec-POMDP solvers. MC-JESP even outperforms some solvers that rely on explicit models. Moreover, through the research conducted to solve infinite-horizon Dec-POMDPs for RRC, we provide tools to generate a best-response POMDP for a given agent knowing others' fixed policies. This is an important foundation for our contributions to HRC.

In HRC, we are interested in scenarios where a human and a robot need to coordinate their behaviors to achieve a task. Indeed, only the robot can be controlled in this setting, while the human may have uncertain objectives and behaviors. Nevertheless, the robot's policy should be robust to adapt to the uncertainties about the human partner. In this case, we cannot directly solve a Dec-POMDP to derive each agent's optimal policy as in an RRC problem because the

computed robot policy is not robust, and the actual human may not behave according to the computed policy. To derive robust robot policies, we analyze various mental models for the robot to model its human partner. We choose to equip the robot with a second-order mental model (2oMM) for modeling the human. Using a 2oMM means that the robot models the human as reasoning about the robot's possible behaviors, and to avoid the chicken-and-egg problem raised by **nested** 2oMMs, the human is here assumed to have a SMM of himself and the robot, *i.e.*, he reasons as if controlling the robot. More specifically, we propose two novel algorithms as follows:

- We propose an offline algorithm consisting of two parts: 1. A method to overcome the chicken-and-egg problem through a shared mental model (SMM) assumption of the considered human, which automatically generates a stochastic human FSC model that encodes possible behaviors for each human objective provided in an initial distribution. The human's action selection function is parameterized and tunable with a softmax temperature to create more rational or erratic human behaviors. 2. A robust robot planning algorithm that relies on a POMDP with uncertainties on the human's objective and behavior. Based on the best-response POMDP framework we built for RRC, we demonstrate how to build this robot POMDP in the HRC setting given the HRC task model and a distribution over human FSCs (each stochastic FSC is linked to a human objective). The robust robot policy is derived by solving this robot POMDP. A preliminary version of this work has been published in JFPDA [You et al., 2022], and an improved version with real human experiments has been submitted and accepted in ICRA 2023.

- Based on the offline method, we propose a nested online algorithm for deriving robust robot policies. We construct a robot POMDP generative model with an extended state space in this online method. The human's objective and actions are estimated inside the robot's extended generative model through a nested online planning. Based on POMCP, we then provide an online planning algorithm (top-level) for the robot to plan its actions. Moreover, compared with our offline approach, this online variation relaxes the requirement of explicit models to black-box simulators.

Through experiments with synthetic and real humans, we show that our approaches are robust to uncertain human objectives and induced behaviors. In our works, only a Dec-POMDP is required to present the HRC task, and no prior knowledge of human behaviors is needed. We thus believe that our contributions are generic to tackle different HRC problems.

Beyond the contributions listed above, I also cooperated with other authors and published a review article regarding human-humanoid interaction and cooperation in the journal *Current Robotics Reports* [Vianello et al., 2021].

7.2 Perspectives

At the end of this thesis, we take one step further to have a global view of perspectives for all of our contributions.

In the thesis, we provide several algorithms for solving RRC and HRC problems, and various experiments, including evaluations with real human subjects, are conducted in simulated environments to validate our approaches. In the future, we would like to apply our contributions to real-world scenarios with real robots. Various questions and directions could be further investigated in this interesting but challenging topic, including:

- Different action durations: this thesis assumes that all agents have the same action duration in RRC and HRC. This enables us to use standard POMDP or Dec-POMDP frameworks to

model our problems. However, in real-world applications, actions may have different durations. In this direction, Omidshafiei et al. [2015] propose integrating semi-Markov models and macro actions (solving a Dec-POSMDP). Another related direction is hierarchical planning, where we decompose the task into sub-tasks.

- Large or continuous problems: in real applications, the state, action, and observation spaces may be large or continuous. To our knowledge, continuous problems have been investigated in the POMDP setting, *e.g.*, through POMCPOW [Sunberg and Kochenderfer, 2018], which extends POMCP to solve continuous problems. However, there is still a lack of work addressing such problems in multi-agent settings (Dec-POMDPs) or complex HRC tasks.

- Communication during execution: in this thesis, we do not have any assumption of communication between agents in RRC or HRC. In real scenarios, it may be useful and important that agents can communicate with each other to exchange information in order to have better collaborations. A generic approach is introducing generic communication actions whose purpose is decided by the planning process, which may become expensive and degrade performance if too many communications have been executed [Unhelkar et al., 2020]. Another approach is to seek heuristics to decide when and what to communicate (e.g., detecting situations that need disambiguation to avoid catastrophic decisions).

Second, various questions more specific to human-robot collaboration could be investigated:

- We would like to extend our work to settings with multiple humans and robots. Intuitively, considering multiple robots would mean solving a Dec-POMDP for these robots, the humans being considered as part of the environment. Then, considering multiple humans would require a second-order mental model taking into account the whole group of humans, unless they achieve independent tasks and can thus be considered separately.

- Also, our approach could be more human-aware. It could for example try to account for human fatigue, for instance by letting the robot intervene more often (which maybe at the expense of less human autonomy), and for human emotions, for instance by letting the human be more autonomous and making sure that the human understands the robot's behavior. This last point would naturally lead to considering questions such as trust, acceptability, and explainability of the robot's behavior.

Last but not least, we could improve our methods via a learning approach, e.g., if repeating some tasks with the same human or different humans, we can gather data from interactions. The collected human data can be used to learn human objective distribution, level of rationality, and reward function or policy.

A

Appendix

A.1 Notes about the Candidate POMDP Formalizations

We here look in more details at how, given the Dec-POMDP and FSCs for all agents but i, one can derive a valid best-response POMDP from agent i's point of view. First, note that, in the POMDP formalization:

- we have *no choice* for the observation (o_i^t) and for the action (a_i^t), which are pre-requisites;
- the *only choice* that can be made is that of the variables put in the (extended) state e_i^t;
- the transition, observation and reward functions are *consequences* of this choice.

Also, note that, in a POMDP formalization, the state and observation variables need be distinguished. One cannot write that X is an observed state variable (contrary to the MOMDP formalism). In the following, it is thus important to distinguish the different types of random variables. In particular,

- O_i^t always denotes (of course) the *observation variable*, yet
- $\tilde{O}_i^t$ denotes either an *intermediate variable*[6] or a *state variable* (which, in both cases, is strongly dependent on observation variable O_i^i since their values are always equal).

The main issue when designing our BR POMDP is to verify that the dependencies in the transition, observation, and reward functions are the appropriate ones (see Fig. A.1 (top left)). In the following paragraphs, we consider 3 choices for the *extended state* of the POMDP, check whether they indeed induce valid POMDPs, and derive the induced transition, observation and reward functions when appropriate.

$\underline{e^t = \langle s^t, n_{\neq i}^t \rangle \text{ ?}}$ To show that $e^t = \langle s^t, n_{\neq i}^t \rangle$ does not induce a proper POMDP, let us write the transition function:

$$\begin{aligned} T_e(e^t, a_i^t, e^{t+1}) &\stackrel{\text{def}}{=} Pr(e^{t+1}|e^t, a_i^t) \\ &= Pr(s^{t+1}, n_{\neq i}^{t+1}|s^t, n_{\neq i}^t, a_i^t) \\ &= \sum_{\tilde{o}^{t+1}} Pr(s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}^{t+1}|s^t, n_{\neq i}^t, a_i^t) \end{aligned}$$

[6] Such a variable does not usually appear in the POMDP formalism, but is required here to compute the transition and observation functions based on the Dec-POMDP model and the FSCs.

$$
\begin{aligned}
&= \sum_{\tilde{o}^{t+1}} Pr(n_{\neq i}^{t+1} | s^t, n_{\neq i}^t, a_i^t, s^{t+1}, \tilde{o}^{t+1}) \cdot Pr(s^{t+1}, \tilde{o}^{t+1} | s^t, n_{\neq i}^t, a_i^t) \\
&= \sum_{\tilde{o}^{t+1}} Pr(n_{\neq i}^{t+1} | s^t, n_{\neq i}^t, a_i^t, s^{t+1}, \tilde{o}^{t+1}) \cdot Pr(\tilde{o}^{t+1} | s^t, n_{\neq i}^t, a_i^t, s^{t+1}) \cdot Pr(s^{t+1} | s^t, n_{\neq i}^t, a_i^t) \\
&= \sum_{\tilde{o}^{t+1}} \left(\prod_{j \neq i} \eta(n_j^t, \tilde{o}_j^{t+1}, n_j^{t+1}) \right) \\
&\quad \cdot O(\langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}, \tilde{o}^{t+1}) \cdot T(s^t, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}),
\end{aligned}
$$

where $\tilde{O}^{t+1}$ is a temporary variable, not a state or an observation variable. Here, as illustrated by Fig. A.1 (top right), the issue is that the actual observation variable O_i^{t+1} is not independent on E^t given E^{t+1} and A^t.

