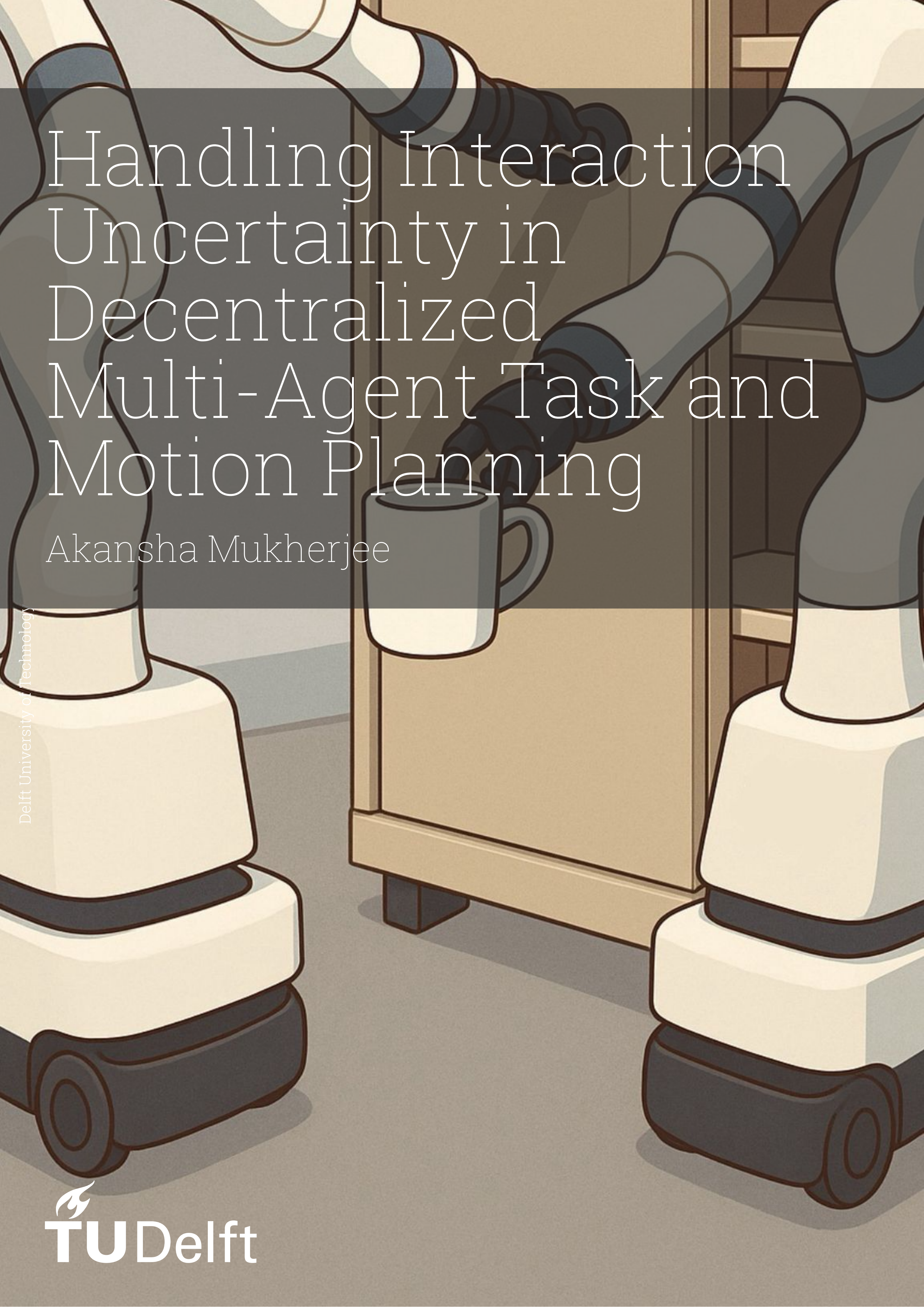


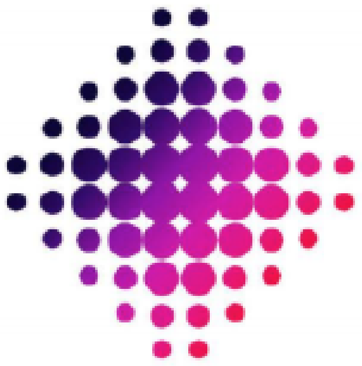
Handling Interaction Uncertainty in Decentralized Multi-Agent Task and Motion Planning

Akansha Mukherjee

Handling Interaction Uncertainty in Decentralized Multi-Agent Task and Motion Planning

by

Akansha Mukherjee



Main Supervisor:	Prof. Javier Alonso-Mora.
Daily Supervisor:	Andreu Matoses-Gimenez.
Faculty:	Robotics, Faculty of Mechanical Engineering, Delft.
Lab:	Autonomous Multi-Robots Lab, TU Delft.

TU Delft

Acknowledgements

I would like to take a moment to acknowledge the people who made this thesis possible. I am grateful to my daily supervisor, Andreu, for his guidance and support throughout the project. Thanks to our weekly meetings reviewing my progress, brainstorming solutions and planning the next steps, I now have a far better understanding of the ”do’s and don’ts” of research than when I started. I would also like to thank my main supervisor, Javier, for taking a keen interest in my work and providing invaluable advice on the technical and non-technical aspects of the project, despite his busy schedule. I am thankful to my friends, near and far, for keeping me company on days I felt unmotivated. In particular, I would like to mention Banhisikha for always being one (or three) phone call(s) away, and Anwesha and Abhyuday for the countless dinners over questionable TV shows. I am also grateful to my Aunt and Uncle for our daily exchanges over text, their advice, and of course, the jokes. Last, but certainly not the least, I would like to thank my parents and grandmother for calling daily to make sure living away from home did not seem so daunting, for their encouragement and for keeping me grounded.

Abstract

We propose a framework that enables an agent to plan effectively under interaction uncertainty in decentralized multi-agent task and motion planning settings for cooperative manipulation tasks. In decentralized systems, each agent computes plans locally, based on local observations and assumptions on the other agents. Decentralization leads to each agent being uncertain about the actions and behavior of the other agents in the scene. We refer to this as ”interaction uncertainty”, as it arises from the implicit interaction between agents. This stems from limitations of the perception system and the inherent ambiguity of actions, meaning that an agent is not certain about the action being performed by the other agent(s). In other words, it is not certain if the other agents' observed actions will obtain desired outcomes. In addition, an agent cannot predict the future behavior of the other agents or how this behavior is influenced by its own actions. The proposed framework addresses these two levels of interaction uncertainty in two-agent settings by leveraging the partial observability of the other agent's short-term intent through ego's perception system, and modeling the behavior of the other agent through an interactive MPD. The uncertainty about the other agent's short- and long-term intent is represented using probabilistic effects of joint actions. The focus of the thesis is on high-level, symbolic uncertainty in action recognition and preferences of the other agent, but may be extended to its low-level, geometric counterpart using existing methods in task and motion planning. Experiments are performed for different behavior models of the unknown agent, from the perspective of the protagonist or ego agent and are compared to baselines from sequential centralized planning. For most settings, the agents are able to eventually find the symbolic goal. Although the experiments are performed for a simple goal with two agents, the consistent convergence to the symbolic goal indicates that this approach may be extended to real-world settings with more complex ambiguity between actions and more agents acting collaboratively. The code for this thesis is available at a public repository.

Contents

1 Introduction

1.1. Motivation

Multi-agent systems (MAS) are a powerful way to solve complex task and motion planning problems in dynamic, unstructured environments. Unlike single-agent systems which are limited by the capabilities of one robot, multi-agent systems distribute tasks across multiple decision-makers or *agents* that jointly influence their environment. Collaboration between agents offers the advantages of improved safety and reliability. MASs also improve efficiency by parallelizing independent sub-tasks, reducing overall execution time, energy consumption and resource utilization. This is especially important in time-sensitive applications such as warehouse automation, disaster response, and so on. Centralized MASs operate with a single ”puppeteer” planner that governs the behavior of all agents in the system. However, with an increase in the number of agents, the size of the joint state and action space grows exponentially, making centralized planning intractable. Further, centralized approaches require constant synchronization with all agents, which is a bottleneck in environments with unreliable networks and high latency. Decentralized MASs enable individual agents to plan asynchronously based on their local observations, thereby reducing the computational burden on a single planner.

However, decentralization is not without its challenges, one of the major ones being the uncertainty due to the presence of other agents in the system. We will call this ”interaction uncertainty”, as it arises due to implicit interactions between the agents. Since agents plan and act independently, they must reason under uncertainty about what other agents are currently doing, what they may do in the future and how this changes in response to the other agents' actions. In order to ensure safe execution without conflicts, deadlocks or redundant actions, an agent must reason about these uncertainties during planning. The many benefits of decentralized multi-agent systems can only be realized if agents are able to reason robustly in the presence of others. This thesis proposes a framework to address this gap by quantifying the interaction uncertainty surrounding other agents and incorporating it in the task and motion planning process, enabling decentralized teams to coordinate more effectively without explicit communication.

1.2. Contribution

This thesis presents a planning framework that incorporates interaction uncertainty about other agent into the planning process of the *ego agent* in a two-agent setting. The ego agent reasons about two levels of uncertainty surrounding the other agent: (i) **current action uncertainty (short-term intent)**: the outcome of the other agent's ongoing action is uncertain, and (ii) **behavior uncertainty (long-term intent)**: the ego agent is uncertain about the other agent's future plan and how it may change in response to ego's actions. In addition, the ego agent also accounts for uncertainty in the outcomes of its own actions. The key contributions of this thesis are:

C1: **Modeling current action uncertainty:** In a two-agent interaction, the other agent's current action - and thus the resulting intermediate state of the world - are only *partially observable* to the ego agent through its action recognition system. A *confusability* assumption constrains the set of actions that can be mistaken for one another in terms of their outcomes. Instead of maintaining a full belief over possible actions, the uncertainty is represented as the set of possible outcomes of the most likely observed action of the other agent.

C2: **Modeling behavior uncertainty:** Uncertainty about the other agent's behavior - including preferences and the effect of ego's actions on its future plan - is learned through interaction under a *subintentional fictitious play* assumption. This enables the ego agent to anticipate how the other agent will act, given the state of the world and its own action.

C3: **Bayes-optimistic model learning:** We extend the Bayes optimistic model learning algorithm proposed in [8] to decentralized multi-agent settings. Optimistic symbolic plans are generated assuming cooperative behavior from the other agent, and desirable action outcomes, yielding short action sequences to achieve the goal. High-information transitions sampled from these plans are used to approximate the true probabilities of joint actions in an interactive MDP by focusing on task-relevant regions of the planning space.

2 Research Objectives

2.1. Research Question

How can an ego agent generate a decentralized task and motion plan under interaction uncertainty, arising from partial observability of other agents' current actions and unpredictability of their future behaviors, to enable reliable multi-agent collaboration?

The following sub-questions are relevant:

- How can uncertainty about the current actions of other agents be modeled and solved?
- How can uncertainty about the future intentions of other agents be incorporated into the planning process?

2.2. Problem Statement

Given a symbolic goal, the capabilities of the agents in the scene and a deterministic state of the environment, the goal is to obtain task and motion plans for the protagonist (ego) agent in a decentralized fashion while accounting for the interaction uncertainty due to decentralization. In order to isolate the interaction uncertainty, the ego-agent makes the following assumptions about all agents in the scene:

A1: Fully observable world state

(a) State of manipulable objects

(b) State of all agents

A2: Capabilities of agents

(a) Known *a priori* by other agents

(b) Stationary

A3: Behavior model of agents

(a) Follows some probability distribution

(b) Not known *a priori* by other agents (decentralized planning)

(c) Can be learned through interaction

(d) Stationary

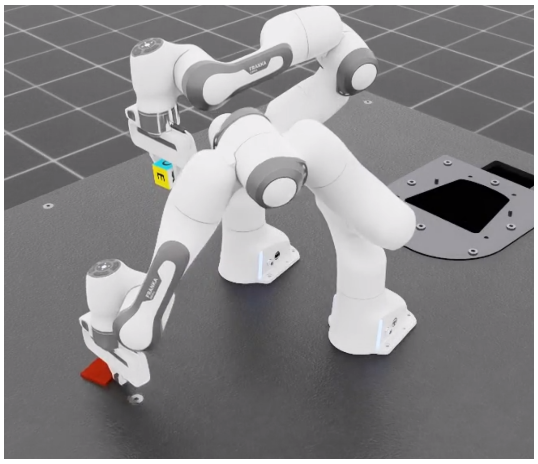
A4: Current action of agents

(a) Partially observable to other agents

(b) Depends on world state and latest actions of other agents

(c) Observed at least once by other agents, mid-execution

The proposed framework is evaluated in two scenarios (see Figure 2.1): the first is a collaborative cleaning task where two manipulators coordinate to lift a block, clean the surface beneath it, and replace the block. The second involves a mug retrieval task, where two (mobile) manipulators work together to open a cabinet door, retrieve a mug, and close the door.


(a) Cleaning scenario

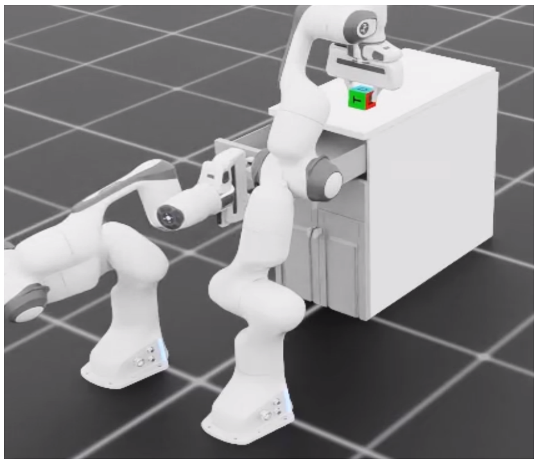

(b) Mug retrieval scenario

Figure 2.1: Multi-agent collaboration scenarios

A decentralized TAMP algorithm runs on the *ego-robot*. The other agent can interact with the environment to change the state of the world and is a source of uncertainty for the ego agent. The uncertainty arises from the fact that the ego-agent cannot gauge the current action of the other agent or predict its behavior with certainty. The ego agent's goal is to solve for actions while accounting for this uncertainty.

3 Related Work

Performing long-horizon tasks autonomously requires an agent, or a collection of agents, to reason about *what* to do and *how* to do it. Simultaneously reasoning about the discrete sequence of tasks and the continuous motions required to accomplish each of the tasks is a *hybrid* planning problem, often addressed using integrated task and motion planning (TAMP). The real world is replete with several forms of uncertainty in the environment of an agent, as well as the effects of the actions executed by it, which must be incorporated in the planning stage. In multi-agent settings, the collaboration between two or more agents is an additional source of uncertainty. In this section, the literature on TAMP is reviewed in light of approaches for single- and multi-agent settings, with and without uncertainty. For a more extensive review of the literature, please refer to the literature study conducted before the thesis.

3.1. Task and Motion Planning (TAMP)

The problem of planning for a robot that operates in environments containing a large number of objects, taking actions to move itself through the world, as well as to change the state of the objects, is known as *task and motion planning* (TAMP) [14]. It is the joint symbolic and geometric planning in a way that captures the complex interdependence between the task-level and motion-level aspects of the problem [43]. Several approaches employ discrete AI task planning [16] to obtain a high-level, symbolic task plan, which is then refined to a low-level, geometric motion plan, if feasible. The complex, often intractable planning problem is broken into a hybrid symbolic search and a set of local motion planning problems, where solving each subproblem is tractable [46].

3.1.1. Deterministic TAMP

Typical TAMP formulations assume a fully observable environment or world and deterministic action outcomes. These approaches often address long-horizon problems with sparse reward through temporal abstraction, factored action models and primitive controller design [14, 13, 7]. A TAMP solution is a finite sequence of action instances $\pi = [a_1, ..., a_k]$ where each a_i includes assigned values for all (symbolic and continuous) parameters satisfying the action's constraints. Selecting the action templates and the values for the discrete template variables specifies the form of the solution (plan skeleton). The unassigned continuous variables are solved for such that they satisfy the constraints imposed by the skeleton, to obtain a solution plan for the TAMP problem.

TAMP problems typically use PDDL [23, 41, 35], STRIPS [11] or some extension of these languages [35] to formalize planning problems. Some key concepts relevant to formulating and solving TAMP problems are introduced in [38, 27, 7]. There are three main classes of methods to solve TAMP problems: (i) constraint-based TAMP [9, 40, 30, 14], (ii) sampling-based TAMP [13, 7] and (iii) optimization-based TAMP [42].

3.1.2. TAMP under Uncertainty

The surroundings of an agent in the real world are often only partially observable due to occlusions, sensor noise, etc. That is, the current state of the world is not known deterministically. Further, actions often have uncertain effects due to the lossiness of symbolic abstractions, controller imperfections, etc. That is, the future state of the world cannot be predicted with certainty. The uncertainty may be at the continuous controller level, i.e. low-level geometric uncertainty in the poses of objects, their mass, coefficient of friction, etc. On the other hand, there may exist high-level, symbolic uncertainty in the predicates that are currently active or become active as a result of some action. The high-level uncertainty may have a low-level source - whether an object is inside a box depends on the geometric pose of the box and the object; or it may be inherently symbolic - the material of an object, for instance.

Future state uncertainty is addressed using Markov Decision Processes (MDP) to capture probabilistic transition dynamics [22, 37]. Current state uncertainty is usually handled by planning in the belief space, where each belief is a probability distribution over the possible world states [8, 25, 1, 20, 15]. Other forms of uncertainty include temporal uncertainty due to scheduling mismatches [44], lossy abstractions [38], interaction uncertainty [33].

3.2. Multi-Agent Task and Motion Planning

Real-world scenarios often require several agents to collaborate in order to achieve a task or set of tasks. For more than one agent, the planning may proceed in a centralized or decentralized fashion.

3.2.1. Deterministic Multi-Agent TAMP

Multi-agent pick-up and delivery (MAPD) is one of the main problems addressed in the literature on multi-agent TAMP [29, 5]. [43] extends the framework from [42] for multi-agent centralized settings with high-dimensional kinematics. [2] is a centralized, knowledge-based approach to TAMP that combines a heuristic task planner, a physics-based motion planner and reasoning over ontologies that encode the knowledge about the problem.

3.2.2. Multi-Agent TAMP under Uncertainty

Multi-agent path finding under uncertainty (MAPFU) [44] presents an approach to solve the multi-agent path finding (MAPF) problem while considering uncertainty only in the low-level motion planning stage. [34] proposes an algorithm for multi-agent task planning under interaction uncertainty for competitive goals. [5] presents a human-aware robot task planning approach based on a hierarchical task model, considering the uncertainty at the symbolic task level alone. [10] proposes a tiered approach interleaving task planning, scheduling, assignment and motion planning for multi-agent systems.

