Contents

Acronyms

AI Artificial Intelligence. 1

HRC Human-Robot Collaboration. 1, 2

HRI Human-Robot Interaction. 7

Introduction

Contents

Human-Robot Collaboration (HRC) is a growing field in robotics and Artificial Intelligence (AI) research that aims to enable safe and effective teamwork between humans and robots. This field concerns fully autonomous robots. Hence, it excludes exoskeletons or teleoperated robots such as surgery manipulators or remotely operated (aerial) vehicles.

The work presented is assumed to be in the Joint Action context, which is described in [Sebanz 2006b] as "any form of social interaction whereby two or more individuals coordinate their actions in space and time to bring about a change in the environment". Various relationships can exist between humans and robots, e.g., collaborators, companions, guides, tutors, or social interaction partners. In all these cases, we think the robot should help and facilitate the human in terms of physical effort, mental workload, and even emotional state.

Industrial robots are already popular in factories because they are fast, accurate, reliable, and never get tired, which makes them ideal for repetitive factory tasks. However, such robots are usually in dedicated areas where humans cannot enter for safety reasons. Hence, it is still an open challenge to endow robots with enough reliable reasoning capabilities and compliant motion control to allow efficient and trusted direct collaboration between humans and robots.

Moreover, HRC can also occur in various contexts that must be considered, ranging from co-worker robots in factories to householder robots for our everyday lives and including service robots in public places like restaurants and shops.

Focused on the decisional aspect of HRC, this work aims to design autonomous robots able to make explainable, acceptable, and efficient decisions to collaborate with humans.

Human-Robot Collaboration (HRC) is a multidisciplinary field that opens several technical challenges to address. The ideal collaborative robot should be the aggregation of solutions for each of the challenges listed below:

- **Navigation**: The robot should be able to acceptably and efficiently move in a human-populated environment. This implies being mechanically designed for it, anticipating and planning correct trajectories, and being able to adapt and follow these trajectories in real time. The robot should not move threateningly and should account for humans.

- **Manipulation**: The robot should be able to manipulate objects to interact with its environment. Hence, the robot should have an actuator like an arm

and a gripper and should be able to exhibit motions that are efficient and safe to nearby humans.

- **Communication**: To achieve congruent interaction and collaboration, collaborative agents must communicate. This implies that the robot should be able to communicate information to the human and understand the one received from the latter. These communications can be verbal or non-verbal.
- **Perception**: It is mandatory for the robot to have a reliable perception scheme to know the position of near objects, obstacles, and humans. First, relevant sensors must be used and placed on the robot or in the environment itself. After, by reasoning on the sensory data, relevant facts must be extracted about the robot's environment, such as objects' positions, spatial relations, reachable objects, human knowledge, intentions, or the state of the current goal.
- **Decision-making**: This aspect is the focus of my work and implies that the robot should be able to make relevant decisions to be collaborative, that is, planning its actions to solve a collaborative task. It also implies being able to supervise the task execution and make online decisions to adapt to uncertainties in real time, which includes human actions and commands.

Contributions and manuscript organization

My work addressed the **decision-making** aspect of HRC and led to three main contributions summarized below.

I began by participating in the development of a novel human-aware task planner called HATP/EHDA [Buisan 2022]. This approach considers two distinct agent models: an uncontrollable one to estimate the human's behavior and a controllable one to plan the robot's actions accordingly. The agent models include distinct beliefs, agendas, and action models. As a result, the planner can anticipate and even elicit human decisions and actions, but it never compels them. For these reasons, we believe this approach is promising to address HRC, and I built two of my main contributions upon this approach.

My first contribution is to propose some models and algorithms of the Theory of Mind and to integrate them into the deliberation process of HATP/EHDA. Despite considering distinct beliefs, they were only updated according to the description of the action effect provided in the estimated human action model. This means, for instance, that modeling the human observing and learning a new fact by entering a room had to be manually scripted in the 'move' action description. With this contribution, we propose that an agent can learn by observing their surroundings or an action execution. Additionally, we propose a way to detect false human beliefs during the planning process, which may be detrimental to the task. Eventually, fix these relevant belief divergences by planning minimal verbal robot communication or delaying robot action that the human will initially not observe.

My second contribution brings planning and execution closer by addressing the turn-taking assumption of HATP/EHDA and exploring parallel executions. We formulated a step-based model of compliant and concurrent joint action. This model describes how the two agents should coordinate as well as four possible online human decisions about the execution: (1) the human decides to be passive and let the robot act alone, (2) the human acts alone and the robot is passive, (3) the human starts acting then the robot adapts and acts in parallel, and finally (4) the human deliberately let the robot decide and start acting before complying with it. This model guides the deliberation process to explore relevant courses of concurrent actions. After exploration, the robot's behavioral policy is extracted using a plan evaluation based on estimations of human preferences. Eventually, by following the produced policy, the robot can comply concurrently with any human online decisions and aims to satisfy human preferences, which can be updated online.

My last contribution is distinguishable from the previous two because it addresses navigation decision-making instead of task planning. Indeed, I designed and implemented two systems simulating interactive social navigating agents endowed with decision-making processes. Additionally, these systems include evaluation processes to record data, compute metrics, and plot navigation performances. Consequently, these systems generate challenging situations that permit challenging, debugging, and evaluating robot navigation schemes in intricate human-populated scenarios. The first system, Intelligent Human Simulator (InHuS), simulates one human agent endowed with complex reasoning processes. The second system, Intelligent Multi Human Simulator (IMHuS), simulates several agents with social group behaviors but with less individual reasoning capabilities.

These contributions are detailed in the rest of this manuscript, structured as follows.

TODO: update structure description with less details since contributions are already described above.

Chapter 1 provides more details about the context of my PhD by discussing related fields and works of HRC. This chapter eventually highlights the gap in the literature that motivates my PhD work.

Part I gathers all my work concerning task planning for HRC. It represents a major portion of my PhD work and includes the chapters 2, 3, 4, 5, and 6.

Chapter 2 familiarizes the reader with the HATP/EHDA task planner. Indeed, I participated in its development and this planner has been the keystone of my two contributions to task-planning for HRC. Hence, the reader should understand both this task planner's motivation and methods.

Chapter 3 introduces my first main contribution which incorporates Theory of Mind concepts in HRC task-planning, more precisely, in the HATP/EHDA planning process. Some models and algorithms are proposed and evaluated to better estimate human beliefs in order to better anticipate their potential actions. As a result, we can identify when the human has a false belief about a fact evaluated as relevant for the task. In such cases, the robot can proactively inform the human to correct the false belief or the robot can purposely delay its action to make sure the human

sees its execution and infer the corresponding fact, avoiding verbal communication.

In Chapter 4, as my second main contribution, I address the lack of concurrent actions of HATP/EHDA to improve fluency in collaboration. Inspired by the Joint Action literature, we designed a model of concurrent and compliant execution for HRC in the form of an automaton. We also propose to evaluate plans based on an estimation of the human inner preferences. A novel task planning approach taking into account the mentioned execution model and plan evaluation is proposed. This approach generates concurrent robot policies compliant with human online decisions and preferences.

Chapter 5 is a technical description of an interactive simulator I developed to execute the robot policy generated by the approach described in the previous chapter. This simulator proposes an execution scheme based on the model of execution to run and supervise the robot policy in a 3D simulator. It also allows a human operator to perform actions through intuitive mouse control. Hence, this simulator offers a way to collaborate with a robot following the approach we designed and is used in the next chapter to evaluate the approach.

Chapter 6 presents a user study validating the approach proposed in Chapter 4 using the simulator described in Chapter 5. For this purpose, several scenarios have been designed using a BlocksWorld task, and human participants were asked to collaborate with the simulated robot to evaluate its behavior. We compared our approach with a baseline consisting of a robot following a simpler version of the model of execution.

Conclusions regarding part I follow this chapter before starting the second part of this thesis.

Part II concerns decision-making in navigation and how to simulate interactive social navigating agents. Despite not being my main research topic, this subject represents significant work in my PhD. This part includes the two last chapters: 7 and 8.

Chapter 7 describes my third contribution in which I address the decision-making challenge in navigation to allow the robot to move in a human-populated environment acceptably and efficiently. I developed a system producing an "intelligent" human avatar that, while being reactive, can make rational decisions about navigation tasks. This system serves as a benchmarking and testing tool for robot navigation systems to be challenged. This way, robot navigation systems can be evaluated, tuned, and stress tested in simulation allowing them to run mature real-life experiments faster.

Chapter 8 presents an additional work with the same rationales as the approach described in Chapter 7 and is largely inspired by it. However, this work addresses a major limitation of the previous one which is not being able to simulate several human agents. This other approach can choreograph several agents with group movements and social behaviors. Despite having limited individual decision-making compared to the previous approach, this additional work generates pertinent and challenging intricate situations with several agents which is beneficial to the social robotic research.

Conclusions concerning both Chapter 7 and 8 end part II.

Eventually, I share general conclusions regarding all of my PhD work in a dedicated part before providing additional materials in the appendix.

List of Publications

As main author

- Anthony Favier, Phani-Teja Singamaneni, Rachid Alami. Simulating Intelligent Human Agents for Intricate Social Robot Navigation. Social Robot Navigation workshop - Robotics: Science and Systems (RSS'21), Jul 2021, Washington, United States.
- Anthony Favier, Phani-Teja Singamaneni, Rachid Alami. An Intelligent Human Avatar to Debug and Challenge Human-aware Robot Navigation Systems. Late Breaking Report - 2022 ACM/IEEE International Conference on Human-Robot Interaction (HRI'22), Mar 2022, Sapporo, Japan.
- Anthony Favier, Shashank Shekhar, Rachid Alami. Robust Planning for Human-Robot Joint Tasks with Explicit Reasoning on Human Mental State. AI-HRI Symposium at AAAI Fall Symposium Series (FSS'22), Nov 2022, Arlington, United States.
- Anthony Favier, Shashank Shekhar, Rachid Alami. Anticipating False Beliefs and Planning Pertinent Reactions in Human-Aware Task Planning with Models of Theory of Mind. PlanRob Workshop - International Conference on Automated Planning and Scheduling (ICAPS'23), Jul 2023, Prague, Czech Republic.
- Anthony Favier, Shashank Shekhar, Rachid Alami. Models and Algorithms for Human-Aware Task Planning with Integrated Theory of Mind. IEEE International Conference on Robot and Human Interactive Communication (RO-MAN'23), Aug 2023, Busan, South Korea.
- Anthony Favier, Phani Teja Singamaneni, Rachid Alami. Challenging Human-Aware Robot Navigation with an Intelligent Human Simulation System. Social Simulation Conference (SSC'23), Sep 2023, Glasgow, France.
- Anthony Favier, Rachid Alami. Planning Concurrent Actions and Decisions in Human-Robot Joint Action Context. Symbiotic Society with Avatars workshop - ACM/IEEE International Conference on Human-Robot Interaction (HRI'24), Mar 2024, Boulder, United States.

As co-author

- Guilhem Buisan, Anthony Favier, Amandine Mayima, Rachid Alami. HATP/EHDA: A Robot Task Planner Anticipating and Eliciting Hu-

man Decisions and Actions. IEEE International Conference On Robotics and Automation (ICRA 2022), May 2022, Philadelphia, United States. ⟨10.1109/ICRA46639.2022.9812227⟩.

- Phani-Teja Singamaneni, Anthony Favier, Rachid Alami. Towards Benchmarking Human-Aware Social Robot Navigation: A New Perspective and Metrics. IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), 2023, Aug 2023, Busan, South Korea.

- Phani-Teja Singamaneni, Anthony Favier, Rachid Alami. Human-Aware Navigation Planner for Diverse Human-Robot Contexts. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sep 2021, Prague (online), Czech Republic.

- Phani-Teja Singamaneni, Anthony Favier, Rachid Alami. Invisible Humans in Human-aware Robot Navigation. IEEE International Conference on Robotics and Automation (ICRA 2022), May 2022, Philadelphia, United States.

- Phani-Teja Singamaneni, Anthony Favier, Rachid Alami. Watch out! There may be a Human. Addressing Invisible Humans in Social Navigation. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022), Oct 2022, Kyoto, Japan.

- Olivier Hauterville, Camino Fernández, Phani-Teja Singamaneni, Anthony Favier, Vicente Matellán, et al.. IMHuS: Intelligent Multi-Human Simulator. IROS2022 Workshop: Artificial Intelligence for Social Robots Interacting with Humans in the Real World, Oct 2022, Kyoto, Japan.

- Olivier Hauterville, Camino Fernández, Phani-Teja Singamaneni, Anthony Favier, Vicente Matellán, et al.. Interactive Social Agents Simulation Tool for Designing Choreographies for Human-Robot-Interaction Research. ROBOT2022: Fifth Iberian Robotics Conference, Nov 2022, Zaragoza, Spain.

Chapter 1

Human-Robot Collaboration Context

Contents

1.1 Multidisciplinarity of Human-Robot Interaction

In [Bartneck 2020], the Human-Robot Interaction (HRI) is considered unique because of the interaction of humans with social robots, which is at the core of this multidisciplinary research field. These interactions usually include physically embodied robots, and their embodiment makes them inherently different from other computing technologies. Moreover, social robots are perceived as social actors bearing cultural meaning and strongly impacting contemporary and future societies. Saying that a robot is embodied does not mean it is simply a computer on legs

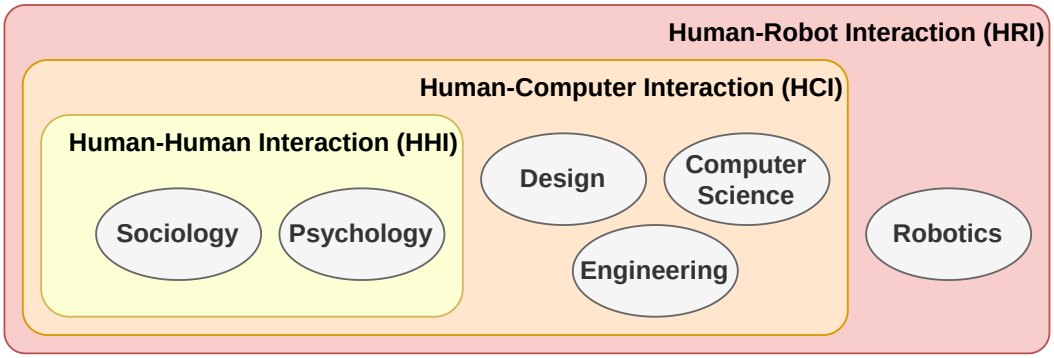


Figure 1.1: Multidisciplinarity of the Human-Robot Interaction field.

or wheels. Instead, we must understand how to design that embodiment, both in terms of software and hardware, as it is commonplace in robotics, and in terms of its effects on people and the kinds of interactions they can have with such a robot.

Overall, HRI focuses on developing robots that can interact with people in various everyday environments. This opens up technical challenges resulting from the dynamics and complexities of humans and the social environment. This also opens up design challenges—related to robotic appearance, behavior, and sensing capabilities—to inspire and guide interaction. From a psychological perspective, HRI offers the unique opportunity to study human affect, cognition, and behavior when confronted with social agents other than humans. In this context, social robots can be research tools to study psychological mechanisms and theories.