$\underline{e^t = \langle s^t, n_{\neq i}^t, \tilde{o}_i^t \rangle \text{ ?}}$ We correct the first attempt by adding a *state* variable $\tilde{O}_i^t$, thus defining $e^t = \langle s^t, n_{\neq i}^t, \tilde{o}_i^t \rangle$, hence:

$$
\begin{aligned}
T_e(e^t, a_i^t, e^{t+1}) &\stackrel{\text{def}}{=} Pr(e^{t+1} | e^t, a_i^t) \\
&= Pr(s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}_i^{t+1} | s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t) \\
&= \sum_{o_{\neq i}^{t+1}} Pr(s^{t+1}, n_{\neq i}^{t+1}, o_{i-1}^{t+1}, \tilde{o}_i^t | s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t) \\
&= \sum_{o_{\neq i}^{t+1}} Pr(n_{\neq i}^{t+1} | s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t, s^{t+1}, o^{t+1}) \cdot Pr(s^{t+1}, o_{\neq i}^{t+1}, \tilde{o}_i^{t+1} | s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t) \\
&= \sum_{o_{\neq i}^{t+1}} Pr(n_{\neq i}^{t+1} | s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t, s^{t+1}, o^{t+1}) \\
&\quad \cdot Pr(o_{\neq i}^{t+1}, \tilde{o}_i^{t+1} | s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t, s^{t+1}) \cdot Pr(s^{t+1} | s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t) \\
&= \sum_{o_{\neq i}^{t+1}} \left(\prod_{j \neq i} \eta(n_j^t, o_j^{t+1}, n_j^{t+1}) \right) \\
&\quad \cdot O(\langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}, o^{t+1}) \cdot T(s^t, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}).
\end{aligned}
$$

The formula above does not raise the same issue as $\tilde{O}_i^t$ is a *state* variable, and the observation variable O_i^t now does not depend on the previous state given the new one and the action (*cf.* Fig. A.1 (bottom right)). Also, we have the following observation function:

$$
\begin{aligned}
O_e(a_i^t, e_i^{t+1}, o_i^{t+1}) &\stackrel{\text{def}}{=} Pr(o_i^{t+1} | a_i^t, e_i^{t+1}) \\
&= Pr(o_i^{t+1} | a_i^t, \langle s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}_i^{t+1} \rangle) = \mathbf{1}_{o_i^{t+1} = \tilde{o}_i^{t+1}},
\end{aligned}
$$

and the trivial reward function:

$$
r_e(e^t, a_i^t) = r(s^t, a_i^t, \psi_{\neq i}(n_{\neq i}^t)).
$$

$\underline{e^t = \langle s^t, n_{\neq i}^{t-1}, \tilde{o}_{\neq i}^t \rangle \text{ ?}}$ In this work, we have also considered a third choice for the extended state, defined as $e^t = \langle s^t, n_{\neq i}^{t-1}, \tilde{o}_{\neq i}^t \rangle$, where $\tilde{O}_{\neq i}^t$ is a state variable (not an observation variable) corresponding to other agents' observations at time t:

$$
T_e(e^t, a_i^t, e^{t+1}) \stackrel{\text{def}}{=} Pr(e^{t+1} | e^t, a_i^t)
$$

$$
\begin{aligned}
&= Pr(s^{t+1}, n^t_{\neq i}, \tilde{o}^{t+1}_{\neq i} | s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i) \\
&= Pr(\tilde{o}^{t+1}_{\neq i} | s^{t+1}, n^t_{\neq i}, s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i) \cdot Pr(s^{t+1}, n^t_{\neq i} | s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i) \\
&= Pr(\tilde{o}^{t+1}_{\neq i} | s^{t+1}, n^t_{\neq i}, s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i) \cdot Pr(s^{t+1} | n^t_{\neq i}, s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i) \\
&\quad \cdot Pr(n^t_{\neq i} | s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i) \\
&= Pr(s^{t+1} | s^t, n^t_{\neq i}, a^t_i) \cdot Pr(n^t_{\neq i} | n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}) \cdot Pr(\tilde{o}^{t+1}_{\neq i} | s^{t+1}, n^t_{\neq i}, a^t_i) \\
&= Pr(s^{t+1} | s^t, n^t_{\neq i}, a^t_i) \cdot Pr(n^t_{\neq i} | n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}) \cdot \sum_{\tilde{o}^{t+1}_i} Pr(\tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i | s^{t+1}, n^t_{\neq i}, a^t_i) \\
&= T(s^t, \langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}) \cdot \prod_{j \neq i} \eta(n^{t-1}_j, o^t_j, n^t_j) \\
&\quad \cdot \sum_{\tilde{o}^{t+1}_i} O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle).
\end{aligned}
$$

The formula above does not raise issues. As illustrated by Fig. A.1 (bottom right), the actual observation variable O^{t+1}_i is independent on E^t given E^{t+1} and A^t. Also, we have the following observation function:

$$
\begin{aligned}
O_e(a^t_i, e^{t+1}_i, o^{t+1}_i) &\stackrel{\text{def}}{=} Pr(o^{t+1}_i | a^t_i, e^{t+1}_i) \\
&= Pr(o^{t+1}_i | a^t_i, \langle s^{t+1}, n^t_{\neq i}, \tilde{o}^{t+1}_{\neq i} \rangle) \\
&= \frac{Pr(\tilde{o}^{t+1}_{\neq i}, o^{t+1}_i, s^{t+1}, n^t_{\neq i}, a^t_i)}{Pr(\tilde{o}^{t+1}_{\neq i}, s^{t+1}, n^t_{\neq i}, a^t_i)} \\
&= \frac{Pr(\tilde{o}^{t+1}_{\neq i}, o^{t+1}_i | s^{t+1}, n^t_{\neq i}, a^t_i)}{\sum_{o^{t+1}_i} Pr(\tilde{o}^{t+1}_{\neq i}, o^{t+1}_i | s^{t+1} n^t_{\neq i}, a^t_i)} \\
&= \frac{O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, o^{t+1}_i \rangle)}{\sum_{\tilde{o}^{t+1}_i} O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle)}.
\end{aligned}
$$

We can see that the denominator is identical to the last part of the transition function. In practice, when computing transition probabilities, we will store the values for

$$
\sum_{\tilde{o}^{t+1}_i} O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle)
$$

so as to reuse them when computing the observation function. In the end, the reward function is obtained with:

$$
\begin{aligned}
r_e(e^t, a^t_i) &= r(\langle s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i} \rangle, a^t_i) \\
&= \sum_{n^t_{\neq i}} Pr(n^t_{\neq i} | n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}) \cdot r(s^t, n^{t-1}_{\neq i}, a^t_i) \\
&= \sum_{n^t_{\neq i}} \Big(\prod_{j \neq i} \eta(n^{t-1}_j, \tilde{o}^t_j, n^t_j) \Big) \cdot r(s^t, \langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle).
\end{aligned}
$$

Different formalizations lead to different state spaces with different sizes, so that the time and space complexities of the generation of this best-response POMDP or of its solving will depend on the choice of formalization. Which choice is best shall depend on the Dec-POMDP at hand,

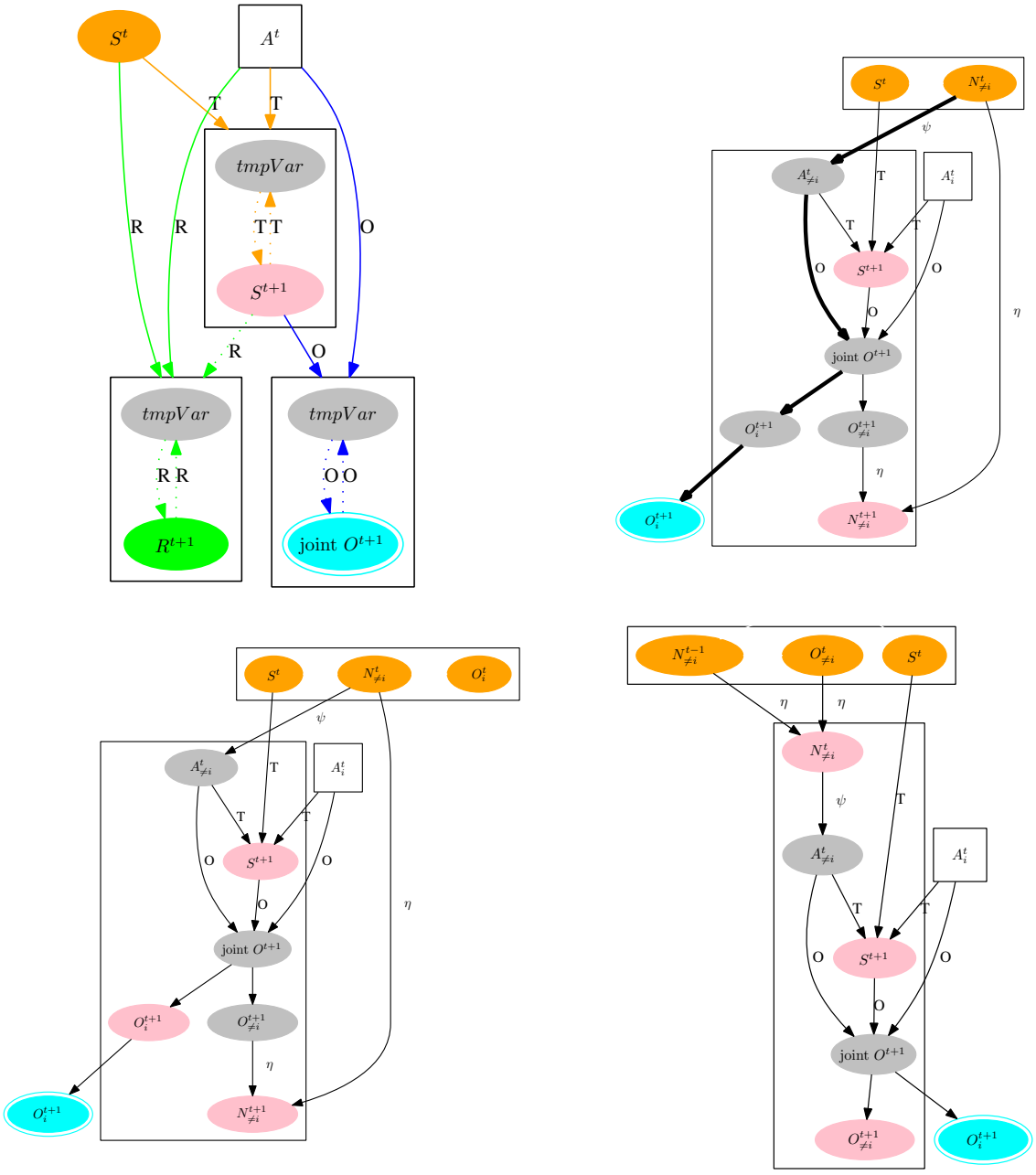

Figure A.1: (top left) Standard POMDP dependencies, including intermediate variables used to compute the transition function (only); and candidate Best-Response POMDP formalizations with: (top right) $e_i^t \stackrel{\text{def}}{=} \langle s^t, n^t_{\neq i} \rangle$, (bottom left) $e_i^t \stackrel{\text{def}}{=} \langle s^t, n^t_{\neq i}, o^t_i \rangle$, and (bottom right) $e_i^t \stackrel{\text{def}}{=} \langle s^t, n^{t-1}_{\neq i}, o^t_{\neq i} \rangle$.

and possibly on the current FSCs. We have opted for the simplest of the two formalizations presented above (which we call the "MOMDP formalization" because one state variable is fully observed), but observed little differences in practice in our experiments.