3.3. Gap in the literature

The articles reviewed during the literature study have been categorized on the basis of several properties relevant to the objectives of this thesis. These include the nature of the task, the nature of uncertainty considered, the presence and type of multi-agent interactions, etc. The main focus of the thesis is the research gap identified during the literature review.

3.3.1. Summary of the literature

Citation	Task	Motion	Uncertainty				Multi-Agent		Nature of Tasks	
			World	Action	Interaction	Temporal	Centralised	Decentralised	Manipulation	Navigation
[14], [13]	✓	✓							✓	✓
[9], [30], [42]	✓	✓							✓	
[7]	✓	✓	✓						✓	
[38], [37]	✓	✓		✓					✓	
[1]	✓	✓	✓	✓			✓		✓	✓
[20], [24], [15]	✓	✓	✓	✓					✓	✓
[8]	✓	✓	✓						✓	
[33]	✓	✓		✓						✓
[29], [5]	✓	✓					✓			✓
[43]	✓	✓					✓		✓	✓
[2]	✓	✓					✓		✓	✓
[44]	✓	✓			✓	✓	✓			✓
[34]	✓	✓			✓			✓		✓
[6]	✓							✓	✓	
[10]	✓	✓	✓	✓	✓	✓	✓		✓	
Thesis	✓	✓		✓	✓	✓		✓	✓	

Table 3.1: Summary of the reviewed literature.

Task and motion planning The first two columns identify whether the approaches address task planning, motion planning or both.

Nature of uncertainty An agent may be uncertain about the state of the world, the outcomes of its actions, the uncertainty about the behavior and actions of other agents and the completion times for actions.

Type of multi-agent systems Multi-agent systems may be centralized, where a central planner plans for all agents in the scene, or decentralized, where each agent plans independently.

Nature of tasks Manipulation tasks typically have a longer horizon than navigation tasks, requiring more complex reasoning. Manipulation tasks may be further categorized as fixed-base manipulation tasks or mobile-manipulation tasks. The latter may involve some degree of navigation. Navigation tasks do not cover any instances of manipulation.

3.3.2. Research Gap

Although decentralized multi-agent systems have been explored in prior work [34, 6], some limitations remain. The former considers two agents with conflicting objectives in a navigation context; however, their approach does not extend to cooperative tasks involving complex long-horizon manipulation. Moreover, their method requires a behavioral model for the other agent to be assumed. The latter proposes a task planning framework that partitions tasks between agents based on strong assumptions about the other agent's behavior. While [10] considers interaction between tasks using synergy, it does so in a centralized fashion, not a decentralized one.

Existing research on multi-agent task and motion planning (TAMP) is limited in the exploration of **cooperative manipulation tasks within decentralized settings**. A key challenge in such settings is the interaction uncertainty that arises due to the absence of centralized coordination. This includes uncertainty about the current and future actions of other agents, the impact of the actions of the ego agent on others, the influence of other agents on each other, and the uncertainty due to the completion time of the other agents' actions.

4 Preliminaries

Sequential decision-making problems under uncertainty are often modeled as Partially Observable Markov Decision Processes (POMDPs) [25]. Interactive POMDPs (IPOMDPs) extend this formulation to explicitly represent uncertainty about the behavior of other agents [17]. Although deterministic symbolic planning is unsuitable for stochastic environments, it can be used to generate optimistic plans that guide model learning by focusing on task-relevant parts of the problem. These plans can be simulated to approximately learn the transition dynamics. PPDDL, a symbolic planning language for domains with probabilistic effects, provides a convenient specification format for such problems [35]. Recent work using Bayes-optimistic model learning for single-agent planning in stochastic environments show promising results [8]. Building on these ideas, this thesis proposes a framework that learns transition probabilities over the other agent's behavior, its current action outcomes, and the ego agent's own action outcomes in a goal-directed way.

4.1. Markov Decision Process

A Markov Decision Process (MDP) is a model that captures the characteristics of sequential decision-making under uncertainty [36]. The MDP may be defined as a tuple $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, h, \gamma >$.

Symbol	Name	Description
$\mathcal{S}$	State space	The set of all possible states the environment can be in.
$\mathcal{A}$	Action set	The set of actions the agent can execute.
$\mathcal{T}$	Transition model	$\mathcal{T}(s' \mid s, a) = P(s' \mid s, a)$ specifies the probability of transitioning to state s' when action a is taken in state s. Equivalent to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$.
$\mathcal{R}$	Reward function	$\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. The reward $\mathcal{R}(s, a, s')$ specifies the numerical reward received for taking action a in state s and transitioning to state s'.
h	Horizon	The number of time steps in the future over which rewards are accumulated.
γ	Discount factor	$\gamma \in (0, 1]$ such that the weight of future rewards is lower compared to immediate ones.

Table 4.1: Components of a Markov Decision Process (MDP).

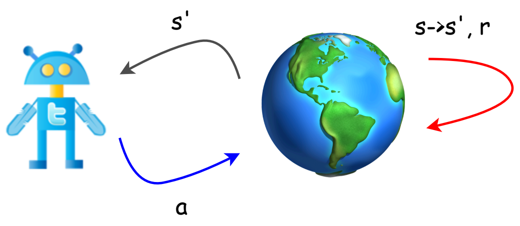


Figure 4.1: Schematic representation of an MDP. At every stage, the agent takes an action and observes the resulting state s'

The agent's policy $\pi : \mathcal{S} \to \mathcal{A}$ determines which action to take at a given state. By the Markov property, future states are assumed to be conditionally independent of past states, given the present state and action: $p(s_{t+1}|s_t, s_{t-1}, ..., s_1, a_t, a_{t-1}, ..., a_1) = p(s_{t+1}|s_t, a_t)$. Thus, the agent's policy only considers the current state in its decision making. The agent seeks to find a policy that maximizes its *utility*, i.e., the expected sum of rewards for horizon $h \geq 1$ and a discount factor $\gamma \in (0, 1]$.

$$U^* = \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=1}^{h} r_t \gamma^{t-1} \right]. \tag{4.1}$$

Solving an MDP amounts to finding an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ such that

$$\pi^* \in \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=1}^{h} r_t \gamma^{t-1} \right]. \tag{4.2}$$

4.2. Partially Observable Markov Decision Process

Noisy or limited sensors often prevent the agent from observing the true state of the environment. Partially Observable Markov Decision Processes (POMDPs) model this uncertainty [26] by extending MDPs to include observations and their likelihood given the state of the environment. An agent does not know the true state of the world but has a *belief* over states, i.e. it uses the history of observations to estimate the probability of each state, and uses this information to decide on an action. A POMDP of agent i is given as $< \mathcal{S}, \mathcal{A}_i, \mathcal{T}_i, \Omega_i, \mathcal{O}_i, \mathcal{R}_i, h, \gamma >$.

Symbol	**Name**	**Description**
$\mathcal{S}$	State space	The set of all possible states of the environment.
$\mathcal{A}_i$	Action space	The set of actions agent i can execute.
$\mathcal{T}_i$	Transition model	$\mathcal{T}_i(s' \mid s, a) = P(s' \mid s, a)$ specifies the probability of transitioning to state s' when when action a is taken in state s.
Ω_i	Observation space	The set of all possible observations for agent i.
$\mathcal{O}_i$	Observation function	$\mathcal{O}_i : \mathcal{S} \times \mathcal{A}_i \times \Omega_i \to [0, 1]$. $\mathcal{O}_i(o \mid s', a)$ specifies the probability that agent i receives observation o after taking action a and transitioning to state s'.
$\mathcal{R}_i$	Reward function	$\mathcal{R}_i(s, a, s')$ specifies the reward received by agent i for taking action a in state s and reaching state s'.
h	Horizon	The number of time steps in the future over which rewards are accumulated.
γ	Discount factor	$\gamma \in (0, 1]$ that reduces the weight of future rewards relative to immediate ones.

Table 4.2: Components of a Partially Observable Markov Decision Process (POMDP) for agent i.

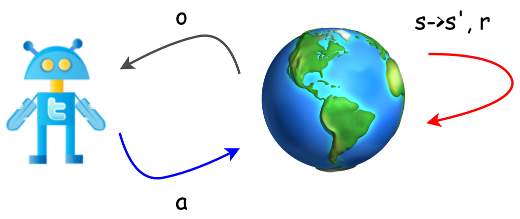


Figure 4.2: Schematic representation of a POMDP. Instead of observing the resulting state s', the agent receives an observation $o \sim O(.|s, a)$

4.3. Interactive Partially Observable Markov Decision Process

Interactive POMDPs (I-POMDPs) address the MAS from the subjective perspective of the protagonist agent, by explicitly modeling the other agent in order to plan while accounting for their anticipated behavior [17]. Each agent computes its plans locally. An I-POMDP of agent i is given as $< \mathcal{IS}_i, \mathcal{A}, \mathcal{T}_i, \Omega_i, \mathcal{O}_i, \mathcal{R}_i, h, \gamma >$.

Symbol	Name	Description
$\mathcal{IS}_i$	Interactive state space	The set of interactive states for agent i, $\mathcal{IS}_i = \mathcal{S} \times \mathcal{M}_j$, where $\mathcal{S}$ is the physical state space and $\mathcal{M}_j$ is the set of possible models of the other agent j.
$\mathcal{M}_j$	Model space of agent j	Each model $m_j \in \mathcal{M}_j$ consists of a history of agent j (used to estimate its belief) and a function mapping this history to a probability distribution over j's possible actions.
$\mathcal{A}$	Joint action space	The set of joint moves for all agents, $\mathcal{A} = \times_j \mathcal{A}_j$.
$\mathcal{T}_i$	Transition model	$\mathcal{T}_i(is' \mid is, a) = P(is' \mid is, a)$ specifies the probability of transitioning from interactive state is to is' when the joint action a is taken.
Ω_i	Observation space	The set of possible observations for agent i.
$\mathcal{O}_i$	Observation function	$\mathcal{O}_i(o \mid is', a)$ specifies the probability that agent i receives observation o after joint action a is taken and the environment transitions to interactive state is'.
$\mathcal{R}_i$	Reward function	$\mathcal{R}_i(is, a, is')$ gives the reward to agent i for taking joint action a in interactive state is and transitioning to is'. Rewards depend only on the physical component of the interactive state.
h	Horizon	The number of time steps in the future over which rewards are accumulated.
γ	Discount factor	$\gamma \in (0, 1]$ that reduces the weight of future rewards relative to immediate ones.

Table 4.3: Components of an Interactive Partially Observable Markov Decision Process (I-POMDP) for agent i.

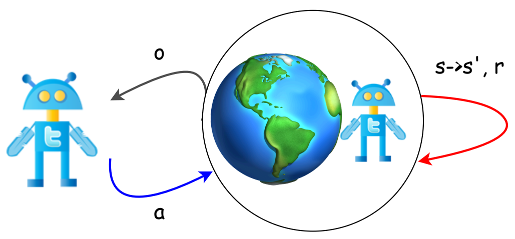


Figure 4.3: Schematic representation of an I-POMDP. The agent reasons about the joint state of the environment and other agent(s).

4.4. Probabilistic Planning and Domain Definition Language

The Planning Domain Definition Language (PDDL) is used to formalize planning problems [23]. It is a deterministic transition system where state variables are boolean facts (predicates). Similar to STRIPS [11], a PDDL action template is specified as a list of free parameters (`:parameters`), a set of preconditions as a logical formula (`:preconditions`) specifying the conditions that must hold in a state for the action to be executable, and a set of effects as a logical conjunction (`:effects`) describing how the action changes the state in which it is executed. A state is a goal state if the goal logical formula holds in that state. Goal specifications, including those that use existential ($\exists$) or universal ($\forall$) quantifiers can be encoded in a PDDL formulation using *axioms* [41]. Probabilistic PDDL (PPDDL) introduces support for probabilistic effects to define probabilistic and decision-theoretic planning problems [35]. Probabilistic effects indicate possible but unknown effects of executing some action.

Listing 4.1: PPDDL-style action with uncertain effects

```
(:action pick
  :parameters (?o - object ?g - grasp)
  :precondition (and (BVPose ?o) (BHandFree))
  :effects (and (not (BVPose ?o)))
  :uconds (and (BClass ?o @glass))
  :ueffects (maybe (Broken ?o) (BGrasp ?o ?g))
)
```

The action template for a ”pick” controller is shown, with a physical object `?o` and a continuous grasp `?g` as parameters. The preconditions require the agent to know the pose of object `?o` on the table with a high probability and believe that its gripper is free. On execution, it is guaranteed that the object's pose on the table will not be known with high confidence. Possible but uncertain effects include the object being broken, and that the robot grabs the object with grasp `?g`. The probability distribution over these outcomes may depend on the material of the object being glass (`:uconds`).

4.5. Combining PPDDL and POMDPs

Action space: When the action space comprises of primitive actions such as joint torques, the horizon for performing meaningful tasks is prohibitively long, making planning intractable. The POMDP is temporally abstracted to obtain a belief space controller MDP, where the action space comprises of low-level short-horizon controllers that operate on beliefs.

State space abstraction: The state space of a POMDP is the space of continuous probabilities or beliefs. Belief-state abstraction is performed to group together operationally similar beliefs using propositions [25], discretizing the state space.

Sparse abstract belief-space controller MDP: Preconditions may be specified for each controller, listing the symbolic predicates that must be active in an abstract belief state for it to be applicable using PPDDL [35]. This results in a sparse, abstract belief-space MDP.

Symbolic planning (PPDDL): A deterministic planner uses the preconditions and probabilistic effects to search for a sequence of controllers to the goal state using all-outcomes determinization [39, 45]. This plan is overly optimistic, being untethered from geometric and physical constraints, but is used to guide model learning.

Goal-directed model learning: High-information transitions are simulated from the optimistic symbolic plans to learn the approximate transition dynamics by focusing exploration on task-relevant parts of the abstract belief space.

Solving for optimal actions: Having approximately learned the dynamics, a probabilistic planner is used to solve for a risk and uncertainty aware policy in the abstract belief space [21].

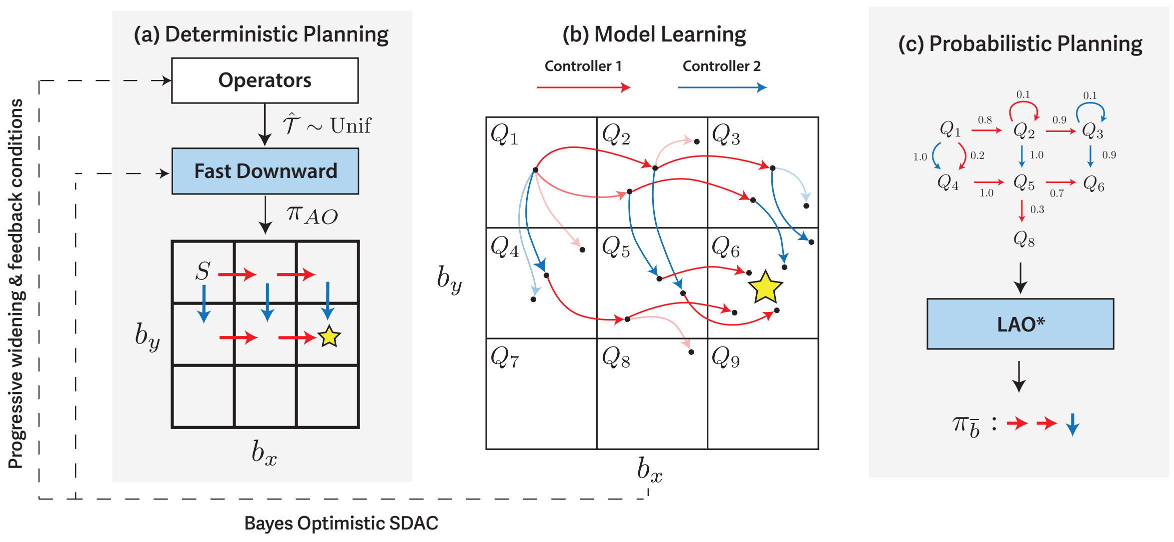


Figure 4.4: The TAMPURA pipeline.

Partially Observable Task and Motion Planning with Uncertainty and Risk Awareness (TAMPURA) is an approach that combines these methods for Bayes optimistic model learning in single-agent settings [8]. First, the continuous belief-space is abstracted, with 9 states. The preconditions and possible but uncertain effects of controller execution are specified. All-outcomes determinization is performed and a symbolic planner solves for several plans to the goal. Next, learning is performed to approximately learn the transition model. Lastly, the MDP is solved for a policy that is uncertainty and risk-aware.

5 Method

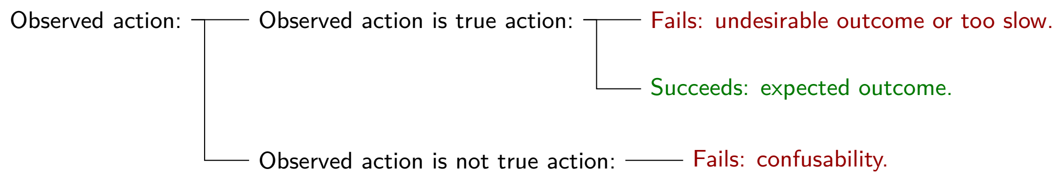
The formalism of POMDPs may be extended to a decentralized MAS as decentralized POMDPs (Dec-POMDPs) where each agent acts on basis of local observations [31]. However, this framework assumes all agents are fully cooperative, with a common reward, and the joint optimal solution is computed centrally. We consider the MAS from the subjective perspective of the ego agent using an interactive POMDP (IPOMDP) formulation, where the ego agent maintains a model of the other agent to make decisions based on its expected behavior [17].

5.1. Interaction uncertainty

Decentralization of a multi-agent system causes the ego-agent to be uncertain about the role of the other agent(s) in the execution of the plan. With two agents in the scene, the ego agent has two levels of uncertainty surrounding the other agent: (i) current action uncertainty and (ii) behavior uncertainty.

5.1.1. Current action uncertainty

Due to imperfections in the ego agent's perception system and the inherent ambiguity of some actions, the current action of the other agent is only partially observable. The ego agent cannot be certain whether the action it perceives is the other agent's true action or whether that action will achieve its expected outcome. That is, the ego agent is not certain about the intermediate state of the world that will result from the observed action. The possibilities are illustrated below. All failure cases are equivalent to the ego agent.


Observed action:
Observed action is true action:
Fails: undesirable outcome or too slow.
Succeeds: expected outcome.
Observed action is not true action:
Fails: confusability.