As a result, by taking inspiration from Human-Human Interaction (HHI) and Human-Computer Interaction (HCI), HRI is an endeavor that brings together ideas from a wide range of disciplines such as Engineering, Computer Science, Robotics, Psychology, Sociology, and Design by taking inspiration from Human-Human and Human-Computer Interaction. In the following, we discuss some related aspects and works of the mentioned disciplines by categorizing them in HHI, HCI, and HRI as depicted in figure 1.1.

1.1.1 Human-Human Interaction

Many works dealing with interacting with humans take inspiration from Human-Human Interaction (HHI), including research in sociology and psychology. HHI refers to the communication and collaboration between two or more individuals, where humans engage in various forms of social, cognitive, and emotional exchanges. Such interaction can occur through verbal and non-verbal communication, such as speech, gestures, facial expressions, and body language.

TODO: detail more each aspect with words

Communication theories, both verbal or not: Albert Meharbian's 7-38-55 rule [Mehrabian 1967], and the Grice's four maxims of conversation called the Gricean maxims: quantity, quality, relation, and manner. These four maxims describe specific rational principles observed by people who follow the cooperative principle in

pursuit of effective communication [Grice 1975]. [Smith 1998], [Cherry 1966])

Joint Action theories, including collaboration and teamwork ([Cohen 1991, Cohen 1970, Levesque 1990])

Social psychology (Stanley Milgram, Philip Zimbardo, and Solomon Asch)

conflict resolution and negotiation (Roger Fisher and William Ury)

emotional intelligence (Daniel Goleman)

cross-cultural communication.

We must first understand how humans interact with each other before making robots able to interact correctly with humans. Nevertheless, mimicking humans perfectly is questionable since robots fundamentally differ from humans. Robots are created by humans to be helped and assisted. Thus, HHI should inspire robot design, but additional research is mandatory to determine how to create appropriate interactive and collaborative robots.

1.1.2 Human-Computer Interaction

A first step of artificial interaction and collaboration is the field of Human-Computer Interaction. Human-Computer Interaction (HCI) is the field of study that focuses on optimizing how users and computers interact by designing interactive computer interfaces that satisfy users' needs. It is a multidisciplinary subject covering computer science, behavioral sciences, cognitive science, ergonomics, psychology, and design principles. Today, HCI focuses on designing, implementing, and evaluating interactive interfaces that enhance user experience using computing devices. This includes user interface design, user-centered design, and user experience design.

This field is made up of four key components. The User along with their needs, goals, interaction patterns, cognitive capabilities, emotions, and experiences. The Goal-Oriented Task which is the objective or goal the user has in mind. The Interface is about the overall user interaction experience through senses such as touch, click, gesture, voice, display size, colors. The Context must be taken into account because it influences the interaction.

To produce easy to interact with robots, the study of HCI is relevant and also serves as an inspiration to design intuitive, user-friendly interactive robots.

1.1.3 Human-Robot Interaction

Human-Robot Interaction (HRI) is a field of study that explores the design, development, and evaluation of robots that interact with humans in various settings. HRI aims to create robots that can effectively and seamlessly collaborate with humans in domestic environments, workplaces, or other contexts. HRI can be categorized in several domains, not necessarily exclusive. Here are some examples:

Social robotics focuses on social interactions with humans and, thus, explores how robots can understand and respond to human emotions, social cues, and communication styles. A significant amount of work is dedicated to HRI in Healthcare to assist patients, especially the elderly and children with conditions. Those works

are also usually linked to emotion-aware robotics focused on recognizing and responding to human emotions using affective computing techniques. A common application is storytelling for children to convey ideas, feelings, or culture.

Human-Centered Robotics emphasizes the importance of considering human needs and preferences. This subfield often involves user studies to ensure and identify if and how robots are user-friendly and can seamlessly integrate into human environments.

Robot Ethics is another central subfield focused on considerations such as privacy, safety, responsibility/accountability, and the impact of robots on society.

Explainable AI and transparency are a growing interest in making decision-making processes more understandable to humans, and thus, help robots be legible, predictable, and acceptable.

Computational HRI, as described in [Thomaz 2016], is the subset of HRI concerned explicitly with the algorithms, techniques, models, and frameworks necessary to build robotic systems that engage in social interactions with humans. This thesis is part of this category because it is focused on developing task-planning algorithms and models relevant to a collaborative robot.

Human-Robot Collaboration or Collaborative Robotics focuses on developing robots that work alongside humans in shared workspaces, usually as a team. HRC is the main topic of my thesis and is a vast subject worth delving into. Hence, the following section is dedicated to providing more details about HRC.

1.2 Human-Robot Collaboration

Human-Robot Collaboration (HRC) refers to the synergy and cooperation between humans and robots in shared environments to achieve common goals. In HRC, humans and robots work together, often leveraging their complementary strengths to enhance overall performance and efficiency. According to human desires, the robot can also act in a way that eases and facilitates the human part of the task. This collaborative approach involves close interaction, communication, and coordination between human and robotic agents.

1.2.1 Inspirations & Theories informing HRC

This interdisciplinary field takes inspiration from various theories and fields as introduced earlier. Nevertheless, three main inspirations can be highlighted:

Belief Desire Intention Model:

The belief-desire-intention (BDI) model was originally developed by Michael Bratman [Bratman 1987]. This model is used in intelligent agents research to describe and model intelligent agents. Straightforwardly, the BDI model is characterized by the implementation of the three notions appearing in its name, i.e., an agent's beliefs (knowledge of the work in the perspective of the agent), desires (objective

or goal to accomplish), and intentions (the planned course of actions to achieve the agent's desire).

Shared Cooperative Activity:

Shared cooperative Activity defines prerequisites for an activity to be considered shared and cooperative. The main ones are mutual responsiveness, commitment to the joint activity, and commitment to mutual support. A good example to clarify these prerequisites is a scenario where agents move a table together. Mutual responsiveness ensures that the agents' movements are synchronized. The commitment to the joint activity reassures each agent that the others will not drop their side and quit the joint activity. Finally, the commitment to mutual support deals with possible breakdowns due to one agent's inability to perform part of the plan.

Joint Intention & Action Theory:

Joint Intention Theory proposes that for joint action to emerge, team members must communicate to maintain a set of shared beliefs and to coordinate their actions toward the shared plan [Cohen 1991]. In collaborative work, agents should be able to count on the commitment of other members. Therefore, each agent should inform the others when they conclude that a goal is achievable, impossible, or irrelevant [Hoffman 2004].

1.2.2 Key Aspects

In order to better picture the implications of the above theories, some key aspects of a seamless collaboration are listed and commented on below. This list is not exhaustive, but it highlights some skills that humans naturally exhibit and that a robot must be endowed with to collaborate with them.

Specialization of Roles: There are several human-robot relationships, including supervisor-subordinate, partner-partner, teacher-learner, and leader-follower. These roles can be predefined and fixed during the whole collaboration. The role distribution can also be flexible using weighting functions that allow a continuous change between the roles to adapt to every context and situation.

Establishing shared goal(s): Through direct discussion or inference, agents must determine the shared goals they are trying to achieve. However, a shared goal isn't always necessary and can be established in the middle of a task execution either by the human or the robot.

Allocation of subtasks: After deciding how to achieve their goals, agents must determine what actions and subtasks will be done by each agent and how to coordinate each other. This can either be done explicitly before starting the task or be reactively done on the fly.

Progression tracking: Agents must be able to track progress toward their goals. That is, they must be able to determine what has been achieved, by whom, and what remains to be done.

Communication: Any collaboration requires communication, verbal or not. Most of the mentioned aspects can or must involve communication. However, it is essential to identify what and how to communicate during the collaboration. Communicating too much can be annoying, while not enough can induce confusion and harm collaboration.

Adaption and learning: On a short-term scale, agents must adapt to each other and the environment. In the longer term, agents must also learn from other partners and the acquired experience.

Ergonomics: It should be intuitive to collaborate and communicate with the robot. This aspect must be taken into account when designing both the hardware and the software of the robot. Ergonomics is a central aspect of Human-Compute Interaction. Thus, many works from this field can be used in our context or serve as inspiration.

Explainability: This aspect is important for seamless collaboration as the human should be able to understand what the robot is doing and why. This topic is getting more and more attention and is often referred to as Explainable AI. This is especially relevant to counter the *black box* tendency of machine learning where it's impossible to explain a specific decision. Being explainable often enhances predictability, which is also essential for a collaborative robot.

1.2.3 Architectures & Complete systems

It is important to remember that since a collaborative robot is issued from an interdisciplinary field, its different functionalities and capabilities are usually separated into several dedicated components. These components interact and communicate with each other, forming a complete architecture. This architecture covers all aspects relevant to exhibiting the robot's behavior, from sensory perception to physical motions, including reasoning processes. Despite developing distinct robotic components, the robot must be considered a whole. Each component cannot be studied entirely independently of other aspects of the complete system.

As a result, some works are dedicated to designing robotic architectures, especially cognitive architectures, which are the most promising ones for designing collaboration robots.

Example of cognitive architecture [cite SOAR Laird et al, CORTEX Bustos et al, ORO/SHARY/SPARK.. Lemaignan, perspective-aware Lemaignan] **TODO**

1.3 Models for interaction

It has been shown previously that a robotic agent interacting with a human needs to coordinate its actions with them. Moreover, joint action theory exhibits that humans interacting together represent the task as a whole, and plan not only for their actions but also for the actions of other agents. Thus, we think that for a human to perform the most efficient and satisfactory joint task with a robot, this robot must explicitly model human actions and plan not only for its actions but

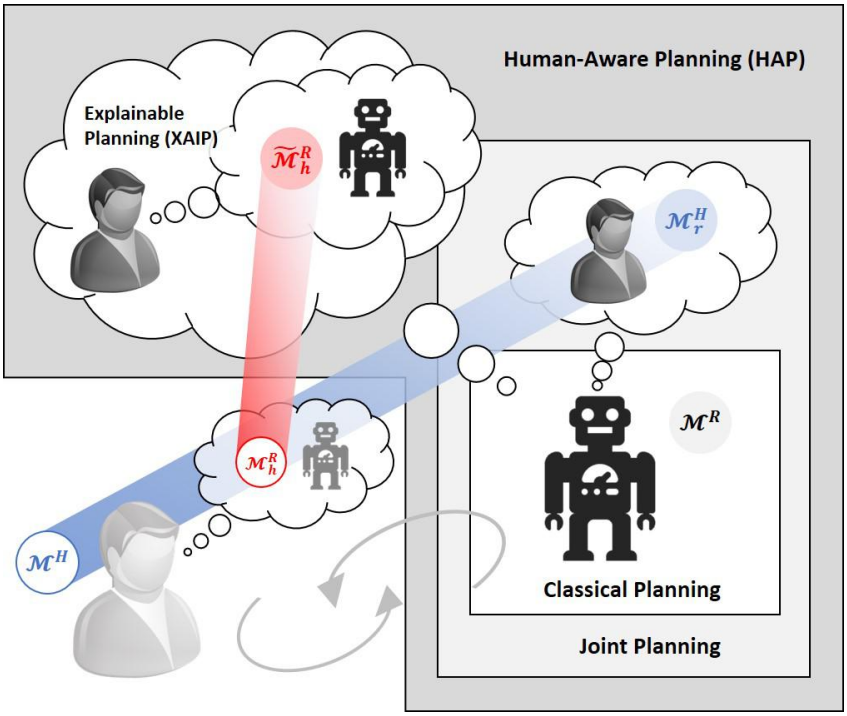


Figure 1.2: Agent models from Chakraborti *et al.* notations.

also for the human ones. This is why we present in this section some notations to clarify the different models used in this thesis, and then we present in more details how to model tasks.

1.3.1 Human and Robot agents

Chakraborti *et al.* introduced notations to differentiate between the models used in this thesis in their work [Chakraborti 2015], also used in Buisan's thesis [Buisan 2021]. These notations summarize and differentiate elegantly the different models manipulated in the HRI/HRC field. These notations are depicted in figure 1.2. At the bottom are depicted the human agent (on the left) and the robot agent (on the right). When solving a task alone, the robot uses its own model referred to as $\mathcal{M}^R$ and this is considered as Classical Planning. Then, $\mathcal{M}_r^H$ is an estimation by the robot of the model of the human. Finally, $\tilde{\mathcal{M}}_h^R$ is an estimation of the robot model the human has. These models are likely to include knowledge about the world in the agent's perspective, an action model describing the agent's capabilities, and an agenda capturing the goal and motivation of the agent.

It is important to keep in mind that when discussing a task planner it is considered as part of the robot. Thus, $\mathcal{M}^R$ is considered as the ground truth for the robot. As a consequence, if there is a belief divergence between $\mathcal{M}_r^H$ and $\mathcal{M}^R$, we always consider that $\mathcal{M}^R$ is the truth, otherwise, it would make no sense to keep this information in $\mathcal{M}^R$ while having access to the one in $\mathcal{M}_r^H$.

1.3.2 Task modeling

A common way of representing human activity (MrH) and interaction with computers at a high abstraction level is by using task models. The hierarchical structure of human activity was first exploited by Annett and Duncan [Annett 1967]. They state that tasks can be described at several levels of abstraction until a certain criterion is met. Each task can thus be refined into subtasks detailing the procedure followed by the human to achieve the higher level task. Task modeling has then evolved to introduce interaction with systems, produced and needed information, potential errors and a wide variety of operator specifying how tasks interact with each other during their execution. Task models are now commonly used in user-centered and user-interface design processes. Most advanced notations include ConcurTaskTrees [Paternò 2004] and HAMSTERS [Martinie 2019]. These models are used to design or evaluate interactive systems. They allow the designer to better understand the user task or to study the user workflow using their system. However, these models contain too little information for a system to be able to reason and make decisions on them (either in planning or acting).

1.3.3 Hierarchical models

In classical planning, each action of an agent is atomic and needs some conditions to hold in the environment to be executed, and then it changes the environment when applied. The planning process has then to find the right sequence of actions, being applicable one after the other to change the environment to reach a specific goal state.

However, humans tend to work more abstractly and decompose tasks hierarchically into smaller tasks until the action level is reached. In practice, using Hierarchical Task Networks (HTNs) allows the domain designer to help the plan search by inserting expert knowledge via a hierarchy linking the actions [Erol 1996]. A task network consists of tasks organized in a fully or partially ordered manner, and each task can be either abstract or primitive. Primitive tasks are elementary tasks that can be achieved by performing one associated action. On the other hand, abstract tasks are tasks that first need to be decomposed into other subtasks, "more primitive". The planner's goal is not to find the sequence of actions to reach a goal but to select recursively for each task the suitable decomposition ending (if possible) with a network of actions applicable from the initial state. Ghallab, Nau, and Traverso name such a process as *planning with refinement methods* [Ghallab 2016]. This planning hierarchy allows the domain designer to guide the search by inserting some expertise into the model and enhancing explainability as the decompositions often offer a semantic to their subtasks. The 'why' of a subtask can usually be answered by going up in the hierarchy, while the 'how' is answered by going down.