Bibliography

Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab, and Félix Ingrand. An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337, 1998.

Alexandre Albore, Héctor Palacios, and Hector Geffner. A translation-based approach to contingent planning. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

Christopher Amato. *Increasing scalability in algorithms for centralized and decentralized partially observable Markov decision processes: Efficient decision-making and coordination in uncertain environments*. University of Massachusetts Amherst, 2010.

Christopher Amato and Shlomo Zilberstein. Achieving goals in decentralized pomdps. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 593–600, 2009.

Christopher Amato, Jilles Steeve Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon dec-pomdps. In *Nineteenth International Conference on Automated Planning and Scheduling*, 2009.

Christopher Amato, Blai Bonet, and Shlomo Zilberstein. Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Christopher Amato, Daniel Bernstein, and Shlomo Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. *Proceedings of the Conference on Uncertainty in Artificial Intelligence, (UAI-07)*, 06 2012.

Mauricio Araya-López, Vincent Thomas, Olivier Buffet, and François Charpillet. A closer look at MOMDPs. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2010.

Matthew Arnold, Rachel K. E. Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, Aleksandra Mojsilovic, Ravi Nair, Karthikeyan Natesan Ramamurthy, Darrell Reimer, Alexandra Olteanu, David Piorkowski, Jason Tsay, and Kush R. Varshney. Factsheets: Increasing trust in AI services through supplier's declarations of conformity, 2018. URL `https://arxiv.org/abs/1808.07261`.

Hagai Attias. Planning by probabilistic inference. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, pages 9–16. PMLR, 2003. Reissued by PMLR on 01 April 2021.

Andrea Bauer, Dirk Wollherr, and Martin Buss. Human-robot collaboration: a survey. *I. J. Humanoid Robotics*, 5:47–66, 03 2008. doi: 10.1142/S0219843608001303.

Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684, 1957. URL `http://www.jstor.org/stable/24900506`.

Ron Berenstein and Yael Edan. Human-robot cooperative precision spraying: Collaboration levels and optimization function. *IFAC Proceedings Volumes*, 45(22):799–804, 2012. ISSN 1474-6670. doi: https://doi.org/10.3182/20120905-3-HR-2030.00084. URL `https://www.sciencedirect.com/science/article/pii/S1474667016337077`. 10th IFAC Symposium on Robot Control.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. of Operations Research*, 27(4), 2002.

Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 1287–1292. Morgan Kaufmann Publishers Inc., 2005.

Daniel S. Bernstein, Christopher Amato, Eric A. Hansen, and Shlomo Zilberstein. Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research*, 34:89–132, 2009. doi: 10.1613/jair.2667.

Aaron Bestick, Ruzena Bajcsy, and Anca Dragan. Implicitly assisting humans to choose good grasps in robot to human handovers. In *2016 Int. Symp. on Experimental Robotics*, 2017. ISBN 978-3-319-50114-7. doi: 10.1007/978-3-319-50115-4_30.

Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *J. Artif. Int. Res.*, 11(1):1–94, 1999. ISSN 1076-9757.

Cameron Browne, Edward J. Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Trans. Comp. Intell. AI Games*, 4(1): 1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6145622`.

Jonathan Cacace, Alberto Finzi, Vincenzo Lippiello, Giuseppe Loianno, and Dario Sanzone. Aerial service vehicles for industrial inspection: Task decomposition and plan execution. *Applied Intelligence*, 42:49–62, 01 2015. doi: 10.1007/s10489-014-0542-0.

Sotirios P. Chatzis and Dimitrios Kosmopoulos. A non-stationary infinite partially-observable Markov decision process. In Stefan Wermter, Cornelius Weber, Włodzisław Duch, Timo Honkela, Petia Koprinkova-Hristova, Sven Magg, Günther Palm, and Alessandro E. P. Villa, editors, *Artificial Neural Networks and Machine Learning (ICANN)*, pages 355–362, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11179-7.

Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. Planning with trust for human-robot collaboration. In *2018 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 307–315, 2018.

Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. Trust-aware decision making for human-robot collaboration: Model learning and planning. *J. Hum.-Robot Interact.*, 9(2), January 2020. doi: 10.1145/3359616.

Alessandro Cimatti and Marco Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

Jean-Alexis Delamer, Yoko Watanabe, and Caroline P. Carvalho Chanel. Solving path planning problems in urban environments based on a priori sensors availabilities and execution error propagation. In *AIAA Scitech 2019 Forum*, page 2202, 2019.

Eric V Denardo. Contraction mappings in the theory underlying dynamic programming. *Siam Review*, 9(2):165–177, 1967.

Jilles Dibangoye and Olivier Buffet. Learning to act in decentralized partially observable MDPs. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1233–1242. PMLR, 2018.

Jilles Steeve Dibangoye, Olivier Buffet, and François Charpillet. Error-bounded approximations for infinite-horizon discounted decentralized POMDPs. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, volume 8724 of *Machine Learning and Knowledge Discovery in Databases*, pages 338 – 353, Nancy, France, September 2014. doi: 10.1007/978-3-662-44848-9_22. URL `https://hal.inria.fr/hal-01096610`.

Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, 55:443–497, 2016.

Chuong B Do and Serafim Batzoglou. What is the expectation maximization algorithm? *Nature biotechnology*, 26(8):897–899, 2008.

Prajant Doshi and Piotr Gmytrasiewicz. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24, 07 2005.

Kai-ping Feng and Fang Yuan. Static hand gesture recognition based on hog characters and support vector machines. In *2013 2nd international symposium on instrumentation and measurement, sensor network and automation (IMSNA)*, pages 936–938. IEEE, 2013.

Javier García, José E. Florez, Álvaro Torralba, Daniel Borrajo, Carlos Linares López, Ángel García-Olaya, and Juan Sáenz. Combining linear programming and automated planning to solve intermodal transportation problems. *European Journal of Operational Research*, 227 (1):216–226, 2013. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2012.12.018. URL `https://www.sciencedirect.com/science/article/pii/S0377221712009496`.

J. Goetz, S. Kiesler, and A. Powers. Matching robot appearance and behavior to tasks to improve human-robot cooperation. In *The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003.*, pages 55–60, 2003. doi: 10.1109/ROMAN.2003.1251796.

José Carlos González, Fernando Fernández, Ángel García-Olaya, and Raquel Fuentetaja. On the application of classical planning to real social robotic tasks. In *Proceedings of the 5th Workshop on Planning and Robotics (PlanRob), ICAPS Conference, Pittsburgh, Pennsylvania, USA*, pages 38–47, 2017.

O Can Görür, Benjamin Rosman, Fikret Sivrikaya, and Sahin Albayrak. Social cobots: Anticipatory decision-making for collaborative robots incorporating unexpected human behaviors. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 398–406, 2018.

Marek Grześ, Pascal Poupart, Xiao Yang, and Jesse Hoey. Energy efficient execution of POMDP policies. *IEEE Transactions on Cybernetics*, 45, 2015. doi: 10.1109/TCYB.2014.2375817.

Kelleher R. Guerin, Colin Lea, Chris Paxton, and Gregory D. Hager. A framework for end-user instruction of a robot assistant for manufacturing. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6167–6174, 2015. doi: 10.1109/ICRA.2015.7140065.

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yanlin Han and Piotr Gmytrasiewicz. IPOMDP-Net: A deep neural network for partially observable multi-agent planning using interactive POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6062–6069, 2019. doi: 10.1609/aaai.v33i01.33016062. URL `https://ojs.aaai.org/index.php/AAAI/article/view/4562`.

Eric A. Hansen. An improved policy iteration algorithm for partially observable MDPs. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997.

Eric A. Hansen. Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 211–219, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 155860555X.

Eric A. Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1):35–62, 2001. doi: https://doi.org/10.1016/S0004-3702(01)00106-0.

Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5):503–535, 2009. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2008.10.013. URL `https://www.sciencedirect.com/science/article/pii/S0004370208001926`. Advances in Automated Plan Generation.

Trong Nghia Hoang and Kian Hsiang Low. Interactive POMDP lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 04 2013.

Jörg Hoffmann and Ronen Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proc. ICAPS*, volume 2005, pages 71–80, 2005.

Jörg Hoffmann and Ronen I Brafman. Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, 170(6-7):507–541, 2006.

R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

ISO 10218-1:2011. Robots and robotic devices – safety requirements for industrial robots. Standard, International Organization for Standardization, 2011.

ISO 10218-2:2011. Robots and robotic devices – safety requirements for industrial robots. Standard, International Organization for Standardization, 2011.

Mithun George Jacob, Yu-Ting Li, George A Akingba, and Juan P Wachs. Collaboration with a robotic scrub nurse. *Communications of the ACM*, 56(5):68–75, 2013.

Glenn Jocher. ultralytics/yolov5: v3.1 - bug fixes and performance improvements. `https://github.com/ultralytics/yolov5`, October 2020. URL `https://doi.org/10.5281/zenodo.4154370`.

Yong Gyu Jung, Min Soo Kang, and Jun Heo. Clustering performance comparison using K-means and expectation maximization algorithms. *Biotechnology & Biotechnological Equipment*, 28 (sup1):S44–S48, 2014.

Abir-Beatrice Karami. *Modèles Décisionnels d'Interaction Homme-Robot*. Theses, Université de Caen, December 2011. URL `https://hal.archives-ouvertes.fr/tel-01076430`.

Abir-Beatrice Karami, Laurent Jeanpierre, and Abdel-Illah Mouaddib. Partially observable Markov decision process for managing robot collaboration with human. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 518–521, 2009. doi: 10.1109/ICTAI.2009.61.

Azfar Khalid, Pierre Kirisci, Zied Ghrairi, Klaus-Dieter Thoben, and Jürgen Pannek. A methodology to develop collaborative robotic cyber physical systems for production environments. *Logistics Research*, 9(1):1–15, 2016.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293. Springer-Verlag, 2006. doi: 10.1007/11871842_29. URL `http://dx.doi.org/10.1007/11871842_29`.