This may be modeled using a belief or probability distribution over the possible actions of the other agent:

$$b(a^n) = P(a_{other} = a^n); \sum_{a^n \in \mathcal{A}_{other}} b(a^n) = 1.$$

However, in practice, only a small set of actions are confusable with each other and this uncertainty may be modeled as the uncertainty about the outcome of the most likely current action of the other agent.

5.1.2. Behavior uncertainty

Even if the ego agent were able to mitigate the uncertainty about the current action of the other agent, there remains the uncertainty of the ego agent about the long-term plan of the other agent and how this plan is adapted in response to the ego agent's actions. This requires the ego agent to have a behavior model for the other agent.

Agent j's behavior model may be given as $m_j \in \mathcal{M}_j$ where $m_j =< h_j, f_j >$. $h_j \in H_j$ is the history of the other agent and $f_j : H_j \rightarrow \triangle(A_j)$ is the function of agent j. Intentional models assume rational other agents and are computationally expensive due to recursion [18, 17]. We consider a policy reconstruction-based approach, assuming a subintentional, fictitious play model for the other agent, where its next action depends on the current world state and ego's action [4, 12]. This approach can learn arbitrary models for the other agent through interaction but may require many observations to learn a complex model [3].

5.2. Self uncertainty

The ego agent accounts for the possibility that its own actions may have multiple outcomes due to a non-unique mapping from the abstract predicate space to the continuous pose space. We assume full observability at the symbolic level, ignoring low-level pose uncertainty that is typically modeled using beliefs or probability distributions. The proposed framework treats this as action outcome uncertainty: the ego agent's intended action may fail or produce different results, consistent with TAMP literature [8, 25].

$$\mathbf{q} \sim p_\phi(\mathbf{q} \mid s).$$

Given the fully-known symbolic state s, continuous arguments $\mathbf{q}$ are assumed to follow some known probability distribution p_ϕ. We simulate the action with variables sampled from this distribution [13] and estimate the probability of success experimentally.

5.3. Interactive MDP

Since the world is assumed to be fully observable for all agents in the scene, and the uncertainty about the current action of the other agent is modeled as action outcome uncertainty, we simplify the problem to be fully observable and model it as an *interactive MDP* (see Chapter 4, Section 4.3)

$$IMDP_{ego} =< \mathcal{IS}_{ego}, \mathcal{A}, \mathcal{T}_{ego}, \mathcal{R}_{ego}, h, \gamma > .$$

The most likely action of the other agent, together with the ego agent's chosen action, forms the joint action. The ego agent selects its action by solving the interactive MDP while reasoning about: (i) the possible outcomes of the other agent's action, (ii) the anticipated behavior of the other agent, and (iii) the possible outcomes of its own action. This joint action transitions the environment to a new interactive state [32, 17].

5.3.1. State space

The ego agent uses a modified *interactive state* representation for the other agent, using the partial observability of the other agent's current action:

$$\mathcal{IS}_{ego} = \mathcal{S} \times \mathcal{A}_{other},$$

where $\mathcal{S}$ is the abstract physical state of the world and $\mathcal{A}_{other}$ is the set of actions of the other agent. An interactive state $is \in \mathcal{IS}_{ego}$ is given as $is = (s, a_{other})$, a_{other} being the most likely action of the other agent which determinizes the other agent's action in the joint action.

5.3.2. Action space

The action space from the ego agent's perspective is the space of joint actions:

$$\mathcal{A} = \mathcal{A}_{other} \times \mathcal{A}_{ego}.$$

A joint action $a = (a_{other}, a_{ego})$ is the most likely action the other agent is performing and the ego agent's chosen action. Let $\mathrm{Pre_a}$ be the set of preconditions (predicates) for the joint action, $\mathrm{Pre_{other}}$, $\mathrm{Pre_{ego}}$ being the preconditions for the other and ego agent. Similarly, let $\mathrm{Eff_a}$ be the set of guaranteed effects and $\mathrm{UEff_a}$ be the set of possible but uncertain effects for the joint action, etc.

- **Prohibited:** Joint actions where the expected $\mathrm{ueff} \in \mathrm{UEff_{other}}$ violates a precondition $\mathrm{pre} \in \mathrm{Pre_{ego}}$ are prohibited. Example: both agents attempt to pick the same object.
- **Dependent:** Joint actions are where the expected $\mathrm{ueff} \in \mathrm{UEff_{other}}$ satisfies a precondition $\mathrm{pre} \in \mathrm{Pre_{ego}}$ that would otherwise be false are dependent. Example: the other agent opens a cabinet, enabling the ego agent to grasp a mug inside.
- **Independent:** For all remaining joint actions, $\mathrm{Pre_a} := \mathrm{Pre_{other}} \wedge \mathrm{Pre_{ego}}$, $\mathrm{Eff_a} := \mathrm{Eff_{other}} \wedge \mathrm{Eff_{ego}}$, $\mathrm{UEff_a} := \mathrm{UEff_{other}} \times \mathrm{UEff_{ego}}$.

5.3.3. Transition model

Bayes optimistic model learning is used to ensure the ego agent is initially optimistic about the compliance of the other agent, and believes that the joint actions obtain their desired outcomes [8]. Probabilistic effects of PPDDL action templates are used to indicate to the symbolic planner:

(i) Possible outcomes of the other agent's observed action, arising from current action uncertainty (Section 5.1.1)

(ii) Possible next actions of the other agent, arising from behavior uncertainty (Section 5.1.2)

(iii) Possible outcomes of the ego agent's action, arising from self uncertainty (Section 5.2)

The symbolic planner of the ego agent uses all-outcomes determinization to select favorable outcomes for joint actions, including desirable, cooperative behavior from the other agent [45]. That is, it chooses a sequence of actions and corresponding outcomes that lead to the goal state, regardless of whether these transitions are actually feasible. This results in a deterministic transition model:

$$\mathcal{T}_{AO} : \mathcal{IS}_{ego} \times (\mathcal{A} \times \mathcal{IS}_{ego}) \rightarrow \mathcal{IS}_{ego}. \tag{5.1}$$

This optimism leads to bad policies during execution, because the planner may choose actions that rely on unlikely but useful outcomes. To discourage the planner from choosing such action-outcome pairs, cost weights J are added to actions to penalize high risk pairs. Optimal open loop plans are obtained when $J = -log(p)$, p being the true outcome probability. Partial knowledge about $\mathcal{T}(is_{t+1}|is_t, a_t)$ is modeled using a Beta(α, β) distribution, with $\alpha = 1, \beta = 1$ for the Beta prior. Given simulations of a_t from is_t, the posterior is Beta($\alpha + n_s, \beta + n_f$) where n_s is the number of successes (is_{t+1}) and n_f is the number of failures. To enforce optimism against partial knowledge about the transition model, a Bayesian upper confidence bound (UCB) criterion is used [28].

$$J(\bar{b}_t, c_t, \bar{b}_{t+1}) = log[F^{-1}_{Beta(\alpha+n_s,\beta+n_f)}(1 - \frac{1}{i})], \tag{5.2}$$

where $J(is_t, c_t, is_{t+1})$ is the cost applied to the transition $(is_t, c_t) \rightarrow is_{t+1}$ in all-outcomes planning, F^{-1} is the inverse CDF of the Beta posterior and n_s and n_f are successes and failures. After model learning, the final transition model is described as follows, where the function f of the other agent is incorporated in the overall transition model:

$$\mathcal{T}_{ego} : \mathcal{IS}_{ego} \times \mathcal{A} \times \mathcal{IS}_{ego} \rightarrow [0, 1]. \tag{5.3}$$

5.3.4. Reward model

The reward function for the problem is sparse, i.e., the reward for a goal-state is 1.0, while other states have 0.0 reward. The reward model is described below, where only the physical state $\mathcal{S}$ is used for the reward computation:

$$\mathcal{R}_{ego} : \mathcal{IS}_{ego} \times \mathcal{A} \times \mathcal{IS}_{ego} \rightarrow \{0, 1\}. \tag{5.4}$$

5.3.5. Discount factor and planning horizon

Since the problem has a sparse reward structure, a high value is used for the discount factor $\gamma = 0.95$ to propagate the reward to earlier states in long-horizon problems. At the same time, γ should not be 1.0 so the ego agent prefers a shorter sequence of actions to the goal state, if it is reachable in different ways. The planning horizon h is not fixed, as the length of the action sequence to the goal is unknown.

5.3.6. Solving for a policy

The interactive MDP is solved using LAO*, a cyclic graph heuristic search algorithm that extends basic solution algorithms such as value iteration, policy iteration, etc. The algorithm alternates between *state space expansion*, exploring states reachable under the current policy π, and *policy optimization* via Bellman backups, producing a policy that maximizes expected cumulative reward while avoiding computation on irrelevant states.

5.4. Learning the transition model

The transition model probabilities are learned by simulating high-information (high-entropy) transitions from the deterministic symbolic plans (Section 5.3.3). The ego agent learns simultaneously the probabilities associated with (i) the other agents possible responses, (ii) the outcomes of the observed action and (iii) its own action outcomes. The model learning pseudocode is shown in Algorithm 1.

Algorithm 1 Model learning pseudocode [8]

```
Require: Parameters for Bayesian model learning prior $\alpha, \beta$.
Require: Runtime parameters $I, K, S$.
Require: Planning problem $(is_0, G)$.
Require: State from the past iterations of model learning $m_s$.
 1: if $m_s = \emptyset$ then
 2:     $N \leftarrow 0, D \leftarrow 0$        ▷ Initialize counts of observed precondition-effect outcomes.
 3:     $IS_{enc} \leftarrow \{is_0\}$        ▷ Initialize encountered set of states.
 4: else
 5:     $(N, D, IS_e) \leftarrow m_s$        ▷ Restore previous model state.
 6: end if
 7: if Prob(Plan | $m_s$) < envelope_threshold then        ▷ Check plan probability.
 8:     for $i = 1 \to I$ do
 9:         $(\tau_k)_{k=1}^{K} \leftarrow$ DeterminizedPlanner$(is_0, K, N, D, G)$        ▷ Generate K symbolic plans using model.
10:         $\tau^* \leftarrow \{(is, op, is') \in \text{concat}(\tau_1, \ldots, \tau_K) : is \in IS_{enc}\}$        ▷ Concatenate and filter trajectories.
11:         $\vec{\Psi}_{pre} \leftarrow [[is[\psi] : \psi \in op.UCond] : (is, op, is') \in \tau^*]$        ▷ Preconditions of transitions.
12:         $\vec{\Psi}_{eff} \leftarrow [[is'[\psi] : \psi \in op.UEffs] : (is, op, is') \in \tau^*]$        ▷ Effects of transitions.
13:         $\vec{c} \leftarrow [op.c : (is, op, is') \in \tau^*]$        ▷ List of controllers.
14:         $\vec{n_s} \leftarrow [D[x] : x \in \text{zip}(\vec{\Psi}_{pre}, \vec{c}, \vec{\Psi}_{eff})]$        ▷ Number of successes observed.
15:         $\vec{n_f} \leftarrow [N[\Psi_{pre}, c] - n_s : (\Psi_{pre}, c, n_s) \in \text{zip}(\vec{\Psi}_{pre}, \vec{c}, \vec{n_s})]$        ▷ Number of failures.
16:         $\vec{H} \leftarrow [H(\alpha + n_s, \beta + n_f) : (n_s, n_f) \in \text{zip}(\vec{n_s}, \vec{n_f})]$        ▷ Compute entropy or information.
17:         for $j = 1 \to S$ do
18:             $(is_1, op, is_2) \leftarrow \text{pop}(\tau^*, \arg\max \vec{H})$        ▷ Select most uncertain transition.
19:             $is_2 \leftarrow$ Simulate$(is_1, op.c)$        ▷ Controller simulation.
20:             $IS_{enc} \leftarrow IS_{enc} | \{is_2\}$        ▷ Update encountered states.
21:             $\Psi_{pre} \leftarrow [is_1[\psi] : \psi \in op.UCond]$
22:             $\Psi_{eff} \leftarrow [is_2[\psi] : \psi \in op.UEff]$
23:             $N[\Psi_{pre}, op.c] \leftarrow N[\Psi_{pre}, op.c] + 1$
24:             $D[\Psi_{pre}, op.c, \Psi_{eff}] \leftarrow D[\Psi_{pre}, op.c, \Psi_{eff}] + 1$
25:         end for
26:     end for
27: end if
28: $(\hat{T}, IS_{\text{sparse}}) \leftarrow$ Compile$(D, N)$        ▷ Convert counts to sparse abstract MDP.
29: return $(N, D, IS_{enc}), \hat{T}, IS_{\text{sparse}}$        ▷ Learned model and abstract MDP.
```

5.4.1. Model learning hyperparameters

Model learning is deemed successful when at least one goal state is encountered during training. The model-learning algorithm has four main hyperparameters that determine success:

- `num_skeletons` (K): The number of determinized symbolic plans generated per batch of transitions. For the initial state being close to a goal state, and the other agent being compliant with desired behavior, a low value of `num_skeletons` is sufficient; however for an uncooperative other agent or unrealistic transitions, a higher value is required to consider the other possibilities besides the desired outcomes.
- `batch_size` (S): The number of transitions per batch; must be large enough such that sufficient meaningful samples are transitioned in each batch. For a non-compliant other agent, it should be adjusted such that the number of batches ($\frac{\texttt{num\_samples}}{\texttt{batch\_size}}$) and thus the number of times a new set of symbolic plans is generated with risk-informed action costs is higher.
- `num_samples` ($I * S$): The total number of transitions sampled (simulated) during model learning; should be high enough to learn a close approximation of the true transition model. An increase in problem complexity necessitates a larger value of `num_skeletons` to account for different ways to reach a goal state. The total number of transitions increases, requiring a higher value for `num_samples`.
- `envelope_threshold`: The minimum probability of being at the current state, given the executed sequence of actions, such that the previously learned model may be reused. A low value encourages model reuse, speeding up computation. However, if this value is too low, the planner may have a false sense of confidence about its current state, trusting the previous model when it should relearn a more reliable model from the current state.

5.4.2. Interaction uncertainty

The ego agent must learn the probabilities associated with the (i) action outcomes (Section 5.1.1) and (ii) behavior (Section 5.1.2) of the other agent. Ideally, the ego agent would learn this through direct physical or simulated interactions with the other agent, observing its actions via the perception system and evaluating the actual outcomes at the next step. In the absence of a perception system, we assume access to a query function that represents the other agent's behavior or policy. This function must indicate the apparent action of the other agent as well as its true outcome. A human operator could provide these inputs, serving as the query function.

5.4.3. Self uncertainty

Self uncertainty represents the outcome uncertainty surrounding ego's own actions (Section 5.2). The ego agent can obtain a realistic estimate of its own action outcome uncertainty by querying a physics-based simulator. Consider the action `pick_ego`: the probability of success is learned from querying the IsaacLab simulator [1]. First, we must assume a probability distribution for the x and y coordinates of the 2D pose of the object on the table, given its fully-known symbolic location s_{region}:

$$\mathbf{q} \sim p_\phi(\mathbf{q} \mid s_{region}),$$

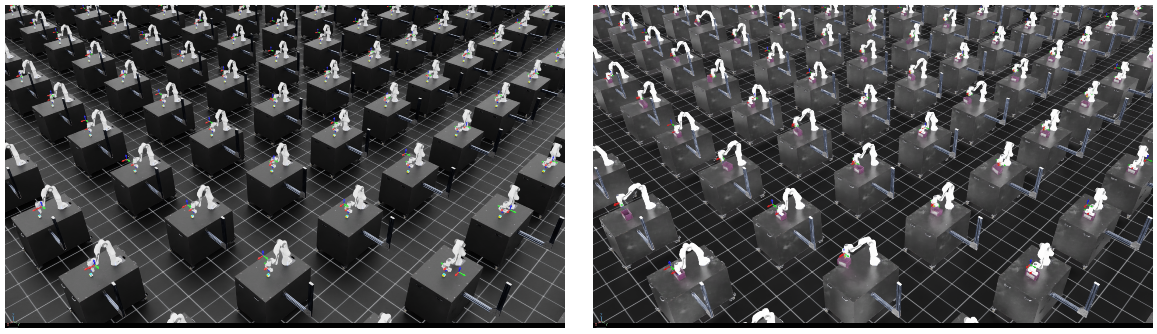
where $\mathbf{q}$ is x or y. We assume that the ego robot has a perception system that can get the exact pose of the object in the scene with a *uniform random* error, while its orientation is known with certainty:

$$x_{obs} = x_{true} + \Delta x, \quad \Delta x \sim \mathcal{U}(-0.010,\ 0.010).$$
$$y_{obs} = y_{true} + \Delta y, \quad \Delta y \sim \mathcal{U}(-0.025,\ 0.025).$$

During model learning, the simulator is queried only once for each action with a sufficient batch of instances to estimate $p_{success}$, as repeated queries are expensive. This is an implementation limitation, not a methodological one. A batch of 1000 simulations are run in parallel, with a Franka Emika-Panda arm [2] mounted on a table, trying to pick a small cube. Different probabilities are learned for picking an object from the neutral (`region_stable_mug`) region and for picking from the cabinet (`region_mug`) (mug-retrieval scenario). The KLT bin asset in IsaacLab

[1] https://isaac-sim.github.io/IsaacLab/main/index.html
[2] https://robodk.com/robot/Franka/Emika-Panda

is used to represent a cabinet which may hinder the motion of the manipulator, leading it to fail more often than it would for the unobstructed pick operation.



(a) The simple pick scenario **(b)** The cabinet pick scenario

Figure 5.1: Simulation settings for different pick actions: mug-retrieval scenario.

The success/failure (1/0) outcomes are assumed to be *Bernoulli distributed* $X_i \sim \text{Bernoulli}(p_{success})$, where $p_{success}$ is the probability of success. The empirical estimate of $p_{success}$ is given below and is used to simulate Bernoulli-distributed outcomes with a random number generator.

$$\hat{p}_{success} = \frac{\texttt{num\_successes}}{\texttt{num\_simulations}}.$$

Low-level motion planning is abstracted away to focus on interaction uncertainty, but may be incorporated to generate hybrid task and motion plans. Transitions may be sampled from a physics-based simulator to learn their quality and feasibility as transition probabilities [13, 8].

5.5. Example

To better understand the role of each kind of uncertainty in isolation, an example is illustrated for the mug retrieval scenario. For simplicity, we assume the other agent (robot_2) can only open or close the door. The initial state and action (top left) and possible next states are shown in Figure 5.2, with probabilities in red indicating infeasible states (probability 0.0).