To better picture HTN models, a symbolic example is depicted in figure 1.3. Rectangles represent abstract tasks. Each method describes a possible way to decompose an abstract task if its preconditions are met. Methods' preconditions are

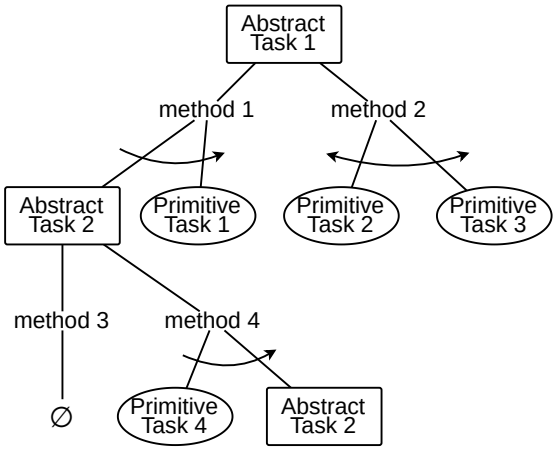


Figure 1.3: An HTN example. Rectangles represent abstract tasks. Each method describes a possible way to decompose an abstract task if its preconditions are met. Methods can decompose abstract tasks into primitive tasks (ellipses) and/or into other abstract tasks. The obtained subtasks can be fully ordered, such as with *method 1* (represented with a one-way arrow), or partially ordered, like *method 2* (represented with a two-way arrow). Note that methods can also decompose a task into nothing like *method 3*, for instance, when the task is already done, and they can be recursive like *method 4*.

not necessarily mutually exclusive. Hence, as mentioned above, it is the planner's job to select the most suitable one when several ones are applicable. Methods decompose abstract tasks into primitive tasks, represented with ellipses, and/or into other abstract tasks. The obtained subtasks can be fully ordered, such as with *method 1* (represented with a one-way arrow) where *Abstract Task 2* has to be completed before *Primitive Task 1*. Methods can also be partially ordered as with *method 2* (represented with a two-way arrow) where *Primitive Task 2* and *Primitive Task 3* can be achieved in any order. Note that methods can also decompose a task into nothing like *method 3*, for instance, when the task is already done. Moreover, methods can be recursive like *method 4*.

Now, let us look at an example of a refinement process. We consider an initial task network only composed of *Abstract Task 1*, and we refine it using the HTN described in fig 1.3 until the task network only contains primitive tasks (actions). Initially, *method 1* and *method 2* are both applicable and thus, are candidates to decompose *Abstract Task 1*. Applying *method 2* leads directly to a fully refined solution task network, including two partially ordered primitive tasks (*Primitive Task 2* and *Primitive Task 3*). On the other hand, *method 1* can be applied, leading to a different task network that still includes abstract tasks. Then, *Abstract Task 2* is recursively decomposed using M4 twice. This could correspond to scenarios like filling a box with two balls. Thus, M4 is only applicable until two more balls are in the box. Eventually, only M3 is applicable and refines *Abstract Task 2* into nothing, leading to another solution task network.

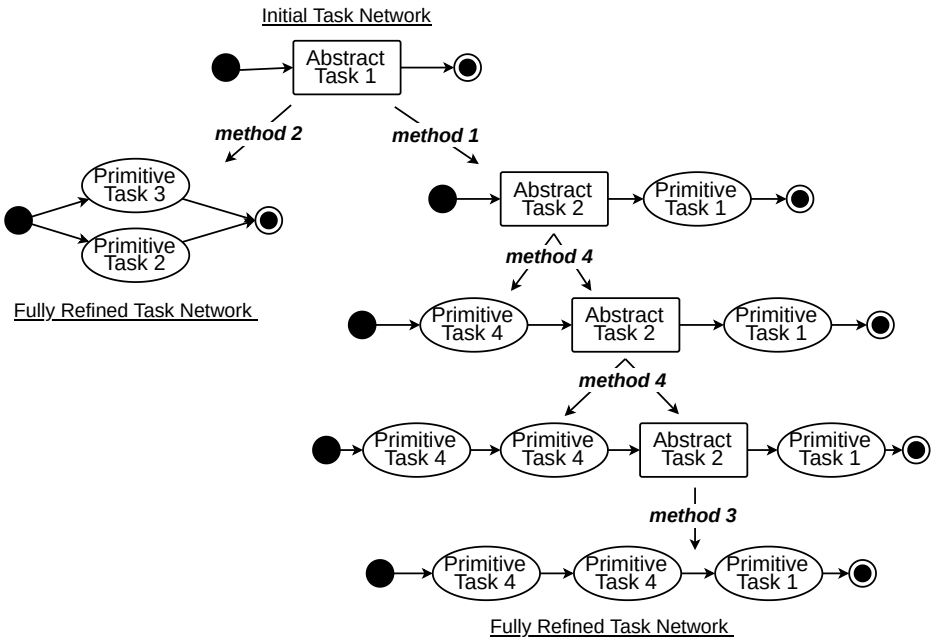


Figure 1.4: Two possible decompositions of a task network using the HTN described in fig 1.3. Both *method 1* and *method 2* can be applied, leading to two different solutions.

1.4 Task Planning

1.4.1 Classical Planning

As put by Ghallab, Nau and Traverso, "the purpose of planning is to synthesize an organized set of actions to carry out some activity" [Ghallab 2016]. Classical planning is a type of planning that assume deterministic and fully observable environments. It involves representing the world as a set of states and actions, with plans derived through state space search algorithms. Actions have preconditions and effects, and planning problems entail finding a sequence of actions transforming an initial state into a goal state. Classical planning algorithms, including STRIPS, Graphplan, and Fast Downward, utilize heuristics to guide the search efficiently. While well-suited for domains with clear and deterministic dynamics, classical planning may face challenges in handling uncertainty or partial observability, leading to the development of alternative planning approaches for such scenarios.

1.4.2 Planning for HRC

Classical planning has been vastly studied and can now solve efficiently various problems. Yet, the intricate nature of HRC scenarios demands sophisticated task-planning methodologies capable of adapting to dynamic environments, understanding human intent, and promoting a fluent exchange of information. Hence, several subfields of task-planning have emerged and are used in HRC. Here are a few ex-

amples:

- **Hierarchical Task Planning** is a technique that organizes tasks in a hierarchical structure like presented in the previous section, allowing for the representation of complex tasks at various abstraction levels. This approach enhances modularity, flexibility and explainability in task planning, accommodating intricate collaborative scenarios.
- **Mixed-Initiative Planning** leverages the strengths of both humans and robots by allowing for a dynamic allocation of decision-making authority. This technique promotes collaborative decision-making, enabling the system to adapt to the expertise and preferences of each agent involved in the collaborative task.
- **Human-Centric Task Planning** focuses on incorporating human factors into the planning process. This involves understanding human capabilities, preferences, and cognitive load to optimize task plans that align with the natural workflows and expectations of human collaborators.
- **Learning-Based Task Planning** has emerged thanks to advancements in machine learning as a frontier in adapting to evolving environments. This technique involves training models to understand patterns in human behavior, enabling the robot to learn and adapt its task planning strategies over time. Such techniques can also be used to predict human behavior and consequently adapt the robot's actions.
- **Probabilistic Task Planning** integrates uncertainty into the planning process, acknowledging the inherent unpredictability of human behavior and environmental factors. By incorporating probabilistic models, this technique enhances the robustness of task plans in dynamic and uncertain collaborative settings.
- **Task and Motion Planning** combines symbolic and geometric reasoning to plan agents' actions. In our context, it can be helpful to consider safety spatial areas near humans and adapt both the robot's motion and decisions.

Overall, **Human-Aware Task Planning** is the process of considering the presence and behavior of humans in the planning and execution of robot tasks. It involves taking into account cues from the shared environment and the dynamics of human-robot interaction. The goal is to generate robot policies that are adaptable, robust, and efficient in crowded and dynamic environments. This field includes having an explicit shared task or common goal between the human and the robot, implying that the two agents will collaborate to reach the goal. It also includes not having an established shared goal, and thus, is closer to what I would call an interaction instead of a collaboration. Yet, both problems are interesting and must be addressed, with a unified approach or not. The approaches presented in this thesis are in between the first three subfields listed above.

1.4.3 Other usage of task planning

Despite being designed for Human-Robot Collaboration, the task planning techniques presented in this work can be used in other interactive contexts.

For instance, instead of planning robot actions, we could plan verbal answers in a dialogue. This approach is used in [De Carolis 2000] and [De Carolis 2001]. Hence, some algorithms and models proposed in this manuscript could be used to anticipate the possible human sentences and plan the relevant robot communications.

Additionally, one could think about a smart environment, e.g., a domotic house, where various sensors and actuators are connected. When humans explore and operate in such an environment, it could be relevant to plan domotic actions according to human behavior, e.g., proactively making coffee, opening stores, activating the robot vacuum cleaner, etc.

1.5 Human-Aware Navigation

In my work, I studied the decision-making challenge mainly in the field of task-planning. Nevertheless, I also worked on decision-making processes for navigation, more precisely, on how to simulate a navigating interactive human agent endowed with decisional capabilities to challenge robotic navigation systems. Hence, to better understand this contribution, this section provides some context on robot navigation, especially on state-of-the-art techniques and existing benchmarking tools.

As stated in [Buisan 2021], robot navigation aims to make the robot base (the whole robot) move from one place to another while avoiding static and moving obstacles. However, other constraints must be added when the robot has to move in an environment where humans are evolving. Humans should not just be avoided as other moving obstacles, and their psychological and mental state must be taken into account. Hence, the robot should neither move threateningly, block the humans, nor induce drastic changes in their motion. Taking all these aspects into account is what is called human-aware robot navigation.

1.5.1 Robot Navigation techniques

State-of-the-art techniques for robot navigation involve two kinds of motion planners: a global and a local planner. The global planner is in charge of finding the best overall trajectory leading the robot to its goal, producing a global plan. This planner usually only takes into account static obstacles described by a given map of the environment. Then, the local planner is in charge of producing velocity commands sent to the motor controllers to follow the produced global plan. To produce the velocity commands the local planner may produce a local plan with only a few seconds of time horizon that follows the global plan while taking into account obstacles detected in real-time by the robot sensors, including moving obstacles. This way the robot should reach its goal while being reactive to moving obstacles.

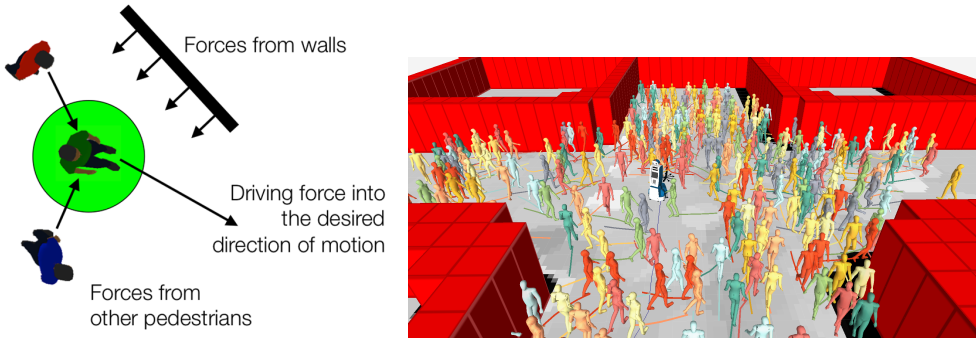


Figure 1.5: Social force model approximating pedestrian motion by a sum of forces.

However, as explained above, such approaches are insufficient in human-populated environments. Humans must be detected and treated differently during the motion planning process. Human-aware approaches detect and track nearby humans and try to estimate their trajectory to plan the robot one accordingly. This is achieved in works like [Singamaneni 2021], where the human trajectory is estimated using goal recognition processes and elastic bands methods. Then, the robot's motion is planned using tuned elastic band methods to account for the robot's goal, the estimated human trajectory, and other social norms.

1.5.2 Benchmarking tools and metrics

Where it is easy to benchmark robot navigation on objective metrics like the time to reach a goal, the distance traveled and the number of collisions [Perille 2020], it is more challenging to benchmark their human-aware properties. First, there is no consensus on the metrics to evaluate a navigation system's human-aware properties. State-of-the-art metrics involve proxemics [Samarakoon 2022]. However, other metrics can be relevant, such as the deviation imposed on human motion and the feeling of threat produced. Also, it is challenging to find a usable system that will effectively challenge a HAN system.

Typical approaches involve reactive-only techniques such as social force models [Helbing 1995, Chen 2018]. The social force model assumes that a sum of different forces can approximate pedestrians' acceleration, deceleration, and directional changes, each capturing a different desire or interaction effect. For instance, as depicted in figure 1.5, one force corresponds to the acceleration towards the desired velocity of motion; second, repulsive forces reflect the agent keeping a certain distance from other agents and obstacles; third, attractive forces represent the goal and motivations of the agent. Eventually, using standard physics equations, the sum of these dynamic forces describes the agents' motion. Such reactive models are easy to use and efficient for crowd simulations. Interestingly, in crowded or evacuation scenarios, social force models exhibit several so-called "self-organization phenomena" such as lane formation, zipper effect, intermittent flow, or turbulence. Unfortunately, such a model can perform very poorly in intricate and non-crowded

scenarios involving some decision-making. Thus, there was a lack of intelligent simulated agents to challenge effectively HAN system. A few recent works also propose simulating human agents endowed with some reasoning processes, but I did not find the time to compare my contribution with them and it remains an interesting possible future work. Nevertheless, this shows that this is a subject of interest. Some related works will be discussed in Chapter 6.

Part I

Human-Aware Task Planning

Chapter 2

A Human Aware Task Planner Emulating Human Decisions and Actions (HATP/EHDA)

Contents

This chapter familiarizes the reader with the HATP/EHDA task planner. I participated in developing this planner, and it became the keystone of two of my main contributions. Hence, the reader must understand this planner's motivation and methods before considering my contributions. First,

2.1 Introduction

I was introduced to task planning with the work of a PhD student from my lab, Guilhem Buisan. I slightly contributed to the original version and then proposed two extensions of his work. However, Guilhem mainly designed and implemented this novel human-aware task planning approach dedicated to HRI, which plans the robot's actions while estimating and emulating the human decisions and actions, namely HATP/EHDA.

We believe this planning approach suits the needs of HRI scenarios well; thus, it became a laboratory to address relevant challenges of task planning for HRC. Two of my main contributions include addressing such challenges and implementing the solutions as extensions of HATP/EHDA. Consequently, it is essential to understand this work's motivation and methods well before introducing my proper contributions. This section introduces, motivates, and explains the HATP/EHDA approach as a background to the other chapters. A detailed description of this prototypical planner is already given in Buisan's thesis [Buisan 2021]. Thus, large parts of this section are directly retrieved from Buisan's thesis, but they are essential to have in mind. Some notations are adapted to match my contributions' descriptions in the following chapters.

2.2 A Hierarchical Agent-Based Task Planner (HATP)

My work and HATP/EHDA are part of a line of work that started with the Hierarchical Agent-based Task Planner (HATP) [Alili 2009, Lallement 2014].

Based on Hierarchical Task Networks (HTN), this planner can elaborate a multi-agent plan based on a single HTN tree. Moreover, it maintains one belief base per agent, allowing the writer to write task decomposition rules and action preconditions and effects in any agent belief base. This approach produces a joint plan that includes actions from all involved agents.