David M. Kreps. *Nash Equilibrium*, pages 167–177. Palgrave Macmillan UK, London, 1989. ISBN 978-1-349-20181-5. doi: 10.1007/978-1-349-20181-5_19. URL `https://doi.org/10.1007/978-1-349-20181-5_19`.

Akshat Kumar and Shlomo Zilberstein. Anytime planning for decentralized POMDPs using expectation maximization. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, page 294–301. AUAI Press, 2010. ISBN 9780974903965.

Akshat Kumar, Shlomo Zilberstein, and Marc Toussaint. Scalable multiagent planning using probabilistic inference. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, page 2140–2146. AAAI Press, 2011. ISBN 9781577355151.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *In Proc. Robotics: Science and Systems*, 2008.

Sonya S Kwak. The impact of the robot appearance types on social interaction with a robot and service evaluation of a robot. *Archives of Design Research*, 27(2):81–93, 2014.

Erwan Lecarpentier and Emmanuel Rachelson. *Non-Stationary Markov Decision Processes a Worst-Case Approach Using Model-Based Reinforcement Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Dongheui Lee, Christian Ott, Yoshihiko Nakamura, and Gerd Hirzinger. Physical human robot interaction in imitation learning. In *2011 IEEE International Conference on Robotics and Automation*, pages 3439–3440. IEEE, 2011.

Yee Yeng Liau and Kwangyeol Ryu. Status recognition using pre-trained yolov5 for sustainable human-robot collaboration (hrc) system in mold assembly. *Sustainability*, 13(21), 2021. ISSN 2071-1050. doi: 10.3390/su132112044. URL `https://www.mdpi.com/2071-1050/13/21/12044`.

Kai Lin, Yihui Li, Jinchuan Sun, Dongsheng Zhou, and Qiang Zhang. Multi-sensor fusion for body sensor network in medical human–robot interaction scenario. *Information Fusion*, 57: 15–26, 2020.

Miao Liu, Christopher Amato, Xuejun Liao, Lawrence Carin, and Jonathan P How. Stick-breaking policy learning in Dec-POMDPs. *Ijcai*, page 7, 2015.

Zhihao Liu, Quan Liu, Lihui Wang, Wenjun Xu, and Zude Zhou. Task-level decision-making for dynamic and stochastic human-robot collaboration based on dual agents deep reinforcement learning. *The International Journal of Advanced Manufacturing Technology*, 115, 08 2021. doi: 10.1007/s00170-021-07265-2.

Honghao Lv, Geng Yang, Huiying Zhou, Xiaoyan Huang, Huayong Yang, and Zhibo Pang. Teleoperation of collaborative robot for remote dementia care in home environments. *IEEE Journal of Translational Engineering in Health and Medicine*, 8:1–10, 2020. doi: 10.1109/JTEHM.2020.3002384.

Liam C. MacDermed and Charles L. Isbell. Point based value iteration with optimal belief compression for Dec-POMDPs. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/c9e1074f5b3f9fc8ea15d152add07294-Paper.pdf`.

Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1):5–34, 2003. ISSN 0004-3702. Planning with Uncertainty and Incomplete Information.

Anthony S Maida. Introspection and reasoning about the beliefs of other agents. In *Proceedings of the Eights Annual Conference of the Cognitive Science Society*, pages 187–195, 1986.

Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Abdullah Mohammed, Bernard Schmidt, and Lihui Wang. Active collision avoidance for human–robot collaboration driven by vision sensors. *International Journal of Computer Integrated Manufacturing*, 30(9):970–980, 2017.

Saeid Mokaram, Jonathan M. Aitken, Uriel Martinez-Hernandez, Iveta Eimontaite, David Cameron, Joe Rolph, Ian Gwilt, Owen McAree, and James Law. A ROS-integrated API for the KUKA LBR iiwa collaborative robot. *IFAC-PapersOnLine*, 50(1):15859–15864, 2017. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2017.08.2331. URL `https://www.sciencedirect.com/science/article/pii/S2405896317331464`. 20th IFAC World Congress.

Masahiro Mori. The uncanny valley. *IEEE Robotics & Automation Magazine*, 19(2):98–100, 2012.

Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711. Citeseer, 2003.

Hugo Nascimento, Martin Mujica, and Mourad Benoussaad. Collision avoidance in human-robot interaction using Kinect vision system combined with robot's model and data. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10293–10298, 2020. doi: 10.1109/IROS45743.2020.9341248.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

Stefanos Nikolaidis and Julie Shah. Human-robot cross-training: Computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*, HRI '13, page 33–40. IEEE Press, 2013. ISBN 9781467330558.

Stefanos Nikolaidis, Anton Kuznetsov, David Hsu, and Siddhartha Srinivasa. Formalizing human-robot mutual adaptation: A bounded memory model. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 75–82. IEEE, 2016.

Stefanos Nikolaidis, Yu Xiang Zhu, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in shared autonomy. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, page 294–302. Association for Computing Machinery, 2017. ISBN 9781450343367. doi: 10.1145/2909824.3020252. URL `https://doi.org/10.1145/2909824.3020252`.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319289276.

Frans A. Oliehoek, Matthijs Spaan, and Nikos Vlassis. Best-response play in partially observable card games. In *Benelearn*, 2005.

Shayegan Omidshafiei, Ali-akbar Agha-mohammadi, Christopher Amato, and Jonathan P. How. Decentralized control of partially observable Markov decision processes using belief space macro-actions, 2015. URL `https://arxiv.org/abs/1502.06030`.

Sylvie C.W. Ong, Shao W. Png, David Hsu, and Wee S. Lee. POMDPs for robotic tasks with mixed observability. In *Robotics: Science and Systems (RSS)*, 2009.

Takayuki Osogami and Makoto Otsuka. Restricted Boltzmann machines modeling human choice. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/3295c76acbf4caaed33c36b1b5fc2cb1-Paper.pdf`.

Makoto Otsuka and Takayuki Osogami. A deep choice model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016. doi: 10.1609/aaai.v30i1.10059. URL `https://ojs.aaai.org/index.php/AAAI/article/view/10059`.

Joni Pajarinen and Jaakko Peltonen. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *Advances in Neural Information Processing Systems 24*, volume 24, 2011.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, page 1025–1030. Morgan Kaufmann Publishers Inc., 2003.

Pascal Poupart and Craig Boutilier. Bounded finite state controllers. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. URL `https://proceedings.neurips.cc/paper/2003/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16, 2002. ISSN 1076-9757. doi: 10.1613/jair.1024.

Alina Roitberg, Alexander Perzylo, Nikhil Somani, Manuel Giuliani, Markus Rickert, and Alois Knoll. Human activity recognition in the context of industrial human-robot interaction. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–10. IEEE, 2014.

Pol Rosello. Fully-nested interactive POMDPs for partially-observable turn-based games. Technical report, CS 238, Stanford University, 2016.

Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 2008.

Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 101–103. Association for Computing Machinery, 1998. ISBN 1581130570. doi: 10.1145/279943.279964. URL `https://doi.org/10.1145/279943.279964`.

John Rust. Using randomization to break the curse of dimensionality. *Econometrica: Journal of the Econometric Society*, pages 487–516, 1997.

Mohammad Safeea and Pedro Neto. Minimum distance calculation using laser scanner and imus for safe human-robot interaction. *Robotics and Computer-Integrated Manufacturing*, 58:33–42, 2019.

Bernard Schmidt and Lihui Wang. Depth camera based collision avoidance via active robot control. *Journal of Manufacturing Systems*, 33(4):711–718, 2014. ISSN 0278-6125. doi: https://doi.org/10.1016/j.jmsy.2014.04.004. URL `https://www.sciencedirect.com/science/article/pii/S0278612514000417`.

Sven Seuken and Shlomo Zilberstein. Improved memory-bounded dynamic programming for decentralized pomdps. *ArXiv*, abs/1206.5295, 2007.

Guy Shani, Joelle Pineau, and Roberta Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27:1–51, 2012.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, page 520–527. AUAI Press, 2004. ISBN 0974903906.

Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/c2aee86157b4a40b78132f1e71a9e6f1-Paper.pdf`.

Edward Sondik. The optimal control of partially observable Markov process over the infinite horizon: Discounted costs. *Operations Research*, 26:282–304, 04 1978. doi: 10.1287/opre.26.2.282.

Matthijs TJ Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of artificial intelligence research*, 24:195–220, 2005.

Zachary N Sunberg and Mykel J Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

Daniel Szer, François Charpillet, and Shlomo Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 2005.

Aaquib Tabrez and Bradley Hayes. Improving human-robot interaction through explainable reinforcement learning. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 751–753, 2019. doi: 10.1109/HRI.2019.8673198.

Aaquib Tabrez, Matthew Luebbers, and Bradley Hayes. A survey of mental modeling techniques in human-robot teaming. *Current Robotics Reports*, 1, 12 2020. doi: 10.1007/s43154-020-00019-0.

Tapio Taipalus and Kazuhiro Kosuge. Development of service robot for fetching objects in home environment. In *2005 International Symposium on Computational Intelligence in Robotics and Automation*, pages 451–456. IEEE, 2005.

Adriana Tapus, Maja J Mataric, and Brian Scassellati. Socially assistive robotics [grand challenges of robotics]. *IEEE robotics & automation magazine*, 14(1):35–42, 2007.

Adriana Tapus, Cristian Tapus, and Maja J Mataric. The use of socially assistive robots in the design of intelligent cognitive therapies for people with dementia. In *2009 IEEE international conference on rehabilitation robotics*, pages 924–929. IEEE, 2009.

Rundong Tian and Eric Paulos. Adroid: Augmenting hands-on making with a collaborative robot. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 270–281. Association for Computing Machinery, 2021. ISBN 9781450386357. doi: 10.1145/3472749.3474749. URL `https://doi.org/10.1145/3472749.3474749`.

Marc Toussaint, Stefan Harmeling, and Amos Storkey. Probabilistic inference for solving (PO) MDPs. Technical report, Technical Report EDI-INF-RR-0934, School of Informatics, University of Edinburgh, 2006.