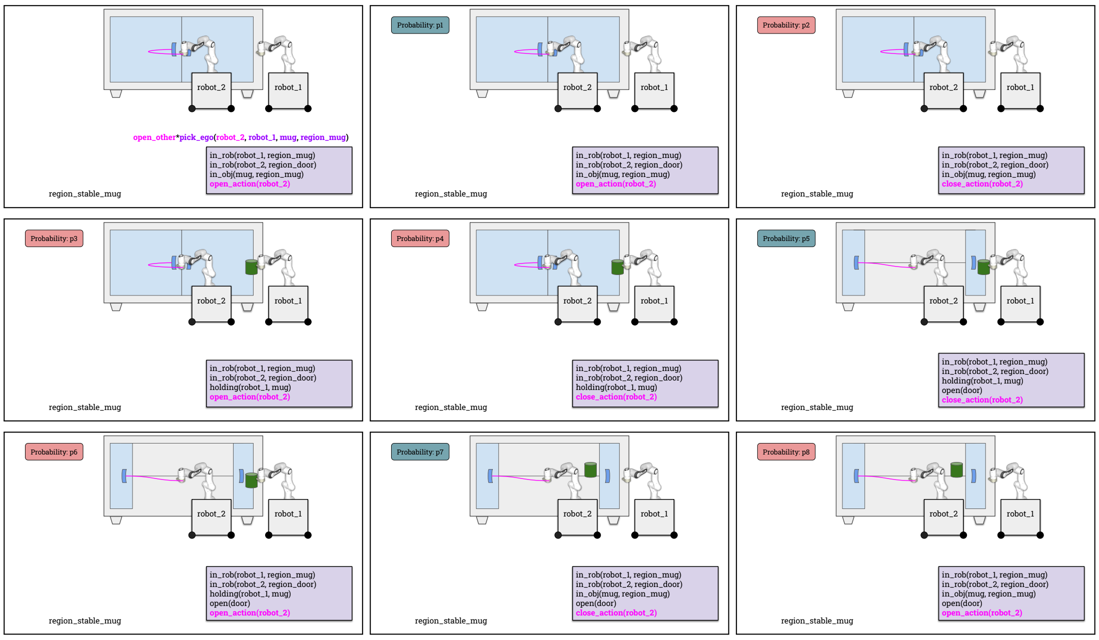


Figure 5.2: Initial state and action (top left) and possible next states. Probability in red means infeasible next state.

Current interactive state of ego Let is_t be the current interactive state, where (i) the other robot (robot_2) is near the door, (ii) the ego agent (robot_1) is near the mug inside a closed cabinet, and (iii) the ego agent observes that robot_2 is opening the door. Thus, s and a_{robot_2} are as follows, such that the interactive state is defined as $is = (s, a_{robot\_2})$:

$$\begin{aligned} s = &\ \texttt{in\_rob(robot\_2, region\_door)}, \texttt{in\_rob(robot\_1, region\_mug)}, \\ &\ \texttt{in\_obj(mug, region\_mug)}, \texttt{not(open(door))} \end{aligned}$$

$$a_{robot\_2} = \texttt{open\_action(robot\_2)}$$

Joint action the ego agent attempts to pick up the mug, making the joint action $a_t = (a_{other}, a_{ego})$:

$$a = \texttt{open\_other*pick\_ego(robot\_2, robot\_1, mug, region\_mug)},$$

Next interactive state Let $is'_{t+1} = (s', a'_{robot\_2})$ be the next interactive state of interest, in which (i) the door is open, (ii) the ego agent is holding the mug, and (iii) the ego agent observes `robot_2` to be closing the door:

$$\begin{aligned} s' = &\ \texttt{in\_rob(robot\_2, region\_door)}, \texttt{in\_rob(robot\_1, region\_mug)}, \\ &\ \texttt{holding(robot\_1, mug)}, \texttt{open(door)} \end{aligned}$$

$$a'_{robot\_2} = \texttt{close\_action(robot\_2)}$$

The transition model specifies the joint outcome probability for this action as the probability: $\mathcal{T}(is_{t+1} \mid is_t, a) = P(is_{t+1} \mid is_t, a) = p_7$, which jointly accounts for current action uncertainty, future action uncertainty and self uncertainty. These may be isolated as illustrated below.

(i) Current action uncertainty The probability that the door will be open in the next state, given that the other agent is observed to be opening the door in the current state. This depends on s and a_{robot_2}, and on a_{robot_1} only if it is a door-related action.

$$\begin{aligned} P(\texttt{open(door)} \in is' \mid is, a) &= \sum_{is' : \texttt{open(door)} \in is'} P(is' \mid is, a) \\ &= p_5 + p_6 + p_7 + p_8 \end{aligned} \tag{5.5}$$

(ii) Future action uncertainty The probability that the next action of the other agent is to close the door, given that the ego agent attempts to pick the mug. This depends on s, a_{robot_1}, and the unobserved outcome of a_{robot_2} at the time of planning: if the other agent failed to open the door, closing it would be infeasible, which is indicated by 0.0 probability for this outcome combination.

$$\begin{aligned} P(\texttt{close\_action(robot\_2)} \in is' \mid is, a) &= \sum_{is' : \texttt{close\_action(robot\_2)} \in is'} P(is' \mid is, a) \\ &= p_2 + p_4 + p_5 + p_7 \end{aligned} \tag{5.6}$$

(iii) Self uncertainty The probability that the ego agent will be holding the mug in the next state, given that it attempted to pick it from the current state. This depends on s, a_{robot_1}, and the unobserved outcome of a_{robot_2}: if the other agent failed to open the door, pick would not succeed, indicated by probability 0.0 for the outcome combination. The other agent only observes ego's action before planning but does not know its outcome yet.

$$\begin{aligned} P(\texttt{holding(robot\_1, mug)} \in is' \mid is, a) &= \sum_{is' : \texttt{holding(robot\_1, mug)} \in is'} P(is' \mid is, a) \\ &= p_3 + p_4 + p_5 + p_6 \end{aligned} \tag{5.7}$$

Given the other agent's most likely action, the ego agent selects its action to maximize the probability of a high-reward outcome from the joint action, while encouraging cooperative behavior from the other agent through its action.

6 Experimental Setup

We evaluate the performance of the proposed framework for two scenarios

S1: **Cleaning scenario** (short-horizon) there is a dex cube (mug) on top of a red plate. The ego agent's goal is to have the dex cube at the same spot, but with the red plate removed. That is, the two robots must collaborate to "clean" the surface under the cube and place the cube again.

S2: **Mug retrieval scenario** (long-horizon) a mug is inside the cabinet with the door closed. The ego agent's goal is to have one of the robot's holding the mug, with the door closed. Two (mobile) manipulators must collaborate to open the door, retrieve the mug and close the door.

6.1. Behavior uncertainty

The following behavior models are explored for the other agent to demonstrate the ability of the framework to plan under interaction uncertainty:

B1: **Random**: a subintentional, no information model, where the other agent is equally likely to be performing any action. The ego agent must be prepared for any contingency during execution and adapt its plan accordingly.

B2: **Human**: a subintentional, fictitious play model where the other agent is somewhat rational, and bases its decisions on the current world state and the ego agent's observed action.

B3: **Inactive**: the other agent is always inactive, meaning the ego agent must perform all the subtasks on its own.

B4: **Symmetric**: the other agent uses the same algorithm as the ego agent and is assumed to behave optimally during execution.

Model learning is deemed successful if the ego agent is able to reach at least one goal state from its current state. For a cooperative other agent, this is easy as the other agent complies with the symbolic plans. For uncooperative (unhelpful or harmful) other agents, the ego agent must learn the true behavior over several samples to overcome optimism.

6.2. Current action uncertainty

The uncertainty surrounding the observed action of the other agent. The observed action is the true action (thus obtains the desired outcome) 90% of the time during model learning.

6.3. Self uncertainty

The ego agent obtains the desired outcome of its chosen action with 90% success during model learning for all actions except "pick", for which the probability of success is learned from the simulator.

7 Cleaning scenario

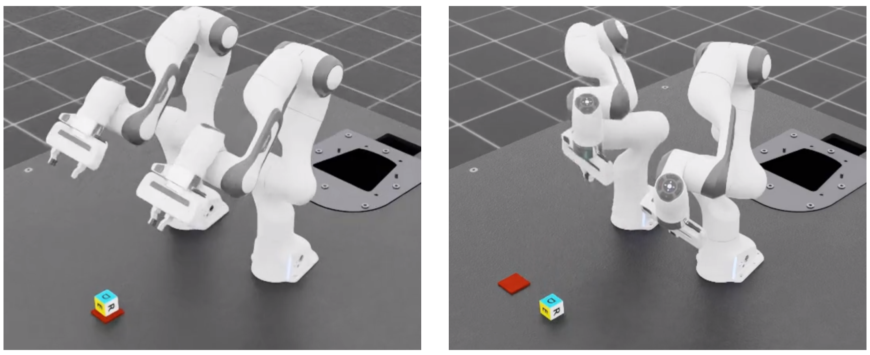
7.1. State space

The state space of the world includes symbolic predicates describing the object's location and the state of two regions. The object may be in one of two regions or held by one of two robots. Each region may be clean or unclean. Thus, the total number of symbolic world states is $4 \times 2 \times 2 = 16$. In addition, the observed action of the other agent may be: pick, place, clean or nothing (with appropriate arguments), which is included in the interactive state. The states are specified using predicates. The green predicates are relevant to the physical state of the world and the blue one indicates the most likely current action of the other agent.

$$\begin{aligned} is_0 = \{&\texttt{in\_obj(mug, region\_mug)},\\ &\texttt{not(clean(region\_mug))},\\ &\texttt{clean(region\_stable\_mug)},\\ &\texttt{nothing\_action(robot\_2)}\}. \end{aligned}$$

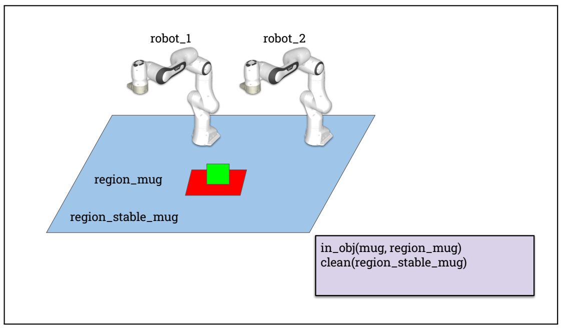
The goal state requires region_mug to be clean, with the mug placed there.

$$\texttt{clean(region\_mug)}, \texttt{in\_obj(mug, region\_mug)} \in is_f.$$

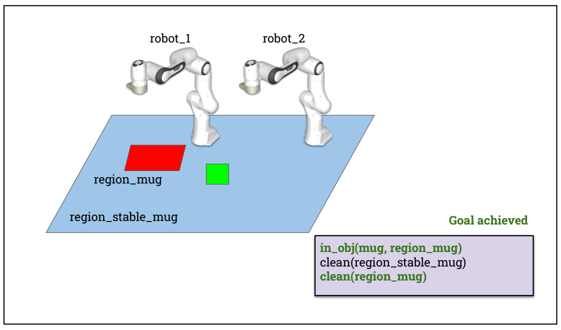


(a) Initial state is_0 **(b)** Goal state is_f

Figure 7.1: Initial and goal states for scenario S1.



(a) Initial state is_0



(b) Goal state is_f

Figure 7.2: Symbolic initial and goal states for scenario S1.

7.2. Action space

Each robot is capable of performing the four actions listed below. The ego agent's attempted action may either succeed and obtain the desired outcome or fail, resulting in no change to the world state. When observing the other agent's action, it may obtain the expected outcome or an alternative outcome associated with confusable actions.

Action	Description	Confusable with
`pick("?rob","?obj","?reg")`	robot "?rob" attempts to pick object "?obj" from region "?reg"	`place`, `nothing`
`place("?rob","?obj","?reg")`	robot "?rob" attempts to place object "?obj" in region "?reg"	`pick`, `nothing`
`clean("?rob","?reg")`	robot "?rob" attempts to clean region "?reg"	`nothing`
`nothing("?rob")`	robot "?rob" does nothing	-

Table 7.1: Actions for each robot and confusable actions of other robot

Joint actions combine the other agent's observed action with the ego agent's chosen action. Prohibited joint actions are those wherein the expected outcome of the other agent's most likely action negates one or more preconditions of ego's action, making it infeasible (see Chapter 5, Section 5.3.2).

Prohibited joint action	Condition
`pick_other*pick_ego("?rob2","?obj2","?reg2","?rob1","?obj1","?reg1")`	"?obj1" = "?obj2"
`pick_other*place_ego("?rob2","?obj2","?reg2","?rob1","?obj1","?reg1")`	"?obj1" = "?obj2"
`place_other*place_ego("?rob2","?obj2","?reg2","?rob1","?obj1","?reg1")`	"?obj1" = "?obj2"
`clean_other*clean_ego("?rob2","?reg2","?rob1","?reg1")`	"?reg1" = "?reg2"
`place_other*clean_ego("?rob2","?obj2","?reg2","?rob1","?reg1")`	"?reg1" = "?reg2"

Table 7.2: Prohibited joint actions

For some actions, the expected outcome of the other agent's most likely action satisfies one or more preconditions of ego's action (see Chapter 5, Section 5.3.2).

Dependent joint action	Condition
`pick_other*clean_ego("?rob2","?obj2","?reg2","?rob1","?reg1")`	"?reg1" = "?reg2"
`place_other*pick_ego("?rob2","?obj2","?reg2","?rob1","?obj1","?reg1")`	"?obj1" = "?obj2"

Table 7.3: Dependent joint actions

7.3. Model learning and execution

During model learning, the ego agent learns a transition model encoding the behavior of the other agent and joint outcome probabilities for the actions.

7.3.1. Random other agent

Model learning. Transition models learned using an other agent that picks any action with equal probability are shown below. The rectangles represent interactive states and ovals represent joint actions. The sum of the numbers on the arrows leaving each action joint action indicate the number of times it was sampled from the interactive state with the blue arrow leading to it. For some interactive states, there are no plans to a goal state (green) passing through them. This is either because the chosen action of the other agent in this interactive state was infeasible from the world state, resulting in symbolic planning failure; or, the deterministic symbolic planner did not generate plans through such interactive states because they were not ”goal-directed”.

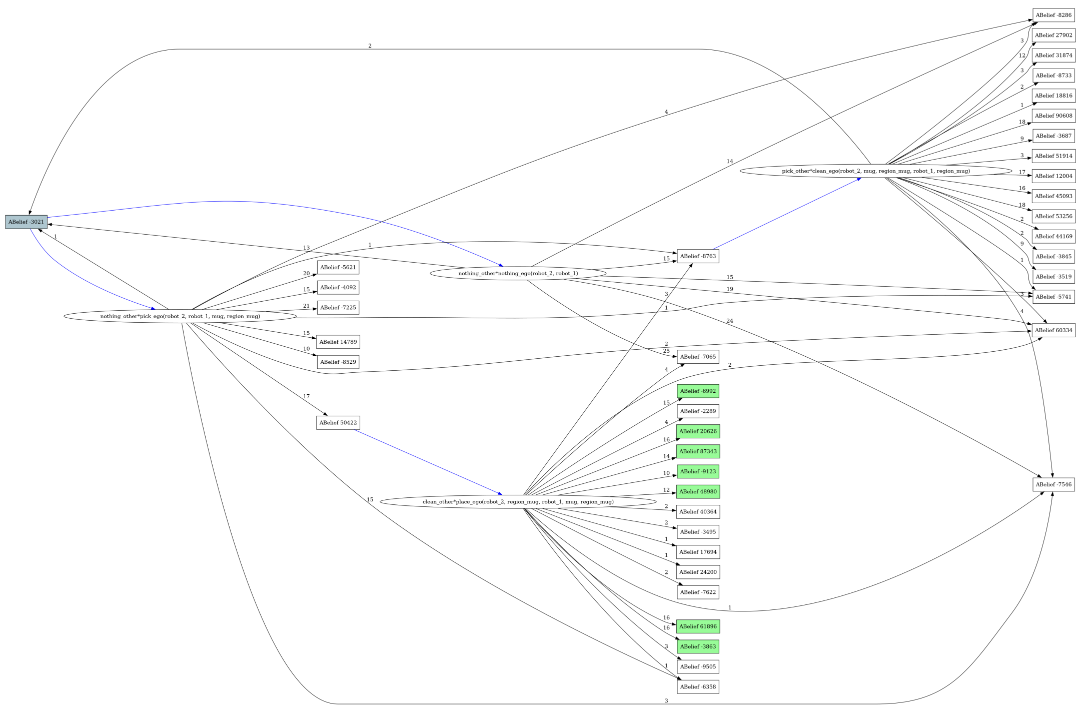


Figure 7.3: Cleaning scenario: transition model for random other agent

Planning begins at the blue interactive state (`ABelief -3021`). Initially, the other agent is known to be idle, doing nothing. The ego agent has two applicable actions: pick the object (mug) from region_mug, or do nothing. On attempting pick, the ego agent may succeed or fail. Moreover, the other agent may pick one of 7 actions at the next step (pick $\times 2$, place $\times 2$, clean $\times 2$, nothing), resulting in 2*7=14 possible outcomes. Depending on the world state after the first joint action, only few of the 6 actions will be applicable. `ABelief 50422` is the interactive state where pick ego succeeded and the other agent appeared to clean region_mug. The clean action will be applicable, and the ego agent attempts place, from which a goal state may be reached if both components of the joint action had the desired outcome. For a joint action, if ego's action has num_{ego} possible outcomes, the other agent's action has num_{other} possible outcomes and $num_{actions}$ denotes the number of actions the other agent can pick at the next state, the total number of possible abstract states as a result of this action is $num_{ego} * num_{other} * num_{actions}$.

Execution. During execution, the other agent behaves randomly and often chooses infeasible actions, resulting in no change to the world state. It is observed that as long as the **other agent takes some helpful action(s)** such that a goal state was reached through such behavior during model-learning, the goal is achieved.