Finally, to make HATP suitable to HRI scenarios, specific mechanisms to compute costs and filter plans have been included. To this end, HATP allows balancing efficiency, wasted time, agent effort, intricacy, and undesirable states or sequences. This way, social rules can be defined to produce a collaborative plan the human will likely accept.

However, HATP assumes a shared goal has been established between humans and robots before planning. The generated plan will be shared with and accepted by the human before the execution. Indeed, HATP does not represent the human as an agent having a separate decision process that may lead to diverging plans without robot communication. Hence, any human deviation from their generated stream of action needs supervision to perform repair action or request a replanning. Such an approach can work well but assumes that communication can easily be done at any point of the plan. Since this isn't always the case, or simply because communication can be costly, any human deviation, even due to inattention, will put the robot in a failure state which needs to be fixed before continuing (replanning).

2.3 Rationales of a new approach HATP/EHDA

The HATP approach produces what is estimated to be the best joint plan for solving the task. Hence, this approach assumes that the human is likely to accept and follow the produced plan. This leaves no room for online human decisions and assumes a fully committed human, through a prior defined shared goal. This

approach also consider one shared task representation and knowledge base. To cater with the limiations of HATP, I participated in the development of a new approach, HATP/EHDA, which try to satisfy several objectives:

1. **Plan without assuming a prior shared goal.** In HRI scenarios, the robot and the human do not always share a goal. The robot can, for example, plan to perform a task around humans that are not involved at first, or it may be requested by a human to do a task without wanting to take part in it. HATP/EHDA can balance between integrating the sharing of a goal with a human (assumed to be collaborative) in the plan and making the robot do the task alone or integrating the eventuality to ask for punctual human help.

2. **Model the human decision processes.** When taking part in a task, a human (assumed willing to collaborate with the robot) will also plan to reach their (potentially shared) goal. HATP/EHDA must be able to account for this to provide plans that are expected and explainable by the human partner.

3. **Help the human decisions, but not compel them.** Unlike HATP, HATP/EHDA should account for human decision-making flexibility. While modeling the human decision processes, it is possible to narrow down the possible human actions, and the generated plans must help the supervision (execution of the plan) avoid replanning or repairing during the execution by considering several human actions.

4. **Model the potential human reactions.** It is possible to predict that the human may react to some situations, interrupting or helping their current task. Two causes have been identified for these reactions. First, they can result from specific world states humans perceive and interpret. Then, they can also originate from explicit communications issued by the robot. These communications can either be a belief alignment, updating the human knowledge and impacting their decisions, a request to perform a specific action, or a request to help the robot with a shared goal, needing the human to plan for it.

5. **Act and decide on the different agents' beliefs.** It is crucial to be able to represent actions as having different effects on the beliefs of the robot or the human. Indeed, some robot actions are partially or not observable by humans; humans cannot know the complete new world state when performing them. Besides, these effects and their observability often depend on the current world state, whose representation must be supported by the planner. Then, planning decisions may require reasoning on both the robot's and human's beliefs. This is especially true with communication actions aiming to align knowledge or ask questions. Finally, some actions of pure decision have no direct effect on the world but only on the internal beliefs of the agents. For example, observation actions will only update the beliefs of the agent doing it.

6. **Decide not only on the world state but also on the decision processes of the agents.** Some decisions made during the planning process require access to the agents' beliefs representing the world state and the estimation of their planning processes. For example, the decomposition of a task by the robot may be impossible if some other task is already performed in its partial plan. Other decisions may also need the estimation of the current human planning process. For example, if it were estimated earlier in the plan that the human would perform a specific task decomposition, the planner would assign a complementary task to the robot.

7. **Adapt to the human experience, trust, and preferences.** We also want the planning process to be adjusted depending on the actual human it is planning with. It must perform its planned search differently, whether the human has the habit of performing this particular task with the robot or not. Moreover, the human model can be adjusted to the human's trust in the robot and to their preferences.

TODO: explicitly give new things compared to HATP

2.4 Related work

TODO: List and develop some human-aware / HRC task planners, regroup them per "categories"

Robot actions must be planned according to several factors, try to find a citation?. All these can be linked to explainable AI this it justifies the robot behavior from the human's perspective.

1. Achieving the goal: classical planning already achieves this efficiently. the robot behavior should be efficient.

2. Human actions: like in any collaboration, the robot should anticipate and plan its own actions according to the expected human behavior. Robot behavior should be accommodating.

3. Human preferences: On top of aiming for the goal, the human can, and usually, have some preferences regarding the task resolution. The robot satisfy and allow to satisfy these preferences.

4. Human uncontrollability: The robot should not compel the human decision and should thus be adaptive to what the human does.

5. Social norms/rules: The human comfort must be satisfied when collaborating, to that end, the robot should follow (implicit) social rules leading to an appropriate, legible robot behavior.

Some works more focused on the adaptive execution, using online planning.

Some on explainability 2.4.0.2

Some on supporting human. Here the robot proactively help the human to accomplish a task but the robot doesn't have a proper task. Often closer to execution because the robot is adaptive and reactive to the human to support them 2.4.0.1 2.4.0.6

Some service robot but on long term support 2.4.0.8. Human activity leads to constrainst on the robot behavior like don't make noise while the human is sleeping

2.4.0.1 Hoffman - 2007 - Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team

Proposes to anticipate human actions to provide efficient robot support. The human is modeled as a First-Order Markov Process and could even be modeled as a Hidden Markov Model. The probabilities constituting the human model are used in the cost evaluation of different plans, eventually leading to a robot action selection producing a robot plan.

2.4.0.2 Chakraborti - 2017 - Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy

They propose to improve the explainability of a robot plan by using both a robot model $\mathcal{M}^{\mathcal{R}}$ and the estimation of the model the human has of it $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. This approach, called model reconciliation, aims to make identical the optimal plans generated using both models, i.e., $\mathcal{M}^{\mathcal{R}}$ and $\mathcal{M}^{\mathcal{R}}_{\mathcal{H}}$. They define a list of operators to modify the different models until the plans match.

Although this approach interestingly improves the explainability of the robot's plan, it does not consider the two agents to collaborate directly. Indeed, the plan produced only contains robot actions.

2.4.0.3 Iocchi (Sanelli et al.) - 2017 - Short-Term Human-Robot Interaction through Conditional Planning and Execution

This work is two-sided. First, they propose a conditional planning system considering uncertainties, primarily due to human agents. This planning approach is based on planner Contingent-FF (cite Hoffman and Brafman 2005). Then, they propose a component to translate these plans into Robust Petri-Net Plan (PNP) to handle their execution. This translation is inspired by work (Iocchi 2016), which improves plans with execution rules. Eventually, this plan are executed with an existing module called PNPRos [cite?].

Interestingly, the expected human actions are transformed into sub-petri nets where the robot elicits the action (e.g., via verbal communication) if the human does not perform it by themselves.

However, this approach does not model either reason on a distinct human agent model. Hence, the human reasoning process, goal, and belief are not modeled. This means that the interaction is limited to requesting the human to perform single

actions without setting a proper high-level joint goal. This can be efficient in some situations, like a service robot providing information and requesting answers to some questions. However, repeated punctual interventions to collaboratively perform a longer task can become unpleasant for the human.

2.4.0.4 Scheutz (Buckingham et al.) - 2020 - Robot Planning with Mental Models of Co-present Humans

This planning approach proposes a unified scheme to cope with collaborative, adversarial, and non-involved human agents. This scheme considers given mental models for each human co-present with the robot, which might interact with the latter. These mental models are queried to estimate a set of actions each agent is likely to perform given a state. The robot's actions are planned according to these estimated actions to reach the robot's goal and potentially achieve the human agents' goal. In doing so, the robot's actions can influence the human ones without explicit communication, helping the robot achieve its goal.

This mental model querying approach is similar to the HATP/EHDA approach. Similarly, it uses an AND/OR graph where OR nodes represent possible robot actions in a state and AND nodes represent the possible human actions estimated by the models. However, despite saying the model can be generic, they use a basic breadth-first search planner to produce a set of minimal-cost plans solving the estimated human goal. Then, they extract the first actions of each plan to produce a set of actions the human is likely to perform. No details are given on the cost evaluation. This approach is likely to no consider sub-optimal human actions, but still relevant, that can be due to inattention. Additionally, the robot actions are selected by using a Min-Max approach on the AND/OR graph, minimizing the worst-case. This approach works well in adversarial setups, and still allow cooperative human interventions. However, it is unclear how these cooperative actions are taken into account in the robot decision process. Assuming that the human will be adversarial, the robot might make "wrong" decisions, possibly preventing an efficient and optimal cooperative execution.

In contrast, HATP/EHDA uses HTN-based mental models, which require expert knowledge to be created, but they predict human behaviors in a more flexible and task-oriented way. Moreover, when selecting the robot actions, HATP/EHDA minimizes the average cost of all possible human decisions, optimizing uniformly for any human decisions. This approach considers humans as congruent, rational, and cooperative but not necessarily involved in the robot's task. Hence, HATP/EHDA doesn't account for adversarial human.

2.4.0.5 Unhelkar / Shah - 2020 - Decision-Making for Bidirectional Communication in Sequential Human-Robot Collaborative Tasks

They propose an approach based on Partially Observable Markov Decision Process (POMDP) called CommPlan. The POMDP is built using a user defined

MDP (Markov Decision Process) representing the collaborative task and an Agent Markov Model (AMM) representing the human decision-making process. Solving this POMDP produces a robot policy which decides when the robot has to communicate about its beliefs, when to question the human about theirs, and when to ask the human to perform an action. Besides, the AMM is not only specified by expert modeler but also refined during the interaction via learning.

However, the approach considers the human model as an oracle on which reasoning is hardly possible **To develop**

2.4.0.6 Darvish - 2021 - A Hierarchical Architecture for Human-Robot Cooperation Processes

In [Darvish 2021], they propose a hierarchical human-robot cooperation architecture called FlexHRC+ designed to provide collaborative robots with an extended degree of autonomy when supporting human operators in high-variability shop-floor tasks. This architecture is organized into three levels: perception, representation, and action. They use hierarchical AND/OR graphs where arcs are sub AND/OR graphs to reduce the complexity of task descriptions, especially when including repetitive subsequences of action (like mounting four table legs). This representation is close to the HTN one used in HATP/EHDA.

While this interesting architecture seems to provide a robust, adaptive, and sometimes proactive collaboration, it is based on online planning, meaning their planning horizon is limited and optimality is not guaranteed.

Additionally, this architecture could complement the HATP/EHDA approach by providing a robust online support to the offline conditional plan produced.

2.4.0.7 2017 - A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes

In [Johannsmeier 2016], the authors propose a task allocation framework for human-robot collaborative assembly line tasks. They propose representing the task through an AND/OR graph and solving the optimal sub-task allocation problem considering a given cost function. Action sequences are extracted from this allocation and then shared with the respective agents to be executed. Hierarchical and concurrent hybrid state machines handle the execution of these sequences. This copes with unpredictable events likely to happen in dynamic and partially known environments, especially in the presence of humans.

Like any industrial task, assembly line tasks allow using a few assumptions about humans. Indeed, in this context, the human is working and thus should be fully committed and focused on the task. Hence, even if humans appreciate flexibility, their priority in this context is task efficiency. Thus, humans will be more likely to accept propositions from robots than in a less demanding context like a household one. Moreover, humans will likely be distracted or change their minds in household

scenarios. Thus, like HATP/EHDA, it is relevant to plan and anticipate several possible human decisions instead of an optimal one.

2.4.0.8 A constraint-based approach for proactive, context-aware human support

more about supporting human

2.4.0.9 A Human Decision-Making Behavior Model for Human-Robot Interaction in Multi-Robot Systems

dd

2.4.0.10 A Human-Aware Method to Plan Complex Cooperative and Autonomous Tasks using Behavior Trees

The paper proposes a novel human-aware method for generating robot plans in industrial environments, allowing the robot to adapt its plan to the human partner and optimize decision-making based on human actions. The method models task-related costs considering distances from the human and the robot to the task, human ergonomics, and task duration. It enables the robot to plan online and obtain different behaviors based on user choice, considering different levels of engagement between robots and humans. The proposed solution demonstrates high potential in increasing robot reactivity and flexibility while optimizing decision-making based on human actions.

2.4.0.11 Improved Task Planning through Failure Anticipation in Human-Robot Collaboration

2.4.0.12 Human-Robot Collaboration using Monte Carlo Tree Search

2.5 Running example

To highlight the potential of our approach, we present a cube-stacking scene as an example. The scene is depicted in four different scenarios in 2.1. The goal consists of stacking the colored cubes on the empty marks to match the colors on the left of the figure. All cubes placed in the middle of the table are reachable from anywhere. However, when close to one side, a cube is only reachable from this specific side. Notice that one of the required red cubes is located on the opposite side of the table and cannot be grabed by the robot in the initial state.

A first human, called $H1$, requests the robot to stack the colored cubes to match the goal pattern given. In the first case (a), $H1$ set a shared collaboration goal by requesting the robot to stack the cubes together with them. However, $H1$ can also just give their request to the robot and leave. In such a case, the robot has three possibilities. First, in (b), the robot can solve the task alone but must move to the other side of the table, which is slow and costly. Secondly, in (c), the robot can ask

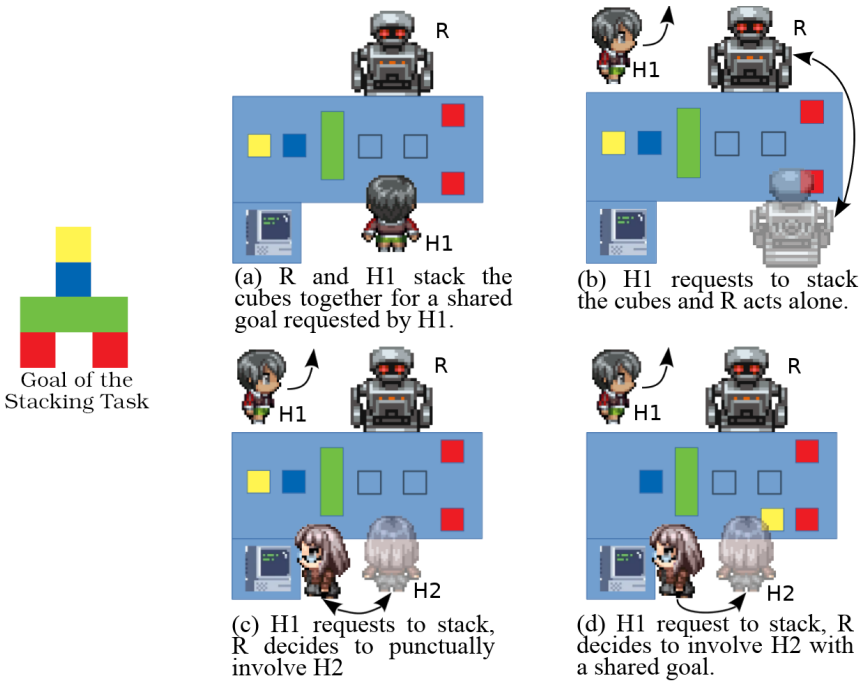


Figure 2.1: Cube stacking scene: A different plan is selected for each scenario, involving nearby humans in the less disturbing way possible

another nearby human, $H2$, for punctual help with the unreachable red cube. $H2$'s reaction can be to put the red cube directly in the stack or only make it reachable to the robot. Finally, the robot can set a shared goal by requesting $H2$ to help it build the stack together.