Rudolph Triebel, Kai Arras, Rachid Alami, Lucas Beyer, Stefan Breuers, Raja Chatila, Mohamed Chetouani, Daniel Cremers, Vanessa Evers, Michelangelo Fiore, et al. Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and service robotics*, pages 607–622. Springer, 2016.

Vaibhav V. Unhelkar, Shen Li, and Julie A. Shah. Decision-making for bidirectional communication in sequential human-robot collaborative tasks. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '20, page 329–341. Association for Computing Machinery, 2020. ISBN 9781450367462. doi: 10.1145/3319502.3374779. URL `https://doi.org/10.1145/3319502.3374779`.

Lorenzo Vianello, Luigi Penco, Waldez Gomes, Yang You, Salvatore Maria Anzalone, Pauline Maurice, Vincent Thomas, and Serena Ivaldi. Human-Humanoid Interaction and Cooperation: a Review. *Current Robotics Reports*, 2(4):441–454, 2021. ISSN 2662-4087. doi: 10.1007/s43154-021-00068-z. URL `https://doi.org/10.1007/s43154-021-00068-z`.

Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, 2018. ISSN 0957-4158. doi: https://doi.org/10.1016/j.mechatronics.2018.02.009. URL `https://www.sciencedirect.com/science/article/pii/S0957415818300321`.

Ales Vysocky and Petr Novak. Human - robot collaboration in industry. *MM Science Journal*, 2016:903–906, 06 2016. doi: 10.17973/MMSJ.2016_06_201611.

Michael L Walters, Dag S Syrdal, Kerstin Dautenhahn, René Te Boekhorst, and Kheng Lee Koay. Avoiding the uncanny valley: robot appearance, personality and consistency of behavior in an attention-seeking home scenario for a robot companion. *Autonomous Robots*, 24(2):159–178, 2008.

Michael L Walters, Kheng Lee Koay, Dag Sverre Syrdal, Kerstin Dautenhahn, and René Te Boekhorst. Preferences and perceptions of robot appearance and embodiment in human-robot interaction trials. *Procs of New Frontiers in Human-Robot Interaction*, 2009.

Fangju Wang. An I-POMDP based multi-agent architecture for dialogue tutoring. In *2013 International Conference on Advanced ICT and Education (ICAICTE-13)*, pages 469–472. Atlantis Press, 2013.

Haoyu Wang, Biao Zhang, Tingshen Zhang, and Austin Jakacky. Tele-operating a collaborative robot for space repairs with virtual reality. In *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 175–180. IEEE, 2019.

Lihui Wang. Collaborative robot monitoring and control for enhanced sustainability. *The International Journal of Advanced Manufacturing Technology*, 81:1433–1445, 12 2015. doi: 10.1007/s00170-013-4864-6.

Weitian Wang, Rui Li, Zachary Max Diekel, Yi Chen, Zhujun Zhang, and Yunyi Jia. Controlling object hand-over in human–robot collaboration via natural wearable sensing. *IEEE Transactions on Human-Machine Systems*, 49(1):59–71, 2018.

Xi Vincent Wang, Zsolt Kemény, József Váncza, and Lihui Wang. Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation. *CIRP Annals*, 66(1):5–8, 2017.

Yorick Wilks and Afzal Ballim. Multiple agents and the heuristic ascription of belief. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'87, page 118–124. Morgan Kaufmann Publishers Inc., 1987.

Feng Wu, Shlomo Zilberstein, and Nicholas R. Jennings. Monte-Carlo expectation maximization for decentralized POMDPs. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 397–403. AAAI Press, 2013. ISBN 9781577356332.

Bo Yang and Zheng Qin. Composing semantic web services with PDDL. *Information Technology Journal*, 9(1):48–54, 2010.

Yang You, Vincent Thomas, Francis Colas, and Olivier Buffet. Solving infinite-horizon Dec-POMDPs using finite state controllers within JESP. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 427–434, 2021a. doi: 10.1109/ICTAI52525.2021.00069.

Yang You, Vincent Thomas, Francis Colas, and Olivier Buffet. Résolution de Dec-POMDP à horizon infini à l'aide de contrôleurs à états finis dans JESP. In *Actes des seizièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-21)*, 2021b.

Yang You, Vincent Thomas, Francis Colas, and Olivier Buffet. Solving infinite-horizon dec-pomdps using finite state controllers within jesp, 2021c.

Yang You, Vincent Thomas, Francis Colas, Olivier Buffet, and Rachid Alami. Planification robuste pour la collaboration homme-robot. In *Actes des dix-septièmes journées francophones planification, décision, apprentissage pour la conduite de systèmes (JFPDA-22)*, 2022.

Andrea Maria Zanchettin and Paolo Rocco. Path-consistent safety in mixed human-robot collaborative manufacturing environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1131–1136, 2013. doi: 10.1109/IROS.2013.6696492.

Andrea Maria Zanchettin, Nicola Maria Ceriani, Paolo Rocco, Hao Ding, and Björn Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 13(2):882–893, 2016. doi: 10.1109/TASE.2015.2412256.

Yan Zhao, Shuxiang Guo, Yuxin Wang, Jinxin Cui, Youchun Ma, Yuwen Zeng, Xinke Liu, Yuhua Jiang, Youxinag Li, Liwei Shi, et al. A CNN-based prototype method of unstructured surgical state perception and navigation for an endovascular surgery robot. *Medical & Biological Engineering & Computing*, 57(9):1875–1887, 2019.

Wei Zheng, Bo Wu, and Hai Lin. POMDP model learning for human robot collaboration. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1156–1161. IEEE, 2018.

K.J Åström. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174 – 205, 1965. ISSN 0022-247X. doi: https://doi.org/10.1016/0022-247X(65)90154-X. URL `http://www.sciencedirect.com/science/article/pii/0022247X6590154X`.

Résumé étendu

1 Introduction

Les robots collaboratifs sont largement utilisés dans la société moderne dans divers domaines. Par exemple, les robots peuvent travailler comme pulvérisateurs pour aider les agriculteurs à se débarrasser des insectes nuisibles [Berenstein and Edan, 2012], fournir des services dans les hôpitaux et les aéroports [Tapus et al., 2007, Triebel et al., 2016], soulever des objets lourds ou effectuer des manipulations d'objets de haute précision dans l'industrie [Vysocky and Novak, 2016]. Nous pouvons classer les contextes d'application des robots collaboratifs en deux catégories :

- collaboration robot-robot (RRC) : plusieurs robots travaillent ensemble pour réaliser une tâche ou un objectif spécifique en collaboration ;
- collaboration homme-robot (HRC) : des humains et des robots travaillent ensemble de manière coopérative ; généralement, un but est spécifié par les utilisateurs humains ou lié aux objectifs des utilisateurs humains.

Cette thèse est financée par le projet ANR *Flying Co-Worker*, dont le but à long-terme est de proposer un robot manipulateur aérien capable d'aider un travailleur humain à réaliser des tâches complexes. Dans cette thèse, nous nous intéressons à la manière de concevoir le système de prise de décision d'un robot de ce type dans un contexte général, aussi bien dans le cadre de collaborations robot-robot que de collaborations homme-robot. En d'autres termes, nous étudions comment le robot doit décider de ses actions afin d'accomplir une tâche collaborative, que les partenaires soient d'autres robots ou des humains. Nous nous plaçons plus particulièrement dans le cadre de problèmes collaboratifs de prise de décision séquentielle, car (1) les tâches de collaboration nécessitent plusieurs étapes pour être accomplies, et car (2) nous cherchons des politiques permettant de maximiser la performance globale de l'équipe. L'objectif est de calculer, pour chaque agent, une politique indiquant les actions qu'il doit effectuer pour chaque situation qu'il est amené à rencontrer.

De plus, on peut décomposer le processus de décision de manière hiérarchique en décisions bas-niveau et décisions haut-niveau [Cacace et al., 2015]. Dans le cadre de la robotique, les décisions bas-niveau correspondent au contrôle de bas niveau, c'est-à-dire aux commandes motrices, et les décisions haut-niveau correspondent à la planification des tâches, c'est-à-dire au choix d'actions à grosse granularité qui déclencheront des appels aux contrôleurs bas-niveau en charge de réaliser l'action (comme se déplacer vers un lieu ou réparer un dispositif). Notre thèse se concentre sur la prise de décision haut-niveau (donc la planification des tâches) pour la robotique à la fois en RRC et en HRC.

Cette thèse utilise des modèles de prise de décision markoviens pour formaliser de tels problèmes de décision et calculer les stratégies des robots collaboratifs. Cependant, les questions de recherche posées en RRC et HRC sont différentes en raison de leurs caractéristiques propres.

Dans une tâche de collaboration robot-robot (RRC), l'ensemble des robots constitue un système multi-agent (MAS) : ils doivent travaillent ensemble pour atteindre un objectif commun. Il s'agit alors d'attribuer des politiques individuelles à chacun des robots du système pour qu'ils sachent ensuite comment se comporter dans leur environnement. Ainsi, nous étudions comment concevoir des algorithmes qui calculent des politiques optimales pour chaque agent afin de maximiser la performance globale lors de l'exécution décentralisée de ces politiques. D'autre part, dans une tâche de collaboration homme-robot, seuls les robots peuvent être contrôlés, mais pas les hommes. Par conséquent, le défi consiste à concevoir un système de prise de décision uniquement pour le robot pour qu'il soit capable de réaliser la tâche collaborative, tout en étant incertain quant aux comportements de l'utilisateur humain ou à ses objectifs.

La table 1 donne une vue synthétique des contributions que nous résumons ci-après. Pour la collaboration robot-robot, cette thèse fournit de nouvelles techniques pour résoudre les problèmes de décision multi-agents dans lesquels les agents n'ont que des observations partielles de l'environnement (Dec-POMDP). Plus spécifiquement, en s'inspirant de JESP [Nair et al., 2003], nous cherchons des solutions à l'équilibre de Nash en optimisant itérativement la politique de chaque agent jusqu'à ce qu'aucune amélioration ne puisse être apportée. De plus, JESP possède un inconvénient majeur : il ne peut résoudre que des problèmes à horizon fini car il utilise des politiques représentées par des arbres sur l'historique des actions-observations passées. Dans cette thèse, pour contourner ce problème, nous utilisons des contrôleurs à états finis pour représenter chaque politique, et proposons deux nouveaux algorithmes (Inf-JESP et MC-JESP) pour résoudre les problèmes à horizon infini. Ces deux algorithmes originaux diffèrent en ce que l'un utilise un modèle explicite tandis que l'autre utilise un modèle génératif (un simulateur de type boîte noire qui permet de modéliser de grands problèmes). Nous démontrons expérimentalement la validité de nos approches sur des problèmes de référence.