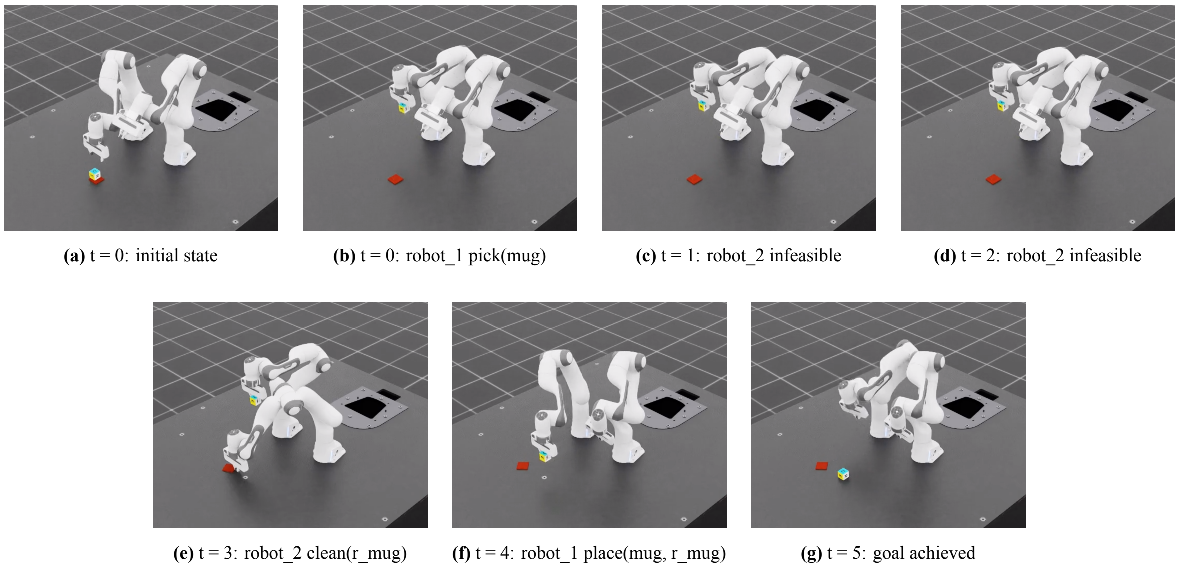

(a) t = 0: initial state **(b)** t = 0: robot_1 pick(mug) **(c)** t = 1: robot_2 infeasible **(d)** t = 2: robot_2 infeasible **(e)** t = 3: robot_2 clean(r_mug) **(f)** t = 4: robot_1 place(mug, r_mug) **(g)** t = 5: goal achieved

Figure 7.4: Cleaning scenario: random other agent demo 1

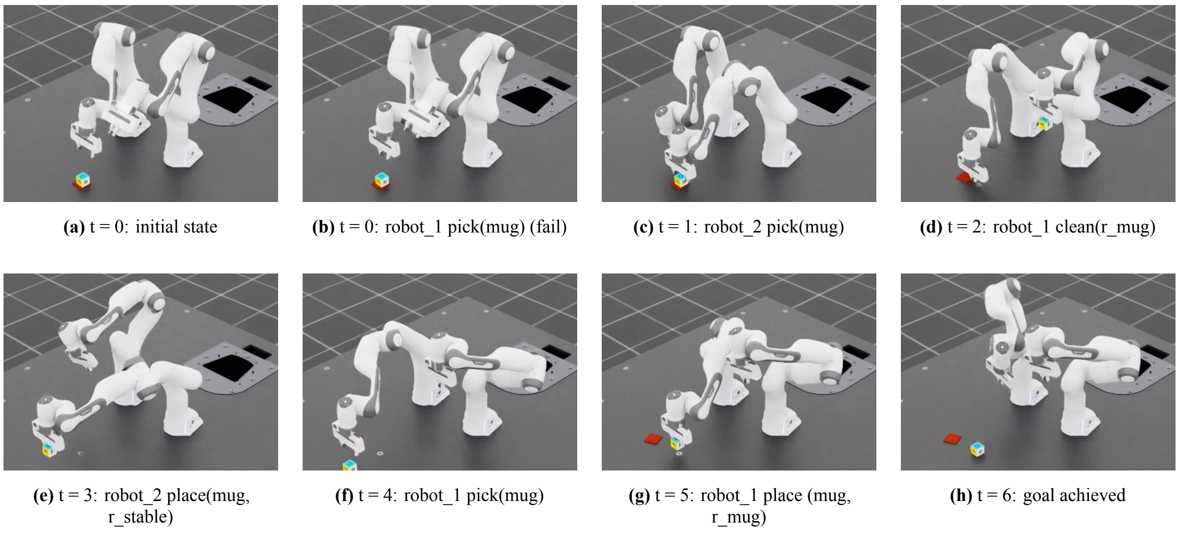

(a) t = 0: initial state **(b)** t = 0: robot_1 pick(mug) (fail) **(c)** t = 1: robot_2 pick(mug) **(d)** t = 2: robot_1 clean(r_mug) **(e)** t = 3: robot_2 place(mug, r_stable) **(f)** t = 4: robot_1 pick(mug) **(g)** t = 5: robot_1 place (mug, r_mug) **(h)** t = 6: goal achieved

Figure 7.5: Cleaning scenario: random other agent demo 2

7.3.2. Human other agent

Model learning. The behavior of the other agent is learned by querying a human user. For a consistent human agent, the transition model is narrow and represents meaningful behavior. For a high number of symbolic plans, the human operator query process is tedious as there are many high entropy transitions, several of them at the early stages of the plan, far from the goal state. These ”earlier” transitions are sampled often and may result in the goal not being encountered during model learning with too few samples. If the human agent is cooperative, fewer plan skeletons are sufficient to encounter the goal during model learning as the agent complies with the optimistic plans.

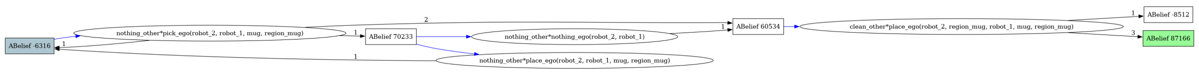
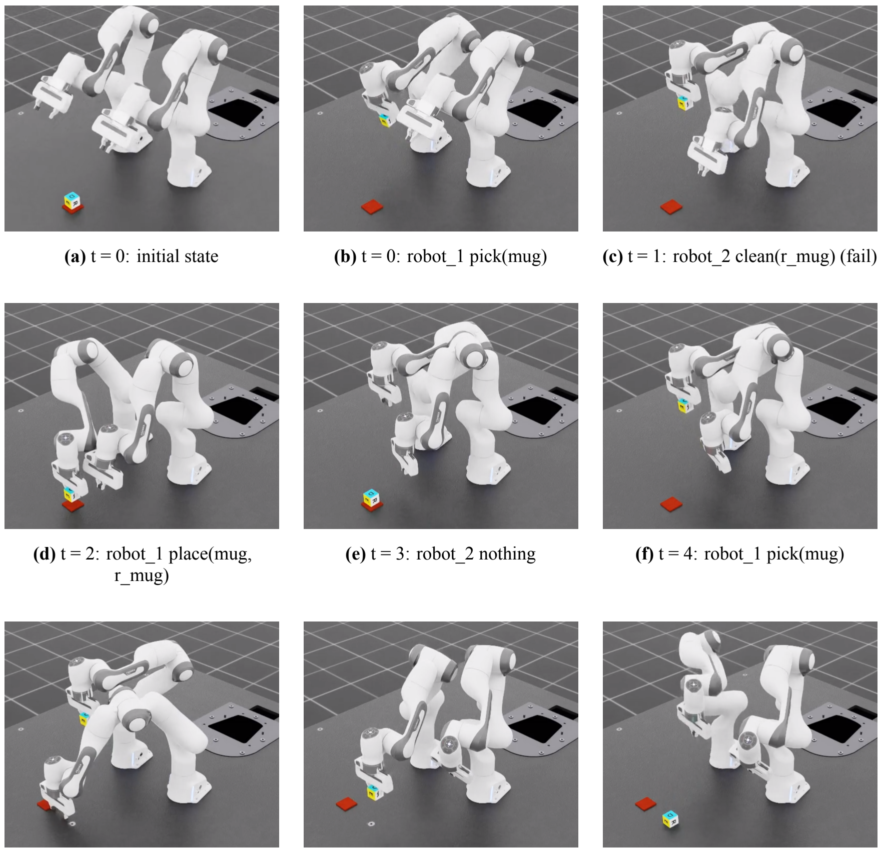

ABelief -6316
nothing_other*pick_ego(robot_2, robot_1, mug, region_mug)
1
1
2
ABelief 70233
nothing_other*nothing_ego(robot_2, robot_1)
nothing_other*place_ego(robot_2, robot_1, mug, region_mug)
1
1
ABelief 60534
clean_other*place_ego(robot_2, region_mug, robot_1, mug, region_mug)
1
3
ABelief -8512
ABelief 87166

Figure 7.6: Cleaning scenario: transition model for human other agent

Execution. An apparently cooperative human agent execution scenario is shown where the operator pretends to perform the cleaning action but without the intended outcome. Since this behavior was encountered during model learning, and a goal state was reached through such behavior, the ego agent is able to **adapt** to it during execution.



(a) t = 0: initial state

(b) t = 0: robot_1 pick(mug)

(c) t = 1: robot_2 clean(r_mug) (fail)

(d) t = 2: robot_1 place(mug, r_mug)

(e) t = 3: robot_2 nothing

(f) t = 4: robot_1 pick(mug)

(g) t = 5: robot_2 clean(r_mug)

(h) t = 6: robot_1 place(mug, r_mug)

(i) t = 7: goal achieved

Figure 7.7: Cleaning scenario: human other agent demo

7.3.3. Inactive other agent

Model learning. The other agent is **consistently inactive**, i.e. it chooses the `nothing` action.

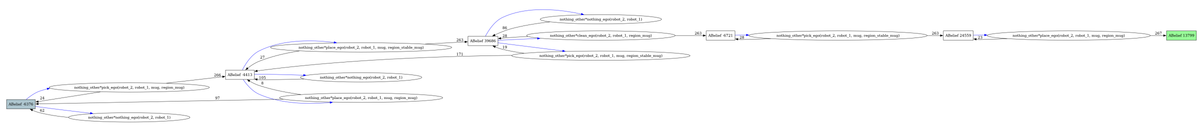
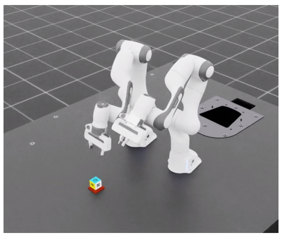
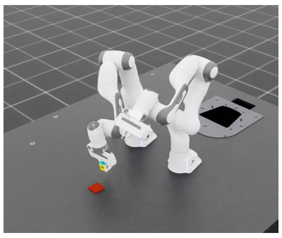
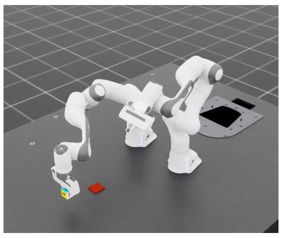

ABelief -6376
nothing_other*pick_ego(robot_2, robot_1, mug, region_mug)
nothing_other*nothing_ego(robot_2, robot_1)
24
62
97
266
ABelief -4413
nothing_other*place_ego(robot_2, robot_1, mug, region_stable_mug)
nothing_other*nothing_ego(robot_2, robot_1)
nothing_other*place_ego(robot_2, robot_1, mug, region_mug)
27
105
8
171
263
ABelief 39686
nothing_other*nothing_ego(robot_2, robot_1)
nothing_other*clean_ego(robot_2, robot_1, region_mug)
nothing_other*pick_ego(robot_2, robot_1, mug, region_stable_mug)
86
28
19
263
ABelief -6721
28
nothing_other*pick_ego(robot_2, robot_1, mug, region_stable_mug)
263
ABelief 24559
23
nothing_other*place_ego(robot_2, robot_1, mug, region_mug)
267
ABelief 13799

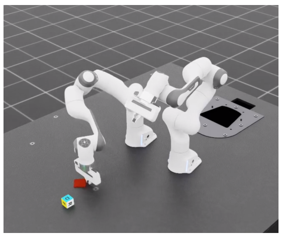
Figure 7.8: Cleaning scenario: transition model for inactive other agent

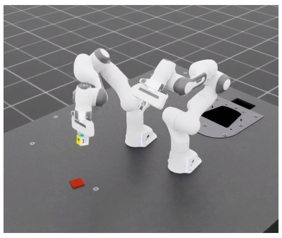
Execution. The ego agent learns that the other agent is inactive and **executes the entire plan on its own**.

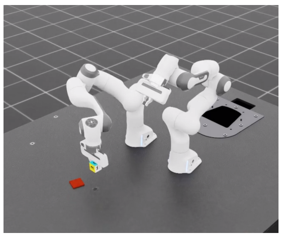

(a) t = 0: initial state


(b) t = 0: robot_1 pick(mug)


(c) t = 1: robot_1 place(mug, r_stable)


(d) t = 2: robot_1 clean(r_mug)


(e) t = 3: robot_1 pick(mug)


(f) t = 4: robot_1 place(mug, r_mug)

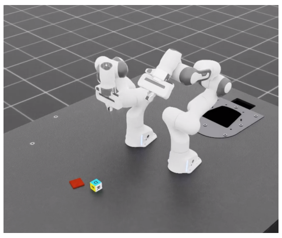
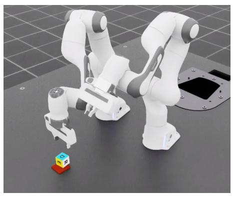

(g) t = 5: goal achieved

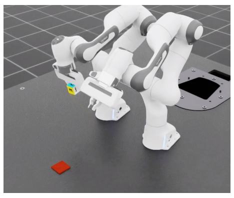
Figure 7.9: Cleaning scenario: inactive other agent demo

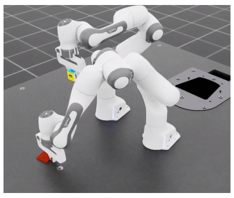
7.3.4. Symmetric other agent

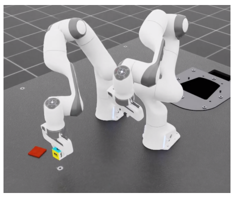
Model learning. We can assume that the other agent behaves in a cooperative way, similar to a centralized planning approach. Alternatively, a random other agent model or a cooperative human agent model may be used: as long as a goal state is encountered during learning through cooperative actions from the other agent, the ego agent is able to successfully adapt to the other's actions during execution. Here, we use a **random other agent for model learning**, as we initially lack the final learned model.

Execution. When both agents use the same learned policy, the agents cooperate to **find and execute the minimum make-span plan consistently**.


(a) t = 0: initial state


(b) t = 0: robot_1 pick(mug)


(c) t = 1: robot_2 clean(r_mug)


(d) t = 2: robot_1 place(mug, r_mug)

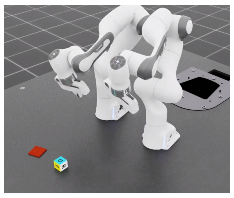

(e) t = 3: goal achieved

Figure 7.10: Cleaning scenario: same algorithm for both agents

8 Mug retrieval scenario

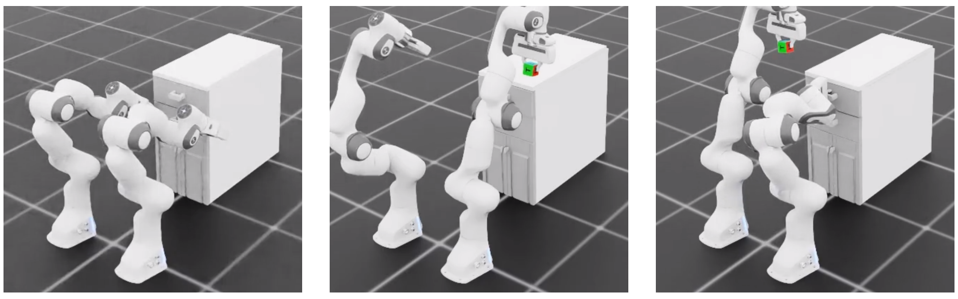
8.1. State space

The state space of the world includes symbolic predicates describing the object's location, the robots' location and the state of door. The object may be in one of two regions or held by one of two robots. Each robot may be in one of three regions and the door may be open or closed. Since a suitable mobile manipulator model is not available on IsaacLab, we use a fixed-base manipulator; the regions are defined by the vertical position of the end-effector. Thus, the total number of symbolic world states is $4 \times 3 \times 3 \times 2 = 72$. In addition, the observed action of the other agent may be: pick, place, open, close, transit, transfer or nothing (with appropriate arguments), which is included in the interactive state.

$$\begin{aligned} is_0 = \{&\texttt{in\_rob(robot\_1, region\_stable\_mug)},\\ &\texttt{in\_rob(robot\_2, region\_stable\_mug)},\\ &\texttt{in\_obj(mug, region\_mug)},\\ &\texttt{not(open(door))},\\ &\texttt{nothing\_action(robot\_2)}\}. \end{aligned}$$

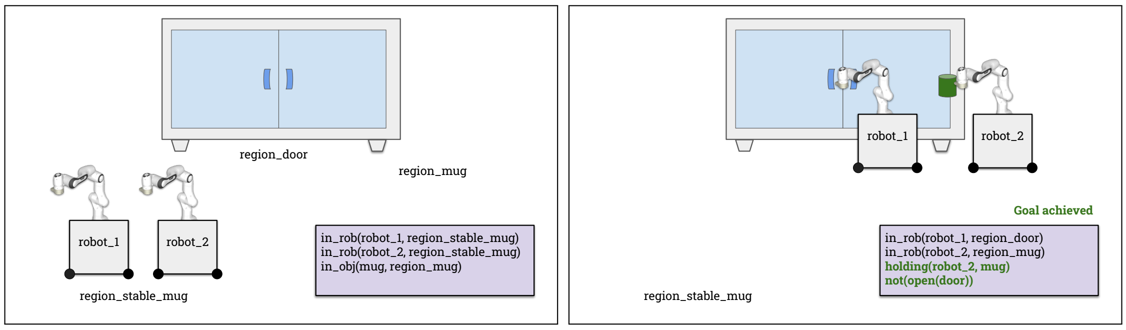
The goal state requires one of the robots to be holding the mug, with the door closed.

$$\begin{aligned} &\texttt{open(door)} \notin is_f,\\ &(\texttt{holding(robot\_1, mug)} \in is_f\\ &\vee\\ &\texttt{holding(robot\_2, mug)} \in is_f) \end{aligned}$$



(a) Initial state is_0 **(b)** Candidate goal state is_f **(c)** Candidate goal state is_f

Figure 8.1: Initial state and candidate goal states for scenario S2.



(a) Initial state is_0 **(b)** Goal state is_f

Figure 8.2: Symbolic initial and goal states for scenario S2. In the goal, either robot could be holding the mug.

8.2. Action space

Each robot is capable of performing the seven actions listed below. The ego agent's attempted action may either succeed and obtain the desired outcome or fail, resulting in no change to the world state. When observing the other agent's action, it may obtain the expected outcome or an alternative outcome associated with confusable actions.

Action	Description	Confusable with
`transit("?rob","?reg1","?reg2")`	robot ”?rob” attempts to move from ”?reg1” to ”?reg2”	`transit`, `nothing`
`transfer("?rob","?reg1","?reg2","?obj")`	robot ”?rob” attempts to transfer ”?obj” from ”?reg1” to ”?reg2”	`transfer`, `nothing`
`pick("?rob","?obj","?reg")`	robot ”?rob” attempts to pick ”?obj” from ”?reg”	`place`, `nothing`
`place("?rob","?obj","?reg")`	robot ”?rob” attempts to place ”?obj” in ”?reg”	`pick`, `nothing`
`open("?rob")`	robot ”?rob” attempts to open door	`close`, `nothing`
`close("?rob")`	robot ”?rob” attempts to close door	`open`, `nothing`
`nothing("?rob")`	robot ”?rob” does nothing	-

Table 8.1: Actions for each robot and confusable actions of other robot

Prohibited joint actions for scenario S2 are listed below (see Chapter 5, Section 5.3.2).