The HATP/EHDA approach explores and evaluates all these kinds of scenarios to produce the robot policy.

2.6 Problem Specification and Solution

2.6.1 Problem Specification

The notations from Buisan's thesis have been adapted to match the notations in this thesis and ease readers' comprehension. We start from classical planning formalization.

We consider a *classical planning domain* (*state-transition system*) $\Sigma = (S, A, \gamma)$, s.t., S is a finite set of states in which the system may be, A is a finite set of actions that the actors may perform, $\gamma : S \times A \rightarrow S$ is a state-transition function. Each state $s \in S$ is a description of the properties of various objects in the planner's environment [Ghallab 2016].

To represent the objects and their properties, we will use two sets B and X: B is a set of names for all the objects, plus any mathematical constants representing the properties of those objects. X is a set of syntactic terms called state variables,

s.t. the value of each $x \in X$ depends solely on the state s.

A *state-variable* over B is a syntactic term $x = sv(b_1, ..., b_k)$, where sv is a symbol called the state variable's name, and each b_i is a member of B and a parameter of x. Each state variable x has a range, $Range(x) \subseteq B$, which is the set of all possible values for x.

Here is the description of the sets B and X for the stacking example given in the introduction:

$B = Entities \cup Locations \cup TableZones \cup Booleans \cup \{\mathsf{nil}\}$
$Entities = Agents \cup Cubes$
$Agents = \{\mathsf{R}, \mathsf{H}\}$ \\ $\mathsf{R} : robot,\ \mathsf{H} : human$
$Cubes = \{\mathsf{red1}, \mathsf{red2}, \mathsf{green1}, \mathsf{blue1}, \mathsf{yellow1}\}$
$Locations = \{\mathsf{base1}, \mathsf{base2}, \mathsf{bridge}, \mathsf{top1}, \mathsf{top2}\}$
$TableZones = \{\mathsf{middle}, \mathsf{side\_h}, \mathsf{side\_r}\}$
$Booleans = \{\mathsf{true}, \mathsf{false}\}$

$X = \{at(e), holding(a), solution(l) \mid e \in Entities, a \in Agents, l \in StackLocations\}$
$Range(holding(a) \mid a \in Agents) = Cubes \cup \{\mathsf{nil}\}$
$Range(at(a) \mid a \in Agents) = \{\mathsf{side\_h}, \mathsf{side\_r}\}$
$Range(at(c) \mid c \in Cubes) = TableZones \cup Locations$
$Range(solution(l) \mid l \in StackLocations) = Cubes$

TODO: Make proper definitions? Here to much lost in the text... and easier in next chapter to say "we redefine def.x"

A *variable value assignment* function over X is a function val that maps each $x_i \in X$ into a value $z_i \in Range(x_i)$. With $X = \{x_1, ..., x_n\}$, this function can be written as a set of assertions: $val = \{x_1 = z_1, \ldots, x_n = z_n\}$.

An *action* is a tuple $\alpha = (head(\alpha), pre(\alpha), \mathit{eff}(\alpha))$ where $head(\alpha)$ is a syntactic expression of the form $act(z_1, ..., z_k)$ where act is a symbol called the *action name* and $z_1, ..., z_k$ are variables called parameters. $pre(\alpha) = \{p_1, ..., p_m\}$ is a set of preconditions, each of which is a literal. And $\mathit{eff}(\alpha) = \{e_1, ..., e_n\}$ is a set of effects, each of which is an expression of the form: $sv(t_1, ..., t_j) \leftarrow t_0$ with t_0 being the value to assign to the state variable $sv(t_1, ..., t_j)$. We note $agt(\alpha)$ the agent performing the action α.

The problem specification of HATP/EHDA is a pair of two distinct human and robot models as follows: $P = \{\mathcal{M}^H, \mathcal{M}^R\}$. Each model $\mathcal{M}^\varphi$, for an agent φ, comprises the following:

- **Name** ($name^\varphi$): being the name of the agent. Hence, either "robot" or "human".
- **Beliefs** (val^φ): estimation of the world state from the agent's perspective.

- **Agenda** (d^{φ}): capturing the personal and/or shared goals of the agent, currently implemented as a task list/sequence but could be generalized to partially ordered task networks.

- **Partial Plan** (π^{φ}): storing the current partial plan on an agent. This is empty at the beginning and filled during the planning process.

- **Action Model** (Λ^{φ}): encoding the capabilities of the agent and used to estimate the next actions of the agent given a goal and a world state. Here described by a hierarchical task network (HTN) and thus a set of operators and methods.

- **Triggers** (Tr^{φ}): describes the reactions the agent may have which might update their agenda. That is, the agent may react to a specific world state, event sequence, or an explicit communication. For instance, consider a scenario where suddenly another agent is handing over an object to the agent. This event has nothing to do with the agent's goal, and thus, the next agent action extracted from the Action Model might not consider the other agent. However, a natural reaction to this situation is to grab the handed object. Thanks to the Triggers mechanic, we can in an automated way estimate that whatever the agent was doing, when being handed an object the agent will grab it.

Note that most of the models' elements are static during the planning process. Only the Beliefs (val^{φ}), the Agenda (d^{φ}) and the Partial Plan (π^{φ}) of each agent evolve during the planning process. That is why we define specifically an **agent state** which is part of the agent model: $\sigma^{\varphi} = \{val^{\varphi}, d^{\varphi}, \pi^{\varphi}\}$. And thus, the agent model can be written as $\mathcal{M}^{\varphi} = \{name^{\varphi}, \sigma^{\varphi}, \Lambda^{\varphi}, Tr^{\varphi}\}$.

The planner uses two agent models, one for the human and one for the robot. Despite their identical structure, there is a fundamental difference between the two models: one is a controllable agent and not the other. Indeed, the human model is only used to speculate the human decisions and actions in given situations. Then, the robot model is used to plan the robot's actions according to the estimated human actions. Note that human decisions can still be influenced by the robot's actions, but they cannot be compelled. It's also important to remember that the two agents are not equivalent, the robot agent's role is to help, assist and facilitate humans, and thus, to synthesize pertinent, legible and acceptable behavior.

A *state*, or *planning state*, $s_i \in S$ is a tuple composed of two agent states $s_i = (\sigma_i^H, \sigma_i^R)$. The state of the world from the perspective of the robot is captured by the variable value assignment function $val_i^R \in \sigma_i^R$. Since the planner is assumed to be part of the robot, the beliefs of the robot are assumed to be the ground truth, and thus, are sometimes noted as val_i. Similarly, $val_i^H \in \sigma_i^H$ represents the estimation of val_i from the perspective of the human, also referred to as the estimated human beliefs. We say that a state $s_i \in S$ contains *false beliefs*, or *belief divergences*, if $\exists x_j \in X, val_i^H(x_j) \neq val_i^R(x_j)$. Be careful not to get confused, a

state is a state in which the planning problem is while being solved, and each *state* is connected to other states through agent actions. Where the *beliefs* are the state of the world from the perspective of an agent, which is part of the *planning state*. The other elements in the two agent states, namely the agendas and partial plans, indicate respectively what each agent still has to do and which actions they already performed.

For our example, the initial state s_0 would be as follows:

$$
\begin{aligned}
s_0 &= \{\sigma_0^R, \sigma_0^H\} \\
&\quad \sigma_0^R = \{val_0^R, d_0^R, \pi_0^R\}, \quad \sigma_0^H = \{val_0^H, d_0^H, \pi_0^H\} \\
&\quad d_0^R = \{Stack\}, \quad d_0^H = \{\} \\
&\quad \pi_0^R = \pi_0^H = \{\} \\
&\quad val_0^R = val_0^H = \{at(\mathsf{R}) = \mathsf{side\_r},\ at(\mathsf{H}) = \mathsf{side\_h},\ at(\mathsf{red1}) = \mathsf{side\_r}, \\
&\qquad at(\mathsf{red2}) = \mathsf{side\_h},\ at(c) = \mathsf{middle} \mid c \in Cubes \backslash \{\mathsf{red1}, \mathsf{red2}\}, \\
&\qquad holding(\mathsf{R}) = holding(\mathsf{R}) = \mathsf{nil}, \\
&\qquad solution(\mathsf{base1}) = \mathsf{red},\ solution(\mathsf{base2} = \mathsf{red}),\ solution(\mathsf{bridge} = \mathsf{green}), \\
&\qquad solution(\mathsf{top1}) = \mathsf{blue},\ solution(\mathsf{top2} = \mathsf{yellow})\}
\end{aligned}
$$

Note that d_0^H is empty except in scenario (a) where H1 establishes a shared goal. Also, for legibility reasons, the sets B and X have been slightly modified because in fact the cubes are explicitly associated with colors which are used in the task decompositions. Hence, the solution is simply expressed using the colors and not the cube names. Finding the exact cube to place is described in the action models.

The process of estimating the next actions that an agent $\varphi \in Agents$ is likely to perform in a state $s_i \in S$ will be detailed later. Meanwhile, when refining the agent's agenda d^φ based on its belief val_i^φ, we obtain a *refinement* as follows $ref(d_i^\varphi, val_i^\varphi) = \{(a_1, d_1), ..., (a_j, d_j)\}$. A *refinement* contains a tuple for each estimated possible action a_j and the associated new agenda d_j after being refined.

In our cooking example, we obtain the following refinement if the starting agent is the human:
$ref(d_0^H, val_0^H) = \{(add\_salt(), d_1), (move\_to(\mathsf{kitchen}), d_2)\}$

The execution of an action in HATP/EHDA is known by both agents, and thus, for both beliefs we have $\forall x \in X$:

$$
val_{i+1}(x) = \begin{cases} w, & \text{if } x \leftarrow w \in eff(a) \\ val_i(x), & \text{otherwise} \end{cases} \tag{2.1}
$$

2.6.2 Solution

The solution produced by HATP/EHDA is a robot policy mapping each state preceded by a human action to an optimal robot action. It assumes a turn-taking

fashion where agents act one after another. If the robot is starting, a dummy human action is inserted into the plan. In practice, it produces a tree of sequential actions where every human action is succeeded by one optimal robot action. Additionally, whenever it is estimated that the human can perform several actions, a branch is created for possible action, ensuring to cover all possible human choices. Each branch is a possible course of alternating robot and human actions leading to the goal.

TODO: Insert figure for policy: same as 2.2 bottom left.

2.7 Exploration

To start planning, HATP/EHDA must be given the two action models (the robot and the human HTNs), the initial beliefs of both agents (which can differ) and the initial agenda of both agents. The initial agenda of the robot represents the task to decompose, while the agenda of the human represents any task the human is estimated to be committed to. If a shared goal has been established prior to planning between the robot and the human (*e.g.* the human asking to perform a task with the robot), the agenda of both agents will be filled with the same task.

The planning process is done in three parts: (1) First, both HTNs are explored in a turn-taking fashion, resulting in a valid joint plans tree. (2) Then, based on this tree, robot actions are selected according to action, plan-wide and social costs, resulting in a conditional plan, where at each step multiple human actions can be performed but only one robot action is set. (3) Finally, causal and threat links are added between actions of the conditional plan to ease its execution.

The robot HTN exploration is a pretty standard depth-first algorithm. The first task λ from its agenda d^R is popped, then if it is an abstract task $\lambda \in Ab$, all the applicable methods are applied, and their results are prepended to the agenda, thus giving new agents state (with the same beliefs as the previous ones but with the robot agenda updated) and branching our search space. We recursively iterate with the new task popped from the new robot agenda. Eventually, the popped task will be a primitive one $\lambda \in Op$, its function will then be applied to the currently explored agent states. If it returns *false* ($\perp$), the action is not applicable, and the exploration backtracks to another decomposition of an abstract task. However, if the action is applicable (returns a new agents state), the action is added to the robot plan and the triggers are run for each agent, updating their agenda if necessary. Then, the human HTN is queried to get their possible next actions from this new agents state. The possible actions found are added to the human plan, and, for each possible new agents state, we apply the triggers of each agent then we continue the robot HTN exploration. This exploration continues until the robot agenda is empty, or all the branches return *false*. The HTNs exploration process is summarized on Fig. 2.2.

The human HTN exploration differs from classical HTN planner as the goal is not to produce a complete plan, but rather to list all the actions the human is likely to perform in a given agents state. We recursively decompose the first task of

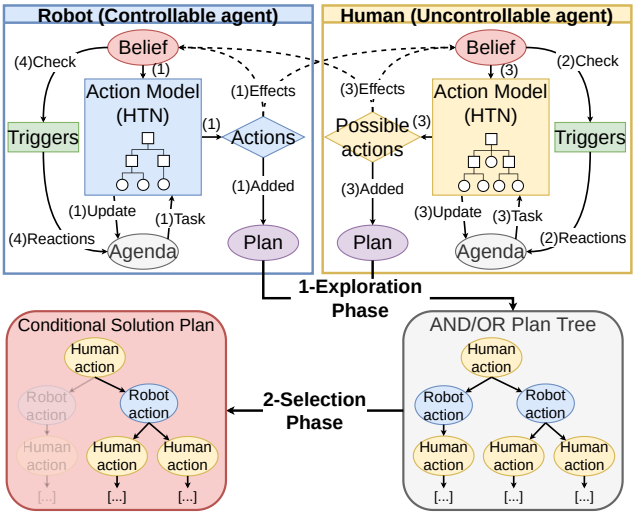


Figure 2.2: The HTNs exploration consists in iterative loops of four steps : (1) Get possible robot actions from the robot HTN, add them to the plan and apply their specific effects on the H & R beliefs, (2) Check Triggers and add the reactions in the corresponding agendas, (3) Get possible human actions based on their estimated beliefs, add them in the plan and apply their effects on the H & R beliefs, (4) Check Triggers again and add the reactions in the corresponding agendas. Here the robot is starting, but the human could with the following ordering: (3)>(4)>(1)>(2).

the human agenda d^H with every applicable method, until we reach an applicable operator. All the operators from all the applicable decompositions are returned to the robot HTN exploration and applied.

Two special cases are handled during the exploration. If the human agenda is empty whereas the robot one is not, the exploration returns a default action *IDLE* for the human (which does not modify agents beliefs nor agendas). This action represents the non-involvement of the human in a task. Besides, if no applicable action is found for the human a default action *WAIT* is returned (which does not modify agents nor agendas). This action represents the impossibility of the human to act in the current situation, making them wait for the robot to proceed. This default action can also be used in a domain to represent the human decision to wait for the robot to act.

Once the robot agenda is emptied, the agents state is set as a success, the plan is added to the valid plans tree and the search can be continued until no decomposition is left for any task.

2.8 Plan Evaluation and Selection

Talk about cost in human-aware task planning. A trade-off between efficiency and social criteria. The robot should be efficient but also behave in an acceptable, legible, and accommodating manner.

Cost evaluation is tricky, objective metrics are easy to use for efficiency, however,

social criteria are harder to compute because they are hard to generalize. These social rules can be very context dependent which make them hard to generalize and thus to take them into account in a reliable manner.