Pour la collaboration homme-robot, nous nous concentrons sur la prise de décision du robot et cherchons à calculer les politiques du robot robustes à des objectifs et comportements humains incertains. Contrairement à d'autres approches [Nikolaidis et al., 2017, Chen et al., 2018], qui nécessitent les connaissances d'experts ou des données collectées pour fournir un modèle humain sur lequel bâtir la stratégie du robot collaboratif, nous fournissons une approche capable de générer, automatiquement et en s'appuyant sur des modèles de prise de décision de Markov, un modèle de comportement humain incertain (une politique) pour chaque objectif humain possible et qui tienne compte du comportement possible du robot. Ensuite, pour dériver des politiques robustes pour le robot, nous formalisons le problème de prise de décision du robot (sous la forme d'un POMDP) en considérant que ce robot fait face à ces comportements humains incertains générés à l'étape précédente. Nous distinguons deux variantes de cette méthode de génération de stratégie robuste. Le premier algorithme est basé sur la connaissance explicite du modèle de la tâche de collaboration et sur le calcul d'une politique robuste du robot en estimant les comportements humains possibles. Cette méthode fonctionne hors ligne, nous calculons donc d'abord une politique de robot robuste et l'assignons au robot avant l'exécution réelle. Le second algorithme fonctionne en ligne, et un simple simulateur de type boîte noire est suffisant. À chaque pas de temps, le robot simule les évolutions possibles de la situation en considérant ce simulateur du monde et le comportement de l'agent, puis décide de son action à partir de ces simulations. Pour évaluer nos méthodes, un scénario de résolution d'une tâche collaborative a été conçu pour mener des expériences et présenter des résultats qualitatifs et quantitatifs.

TABLE 1 : Relations des différentes contributions

Task Type	RRC	HRC
Explicit Model	Inf-JESP	Robust Robot Planning - Offline
Generative Model	MC-JESP	Robust Robot Planning - Online
Research Interest	Optimality	Robustness

2 Contexte - Modèles de prise de décision markoviens

Cette thèse s'appuie principalement sur des modèles de prise de décision markoviens (POMDP et Dec-POMDP) pour dériver des politiques, grâce auxquelles l'agent prendra des décisions à partir d'observations provenant de l'état de l'environnement. Dans de tels modèles, la dynamique de l'environnement est modélisée par une fonction de transition probabiliste et une fonction d'observation probabiliste. La fonction de récompense définit l'objectif des agents, qui doivent maximiser l'espérance de la récompense cumulée pendant l'exécution de la tâche. La caractéristique clé des modèles de prise de décision markoviens, appelée *Propriété de Markov*, est que la distribution de probabilité sur l'état suivant ne dépend que de l'état actuel à l'instant t et des actions entreprises par les agents au même instant, et non de l'historique.

2.1 Processus de décision markoviens partiellement observables

Dans certains cas, l'information complète sur l'environnement n'est pas directement accessible, et l'agent ne dispose que d'une information partielle sur l'état courant [Åström, 1965]. Le modèle POMDP (Partially Observable Markov Decision Process) est conçu pour formaliser de tels problèmes de décision. Un POMDP est défini par un tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, T, O, r, b_0 \rangle$ où :

- $\mathcal{S}$ est un ensemble fini d'états ;
- $\mathcal{A}$ est un ensemble fini d'actions ;
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ désigne la fonction de transition ; $T(s, a, s')$ est la probabilité de transiter vers l'état suivant s' à partir de l'état s si l'action conjointe a est exécutée ;
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ est la fonction de récompense ; $r(s, a)$ donne la récompense immédiate pour l'exécution de l'action a dans l'état s ;
- Ω est un ensemble fini d'observation ;
- $O(o, a, s') = Pr(o|s', a)$ est la fonction d'observation ; elle indique la probabilité de recevoir une observation o étant donné le prochain état s' atteint en effectuant l'action a ;
- b_0 est une distribution initiale sur les états.

À chaque pas de temps t, l'exécution d'une action a dans l'état s induit une récompense $r(s, a)$ et conduit à l'état s' avec une probabilité $T(s, a, s')$. L'agent reçoit ensuite une observation o qui contient une information partielle ou bruitée sur l'état du système.

Dans les POMDP, puisque l'état réel n'est plus accessible par l'agent, toute l'information dont dispose l'agent au moment de l'exécution est l'historique action-observation (AOH), et ses décisions dépendront donc de cet historique. Un POMDP définit un problème consistant à

trouver une politique, c'est-à-dire une fonction de l'historique vers les actions, dont l'exécution maximise l'espérance des récompenses cumulées.

Cependant, dans les problèmes de prise de décision à long terme, le stockage de ces historiques est coûteux. Par conséquent, une autre approche classique consiste à utiliser une distribution de probabilité sur les états possibles, appelée état de croyance (ou croyance) b, pour condenser l'information sur l'état actuel du système. De plus, l'agent doit mettre à jour sa croyance après l'exécution d'une action a et la réception d'une observation o. Ce processus de mise à jour de la croyance est détaillé comme suit, où $b^{a,o}$ est une nouvelle croyance et $b^{a,o}(s')$ est la probabilité d'être dans l'état s' dans cet état de croyance :

$$b^{a,o}\left(s'\right) = Pr\left(s'|o,a,b\right) = \frac{Pr(s',o,a,b)}{Pr(o,a,b)}, \tag{1}$$

$$= \frac{Pr\left(o|s',a,b\right) Pr\left(s'|a,b\right) Pr(a,b)}{\sum_{s'}\sum_{s} Pr(o,s,s',a,b)}, \tag{2}$$

$$= \frac{Pr\left(o|s',a\right) \sum_{s} Pr\left(s'|a,b,s\right) Pr(s|a,b)}{\sum_{s'} Pr\left(o|s',a\right) \sum_{s} Pr\left(s'|a,b,s\right) Pr(s|a,b)}, \tag{3}$$

$$= \frac{O\left(s',a,o\right) \sum_{s} T\left(s,a,s'\right) b(s)}{\sum_{s'} O\left(s',a,o\right) \sum_{s} T\left(s,a,s'\right) b(s)}. \tag{4}$$

La fonction de valeur optimale d'un POMDP est définie par l'équation d'optimalité de Bellman :

$$V^*(b) = \max_{a\in\mathcal{A}} \left[r(b,a) + \gamma \sum_{o\in\Omega} Pr(o|b,a) V^*(b') \right] \tag{5}$$

et une politique optimale est obtenue comme suit :

$$\pi^*(b) = \arg\max_{a\in\mathcal{A}} \left[r(b,a) + \gamma \sum_{o\in\Omega} Pr(o|b,a) V^*(b') \right], \tag{6}$$

où la fonction $r(b,a)$ spécifie l'espérance de récompense immédiate reçue lorsque l'action a est exécutée dans l'état de croyance b :

$$r(b,a) = \sum_{s\in\mathcal{S}} b(s) r(s,a),$$

et la fonction $Pr(o|b,a)$ est la probabilité de recevoir une observation o après avoir exécuté l'action a dans l'état de croyance b :

$$Pr(o|b,a) = \sum_{s'\in\mathcal{S}} O\left(s',a,o\right) \sum_{s\in\mathcal{S}} T\left(s,a,s'\right) b(s).$$

Les solveurs POMDP les plus récents sont des algorithmes à base de points, notamment PBVI [Pineau et al., 2003], HSVI [Smith and Simmons, 2004] et SARSOP [Kurniawati et al., 2008] ; et les méthodes fondées sur l'échantillonnage, telles que POMCP [Silver and Veness, 2010] et DESPOT [Somani et al., 2013]. Ces deux types de méthodes peuvent résoudre des problèmes POMDP complexes comportant des milliers d'états. Dans cette thèse, SARSOP et POMCP sont utilisés comme outils pour dériver efficacement des politiques POMDP.

2.2 Processus de décision markoviens décentralisés partiellement observables

Le formalisme Dec-POMDP est une extension du formalisme POMDP où plusieurs agents interagissent dans le même environnement et partagent la même fonction de récompense. Par rapport à un POMDP, un Dec-POMDP présente les caractéristiques suivantes :

- l'action (jointe) est décomposée en une action par agent,
- l'observation (jointe) est décomposée en une observation par agent, et
- la politique (jointe) est décomposée en des politiques individuelles séparables, une par agent.

Un Dec-POMDP avec $|\mathcal{I}|$ agents est représenté comme un tuple $M \equiv \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, T, O, R, b_0, H, \gamma \rangle$ [Bernstein et al., 2002], où :

- $\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$ est un ensemble fini d'agents ;
- $\mathcal{S}$ est un ensemble fini d'états ;
- $\mathcal{A}^i$ est l'ensemble fini des actions de l'agent i ; $\mathcal{A} = \times_i \mathcal{A}^i$ est l'ensemble fini des actions conjointes ;
- Ω^i est l'ensemble fini des observations de l'agent i ; $\Omega = \times_i \Omega^i$ est l'ensemble fini des observations conjointes ;
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ désigne la fonction de transition ; $T(s, a, s')$ est la probabilité de transiter vers l'état suivant s' à partir de l'état s si l'action conjointe a est exécutée ;
- $O : \mathcal{A} \times \mathcal{S} \times \Omega \to \mathbb{R}$ est la fonction d'observation ; $O(a, s', o)$ est la probabilité d'observer o si l'action conjointe a est effectuée et l'état résultant est s' ;
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ est la fonction de récompense ; $R(s, a)$ donne la récompense immédiate pour l'exécution de l'action commune a dans l'état s ;
- b_0 est la distribution de probabilité initiale sur les états ;
- $H \in \mathbb{N} \cup \{\infty\}$ est l' horizon de temps (éventuellement infini) ;
- $\gamma \in [0, 1)$ est le facteur d'actualisation, lequel définit l'escompte appliqué aux récompenses futures.