Prohibited joint action	Condition
`pick_other*pick_ego("?rob2","?obj2","?reg2","?rob1","?obj1","?reg1")`	”?obj1” = ”?obj2”
`pick_other*place_ego("?rob2","?obj2","?reg2","?rob1","?obj1","?reg1")`	”?obj1” = ”?obj2”
`pick_other*transfer_ego("?rob2","?obj2","?reg2","?rob1","?reg1","?reg2","?obj1")`	”?obj1” = ”?obj2”
`place_other*place_ego("?rob2","?obj2","?reg2","?rob1","?obj1","?reg1")`	”?obj1” = ”?obj2”
`place_other*transfer_ego("?rob2","?obj2","?reg2","?rob1","?reg1","?reg2","?obj1")`	”?obj1” = ”?obj2”
`open_other*open_ego("?rob2","?rob1")`	-
`close_other*close_ego("?rob2","?rob1")`	-
`close_other*pick_ego("?rob2","?rob1","?obj1","?reg1")`	”?reg1” = region_mug
`close_other*place_ego("?rob2","?rob1","?obj1","?reg1")`	”?reg1” = region_mug

Table 8.2: Prohibited joint actions

Dependent joint actions for scenario S2 are listed below (see Chapter 5, Section 5.3.2).

Dependent joint action	Condition
`open_other*pick_ego("?rob2","?rob1","?obj1","?reg1")`	”?reg1” = region_mug
`open_other*place_ego("?rob2","?rob1","?obj1","?reg1")`	”?reg1” = region_mug
`place_other*pick_ego("?rob2","?obj1","?reg1","?rob1","?obj2","?reg2")`	”?obj1” = ”?obj2”

Table 8.3: Dependent joint actions

8.3. Model learning and execution

8.3.1. Random other agent

Model learning. Transition models learned using an other agent that picks any action with equal probability are shown below.

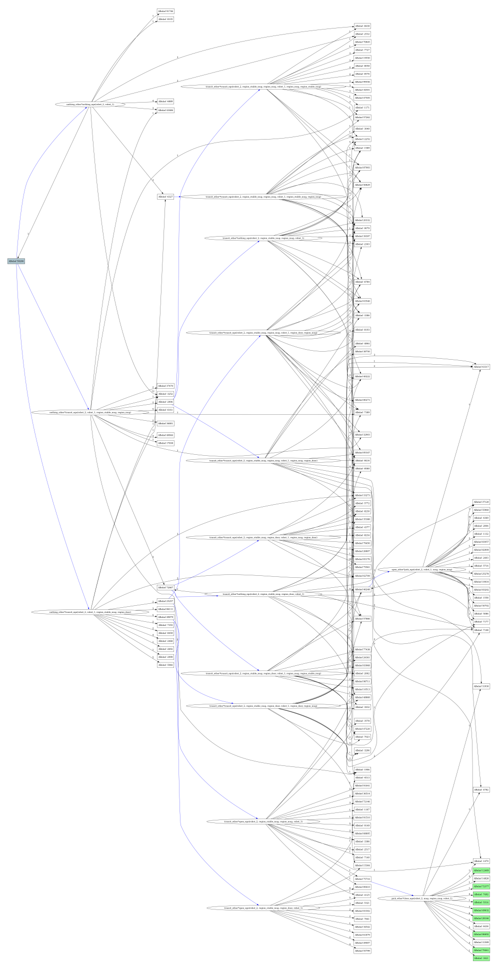


Figure 8.3: Mug retrieval: transition model for random other agent

There are several abstract states (rectangles) from which there are no paths to a goal state (green). These are either infeasible actions from the other agent for which symbolic plans cannot be generated. Or they are states where the other agent's action is unhelpful (not goal-directed) or has undesirable outcomes, and the symbolic planner does not have plans that pass through such states. The other agent's random action often ends up being goal-directed, with goal-directed symbolic plans passing through such behavior-outcome combinations of the other agent, meaning the optimism about the other agent is valid.

Execution. The other agent acts randomly and is equally likely to attempt any action, most of which are infeasible (or unhelpful) for scenario S2. In essence, the **other agent is consistently inactive**, a situation not handled during model learning, since repeatedly sampling a transition occasionally produced feasible, beneficial actions that led to the goal, while infeasible branches were abandoned. In execution, infeasible actions are replaced by `nothing()`, and the ego agent expects the other agent will eventually cooperate. For unhelpful actions, ego adapts its plan, but still expects the other agent will cooperate at some point. However, helpful actions are very unlikely, causing **deadlocks** and preventing the plan from reaching the goal on most occasions ($\geq 90\%$ of the time).

8.3.2. Human other agent

Model learning. The behavior of the other agent is learned by querying a human user. Alternately, a query function may be used to simulate a bias similar to human bias. In the MDP below, the other agent prefers performing door related tasks like transit to the door, open and close when suitable. The ego agent learns this preference by sampling several transitions to overcome the optimism that the other agent will perform the tasks related to retrieving the mug. For a consistent human agent, the transition model is narrow and represents meaningful behavior.

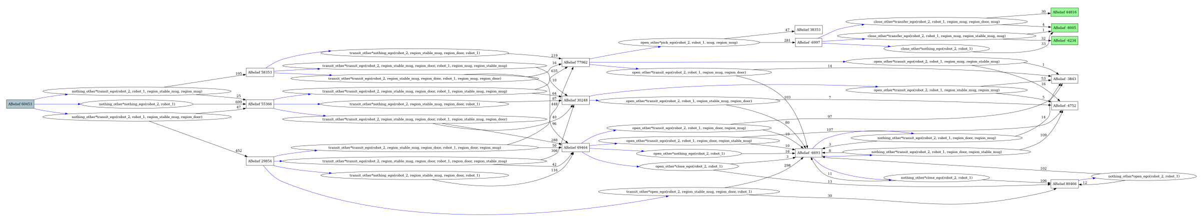
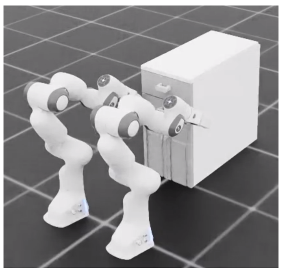
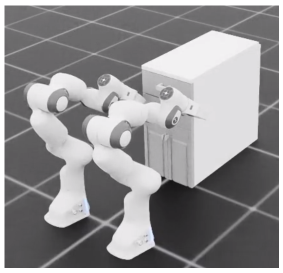


Figure 8.4: Mug retrieval: transition model for biased other agent

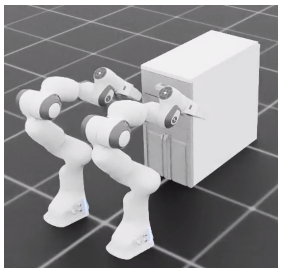
Execution example. The human agent is somewhat **uncooperative** at first, moving to the door along with the ego agent. The ego agent **adapts** its plan to move to the mug and retrieve it.



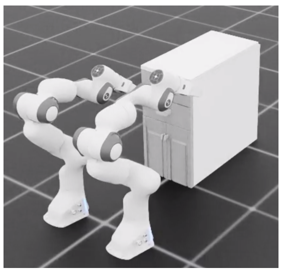
(a) t = 0: initial state



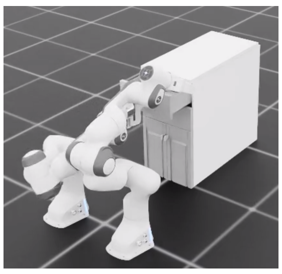
(b) t = 0: robot_1 transit(r_door)



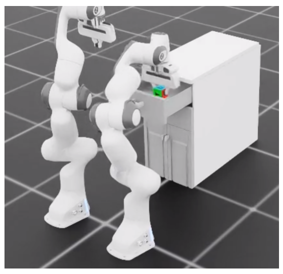
(c) t = 1: robot_2 transit(r_door)



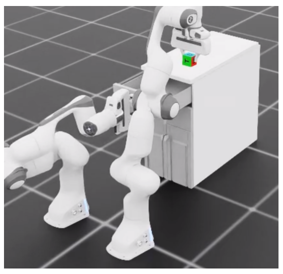
(d) t = 2: robot_1 transit(r_mug)



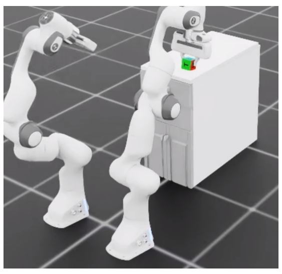
(e) t = 3: robot_2 open door



(f) t = 4: robot_1 pick(mug)



(g) t = 5: robot_2 close door



(h) t = 6: goal achieved

Figure 8.5: Mug retrieval scenario: human other agent demo

8.3.3. Inactive other agent

Model learning. The other agent is consistently inactive, i.e. it chooses the `nothing` action, requiring the ego agent to achieve the goal without any assistance from the other agent.

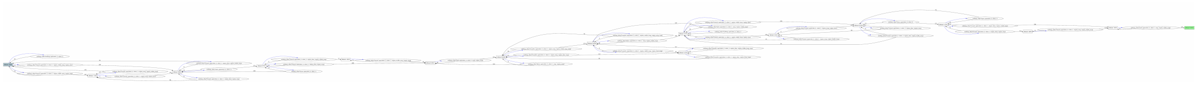
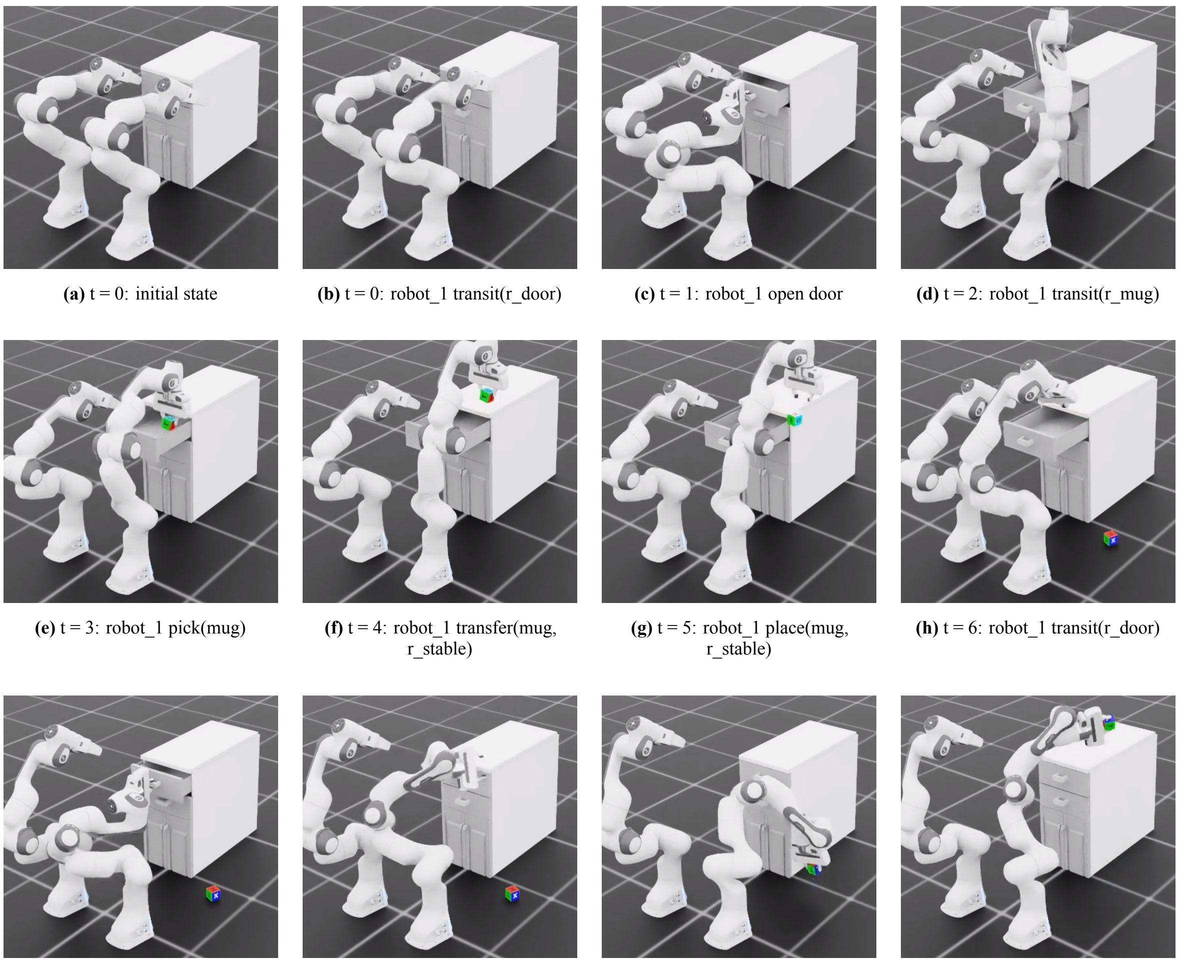


Figure 8.6: Mug retrieval: MDP for inactive other agent

Execution. The ego agent learns that the other agent is inactive and **executes the entire plan on its own**.



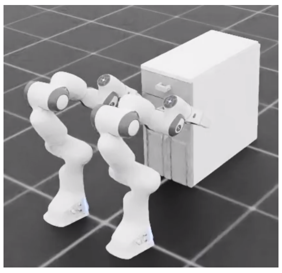
(a) t = 0: initial state
(b) t = 0: robot_1 transit(r_door)
(c) t = 1: robot_1 open door
(d) t = 2: robot_1 transit(r_mug)
(e) t = 3: robot_1 pick(mug)
(f) t = 4: robot_1 transfer(mug, r_stable)
(g) t = 5: robot_1 place(mug, r_stable)
(h) t = 6: robot_1 transit(r_door)
(i) t = 7: robot_1 close door
(j) t = 8: robot_1 transit(r_stable)
(k) t = 9: robot_1 pick(mug)
(l) t = 10: goal achieved

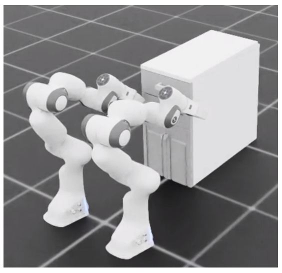
Figure 8.7: Mug retrieval scenario: inactive other agent demo

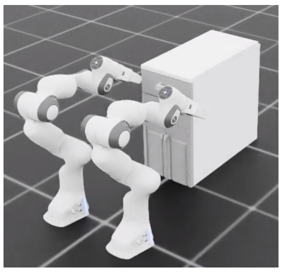
8.3.4. Symmetric other agent

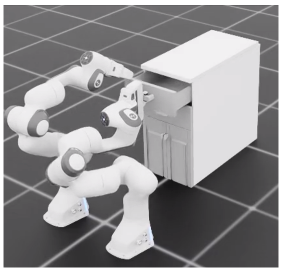
Model learning. We can assume that the other agent behaves in a cooperative way. Alternatively, a random other agent model or a cooperative human agent model may be used: as long as a goal state is encountered during learning through cooperative actions from the other agent, the ego agent is able to adapt successfully during execution. Here, we use a **random other agent for model learning** for the same reasons as in the previous scenario.

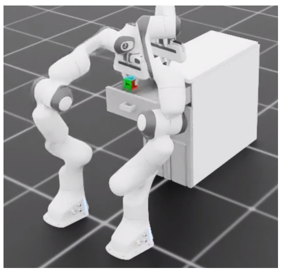
Execution. When both agents use the learned policy, the agents cooperate to **find and execute the minimum make-span plan consistently**.

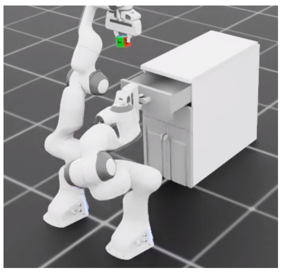

(a) t = 0: initial state


(b) t = 0: robot_1 transit(r_door)


(c) t = 1: robot_2 transit(r_mug)


(d) t = 2: robot_1 open door


(e) t = 3: robot_2 pick(mug)


(f) t = 4: robot_1 close door

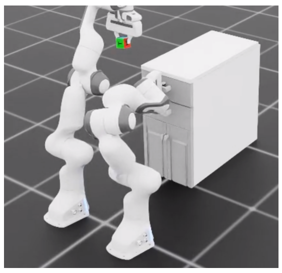

(g) t = 5: goal achieved

Figure 8.8: Mug retrieval scenario: same algorithm for both agents

9 Results

This chapter evaluates the proposed decentralized approach against a **sequential, non-deterministic centralized planner benchmark**, where two agents take turns executing actions. While decentralized multi-agent TAMP under interaction uncertainty is an emerging area of research, existing methods in the literature are limited and not readily adaptable for direct comparison. The centralized sequential planner serves as a meaningful baseline: it has a global view of the system and addresses only one layer of uncertainty at a time, i.e., the uncertainty in the current agent's action outcome. This contrasts with the more complex uncertainty modeling required in joint centralized planning (which must reason over joint action-outcome uncertainty), and the decentralized setting, which has an additional layer of uncertainty due to the behavior of the other agent. We compare the quality of solution and computational performance of the decentralized approach with its centralized, sequential counterpart.

9.1. Baseline

We compare the centralized baseline and the decentralized implementation based on the quality of the solution, quantified as the **average number of steps to reach the goal state during execution**. While the total cost of the plan would be a more accurate metric, we use the number of steps as a proxy, since low-level motion planning is not the focus of this thesis. The ”behavior” of the other agent in the decentralized scenarios is equivalent to the ”skill” of the other agent for the non-deterministic centralized scenario, i.e., each agent's relative ability to perform some tasks better. The ego agent's decentralized planner must learn about the behavior of the other agent through interaction, while the centralized planner learns the (in)ability of an agent to perform some actions through simulations. More specifically, we compare the performance of decentralized planning (ours) against a **centralized planning baseline**. In centralized planning, each action of an agent has more than one possible outcome and the agents take turns to act (sequential), meaning that the planner reasons about one level of uncertainty at a time. The ”skill” of an agent performing some task is quantified through the probability of success when the agent performs the task, and is estimated during model learning. During execution, the planner selects optimal actions for each agent, given their skill set.