Plan cost is a mix of all the following:

- length of the plan (nb of action or temporal duration if available)
- sum of individual action cost: the cost of each action can be estimated to translate the effort required to perform it. It can reflect several aspect and constraints such as: pure physical strength required, duration of the action, energy consumption
- Undesired states: Common sense and social norms can be used to define several rules defining undesired state. This can cover various aspects such as hygiene or safety. For instance, despite being possible and maybe efficient, we wouldn't like a robot holding a dirty dripping mop in one hand and the sandwich we asked for in the other hand. Another example, we wouldn't like a robot dropping a knife just on the edge of a table or counter because it may fall and be dangerous.
- Undesired sequences of actions: For the same reasons, we can also define undesired sequences of actions. This can express preferences regarding the ordering of different subtasks, e.g., since we don't want the robot to hold our sandwich while cleaning the house, we would also not like the robot to clean first and then make a sandwich because the robot is likely to be dirty while making the sandwich.

I implemented in HATP/EHDA a way to specify and take into account undesired state and action sequences. Detecting any of them in a possible plan would penalize the plan cost of the specified amount. Here, the undesired elements must be specified in the problem specification and are abstracted in the planner. However, as stated above, it is hard to generalize undesired states and action sequences to integrate them directly in the planner to avoid specifying them in the problem specification.

In HATP/EHDA, once the exhaustive exploration has been done, the result is a valid plans tree of alternating robot and human feasible actions along with their current beliefs leading to a task completion. The goal of this second planning step is to select robot actions such as each human action in the plan has only one robot action as a child. To do so, we define a cost function $cost : \sigma \times Op \mapsto \mathbb{R}^+$ representing the cost of an action in a specific state. The data structure is now similar to a two players game tree. However, *MinMax* approaches are not suitable here, as we are not in an adversarial setup but more into a collaborative one. Indeed, trying to minimize the maximal possible cost is assuming that the human will always do the actions leading to the worst plan. This defensive behavior could lead to non optimal plans. We thus propose to explore this tree differently.

Moreover, like in HATP we allow to define *social costs* functions. These functions take a complete human and robot sequence of actions (π^R and π^H) and return

(a) R and H1 build the stack together as a shared goal requested by H1.	**(b) H1 requests to stack the cubes and R acts alone.**
R-PickAndPlace(red, base) H1-PickAndPlace(red, base) R-PickAndPlace(green, bridge) H1-PickAndPlace(blue, top) R-PickAndPlace(yellow, top)	R-PickAndPlace(red, base) R-moveTo(red) R-PickAndPlace(red, base) R-moveTo(init) R-PickAndPlace(green, bridge) R-PickAndPlace(blue, top) R-PickAndPlace(yellow, top)
(c) H1 requests R to build the stack, R decides to punctually involve H2.	**(d) H1 requests R to build the stack, R decides to invite H2 to a shared goal.**
R-PickAndPlace(red, base) H2-IDLE R-AskPunctualHelp(red) H2-PickAndPlace(red, base) R-PickAndPlace(green, bridge) H2-IDLE R-PickAndPlace(blue, top) H2-IDLE R-PickAndPlace(yellow, top)	R-PickAndPlace(red, base) H2-IDLE R-AskSharedGoal() H2-PickAndPlace(red, base) R-PickAndPlace(green, bridge) H2-PickAndPlace(blue, top) R-PickAndPlace(yellow, top)

Table 2.1: Execution trace of a selected plan for each scenario.

a cost ($\mathbb{R}^+$) which is added to the cost of the plan previously determined. By doing so, we can penalize non acceptable sequence of robot actions (*e.g.* serving a meal just after taking out the trash) or non satisfactory human required contribution (*e.g.* the robot requesting the human to perform small tasks multiple times instead of giving the big picture of the real task to perform).

The approach we propose for plan selection is to minimize the average cost. It represents the human potentially selecting any course of actions in their stream (while still respecting the action model defined in their HTN). The algorithm is given the root action of the task network previously generated and returns the cost of the conditional plan selected while having selected the robot actions in the task network minimizing the average cost of plans.

2.9 Example

The partial robot and human actions models, as well as their exploration are presented in Fig. 2.3.

Each scenario is commented below with their corresponding selected plan shown in TABLE 2.1.

(a) H1 and R act together First, the human sets a shared goal by asking the robot to stack cubes with him. Since it is a shared goal, Both human and robot agendas are initialized with the “Stack" task. Thus, the robot anticipates that the

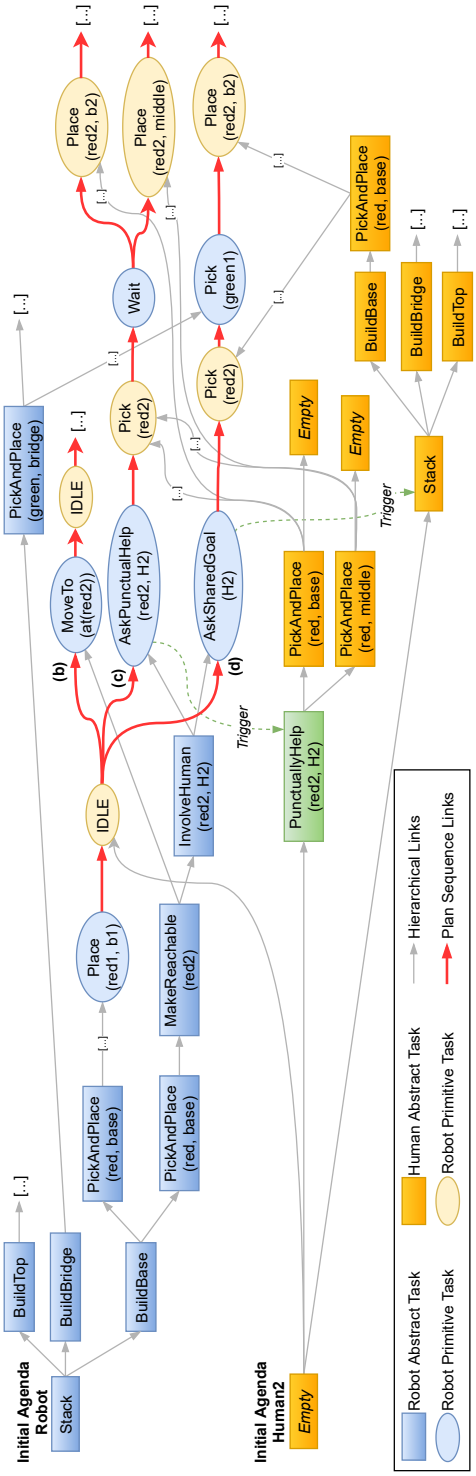


Figure 2.3: Illustration of the incremental exploration of various courses of actions corresponding to scenarios depicted in Fig. 2.1(b), (c) and (d). Since H1 requests the robot to achieve the goal, it is not a shared goal, the robot agenda is filled with the task to achieve and the agenda of H2 starts empty.

human will pick the unreachable second red cube by querying the human action model. The selected plan to collaboratively stack can be seen in TABLE 2.1(a).

(b) Robot acts alone This time the human asks the robot to stack the cubes but then leaves the scene, the robot must act alone. Hence, the only applicable method to make the second red cube reachable is to move to the other side even though the movement action is expensive (we can imagine a table way longer than shown on the figure).

(c) Robot asks punctual help The first human acts the same way but another one is present, not involved in the task. The robot starts exploring its HTN and, thanks to the presence of the other human, a new method is applicable allowing the robot to ask for help. It can either ask for punctual help or involve completely the other human in the task. Of course, just asking help for one cube is less costly than asking to build the whole stack together. However, asking help to someone not already involved in a common task is still expensive since they have to put themselves in the context of the task. Yet, this punctual help is actually less costly for the robot than moving to the other side so the robot chooses to ask for punctual help. Note that we model the fact that after being asked to punctually help, the human can either stack the cube herself or just make it reachable to the robot by placing it in the middle. Only the first branch is shown in TABLE 2.1(c) but the selected plan is in fact conditional with two branches as depicted in Fig. 2.3.

(d) Robot invites Human2 to share a goal Same initial setup but now two cubes are out of reach. Asking for punctual help is still less costly than moving around the table. Nevertheless, this disturbs the human so each new punctual request becomes more expensive than before. Thus, this time setting a shared goal becomes less costly for the robot than asking twice for punctual help.

2.10 Conclusion

what is offered by HATP/EHDA is very interesting, we reason on the human model to plan the robot actions while never compelling the human actions. The plan produced assumes Turn-Taking, thus, no parallel execution of the actions. This isn't a strict constraint as a post-analysis can reason on the causal links of the actions in the plan to extract a partially ordered plan to execute, making the execution more flexible. It is however still a limitation that is addressed in Chapter 4.

Chapter 3

Models and Algorithms for human-aware task planning with integrated theory of mind

Contents

3.1 Introduction

False beliefs tasks are commonly used as tests to acknowledge the presence of Theory of Mind reasoning. The most common test is the *Sally and Anne* one, depicted in fig 3.1, and is used in developmental psychology to examine children's "theory of mind" understanding, which refers to their ability to understand how other people think, feel and behave. The test consist in describing the execution of a simple task involving non-observable objects and co-presence, then children are questioned

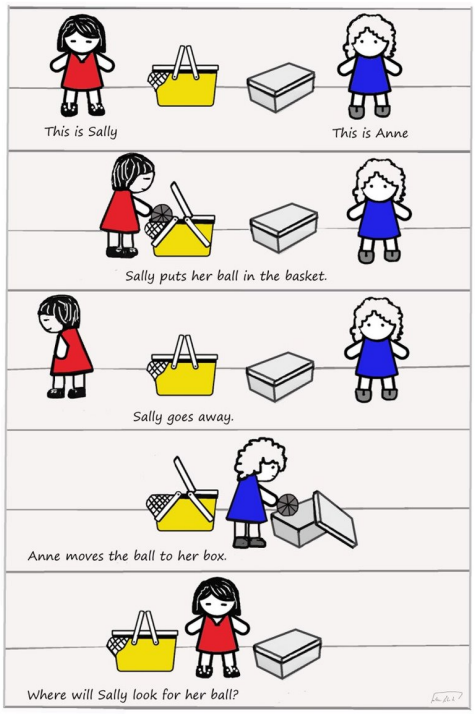


Figure 3.1: Sally and Anne Task

about the belief of one of the character. The task consist in the following. Sally and Anne are two co-present characters near a basket and a box. Sally puts her ball in the basket before going away. Then, Anne moves the ball from the basket to the box, in hindsight of Sally. Eventually, the question is where will Sally look for the ball she left? As an observer of this scene, using Theory of Mind, we can naturally say that Sally will look for her ball in basket because she isn't aware that Anne moved the ball in the box. Very young children are not able to understand that Sally isn't aware of Anne's action and thus, they are likely to answer that Sally will look in the box.

Theory of Mind reasoning is a very important skill for *good* interaction and collaboration between agents. Thus, it is reasonable and desirable to aim at endowing robots with such skill to enhance their interactions with humans. This is the motivation of the contribution presented in this chapter. Here, we propose an extension of HATP/EHDA to be able to maintain in a principled way the human beliefs during the planning process, and to tackle estimated false human beliefs that may be detrimental to the task resolution.

Some works already consider ToM during execution, but this isn't always enough. In some situations it is mandatory to consider ToM beforehand while planning in order for the robot to be proactive during execution, and not only reactive.

We would want the robot to be able to reason and maintain correctly the distinct

human beliefs. Despite modeling distinct beliefs, HATP/EHDA doesn't maintain in a principled way, only in a scripted way (domain specific). Here we propose some models and algorithms to integrate some concept of Theory of Mind in the planning process of HATP/EHDA. This way, the robot can estimate more accurately the human's belief to predict their behavior. Moreover, we propose solutions for the robot when to tackle estimated false human beliefs that may impact the task resolution.

3.2 Related works

This chapter's contribution is related to several topics not mentioned yet. Hence, to better capture the contribution, this section introduces the new topics and relevant related works.

3.2.1 Epistemic planning

Epistemic planning plays a crucial role in human-robot collaboration. Thomas Bolander is one of the main contributors of this field and describe it as follows in [Bolander 2017]. Epistemic Planning is the enrichment of planning with epistemic notions, that is, knowledge and beliefs. The human or robot might have to reason about epistemic aspects such as: Do I know at which post office the parcel is? If not, who would be relevant to ask? Maybe the parcel is a birthday present for my daughter, and I want to ensure that she doesn't get to know, and have to plan my actions accordingly (make sure she doesn't see me with the parcel). The epistemic notions are usually formalized using epistemic logic. Epistemic planning can naturally be en as the combination of automated planning with epistemic logic, relying on ideas, concepts and solutions from both areas.

It enables robots to make plans to achieve the required knowledge and to reason about the knowledge and capabilities of other agents, ensuring effective collaboration and coordination in human-robot interactions [Belle 2023].

Bolander et al. proposed the Dynamic Epistemic Logic (DEL) approach [Bolander 2017] and even implemented some Theory of Mind concepts in it. However, despite being more expressive than HATP/EHDA, like the latter, this approach still doesn't maintain the human beliefs in a principled way. They tend to use domain-specific rules given in the probleme specification, usually with conditional action effects.

Muise et al. worked on multi-agent epistemic planning using a classical planning approach. Since involving nested beliefs is computationally demanding, their work proposes to convert and encode such problems into classical planning problems. Hence, state-of-the-art classical planning techniques can be used to tackle nested beliefs of multiple agents problems.

3.2.2 Theory of Mind in HRC

Epistemic planning helps to plan the correct sequence of actions to perform to reach a desired knowledge, including a desired world state. However, estimating the current knowledge and beliefs of the different agents is challenging. To do so, we have to consider Theory of Mind concepts, especially perspective shift and the notion of co-presence. Some epistemic planning works already integrate of ToM notions, but the subject is worth being discussed a bit more in details. Indeed, in the HRI field, ToM is used in various domains like navigation, dialogue and like here in task planning.

Theory of mind (ToM) refers to the ability to attribute mental states to oneself and others, such as beliefs, desires, and intentions.

TODO: rephrase, less listing of references... Use comment above Robots endowed with ToM abilities are more effective in proactive robotic assistance and are perceived as more socially intelligent by humans [Shvo 2022]. ToM enables robots to infer human desires, beliefs, and intentions, allowing for natural interaction between robots and humans [Yu 2023]. Robots with ToM can anticipate human strategies and incorporate them into their decision models, leading to better team performance [Romeo 2022]. The presence of ToM in robots influences human decision-making behavior and trust, making it more appropriate for human-robot collaboration [Erdogan 2022]. Computational theory of mind, based on abstractions of beliefs into higher-level concepts, facilitates collaboration on decisions and improves the quality of human decisions [Gurney 2022]. However, the lack of a unified construct and consistent benchmarking hinders progress in endowing robots with ToM capabilities.

3.2.3 Communication in HRC

Communication enables effective interaction between humans and robots, promoting inclusivity and reducing obstacles in human-robot interaction. Communication allows robots to share information about their actions and intentions, enhancing transparency and explainability [McMillan 2023]. It helps in establishing trust and understanding between humans and robots, leading to improved teamwork and performance [Verhagen 2022]. Non-verbal gestures and behavior of robots during collaboration can impact the perception of the robot and influence the willingness of humans to cooperate [Arntz 2022]. Communication also allows robots to assess their own skills and limitations, propose alternatives, and adapt the execution of tasks to the capabilities of the collaborators [Ferrari 2022]. Overall, effective communication facilitates mutual knowledge, enables the exchange of information, and allows humans and robots to work together efficiently and successfully.