Chaque agent i peut être équipé d'une politique π^i qui fait correspondre ses historiques d'observation et d'action possibles à des actions. L'objectif est alors de trouver une politique jointe $\pi = \langle \pi^1, \ldots, \pi^{|\mathcal{S}|} \rangle$ qui maximise un critère de performance correspondant à la somme décomptée des récompenses :

$$V_H^\pi(b_0) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(S_t, A_t) \mid S_0 \sim b_0, \pi \right].$$

3 Travaux connexes

3.1 Prise de décision pour la collaboration robot-robot

L'un des principaux intérêts de cette thèse est la prise de décision multi-agent dans le contexte de la collaboration robot-robot. Nous utilisons le cadre Dec-POMDP pour modéliser les problèmes de collaboration robot-robot dans cette thèse car il s'agit du cadre, parmi les modèles de décision markoviens, où plusieurs agents optimisent un critère de performance partagé sous des incertitudes dues à la dynamique de l'environnement et aux observations partielles et bruitées des agents. Cependant, la résolution d'un Dec-POMDP est difficile. Même pour un Dec-POMDP à horizon fini, il a été prouvé que le processus de résolution est NEXP dans le pire des cas [Bernstein et al., 2002] et la résolution d'un Dec-POMDP à horizon infini est indécidable [Oliehoek and Amato, 2016, Madani et al., 2003]. Une différence essentielle par rapport aux POMDP est que l'utilité de la politique d'un agent n'est pas seulement liée à la dynamique de l'environnement, mais dépend également des interactions avec les autres agents et de leurs politiques. Dans cette thèse, nous classons les solveurs Dec-POMDP représentatifs comme suit :

- les algorithmes qui optimisent les politiques de tous les agents simultanément, incluant MAA* [Szer et al., 2005], FB-HSVI [Dibangoye et al., 2016], et PBVI-BB [MacDermed and Isbell, 2013]. MAA* cherche une solution optimale pour les Dec-POMDP à horizon fini en utilisant un schéma A* standard. FB-HSVI et PBVI-BB transforment un Dec-POMDP en un (PO)MDP et le résolvent avec les techniques POMDP. FB-HSVI et PBVI-BB peuvent tous deux être utilisés pour les Dec-POMDP à horizon infini, et FB-HSVI donne des solutions dont l'erreur est bornée à ϵ près en approchant l'horizon infini à l'aide d'horizons finis. oSARSA [Dibangoye and Buffet, 2018] étend l'algorithme FB-HSVI au cadre des modèles génératifs où un Dec-POMDP explicite n'est pas disponible. Cependant, toutes ces méthodes doivent faire face à un problème d'explosion combinatoire lorsqu'elles sont confrontées à de grands problèmes.

- les algorithmes qui alternent entre les agents, incluant JESP [Nair et al., 2003] et Dec-BPI [Bernstein et al., 2005]. Ces algorithmes évitent cette explosion combinatoire en optimisant itérativement la politique de chaque agent en considérant que les politiques des autres sont fixées. Mais l'inconvénient associé est que ces méthodes ne peuvent trouver que des équilibres de Nash au mieux. Dans cette catégorie, JESP est limité aux horizons finis, et Dec-BPI peut être utilisé pour les Dec-POMDP à horizon infini avec n'importe quelle distribution sur les états de départ.

- les algorithmes fondés sur des techniques de type *Expectation-Maximisation* (EM) et qui transforment le problème de planification Dec-POMDP en un problème d'inférence probabiliste [Kumar and Zilberstein, 2010, Kumar et al., 2011, Pajarinen and Peltonen, 2011]. La politique de chaque agent est représentée par un FSC, et ses paramètres sont optimisés à l'aide de l'algorithme EM. Les méthodes basées sur EM peuvent être utilisées pour des Dec-POMDP à horizon infini, et une adaptation des approches de type Monte-Carlo étend leur utilisation à des modèles génératifs. Cependant, les méthodes EM ne garantissent que la convergence vers des optima locaux ou, pire encore, vers des points selle.

À notre connaissance, le schéma JESP n'a été introduit ni dans un cadre à horizon infini ni dans un cadre basé sur un simulateur. Par conséquent, dans cette thèse, deux de nos principales contributions (voir chapitre 4) consistent à : 1. proposer un premier algorithme qui étend l'approche JESP au cadre de l'horizon infini appelé Inf-JESP ; 2. proposer une extension de Inf-JESP qui fonctionne avec des modèles génératifs (MC-JESP).

3.2 Prise de décision pour la collaboration homme-robot

Dans le contexte de la collaboration homme-robot (HRC), contrairement à la collaboration robot-robot, le cadre Dec-POMDP ne peut pas être utilisé directement car le partenaire humain peut se comporter différemment des politiques Dec-POMDP calculées, et ce pour plusieurs raisons :

- les politiques calculées sont trop complexes pour que l'humain puisse les apprendre ;
- l'humain veut conserver son autonomie et ne veut pas suivre aveuglément une politique pré-calculée.

De plus, contrairement à un Dec-POMDP, où tous les agents optimisent une fonction objectif partagée, les objectifs réels des humains dans une tâche HRC peuvent être cachés aux robots. En raison de ces facteurs, dans une tâche HRC, le robot doit tenir compte de l'incertitude sur les objectifs et les comportements humains. Dans cette thèse, inspiré par le travail de Tabrez et al. [2020], nous nous fondons sur le concept de *modèle mental* et sur les catégories qu'il décrit. Théoriquement, en HRC, un robot pourrait posséder un modèle mental d'un humain (qui peut différer des comportements humains réels) parmi l'un des types suivants :

- *Modèle mental de premier ordre (1oMM)* : le robot considère que l'humain se comporte de manière indépendante et ne tient pas compte des actions possibles du robot ;
- *modèle mental de second ordre (2oMM)* : le robot considère que l'humain tient compte du robot, ce qui induit une forme de modélisation récursive jusqu'à une certaine profondeur ;
- *Modèle mental partagé (SMM)* : un SMM suppose que tous les agents de l'équipe ont des attentes communes et raisonnent donc de la même manière, ce qui assure une coordination optimale.

Bien entendu, cette catégorisation s'applique symétriquement à la façon dont l'humain modélise le robot. Nous nous concentrons ici sur les problèmes de dynamique stochastique et d'observabilité partielle, ce qui nous amène à considérer des modèles de décision markoviens. Dans ce contexte, un 1oMM correspond typiquement à un POMDP où l'objectif est de trouver la politique d'un agent d'intérêt, tandis que la politique (connue a priori) de l'autre agent fait partie de la dynamique du système. Par exemple, un "POMDP robot" supposant un comportement humain connu est résolu dans [Unhelkar et al., 2020, Nikolaidis et al., 2017, Chen et al., 2018, 2020]. Les SMM peuvent être formalisés comme des POMDP décentralisés (Dec-POMDP), typiquement utilisés pour optimiser la politique jointe d'une équipe d'agents (avec une fonction de récompense commune). CIRL [Hadfield-Menell et al., 2016] peut être considéré comme un cas particulier où l'humain a une observabilité totale et où le robot n'a qu'une fonction de récompense. L'observabilité de l'état du monde est complète et la seule variable cachée du robot correspond à l'objectif réel de l'homme (sa fonction de récompense), ce qui permet d'utiliser techniques de résolution dédiées proches de la résolution d'un POMDP. Pour leur part, les 2oMM peuvent être formalisés en tant que POMDP interactifs (I-POMDP) [Doshi and Gmytrasiewicz, 2005], où les agents se modélisent les uns les autres de manière imbriquée.

Les 1oMM échoueront dans de nombreuses tâches nécessitant un comportement collaboratif explicite de la part de l'humain, et l'hypothèse principale des SMM est généralement trop forte lorsque des robots et des humains collaborent ensemble. Nous proposons donc d'équiper le robot d'un 2oMM de l'humain. Notre approche de la planification robuste des robots pourrait être formalisée comme un I-POMDP, ce que nous évitons principalement pour simplifier les notations. Cependant, l'utilisation d'un 2oMM doit faire face au paradoxe de l'œuf et de la poule car dériver

le comportement humain requis implique de raisonner sur le comportement du robot que nous essayons de dériver en premier lieu. Ceci est vrai par exemple dans le cas 1. de politiques écrites manuellement et devant décrire le comportement humain [Unhelkar et al., 2020, Nikolaidis et al., 2017], pour lesquelles le concepteur doit raisonner sur le comportement du robot et les réactions de l'humain ; 2. de politiques humaines ou de récompenses humaines apprises (par IRL par exemple) [Russell, 1998, Ng and Russell, 2000], qui nécessitent un robot collaboratif en premier lieu pour démontrer un comportement collaboratif ; ou 3. de politiques humaines planifiées qui nécessitent de disposer d'un modèle de l'environnement incluant le comportement du robot.

Dans cette thèse, nous générons des comportements humains plausibles qui serviront de modèle mental du robot à travers la planification, et abordons le problème de la poule et de l'œuf en se demandant quel serait le comportement de l'humain s'il pouvait aussi contrôler le robot et en supposant ainsi temporairement que l'humain et le robot partagent leurs observations, ce qui revient, pour le robot, à considérer que l'humain adopte un SMM particulier.

4 Contributions

4.1 Recherche d'un équilibre de Nash dans un problème multi-agent à horizon infini

Dans cette thèse, nous modélisons les tâches RRC comme des Dec-POMDP, et nous nous sommes intéressés à la recherche d'une solution correspondant à un équilibre de Nash pour lequel la politique de chaque agent constitue la meilleure réponse aux politiques des autres agents. A cette fin, nous proposons de résoudre les Dec-POMDP par une approche JESP (Joint Equilibrium-Based Search for Policies) [Nair et al., 2003], laquelle recherche des solutions en équilibre de Nash en optimisant itérativement la politique de chaque agent tout en fixant les politiques des autres agents, et ce, jusqu'à convergence. Cependant, JESP présente deux faiblesses :

1. dans JESP, les politiques sont représentées par des arbres ; par conséquent, cet algorithme ne traite que les Dec-POMDP à horizon fini ;

2. les solutions d'équilibre de Nash obtenues par JESP sont des optima locaux mais pas globaux.