9.2. Problem complexity

Within a fixed problem scenario (S1 or S2), the complexity or difficulty is determined by:

- **Behavior model of other agent**. As the other agent's compliance with the desired behavior decreases, or the other agent becomes increasingly uncooperative, the complexity increases. A higher number of transitions must be sampled by the ego agent to overcome its optimism about the other agent's behavior.
- **Goal-directed infeasible action outcomes**. Some joint actions have outcome combinations that are infeasible but helpful in achieving the goal, such as `pick_other*clean_ego` (S1) where the other agent fails to pick the object but ego succeeds in cleaning the surface under it and `open_other*pick_ego` (S2) where the other agent fails to open the door but ego succeeds in picking the mug from the cabinet. A higher number of transitions must be sampled by the ego agent to learn this infeasibility.
- **Minimum horizon length solution**. The minimum number of actions to the goal state, given the behavior of the other agent. This depends on the initial state, the goal state and the behavior model of the other agent. As the minimum horizon length increases, the number of ways to reach the goal increases, increasing the difficulty of the problem.

Scenario S1: we consider three levels of problem complexity (difficulty) for scenario S1, increasing from D1 to D3. robot_1 is the ego agent and robot_2 is the other agent. The cost of the path to the goal is the number of actions to the goal state, including the `nothing` action.

D1: robot_2 behaves optimally. The lowest cost path to the goal is: (i) robot_1 picks the mug from region_mug, (ii) robot_2 cleans region_mug, (iii) robot_1 places mug in region_mug.

D2: robot_2 only performs pick and place (100% success) but not clean. The lowest cost path to the goal is: (i) robot_1 does nothing, (ii) robot_2 picks the mug from region_mug, (iii) robot_1 cleans region_mug, (iv) robot_2 places mug in region_mug.

D3: robot_2 is inactive. The lowest cost path to the goal is: (i) robot_1 picks the mug from region_mug nothing, (ii) robot_1 (temporarily) places the mug in region_stable_mug, (iii) robot_1 cleans region_mug, (iv) robot_1 picks the mug from region_stable_mug, (v) robot_1 places mug in region_mug.

Difficulty	**Initial state**	**Behavior model of robot_2**	**Minimum horizon length**
D1	$is_0 = \{$`clean(robot_1, region_stable_mug)`, `in_obj(mug, region_mug)`, `nothing_action(robot_2)`$\}$.	optimal	3
D2	$is_0 = \{$`clean(robot_1, region_stable_mug)`, `in_obj(mug, region_mug)`, `nothing_action(robot_2)`$\}$.	only performs pick and place; does not perform clean	4
D3	$is_0 = \{$`clean(robot_1, region_stable_mug)`, `in_obj(mug, region_mug)`, `nothing_action(robot_2)`$\}$.	inactive	5

Table 9.1: Difficulty as a function of initial state, other agent behavior, and horizon length

Analysis the execution performance is evaluated over 20 trials on basis of the mean number of steps to the goal and the variance. robot_1 is the ego agent, robot_2 is the other agent. The training and execution distribution for decentralized planning is tabulated below. For centralized planning, the training and execution behavior is aligned with the execution behavior for its decentralized counterpart in each setting.

Difficulty	Training	Execution
D1	robot_1: 90% success for chosen action; robot_2: randomly chooses an action, 90% probability of apparent action being the true action with desired outcome	same algorithm for robot_1 and robot_2; robot_1: 90% success for chosen action; robot_2: 90% probability of apparent action being the true action with desired outcome
D2	robot_1: 90% success for chosen action; robot_2: randomly chooses an action, 100% probability of apparent action being the true action with desired outcome for pick and place, 0% for clean	robot_1: 90% success for chosen action; robot_2: operated by human user, behavior aligned with training behavior
D3	robot_1: 90% success for chosen action; robot_2: inactive, always chooses nothing action	robot_1: 90% success for chosen action; robot_2: inactive, always chooses nothing action

Table 9.2: Training and execution behavior for the two agents for decentralized planning

We observe the following Figure 9.1:

D1: The mean for both centralized and decentralized settings is substantially higher than the minimum number of steps. Each robot has a 90% chance of succeeding at its task, meaning that the probability of a 3-step path to the goal is $(0.9)^3 = 0.729$. Moreover, if robot_2 fails to clean, the centralized planner requires robot_1 to wait for robot_2 to retry and succeed. The decentralized planner of robot_1 instructs it to place the mug, realizing at the next step that clean failed, and it has to pick up the mug again, resulting in a higher variance for the decentralized planner.

D2: There is no uncertainty surrounding robot_2 and the performance is comparable for the centralized and robot_1 using a decentralized planner, as expected. The mean is close to the minimum number of steps as the probability of a 4-step path to the goal is 0.9 and there is only one level of uncertainty surrounding robot_1's actions.

D3: There is no uncertainty surrounding robot_2 and the performance is comparable for the centralized and decentralized planners, as expected. Even though the probability of a 5-step path to the goal is $(0.9)^5 = 0.590$, there is only one level of uncertainty surrounding robot_1's actions and it does not have to wait for robot_2 to retry actions, meaning that the mean is close to the minimum of 5 for both scenarios.

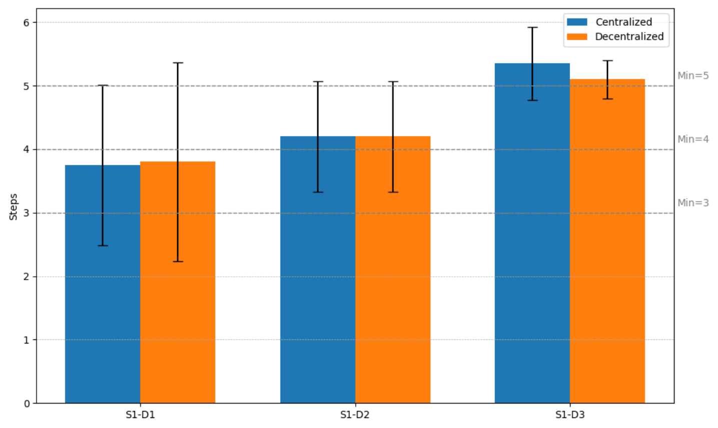


Figure 9.1: Centralized vs. decentralized planner for S1: number of steps to goal during execution

Scenario S2: we consider four levels of problem complexity (difficulty) for scenario S2, increasing from D1 to D4. robot_1 is the ego agent and robot_2 is the other agent. The cost of the path to the goal is the number of actions to the goal state, including the `nothing` action.

- D1: robot_2 behaves optimally. The lowest cost path to the goal is: (i) robot_1 opens the door, (ii) robot_2 picks the mug from region_mug, (iii) robot_1 closes the door.
- D2: robot_2 behaves optimally. The lowest cost path to the goal is: (i) robot_1 transits to region_door, (ii) robot_2 transits to region_mug, (iii) robot_1 opens the door, (iv) robot_2 picks the mug from region_mug, (v) robot_1 closes the door.
- D3: robot_2 is only performs door-related tasks like transit to region_door, opening and closing it, but never performs mug-related tasks like transit to region_mug, pick, place, transfer. The lowest cost path to the goal is: (i) robot_1 transits to region_mug, (ii) robot_2 transits to region_door, (iii) robot_1 does nothing, (iv) robot_2 opens the door, (v) robot_1 picks up the mug, (vi) robot_2 closes the door.
- D4: robot_2 is inactive. The lowest cost path to the goal is: (i) robot_1 transits to region_door, (ii) robot_1 opens the door, (iii) robot_1 transits to region_mug, (iv) robot_1 picks the mug from region_mug, (v) robot_1 transfers the mug to region_stable_mug, (vi) robot_1 (temporarily) places the mug in region_stable_mug, (vii) robot_1 transits to region_door, (viii) robot_1 closes the door, (ix) robot_1 transits to region_stable_mug, (x) robot_1 picks the mug from region_stable_mug.

Difficulty	Initial state	Behavior model	Minimum horizon length
D1	$is_0 = \{$`in_rob(robot_1, region_door)`, `in_rob(robot_2, region_mug)`, `in_obj(mug, region_mug)`, `not(open(door))`, `nothing_action(robot_2)`$\}$.	optimal	3
D2	$is_0 = \{$`in_rob(robot_1, region_stable_mug)`, `in_rob(robot_2, region_stable_mug)`, `in_obj(mug, region_mug)`, `not(open(door))`, `nothing_action(robot_2)`$\}$.	optimal	5
D3	$is_0 = \{$`in_rob(robot_1, region_stable_mug)`, `in_rob(robot_2, region_stable_mug)`, `in_obj(mug, region_mug)`, `not(open(door))`, `nothing_action(robot_2)`$\}$.	only performs door-related tasks: transit(door), open, close	6
D4	$is_0 = \{$`in_rob(robot_1, region_stable_mug)`, `in_rob(robot_2, region_stable_mug)`, `in_obj(mug, region_mug)`, `not(open(door))`, `nothing_action(robot_2)`$\}$.	inactive	10

Table 9.3: Difficulty as a function of initial state, other agent behavior, and horizon length

Analysis the execution performance is evaluated over 10 trials on basis of the mean number of steps to the goal and the variance. The training and execution distribution for decentralized planning is tabulated below. For centralized planning, the training and execution behavior is aligned with the execution behavior for its decentralized counterpart in each setting.

Difficulty	Training	Execution
D1	robot_1: 90% success for chosen action; robot_2: randomly chooses an action, 90% probability of apparent action being the true action with desired outcome	same algorithm for robot_1 and robot_2; robot_1: 90% success for chosen action; robot_2: 90% probability of apparent action being the true action with desired outcome
D2	robot_1: 90% success for chosen action; robot_2: randomly chooses an action, 90% probability of apparent action being the true action with desired outcome	robot_1: 90% success for chosen action; robot_2: operated by human user, 100% probability of apparent action being the true action with desired outcome
D3	robot_1: 90% success for chosen action; robot_2: only chooses transit to door, open or close actions, conditioned on state of the world and ego's attempted action, 100% probability of apparent action being the true action with desired outcome if applicable	robot_1: 90% success for chosen action; robot_2: operated by human user, 100% probability of apparent action being the true action with desired outcome
D4	robot_1: 90% success for chosen action; robot_2: inactive, always chooses nothing action	robot_1: 90% success for chosen action; robot_2: inactive, always chooses nothing action

Table 9.4: Training and execution behavior for the two agents for decentralized planning

We observe the following Figure 9.2:

- D1: as for scenario S1, the mean is higher than the minimum horizon length with a high variance for the decentralized case to deal with instances where robot_1 fails to open or close the door (plan length=5) or robot_1 closes the door when pick has failed, requiring robot_1 open, robot_2 pick, robot_1 close to be repeated (plan length=9).
- D2: similar to S2-D1, but robot_2 is deterministic, meaning that the mean for the decentralized case is closer to the minimum horizon length compared to S2-D1 and variance is lower.
- D3: robot_2 often prematurely closes the door on observing robot_1 performing pick, requiring robot_1 nothing, robot_2 open, robot_1 pick, robot_2 close to be repeated (length=10 or 14). This results in a higher variance (possible plan lengths: 6, 10, 12, 14, etc.) and the mean value being much larger than the minimum for decentralized planning. The centralized sequential planner is able to access action outcomes immediately to make informed decisions, resulting in lower variance and the mean plan length being close to the minimum.
- D4: robot_2 is always inactive. There is no uncertainty surrounding robot_2 and the performance is comparable for the centralized and robot_1 using a decentralized planner, as expected.

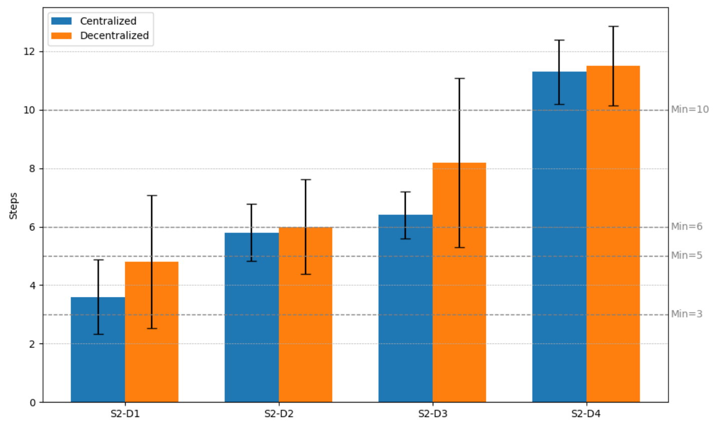
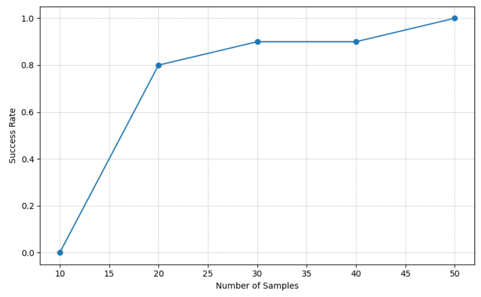


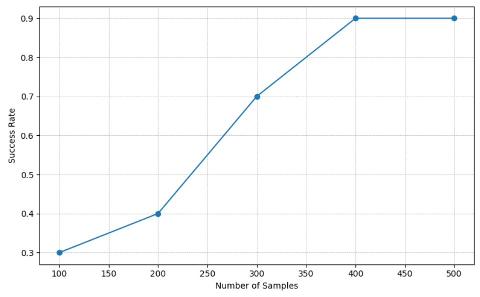
Figure 9.2: Centralized vs. decentralized planner for S2: number of steps to goal during execution

9.3. Effects of hyperparameters on learning

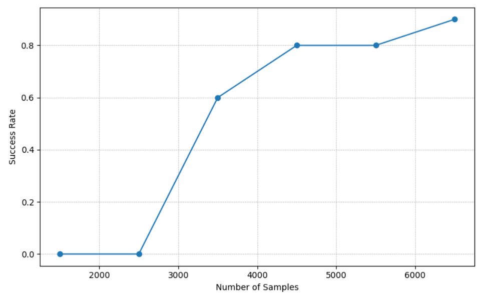
For each setting (D1-D4) for **scenario S2**, with the number of plan skeletons fixed as above, we study the **rate of model learning success as we vary the number of samples** for 10 . A successful learning implies a policy that generates plans which achive the goal 90% of the time. For D1 and D2, `batch_size` is the same as `num_samples`, but for D3 and D4, `batch_size=100` to have a higher number of total plan skeletons ($\frac{\texttt{num\_samples}}{\texttt{batch\_size}} * \texttt{num\_skeletons}$), and therefore transitions, to overcome the optimism surrounding the behavior of the other agent. Optimal behavior is expected for `num_samples` $>>$ `num_transitions_total`.



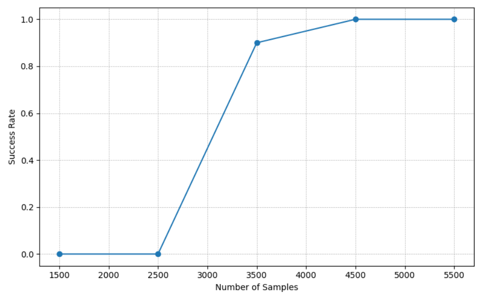
(a) S2-D1



(b) S2-D2



(c) S2-D3



(d) S2-D4

Figure 9.3: Success rate as a function of number of samples for different scenarios.

9.4. Deadlocks and suboptimal behavior

The centralized planner has a global view of all agents in the scene: its sequential nature ensures that each agent's action outcome is observed immediately after execution, and does not suffer from deadlocks. Using joint actions presents the risk of infeasible actions, but it is expected that such cases will be resolved with ease. In the decentralized case, ego observes the other agent's action outcome after its execution, meaning that there is a higher risk of infeasible actions or premature goal-directed actions like closing the door when the other agent is observed to be performing pick, and pick fails (scenario S2). Infeasible actions, suboptimal, unhelpful behavior from the other agent results in deadlocks where the plan is unable to converge to the goal.

When the other agent's execution behavior is compliant with model learning behavior with sufficient probability, such that a goal state was reached through such behavior during model learning, the ego agent can solve for an optimal action, given the observed action of the other agent. However, if the other agent exhibits out-of-distribution behavior during execution, the ego agent may not be able to adapt to it, resorting to inactivity or suboptimal behavior. Further, since the actions are not informed with their true costs, the planner only considers the number of actions to the goal state, which often results in suboptimal behavior from the ego agent.

9.5. Computational performance for learning

Within a scenario, as the planning horizon increases, the number of possible action sequences to reach the goal increases, requiring more plan skeletons to cover these possibilities. Consequently, a higher number of samples is needed to adequately explore the increased number of transitions. As the difficulty increases due to unhelpful behaviors from the other agent or infeasible action outcomes, more samples are needed to counteract overly optimistic assumptions about such transitions. For the decentralized setting, the shorter-horizon cleaning scenario (S1) is less complex than the longer-horizon mug retrieval scenario (S2). This is seen in the time required for successful model learning across 5 trials for each difficulty level in each scenario.

Comparison with sequential centralized planning: The hyperparameter values that guarantee successful model learning are compared for the **sequential centralized** and the **decentralized** case. The model learning time increases with an increase in the number of samples, the number of plan skeletons and number of batches. Solving the MDP is quick and takes 0.15 ± 0.043s for all cases.

Sequential centralized planner: Hyperparameters for $\geq 90\%$ model learning success rate for different skills:

Hyperparameter	D1	D2	D3	D1	D2	D3	D4
`num_skeletons`	10	10	10	10	10	10	10
`num_samples`	50	50	200	50	50	100	200
`batch_size`	50	50	50	50	50	100	50

Table 9.5: Centralized model learning hyperparameters. Clear blue is S1, dark blue is S2.

Decentralized planner: Reasons about three levels of uncertainty, requiring more transitions to be sampled to learn a close approximation of the transition model. The values of hyperparameters for $\geq 90\%$ model learning success rate, and thus learning time are highly sensitive to the problem complexity.

Hyperparameter	D1	D2	D3	D1	D2	D3	D4
`num_skeletons`	10	10	10	10	100	100	100
`num_samples`	50	500	2000	50	500	6500	5000
`batch_size`	50	100	100	50	500	100	100

Table 9.6: Decentralized model learning hyperparameters. Yellow is S1, orange is S2.

The time required to successfully learn the model is an estimate of the difficulty of the problem, as shown in Figure 9.4. The problem difficulty is much higher for decentralized planning compared to its centralized counterpart, due to uncooperative behavior from the other agent and desirable but infeasible outcome combinations.