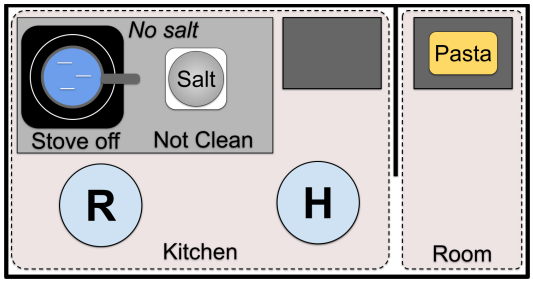


Figure 3.2: Let us consider cooking pasta as a human-robot shared task. The robot has to turn on the stove (*stoveOn*) and clean the counter (*counterClean*), but the latter is not a part of the shared task. The human takes care of fetching the pasta while both agents can add salt into the water (*saltIn*). Before pouring the pasta into the pot the human must know the facts, *stoveOn* and *saltIn*. Unlike *stoveOn*, the facts *saltIn* and *counterClean* are not directly observable. Hence, by acting while the human is away to fetch the pasta, the robot may induce false beliefs which may be detrimental to the shared task (e.g., human adding salt again).

3.3 Maintaining the human beliefs

3.3.1 Enhanced problem specification

We start from the HATP/EHDA probleme specification which is the one described in Chapter 2. For our collaborative cooking example, the sets B and X for the collaborative cooking example are the following:

$$
\begin{aligned}
B &= \mathit{Entities} \cup \mathit{Places} \cup \mathit{Booleans} \cup \{\mathsf{nil}\} \\
&\quad \mathit{Entities} = \mathit{Agents} \cup \mathit{Objects} \\
&\quad \mathit{Agents} = \{\mathsf{R}, \mathsf{H}\} \quad \backslash\backslash\ \mathsf{R} : \mathit{robot},\ \mathsf{H} : \mathit{human} \\
&\quad \mathit{Objects} = \{\mathsf{salt}, \mathsf{pasta}, \mathsf{counter}\} \\
&\quad \mathit{Places} = \{\mathsf{kitchen}, \mathsf{room}\} \\
&\quad \mathit{Booleans} = \{\mathsf{true}, \mathsf{false}\}
\end{aligned}
$$

$$
\begin{aligned}
X &= \{\mathit{at}(e) \mid e \in \mathit{Entities},\ \mathit{saltIn},\ \mathit{stoveOn},\ \mathit{counterClean}\} \\
&\quad \mathit{Range}(\mathit{saltIn} \mid \mathit{stoveOn} \mid \mathit{counterClean}) = \mathit{Booleans} \\
&\quad \mathit{Range}(\mathit{at}(\mathsf{R} \mid \mathsf{H} \mid \mathsf{pasta})) = \mathit{Places} \\
&\quad \mathit{Range}(\mathit{at}(\mathsf{salt} \mid \mathsf{counter})) = \{\mathsf{kitchen}\}
\end{aligned}
$$

Our first contribution starts from here where we augmented the specification by associating each state variable to a location and an observability type. Thus, in addition to the *variable value assignment* function we define two other functions.

First, a *variable observability assignment* function over X is a function *obs* that maps each $x_i \in X$ into an observability type $t_i \in \{\texttt{OBS}, \texttt{INF}\}$: $\mathit{obs} = \{(x_1, t_1), \ldots, (x_n, t_n)\}$. Respectively, when $\mathit{obs}(x_i) = \texttt{OBS}|\texttt{INF}$ then x_i is said to be *observable* | *inferable* in the state s_i.

Then, a *variable location assignment* function over X is a function loc that maps each $x_i \in X$ into a $l_i \in Places \cup \{\texttt{nil}\}$: $loc = \{(x_1, l_1), ..., (x_n, l_n)\}$. $Places \subseteq B$ captures a group of constant symbols such that each member is a predefined area in the environment. Agents are always either "situated" in a place or moving between two places. We consider x_i to be located in every $place \in Places$ if $loc(x_i) = \texttt{nil}$. More details about how the environment should be divided into places will be given shortly when discussing the beliefs updates.

We update the definition of a *state*, or *planning state*, to consider the two functions introduced above s.t. $s_i = \{\sigma_i^R, \sigma_i^H, obs_i, loc_i\}$. This definition still consider the two agent states s.t. $\sigma_i^\varphi = \{val_i^\varphi, d_i^\varphi, \pi_i^\varphi\}$. Now, we keep track in each state of the observability type and location of each state variable, and we can reason on them to update the human beliefs (val^H) accordingly.

We remind that a state $s_i \in S$ contains *false beliefs*, or *belief divergences*, if $\exists x_j \in X, val_i^H(x_j) \neq val_i^R(x_j)$.

The initial state of our cooking example can be described as follows. There is no initial belief diverge, the initial partial plans are empty, and both agents but perform the shared cooking task named $CookPasta$. In addition, the robot must clean the kitchen counter once the cooking is done, hence, the task $CleanCounter$ is added in the initial robot agenda after the shared task. The only *non-obversable* facts are the presence of salt in the pot and if the counter is clean. All entities, including the counter, the stove and the salt, are located in the kitchen, only the pasta are in the adjacent room. More precisely, the initial state s_0 can be written like this:

$$
\begin{aligned}
s_0 &= \{\sigma_0^R, \sigma_0^H,\ obs_0,\ loc_0\} \\
&\quad \pi_0^R = \pi_0^H = () \\
&\quad d_0^R = (CookPasta, CleanCounter) \\
&\quad d_0^H = (CookPasta) \\
&\quad val_0^R = val_0^H = \{at(\mathsf{R}) = at(\mathsf{H}) = \mathsf{kitchen},\ at(\mathsf{pasta}) = \mathsf{room}, \\
&\qquad\qquad saltIn = \mathsf{false},\ stoveOn = \mathsf{false},\ counterClean = \mathsf{false}\} \\
&\quad obs_0 = \{(saltIn, \texttt{INF}),\ (counterClean, \texttt{INF}), \\
&\qquad\qquad (stoveOn, \texttt{OBS}),\ (at(e), \texttt{OBS}) \mid e \in Entities)\} \\
&\quad loc_0 = \{(saltIn, \mathsf{kitchen}),\ (counterClean, \mathsf{kitchen}),\ (stoveOn, \mathsf{kitchen}), \} \\
&\qquad\qquad (at(e), val^0(at(e))) \mid e \in Entities\}
\end{aligned}
$$

3.3.2 State Transitions and Beliefs Updates

We now describe how the agents' beliefs are updated when executing a planned action, and thus, how the transition occurs from one state to another. The key principle is that the human agent learns from observing either an action execution or their environment. We first give the 3 assumptions we made in this approach.

Assumption 1: We do not consider uncertainties. Thus, agents are either wrong or right about the state of the world but never uncertain. This would be an interesting future work.

Assumption 2: We do not consider cases where the robot's beliefs can diverge. Since

the planner is part of the robot, the actual ground truth is unknown. We can only assume that the estimation of the state of the world by the robot is correct and reason on it.

Assumption 3: Coming from the two previous assumptions, we assume that the human only makes deterministic moves when not being observed. Hence, regardless of being co-present, the robot's beliefs are always updated with the action's effects.

In our cooking example, to estimate which actions the human is likely to perform first we have to refine the human agenda. We obtain the following two possible actions, associated to two updated agendas:

$$ref(d_0^H, val_0^H) = \{a_1, d_1, a_2, d_2\} = \{(add\_salt(), d_1), (move\_to(\mathsf{kitchen}), d_2)\}$$

Thus, Assumption 3 indicates that $\forall x \in X$ we always have,

$$val_{i+1}(x) = \begin{cases} w, & \text{if } x \leftarrow w \in \mathit{eff}(a) \\ val_i(x), & \text{otherwise} \end{cases} \tag{3.1}$$

The place associated with a state variable can be modified by the action's effect like an agent moving to another room while holding an object. Currently, we assume in this work that the observability type of each fact is constant during the task. However, the overall approach could be adapted to make the observability types dynamic and this will be discussed later in this Chapter. So, $\forall x \in X$,

$$obs_{i+1}(x) = obs_i(x) \tag{3.2}$$

$$loc_{i+1}(x) = \begin{cases} l, & \text{if } x \leftarrow l \in \mathit{eff}(a) \\ loc_i(x), & \text{otherwise} \end{cases} \tag{3.3}$$

The new agenda of each agent (d_{i+1}^R, d_{i+1}^H) are created by the HTN refinement algorithm, and thus, they are directly retrieved from the obtained refinement. This refinement decomposes abstract tasks in the agenda until the first task is a primitive action. To do so, every applicable method is applied leading to a set of possible actions (and refined task networks).

The new estimated human belief val_{i+1}^H is the two-step result of our Situation Assessment processes that models the human's real-time sensing and reasoning capabilities about their surroundings.

First, let us define the notions of *co-presence* and *co-location* which will be key to maintaining the evolution of agents' beliefs as planning progresses.

Definition 1 ***(Co-presence & Co-location.)*** *In a state $s_i \in S$, two agents, φ_1 and φ_2, are considered to be co-present if $val_i(at(\varphi_1)) = val_i(at(\varphi_2))$. This relation is noted $\varphi_1 \curlywedge_i \varphi_2$ in the rest of the paper. Similarly, we say that an agent φ_1 is co-located with a state variable $x \in X$ if $val_i(at(\varphi_1)) = loc_i(x)$, noted $\varphi_1 \curlywedge_i x$.*

Now we can define two SA processes that will maintain the estimated human beliefs.

Definition 2 ***(Inference Process.)*** *An agent observes the execution of an action by being either co-present with the acting agent, or by being the acting agent. If so, the agent infers the new values of every state variable present in the action's effects.*

Based on the above definition, the human's beliefs are updated as follows when action a is executed in state s_i,

$$val'^{H}_{i+1}(x) = \begin{cases} w, & \text{if } x \leftarrow w \in \mathit{eff}(a) \text{ and} \\ & (H = \mathit{agt}(a) \text{ or } H \curlywedge_i \mathit{agt}(a) \\ & \text{ or } H \curlywedge_{i+1} \mathit{agt}(a)) \\ val^{H}_{i}(x), & \text{otherwise} \end{cases} \tag{3.4}$$

To change its *place* in the environment, agents would use a dedicated *"move"* action, such that its effect only updates the agent's location.

Definition 3 ***(Observation Process.)*** *An agent observes its surroundings and assesses the exact value of each state variable located in the same place (i.e., each state variable the agent is co-located with).*

After applying the effects of an action to obtain val_{i+1} and the human beliefs val'^{H}_{i+1} (using the inference process), the observation process is executed. It updates again the estimated human beliefs with the facts currently observable by the human and provides fully updated human beliefs to store in the state s_{i+1}, $\forall x \in X$:

$$val^{H}_{i+1}(x) = \begin{cases} val_{i+1}(x), & \text{if } H \curlywedge_{i+1} x \text{ and} \\ & \mathit{obs}_{i+1}(x) = \texttt{OBS} \\ val'^{H}_{i+1}(x), & \text{otherwise} \end{cases} \tag{3.5}$$

Note that before starting the planning process, the observation process is executed once on the initial state s_0. This allows us to potentially correct the estimated human beliefs with the facts the human should initially be able to observe.

The definition of the set *Places*, i.e. how the environment is divided into different *places*, is guided by the shape of our state transition function. Hence, a *place* $\in$ *Places* is an area in the environment such that, when situated in it, agents are aware of each other's activity and they can assess every observable fact located in it.

Note that unlike in DEL [Bolander 2021], our knowledge representation is simple and prevents us from expressing agents being *uncertain* about a fact. In line with the classical closed-world assumptions, agents either know the truth or have a false belief w.r.t. the ground truth. We consider a straightforward scenario in which the human is *"unaware"* of non-observed changes in the environment. This results in estimated false human beliefs, helping to detect whether a non-observed robot action can disrupt a seamless collaboration.

3.4 Relevant False human beliefs

In this section, we explain our procedure to detect *when* a false human belief should be corrected and *how*.

3.4.1 Detection

The human and the robot carry individual distinct beliefs, while the two can be aligned, or diverging when the human has a false belief. To produce a legal solution plan the robot is fine with such false human beliefs unless they are qualified as *relevant* (Definition 4). In such cases, the relevant false belief needs to be tackled.

Definition 4 *A* ***relevant false belief*** *is a false belief that influences the next action(s) the human is likely to perform, either in terms of number, name, parameters, or effects.*

This can be written as follows: A state s_i contains a relevant false belief if either (3.6) or (3.7) is true:

$$ref(tn_i^H, val_i^H) \neq ref(tn_i^H, val_i^R) \tag{3.6}$$

$$\{\gamma(s_i, a) \mid \forall a \in ref(tn_i^H, val_i^H)\} \neq \{\gamma(s_i, a) \mid \forall a \in ref(tn_i^H, val_i^R)\} \tag{3.7}$$

We consider that as soon as a false belief has an effect on human actions it should be tackled. An interesting future work could be to check in a principled way the overall positive and detrimental impacts of this false belief on collaboration. But it is out of the scope of this work.

3.4.2 Resolution with minimal communication

A state containing a false human belief marked as *relevant* must be handled. The first way to do it is by planning communication actions such that the robot communicates only the required facts to the human. This allows to correct false human beliefs that are relevant, but false beliefs that are *"non-relevant"* will remain.

3.4.2.1 Modeling Communication Actions

We propose a generic communication action schema (ca) in this context. An agent φ_i can *communicate* an assertion $x = z$ (with $x \in X$ and $z \in \text{Range}(x)$) *via* the action $ca_{\varphi_i,\varphi_j}(x, z)$ if $val^{\varphi_i}(x) = z$ and $val^{\varphi_j}(x) \neq z$. The effect of $ca_{\varphi_i,\varphi_j}(x, z)$ corresponds to $val^{\varphi_j}(x) \leftarrow z$. Such actions are considered equally costly and instantaneous.

3.4.2.2 Communicate Only the Required Facts

Definition 4 indicates if there is at least one diverging state variable in the human beliefs causing adverse effects, but without identifying which one(s). Hence, we explain a subroutine below with the three steps, describing how we first identify the pertinent state variables to align, and then how the corresponding communication actions are created and inserted into the robot's plan.

1. *Store* each state variable whose value differs in the human beliefs from the robot beliefs: $X_{diff} = \{x \mid x \in X, val_i^H(x) \neq val_i^R(x)\}$.
2. *Build*, for each stored state variable $x \in X_{diff}$, a communication action $ca_{R,H}(x, val_i^R(x))$, all stored in a set CA_{diff}.
3. *(Breadth-First Search.)* The *source* is s_i. Applying each $ca \in CA_{diff}$ generates a new state by aligning *exactly* one state variable in the human beliefs s.t. $s_i' = \gamma(s_i, ca)$. The search continues until the first state s_i' selected to expand doesn't contain a relevant false belief. The communication actions used from the root until this selected state are *retrieved* in a set CA.