Cette thèse propose deux nouveaux algorithmes qui répondent au premier inconvénient de JESP en l'étendant au cadre de l'horizon infini. Pour ce faire, nos méthodes ne s'appuient pas sur des arbres de politique mais sur des contrôleurs à états finis (FSC), lesquels représentent des politiques à horizon infini, et nous essayons de contrôler leur taille.

Notre premier algorithme, appelé *Infinite-Horizon JESP* (Inf-JESP), résout les Dec-POMDP avec un modèle explicite du Dec-POMDP correspondant au problème collaboratif à résoudre. L'algorithme principal de Inf-JESP optimise de manière itérative la politique d'un agent représentée sous la forme d'un FSC tout en fixant les FSC des autres agents jusqu'à ce qu'aucune amélioration ne soit possible. Plus précisément, à chaque itération, nous combinons les politiques FSC des autres agents avec un Dec-POMDP pour générer un POMDP avec un espace d'état étendu. Chaque état étendu $e^t \in \mathcal{E}$ contient ainsi :

- s^t, l'état actuel du Dec-POMDP représentant le problème collaboratif à résoudre ;
- $n^t_{\neq i} \equiv \langle n^t_1, \dots, n^t_{i-1}, n^t_{i+1}, \dots, n^t_n \rangle$, les nœuds internes des automates des $|\mathcal{I}| - 1$ autres agents (agents $\neq i$) à l'instant t, et ;

- o_i^t, l'observation actuelle de l'agent i.

Nous avons donc $\mathcal{E} \stackrel{\text{def}}{=} \mathcal{S} \times N_{\neq i} \times \Omega_i$. Étant donné une action a_i^t, l'état de ce POMDP $e^t \equiv \langle s^t, n_{\neq i}^t, o_i^t \rangle$ évolue selon les étapes suivantes :

1. chaque agent $j \neq i$ choisit son action a_j en fonction de son automate et de son nœud interne actuel ;
2. s^t évolue vers s^{t+1} selon T et l'action jointe sélectionnée par les agents $a^t \equiv \langle a_i^t, a_{\neq i}^t \rangle$; et
3. les nœuds internes des automates des agents j (i inclus) évoluent simultanément selon les observations o_j^{t+1} reçues, inférées à partir de s^{t+1} et de l'action jointe a^t ; ces observations peuvent ne pas être indépendantes les unes des autres, de telle sorte que les automates peuvent évoluer de manière dépendante les uns des autres.

Le choix de e^t comme état étendu fait que ce système correspond effectivement à un POMDP du fait des propriétés suivantes :

- cela induit un processus décisionnel de Markov contrôlé par l'action de l'agent i (cf. la dynamique ci-dessous) ;
- l'observation o_i^t ne dépend que de a_i^t et de e^{t+1} ; et
- la fonction de récompense ne dépend que de e^t et de a_i^t.

En effet, dériver les fonctions de transition, d'observation et de récompenses de ce POMDP conduit à :

$$\begin{aligned}
T_e(e^t, a_i^t, e^{t+1}) &= Pr(e^{t+1}|e^t, a_i^t) \\
&= \sum_{o_{\neq i}^{t+1}} T(s^t, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}) \cdot \eta_{\neq i}(n_{\neq i}^t, o_{\neq i}^{t+1}, n_{\neq i}^{t+1}) \\
&\quad \cdot O(s^{t+1}, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, \langle o_{\neq i}^{t+1}, o_i^{t+1} \rangle), \\
O_e(a_i^t, e_i^{t+1}, o_i^{t+1}) &= Pr(o_i^{t+1}|a_i^t, e_i^{t+1}) \\
&= Pr(o_i^{t+1}|a_i^t, \langle s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}_i^{t+1} \rangle) \\
&= \mathbf{1}_{o_i^{t+1} = \tilde{o}_i^{t+1}}, \\
r_e(e^t, a_i^t) &= r(s^t, a_i^t, \psi_{\neq i}(n_{\neq i}^t)),
\end{aligned}$$

où $\eta_{\neq i}(n_{\neq i}^t, o_{\neq i}^{t+1}, n_{\neq i}^{t+1}) = \prod_{j \neq i} \eta(n_j^t, \tilde{o}_j^{t+1}, n_j^{t+1})$ and $\psi_{\neq i}(n_{\neq i}^t) = a_{\neq i}^t$. Ensuite, pour tirer parti des solveurs POMDP modernes à base de points, nous proposons un algorithme (voir algorithme 7) pour générer l'automate à états fini d'un agent à partir d'un ensemble d'α-vecteurs. De plus, des méthodes d'initialisation heuristiques sont également fournies pour donner de bonnes politiques de départ.

4.2 Planification robuste des robots pour la collaboration homme-robot

Cette thèse aborde le problème de l'incertitude sur les objectifs humains et les comportements induits dans la collaboration homme-robot en observabilité partielle. La contribution peut être principalement divisée en trois parties :

- Nous discutons des avantages et des inconvénients des différents modèles mentaux dans les applications de collaboration homme-robot. Dans cette thèse, nous choisissons d'équiper le robot d'un modèle mental du second ordre (2oMM) pour modéliser le partenaire humain. Nous discutons du paradoxe de la poule et de l'œuf soulevé par l'utilisation de 2oMM imbriqués et fournissons deux méthodes pour surmonter cet obstacle. La première méthode génère automatiquement un FSC humain, en agrégeant plusieurs comportements possibles pour un objectif humain donné (voir algorithme 14). La deuxième méthode fournit, en ligne, une distribution de probabilité des actions humaines étant donné un objectif humain à partir d'un planificateur fondé sur des techniques d'échantillonnage (voir algorithme 16). Pour ces deux méthodes, les paramètres peuvent être réglés pour ajuster la diversité des comportements humains générés par une fonction softmax ;

- Nous proposons un algorithme de planification de robot robuste hors ligne dans la section 6.1.2, lequel repose sur un modèle POMDP explicite avec des incertitudes sur l'objectif et le comportement de l'humain. Nous détaillons formellement comment construire ce POMDP, centré sur le point de vue du robot, à partir des modèles de tâches (Dec-POMDP) et d'une distribution sur les politiques stochastiques humaines (FSCs), une par objectif possible. Une politique d'un robot robuste est ensuite obtenue hors ligne en résolvant ce POMDP ;

- Nous proposons également un algorithme de planification en ligne de robot robuste pour les tâches HRC (voir algorithme 17). Par rapport à la méthode hors ligne qui repose sur un POMDP explicite, notre approche en ligne ne nécessite qu'un modèle génératif et peut donc s'adapter à des problèmes de grande taille. Nous détaillons le processus de construction d'un modèle génératif POMDP valide pour le robot avec un espace d'état étendu qui inclut l'état interne de l'humain. Nous fournissons également un algorithme de planification de robot en ligne basé sur POMCP. En utilisant ce modèle génératif POMDP, le robot planifie ses actions à chaque pas de temps en estimant, en ligne, l'objectif réel de l'humain et ses actions futures possibles.

Grâce à des expériences menées avec des humains synthétiques et réels, nous démontrons que nos approches sont robustes aux comportements humains incertains avec différents objectifs. Nous pensons que ce travail est important pour les contextes de collaboration où le robot et l'humain doivent raisonner sur leurs actions possibles mutuelles, et où la prise en compte de politiques humaines myopes ou déterministes est insuffisante pour générer des politiques robustes pour le robot. De plus, notre approche ne nécessite qu'un Dec-POMDP décrivant la tâche de collaboration homme-robot, sans avoir besoin d'un comportement humain préalable. Cela rend notre méthode générique pour aborder divers problèmes de collaboration homme-robot.

5 Conclusion

Dans cette thèse, nous étudions les questions de prise de décision des robots dans deux domaines de recherche, la collaboration robot-robot (RRC) et la collaboration homme-robot (HRC).

Dans la RRC, plusieurs robots sont censés travailler en collaboration pour réaliser une tâche. Plus précisément, cette thèse se concentre sur les problèmes de RRC où chaque robot ne dispose que d'observations partielles dans un environnement incertain. En s'appuyant sur les modèles de prise de décision markoviens, nous utilisons le cadre Dec-POMDP pour modéliser de tels problèmes RRC et proposons deux nouveaux algorithmes (Inf-JESP et MC-JESP) pour résoudre les Dec-POMDP. Nous observons, à travers des expériences avec des problèmes de référence de

l'état de l'art, que Inf-JESP et MC-JESP sont compétitifs par rapport aux solveurs Dec-POMDP existants. MC-JESP surpasse même certains solveurs qui reposent sur des modèles explicites. De plus, grâce à ces recherches menées pour résoudre les Dec-POMDP à horizon infini pour RRC, nous fournissons des outils pour générer un POMDP à partir duquel construire la meilleure réponse d'un agent donné aux politiques supposées fixes et connues des autres agents. Ceci constitue une base importante pour nos contributions en HRC.

Dans le cadre de la HRC, nous nous intéressons aux scénarios où un humain et un robot doivent coordonner leurs comportements pour réaliser une tâche. Dans ce contexte, désormais, seul un agent, le robot, peut être contrôlé, tandis que l'humain peut avoir des objectifs et des comportements incertains. Dans ce cas, nous proposons deux nouveaux algorithmes pour dériver des politiques robustes pour les robots en raisonnant uniquement sur le descriptif de la tâche collaborative et sans nécessiter d'information préalable sur le comportement de l'humain. Nous montrons que nos approches sont robustes à des objectifs humains incertains et des comportements induits par ces objectifs. Nous avons observé expérimentalement que cette approche fournit aussi des décisions adaptées lorsque le robot doit interagir, en simulation, avec des sujets humains.

Dans le futur, nous aimerions étendre ces travaux dans différentes dimensions entre autres en abordant des problèmes continus et en considérant explicitement les durées des actions, ce qui permettrait d'appliquer nos contributions à des scénarios réels impliquant de vrais robots.