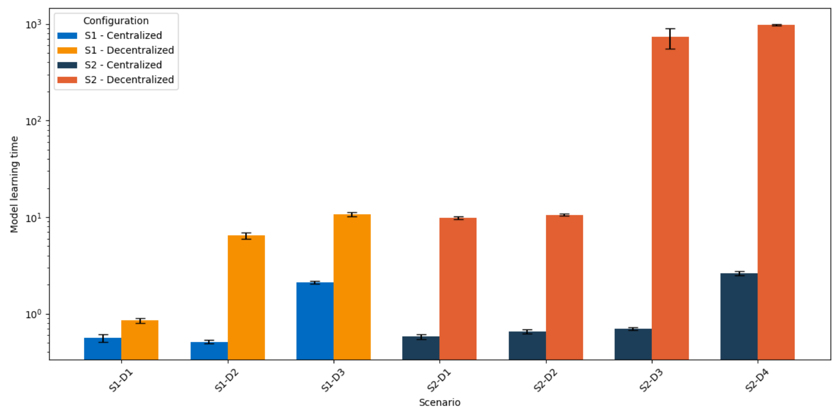


Figure 9.4: Model-learning time across different difficulties for scenario S1 and S2

10 Conclusion

10.1. Discussion

This thesis proposes an approach to handling interaction uncertainty for multi-agent task and motion planning, where the protagonist (ego) agent must coordinate with another agent to achieve a shared goal without a prior model of the other agent's behavior or the dynamics of the environment, by learning these as an interactive Markov Decision Process (I-MDP). Unlike intentional models that require complex reasoning, short-term intent can be directly inferred through action recognition, allowing the ego agent to optimally select actions based on the probability of different outcomes while also promoting cooperative behavior. The ego agent is initially optimistic about the other agent's cooperation, meaning it is able to learn simple, compliant behaviors with ease. It is also able to learn more complex, longer term behaviors such as preferences for subtasks or inactivity over several samples. Experimental results show that a random other agent behavior model prepares the ego agent to adapt to a wide range of execution behavior including compliance, apparent compliance or even redundant, unhelpful behaviors, as long as the other agent is cooperative eventually. Learning complex behaviors requires extensive interaction to overcome the initial optimism about the other agent's compliance: in some cases, different behaviors from the other agent can yield desirable outcomes, leading to a significant increase in computational expense. However, once the model stabilizes, solving the MDP is efficient. During execution, the resulting plan lengths are close to those produced by a centralized sequential centralized planner and to the theoretical minimum makespan. This confirms that the proposed approach can combine the autonomy of decentralized control with performance approaching that of centralized planning.

Comparison with MDPs and game theory. The I-MDP formulation uses an explicit modeling of the other agent through partial observability to predict its behavior, as opposed to the MDPs, which treat the other agent as environmental noise. Each agent is autonomous and self-interested, locally computing their optimal actions using a decision-theoretic framework based on their beliefs while interacting with others. This is in contrast with classical game theoretic approaches that focus more on stability (equilibrium) and describe agent's optimal actions only if, and when, an equilibrium has been reached (incompleteness). In contrast, using an I-MDP, each agent computes its own optimal action by maximizing expected reward while accounting for the uncertainty about the environment and the other agent. Unlike many decision-theoretic formulations, game-theoretic approaches do not require assuming stationary behavior of the other agent, which can be advantageous scenarios where the other agent is adversarial or adaptive. However, I-MDP the approach draws inspiration from the subjective approach to probability in games by predicting the actions of the other agents and using this to inform the choice of its own actions [17, 12].

10.2. Limitations and future work

While the proposed approach demonstrates promising results, it is not without limitations. These shortcomings highlight important directions for future research in improving the robustness and applicability of the method. Addressing them could significantly enhance its effectiveness in real-world multi-agent planning and execution scenarios.

Infeasible outcome combinations: Certain joint actions present the symbolic planner with outcome combinations that are desirable in theory but are infeasible in practice, thereby increasing the computational effort needed to overcome the planner's initial optimism. A more sophisticated planning framework could mitigate this by explicitly prohibiting such outcome combinations, ensuring they are excluded from consideration during symbolic planning.

Perception system and sequential execution: At present, the system does not use a perception module: instead of learning the other agent's behavior through observation and interaction, a query function is used to acquire its behavior. As a result, the other agent explicitly communicates its chosen action to the ego agent, and vice versa, resulting in sequential execution. This contrasts with the goal of enabling parallel or staggered execution, but the approach remains applicable in such settings, as the ego agent does not observe the other agent's action outcomes prematurely, and its decision is only informed by its observed action. Using a working action recognition module could make the interaction more realistic from the perspective of both agents and enable staggered execution [19].

Subintentional behavior model for the other agent: The ego agent assumes a subintentional fictious play behavior model for the other agent, reconstructing its policy through interaction. This reduces the applicability of the approach in scenarios with intentional or adaptive agents. For an intentional other agent, finitely nested recursive reasoning using a POMDP may yield better results. However, this would not account for the possibility of the other agent's observed action not obtaining the expected outcome.

Action execution costs: Since motion planning is abstracted away, actions are not informed by their true execution costs. The action-outcome pairs are only weighted by their risk, meaning the ego agent cannot distinguish between actions on the basis of the cost of their execution, which leads to suboptimal behavior when different action sequences have the same length to the goal state. This may be addressed by using an α-cost likelihood determinized stochastic shortest path problem (α-CLDSSPP) for symbolic planning that leverages information about the risk as well as execution costs of different actions using motion planners to generate joint-space trajectories for the robots to achieve high level tasks like pick or transit [25].

Extension to more agents: The proposed framework may be extended to deal with more agents by selecting joint actions such that each agent performs the most likely observed action, the outcomes being verified at the next step. This setting must additionally consider the effect of the agent's actions on each other, in addition to ego's influence on them. More levels of uncertainty must be considered, making model learning more challenging for the ego agent. However, having learned the model, the ego agent is expected to behave optimally as long as the other agents' execution behavior is aligned with training behavior.

References

[1] Aliakbar Akbari, Mohammed Diab, and Jan Rosell. “Contingent Task and Motion Planning under Uncertainty for Human–Robot Interactions”. In: *Applied Sciences* 10.5 (Jan. 2020). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 1665. ISSN: 2076-3417. DOI: 10.3390/app10051665. URL: https://www.mdpi.com/2076-3417/10/5/1665 (visited on 12/26/2024).

[2] Aliakbar Akbari, Muhayyuddin, and Jan Rosell. “Knowledge-oriented task and motion planning for multiple mobile robots”. In: *Journal of Experimental & Theoretical Artificial Intelligence* 31.1 (Jan. 2, 2019), pp. 137–162. ISSN: 0952-813X, 1362-3079. DOI: 10.1080/0952813X.2018.1544280. URL: https://www.tandfonline.com/doi/full/10.1080/0952813X.2018.1544280 (visited on 02/14/2025).

[3] Stefano V. Albrecht and Peter Stone. “Autonomous agents modelling other agents: A comprehensive survey and open problems”. In: *Artificial Intelligence* 258 (May 1, 2018), pp. 66–95. ISSN: 0004-3702. DOI: 10.1016/j.artint.2018.01.002. URL: https://www.sciencedirect.com/science/article/pii/S0004370218300249 (visited on 07/24/2025).

[4] Dipyaman Banerjee and Sandip Sen. “Reaching pareto-optimality in prisoner’s dilemma using conditional joint action learning”. In: *Autonomous Agents and Multi-Agent Systems* 15.1 (Aug. 2007). Publisher: Springer Science and Business Media LLC, pp. 91–108. ISSN: 1387-2532, 1573-7454. DOI: 10.1007/s10458-007-0020-8. URL: http://link.springer.com/10.1007/s10458-007-0020-8 (visited on 07/24/2025).

[5] Zhe Chen et al. “Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery”. In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 5816–5823. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3074883. arXiv: 2110.14891[cs]. URL: http://arxiv.org/abs/2110.14891 (visited on 12/27/2024).

[6] Yujiao Cheng, Liting Sun, and Masayoshi Tomizuka. “Human-Aware Robot Task Planning Based on a Hierarchical Task Model”. In: *IEEE Robotics and Automation Letters* 6.2 (Apr. 2021), pp. 1136–1143. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3056370. URL: https://ieeexplore.ieee.org/document/9345470/ (visited on 02/13/2025).

[7] Aidan Curtis et al. *Long-Horizon Manipulation of Unknown Objects via Task and Motion Planning with Estimated Affordances*. Aug. 10, 2021. DOI: 10.48550/arXiv.2108.04145. arXiv: 2108.04145[cs]. URL: http://arxiv.org/abs/2108.04145 (visited on 12/26/2024).

[8] Aidan Curtis et al. *Partially Observable Task and Motion Planning with Uncertainty and Risk Awareness*. Oct. 6, 2024. DOI: 10.48550/arXiv.2403.10454. arXiv: 2403.10454[cs]. URL: http://arxiv.org/abs/2403.10454 (visited on 12/26/2024).

[9] Neil T Dantam et al. “An incremental constraint-based framework for task and motion planning”. In: *The International Journal of Robotics Research* 37.10 (Sept. 2018), pp. 1134–1151. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364918761570. URL: https://journals.sagepub.com/doi/10.1177/0278364918761570 (visited on 02/05/2025).

[10] Marco Faroni et al. “Optimal task and motion planning and execution for human-robot multi-agent systems in dynamic environments”. In: *IEEE Transactions on Cybernetics* 54.6 (June 2024), pp. 3366–3377. ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TCYB.2023.3263380. arXiv: 2303.14874[cs]. URL: http://arxiv.org/abs/2303.14874 (visited on 01/21/2025).

[11] Richard E. Fikes and Nils J. Nilsson. “Strips: A new approach to the application of theorem proving to problem solving”. In: *Artificial Intelligence* 2.3 (Dec. 1971), pp. 189–208. ISSN: 00043702. DOI: 10.1016/0004-3702(71)90010-5. URL: https://linkinghub.elsevier.com/retrieve/pii/0004370271900105 (visited on 02/03/2025).

[12] Drew Fudenberg and David Levine. “Learning in games”. In: *European Economic Review* 42.3 (May 31, 1998), pp. 631–639. ISSN: 0014-2921. DOI: 10.1016/S0014-2921(98)00011-7. URL: https://www.sciencedirect.com/science/article/pii/S0014292198000117 (visited on 07/25/2025).

[13] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. *PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning*. Mar. 23, 2020. DOI: 10.48550/arXiv.1802.08705. arXiv: 1802.08705[cs]. URL: http://arxiv.org/abs/1802.08705 (visited on 12/26/2024).

[14] Caelan Reed Garrett et al. *Integrated Task and Motion Planning*. Oct. 2, 2020. DOI: 10.48550/arXiv.2010.01083. arXiv: 2010.01083[cs]. URL: http://arxiv.org/abs/2010.01083 (visited on 12/26/2024).

[15] Caelan Reed Garrett et al. *Online Replanning in Belief Space for Partially Observable Task and Motion Problems*. Mar. 23, 2020. DOI: 10.48550/arXiv.1911.04577. arXiv: 1911.04577[cs]. URL: http://arxiv.org/abs/1911.04577 (visited on 12/26/2024).

[16] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge: Cambridge University Press, 2016. ISBN: 978-1-107-03727-4. DOI: 10.1017/CBO9781139583923. URL: https://www.cambridge.org/core/books/automated-planning-and-acting/E6DE5715A2190651352DFB0869916BC3 (visited on 02/23/2025).

[17] P. J. Gmytrasiewicz and P. Doshi. "A Framework for Sequential Planning in Multi-Agent Settings". In: *Journal of Artificial Intelligence Research* 24 (July 1, 2005). Publisher: AI Access Foundation, pp. 49–79. ISSN: 1076-9757. DOI: 10.1613/jair.1579. URL: https://jair.org/index.php/jair/article/view/10414 (visited on 07/23/2025).

[18] Piotr J Gmytrasiewicz and Edmund H Durfeet. "A Rigorous, Operational Formalization of Recursive Modeling". In: (1995).

[19] Nishad Gothoskar et al. *Bayes3D: fast learning and inference in structured generative models of 3D objects and scenes*. Dec. 14, 2023. DOI: 10.48550/arXiv.2312.08715. arXiv: 2312.08715[cs]. URL: http://arxiv.org/abs/2312.08715 (visited on 02/24/2025).

[20] Dylan Hadfield-Menell et al. "Modular task and motion planning in belief space". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany: IEEE, Sept. 2015, pp. 4991–4998. ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7354079. URL: http://ieeexplore.ieee.org/document/7354079/ (visited on 01/31/2025).

[21] Eric A. Hansen and Shlomo Zilberstein. "LAO□: A heuristic search algorithm that finds solutions with loops". In: *Artificial Intelligence* 129.1 (June 2001), pp. 35–62. ISSN: 00043702. DOI: 10.1016/S0004-3702(01)00106-0. URL: https://linkinghub.elsevier.com/retrieve/pii/S0004370201001060 (visited on 07/31/2025).

[22] Jorg Hoffmann and Ronen I Brafman. "Contingent Planning via Heuristic Forward Search with Implicit Belief States". In: ().

[23] *https://www.cs.cmu.edu/~mmv/planning/readings/98aips-PDDL.pdf*. URL: https://www.cs.cmu.edu/~mmv/planning/readings/98aips-PDDL.pdf (visited on 01/24/2025).

[24] *Integrated task and motion planning in belief space*. DOI: 10.1177/0278364913484072. URL: https://journals.sagepub.com/doi/epdf/10.1177/0278364913484072 (visited on 01/20/2025).

[25] Leslie P. Kaelbling and Tomas Lozano-Perez. *Integrated Robot Task and Motion Planning in the Now:* Fort Belvoir, VA: Defense Technical Information Center, June 29, 2012. DOI: 10.21236/ADA564092. URL: http://www.dtic.mil/docs/citations/ADA564092 (visited on 01/20/2025).

[26] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1 (May 1, 1998), pp. 99–134. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(98)00023-X. URL: https://www.sciencedirect.com/science/article/pii/S000437029800023X (visited on 07/23/2025).

[27] Leslie Pack Kaelbling and Tomas Lozano-Perez. "Hierarchical task and motion planning in the now". In: *2011 IEEE International Conference on Robotics and Automation*. 2011 IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China: IEEE, May 2011, pp. 1470–1477. ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5980391. URL: http://ieeexplore.ieee.org/document/5980391/ (visited on 01/20/2025).

[28] Emilie Kaufmann, Olivier Cappe, and Aurelien Garivier. "On Bayesian Upper Confidence Bounds for Bandit Problems". In: ().

[29] Minghua Liu et al. “Task and Path Planning for Multi-Agent Pickup and Delivery”. In: ().

[30] Tomas Lozano-Perez and Leslie Pack Kaelbling. “A constraint-based method for solving sequential manipulation planning problems”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014). Chicago, IL, USA: IEEE, Sept. 2014, pp. 3684–3691. ISBN: 978-1-4799-6934-0 978-1-4799-6931-9. DOI: 10.1109/IROS.2014.6943079. URL: http://ieeexplore.ieee.org/document/6943079/ (visited on 02/05/2025).

[31] R Nair et al. “Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings”. In: ().

[32] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. ISSN: 2196-548X, 2196-5498. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-28927-4 978-3-319-28929-8. DOI: 10.1007/978-3-319-28929-8. URL: http://link.springer.com/10.1007/978-3-319-28929-8 (visited on 07/21/2025).

[33] Camille Phiquepal and Marc Toussaint. “Combined Task and Motion Planning under Partial Observability: An Optimization-Based Approach”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019 International Conference on Robotics and Automation (ICRA). Montreal, QC, Canada: IEEE, May 2019, pp. 9000–9006. ISBN: 978-1-5386-6027-0. DOI: 10.1109/ICRA.2019.8793260. URL: https://ieeexplore.ieee.org/document/8793260/ (visited on 01/22/2025).

[34] Camille Phiquepal and Marc Toussaint. “Multi-Agent Task and Motion Planning: An Optimization based Approach”. In: ().

[35] *Probabilistic PDDL*. URL: https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume24/younes05a-html/node2.html (visited on 07/24/2025).

[36] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. In collab. with Ernest Davis and Douglas Edwards. Third edition, Global edition. Prentice Hall series in artificial intelligence. Boston Columbus Indianapolis: Pearson, 2016. 1 p. ISBN: 978-0-13-604259-4 978-1-292-15397-1.

[37] Naman Shah et al. *Anytime Integrated Task and Motion Policies for Stochastic Environments*. May 29, 2020. DOI: 10.48550/arXiv.1904.13006. arXiv: 1904.13006[cs]. URL: http://arxiv.org/abs/1904.13006 (visited on 12/26/2024).

[38] Tom Silver et al. *Learning Symbolic Operators for Task and Motion Planning*. July 15, 2021. DOI: 10.48550/arXiv.2103.00589. arXiv: 2103.00589[cs]. URL: http://arxiv.org/abs/2103.00589 (visited on 12/26/2024).

[39] David Speck. “SymK - A Versatile Symbolic Search Planner”. In: ().

[40] Neil T. Dantam et al. “Incremental Task and Motion Planning: A Constraint-Based Approach”. In: *Robotics: Science and Systems XII*. Robotics: Science and Systems 2016. Robotics: Science and Systems Foundation, 2016. ISBN: 978-0-9923747-2-3. DOI: 10.15607/RSS.2016.XII.002. URL: http://www.roboticsproceedings.org/rss12/p02.pdf (visited on 01/20/2025).

[41] Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. “In defense of PDDL axioms”. In: *Artificial Intelligence* 168.1 (Oct. 2005), pp. 38–69. ISSN: 00043702. DOI: 10.1016/j.artint.2005.05.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0004370205000810 (visited on 01/30/2025).

[42] Marc Toussaint. “Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning”. In: ().

[43] Marc Toussaint and Manuel Lopes. “Multi-bound tree search for logic-geometric programming in cooperative manipulation domains”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore, Singapore: IEEE, May 2017, pp. 4044–4051. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989464. URL: http://ieeexplore.ieee.org/document/7989464/ (visited on 01/22/2025).

[44] Glenn Wagner and Howie Choset. “Path Planning for Multiple Agents under Uncertainty”. In: *Proceedings of the International Conference on Automated Planning and Scheduling* 27 (June 5, 2017), pp. 577–585. ISSN: 2334-0843. DOI: 10.1609/icaps.v27i1.13866. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/13866 (visited on 12/27/2024).

[45] Sungwook Yoon, Alan Fern, and Robert Givan. “FF-Replan: A Baseline for Probabilistic Planning”. In: ().

[46] Zhigen Zhao et al. “A Survey of Optimization-based Task and Motion Planning: From Classical To Learning Approaches”. In: *IEEE/ASME Transactions on Mechatronics* (2024), pp. 1–27. ISSN: 1083-4435, 1941-014X. DOI: 10.1109/TMECH.2024.3452509. arXiv: 2404.02817[cs]. URL: http://arxiv.org/abs/2404.02817 (visited on 12/27/2024).