Once the above subroutine finishes, the retrieved communication actions in the set $CA = \{ca_{R,H}(x_1, val_i^R(x_1)), ..., ca_{R,H}(x_j, val_i^R(x_j))\}$ must be inserted in the plan for belief alignment. Thus, Definition **?? TODO: from Chapter, probably best to recall the definition? or not?** is redefined to be sound w.r.t. our approach. An edge can now either be a human action o^h or a robot action o^r with a set of communication action CA. At each step, humans perform *Observation*, while the robot executes each communication action $ca \in CA$, making the human's belief to *update instantaneously.*

The set CA is inserted before the diverging human actions and after the closest state where agents are co-present. But it could be interesting to reason with a better plan evaluation system to find the best place to insert this set.

3.4.3 Resolution by delaying non-observed robot action

So far we relied on communication, but depending on the environment (e.g. noisy), communication can be cognitively demanding. Thus, when the relevant false belief is due to a non-observed robot action, we propose to also consider implicit communication by postponing the pertinent robot action until the human is estimated to be observing its execution. This prevents false beliefs from even occurring.

First, a branch using communication is explored and the state variables concerned by the relevant false beliefs are retrieved (through all $ca \in CA$). Then we check if the divergence is produced by a non-observed action. For now, it is done by checking if the relevant divergence concerns only one inferable state variable and if it was not present in the initial state. After, we identify which action creates the divergence by sequential regressing the current branch/trace. Hence, we can identify when the relevant divergence appears and which action should be delayed. Once identified, we create another branch in the plan just before the identified action. In this new branch, DELAY actions are inserted in the robot's plan until the human is co-present. When the human is co-present again, the identified action is inserted and observed by the human. Then the nominal planning process is resumed.

3.5 Result

Referring to the related work section, we are not aware of an implemented planning system that can be used as a baseline. Hence, we use the HATP/EHDA solver to help present our approach's results on three *novel* planning domains.

3.5.0.1 Cooking Pasta Domain

The running example corresponds to a specific problem in this domain. In fact, agents and pasta can initially either be in the kitchen or in the adjacent room, the stove might be on or off and there might be salt or not in the water. In the results, we will focus on the following three state variables from X. Both *stoveOn* (OBS) and *saltIn* (INF) are relevant to the human, unlike *Clean* (INF) which only concerns the robot.

3.5.0.2 Preparing Box Domain

A box with a sticker on it and filled with a fixed number of balls is considered prepared and needs to be sent. Both agents can *fill* the box with balls from a bucket, while only the robot can *paste* a sticker and only the human can *send* the box. The bucket can run out of balls, so when one ball is left, the human *moves* to another room to *grab* more balls and *refill* it. The number of balls in the box is *inferable*, while all other variables are *observable*. In the following, three boxes have been considered.

3.5.0.3 Car Maintenance Domain

The washer fluid (OBS) and engine oil (INF) levels have to be *full* before *storing* the oil gallon in the cabinet (INF). Only the robot can *refill* both the tanks and store the gallon

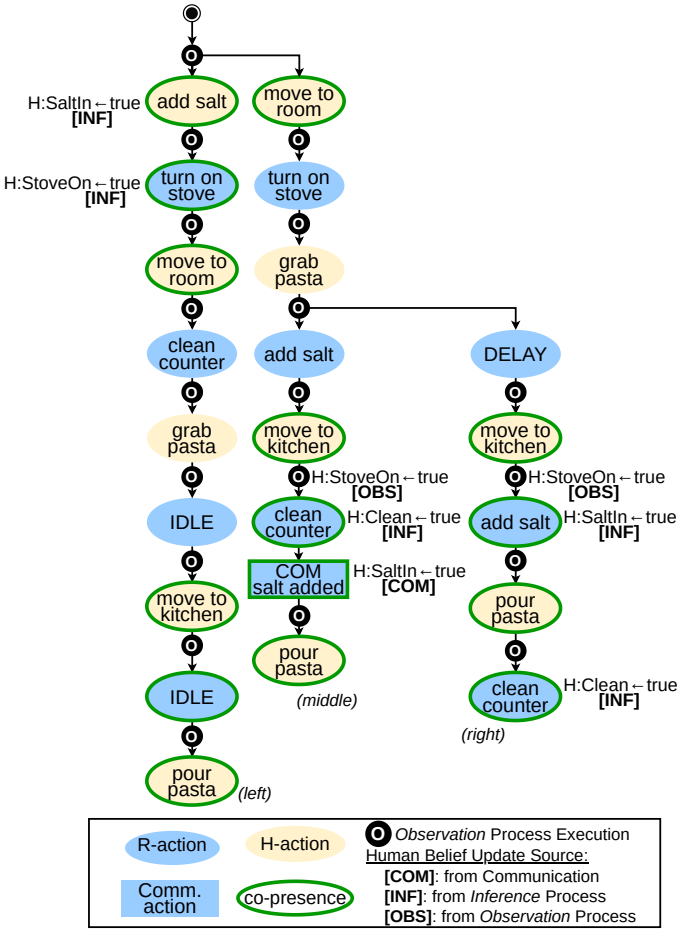


Figure 3.3: Plan obtained for the cooking scenario. 3 branches. Left: The human starts by adding salt. The only false belief is about *"counterClean"* which is not relevant for the human agent, hence no comm is added. Middle: While the human is away the robot turns on the stove and adds salt, creating 2 false beliefs. Once back, we estimate that the human agent will be able to assess the observable fact *"stoveOn"* but not *"saltIn"*. Since the human agent might add salt again due to this false belief, it is relevant and fixed with a communication action. Right: The relevant false belief about *"saltIn"* is avoided by delaying the robot's action until the human is co-present.

while situated at *Front* of the car. *Front-left* and *Front-right* headlights have to be *checked* and a light-bulb has to be *replaced* at *Rear*. Only the human can check and replace lights, and they can start with either of these two tasks. Both agents start at *Front*. The car's hood needs to be *closed* by the human at last.

3.5.1 Qualitative Analysis

Considering the cooking domain, we discuss in detail the plans obtained with our approach to a problem corresponding to the description given in the introduction. I.e., there is no initial human false belief, agents both start in the kitchen, the pasta is in the adjacent

Table 3.1: Success and communication ratio of different approaches.

Domain	**HATP/EHDA**		**Only Comm**	**With Delay**
	S	*S I.Div.B.*	*Comm*	*Comm*
Cooking	18.6%	6.9%	69.5%	65.2%
Box	25.0%	14.3%	79.7%	75.0%
Car	12.5%	0.0%	68.8%	64.1%
Average	18.7%	7.1%	72.6%	68.1%

room, the stove is off, and there is no salt in the water. The resulting plans are shown in Fig. 3.3 and their detailed presentation explains how the approach works in practice. Since human is uncontrollable and has different possible actions, the plan branches and the robot's actions are different in each case.

In (*left*) the human first adds salt and then the robot turns on the stove. In both cases, thanks to the inference process, we estimate that the human will be aware of both facts about the salt (*acting*) and the stove (*co-present*). Then while the human is away to fetch the pasta, the robot cleans the counter and since the human isn't co-present their beliefs aren't updated, containing now a false belief. Once back, since *counterClean* is not *observable* the observation process does nothing and the false belief remains. However, this false belief doesn't affect human actions (non-relevant), hence, there is no need to align human beliefs.

In (*middle* and *right*) the human first fetches the pasta by leaving the kitchen. Let's first focus on the (*middle*) trace. The robot turns on the stove and adds salt while the human is away, creating two false beliefs. When returning to the kitchen, the observation process updates the human beliefs with the observable facts located in the kitchen. This fixes the false belief about *stoveOn*. The robot then cleans the counter, observed by the human. However, without communication, the human's next action will be either "add salt" or "ask the robot", but considering the ground truth the human could directly pour the pasta. Hence, the false belief on *saltIn* is relevant and has to be corrected. To do so a communication is inserted in the robot's plan and a "delay" branch is created (*right*). In this delaying branch, the robot delays the add salt action until the human is co-present in order to make it observed (inference process) by the agent. In addition to this implicit communication, like in (*middle*), the human assesses that the stove is on and hence can directly pour the pasta.

3.5.2 Experimental Results and Analysis

In each domain, the actions and tasks remain the same. So here, a problem is defined by a starting agent (R or H) and a pair of initial beliefs (val_0^R, val_0^H). Initial ground truth ($val_0 \Leftrightarrow val_0^R$) is defined by setting each state variable to an initial value. But, 5 selected state variables can be set to 2 possible values instead of 1. Among these selected ones, 3 can diverge in human belief. This generates 256 pairs of initial beliefs where 12.5% of them include initially aligned beliefs. Then, considering the starting agent, we obtain 512 problems for each domain. Each of the 1536 generated problems has been solved by HATP/EHDA, by *our approach* using first *only communication* and then using also *delay*. The obtained quantitative results appear in TABLE 3.1.

The overall success rate (S) and the one for initially diverging beliefs ($SI.Div.B.$) are shown for the HATP/EHDA solver. As expected, this solver always finds legal plans when dealing with initially aligned beliefs, but the low value of $SI.Div.B.$ reflects how poorly

it handles belief divergences without specifically designed action models. Our approach always finds legal plans so we omitted its success rates in the last two columns, and we can say that it solves a broader class of problems.

Furthermore, considering the initially diverging beliefs and the divergences created along the planning process, more than 87.5% of all problems involve belief divergences. However, when using only verbal communication, only 72.6% of the generated plans include communication actions. This means that *our approach* communicates only when necessary, and not systematically. The amount of communication is even reduced to 68.1% when delaying actions. In the latter case, only delayed branches that do not imply the human to wait are kept.

3.6 Discussion and Limitations

obs type constant: For instance, when an agent places an object in an opaque box, the observable object would no longer be observable despite being in the same place as the agent. Places can be symbolic, thus, with the right modeling one can model the disappearance of an object when being placed in a drawer. Making the obs type dynamic requires more thinking on how their are updated, yet, this would only [ref def] has to be modified and the whole scheme would continue to work.

The underlying scheme allows just a single agent to execute a *"real"* action at a time. However, a post-process can allow the execution of actions concurrently [Crosby 2014], however, note that the domain modeler has modeled $\mathcal{P}_{rh}$ as a sequential joint task. Parallelism is not considered in the current modeling and planning process, which limits the potential for concurrent executions. However, we are working on extending the framework to enable systematic planning with concurrent actions, aligning with [Shekhar 2020].

We believe our modeling-level SA proposals could fit in any other planning approach framing multi-party systems having one controllable agent while can only hypothesize remaining agents' behaviors (e.g., human-centered AI).

Agents' SA models cannot simply refute a false belief, they can only assess new true facts to correct them. E.g., assume the human *wrongly* believes that the pasta is in kitchen. The SA does not help refute this when the agent is in kitchen because appropriate knowledge reasoning w.r.t. *NotAt(Pasta)* in kitchen is not taken into account. However, such issues do not affect the completeness and, if necessary, our approach *tackles* such cases as relevant false beliefs.

We have planned a user study for the future to conform our framework with reality and validate the approach.

We discussed earlier that DEL knowledge representation is more expressive and flexible, and can handle uncertainty. However, it requires an augmented action schema to accurately maintain each agent's beliefs. Think of a specification for *"move"* action manually listing all the environmental facts to be observed by an agent for managing their beliefs. In our case, it is implicitly maintained within a state.

We can consider running a set of rules (e.g., *graph-based ontology*) to bring new interesting facts in the state based on a set of known facts. We believe that this aspect opens up new possibilities in the future to integrate human-aware collaborative planning and ontology.

3.7 Epistemic Extension

The planning process proposed in this contribution is part of the epistemic planning field. However, the uncertainties are limited and focused on the human decisions, in a simplistic manner compared to "classical" epistemic planner. Indeed, state-of-the-art epistemic planners consider epistemic states rather than "simple" beliefs and thus are able to keep track of several possible worlds. More precisely, [cite bolander] this work using the DEL approach are able to keep track and reason on several possible worlds for each agent, and can even manage distinguishable worlds from others. Hence, the planner is able to anticipate that an agent can believe in three different possible worlds and tell that at execution time the agent will be able to distinguish if they are in world 1 or not, but they will not be able to tell distinguish between world 2 and 3 (based on observable facts). Such approaches currently heavily rely on conditional action effects and scripting. That is, when an agent enters a room with an action *move*, the *move* conditional effects will update accordingly the agent's possible worlds (the agent sees a box, notice that someone isn't in the room anymore, and so on...). These situation assessment rules must be encoded into the action model used, and more precisely, in the action's effect which isn't convenient and is domain specific. Our contribution proposes generalized situation assessment rules to maintain the beliefs but at the price to have simplified epistemic states.

In some preliminary work we explored an extension of our approach to handle such classical epistemic representation.

* Describe first findings and models *

* Some results ? *

* Discuss limitations: the current design is computationally very expensive and doesn't scale. Work on this aspect are needed and in progress to continue investigating this extension *

3.8 Conclusion

We propose an extension to a Human-Aware Task Planner called HATP/EHDA. The planner plans and implicitly coordinates the robot's actions with all estimated possible human (uncontrollable) behaviors that are then emulated to generate a new state. Our extension and contribution are, first, to integrate a *Situation Assessment* based reasoning system in the planner. This allows for maintaining distinct agents' beliefs based on what they can/should observe. Compared to existing epistemic planners, this simplifies the action descriptions by focusing on their effects on the world, and not how they influence each agent's beliefs. In addition, we propose to detect false human beliefs and tackle only the necessary ones in a principled way. First, we propose minimal and proactive explicit communication. Second, when pertinent, we propose an implicit communication by postponing the non-observed robot action until the human is co-present to observe it.

The relevance of false belief, when to optimally communicate and parallelization are interesting future works, and we aim at conducting a user study to validate the benefits of the proactive robot behavior that our approach permits.

Chapter 4

A Task Planner Making a Robot Compliant to Human Online Decisions and Preferences

Contents

4.1 Introduction

From HRI paper

In the context of HRC for a shared task [Selvaggio 2021], we believe, based on the literature on joint action [Sebanz 2006a, Sebanz 2009, Clodic 2017, Gordon 2023], that the key towards a seamless interaction is, to consider the human as an uncontrollable agent and to be fully and concurrently compliant with them. The human should not be dictated which action they must perform, as in [Roncone 2017, Buisan 2022], and the robot must comply with possible human decisions and actions during execution.

To collaborate with such humans with their (hidden) preferences, one can devise an online planning scheme coupled with a plan executor. However, in order to maintain real-time performance, online planning generally keeps a restricted horizon. Therefore, decisions taken online may lead to a dead end or may not lead to an optimal solution. Offline planning overcomes these issues.

We propose a new *offline* task planner which extends an existing human-aware planning system addressed in [Buisan 2022]. The new planner is designed to take into account a *Model of Execution*, which is in the form of an automaton and mainly inspired by the joint actions schemes. The model captures humans' latitude in their decisions. The planner's output is the robot's behavioral policy, which describes the robot's action in a state such that the action is congruent and compliant with the human's decision in this state and their (estimated) preferences, and that it is also legal to be executed in parallel. Our framework also allows humans to share their (new) preferences at any time during execution while the robot's policy is adapted online to that. In addition, our approach considers social signals to enhance execution by minimizing uncertainties. Both humans and robots issue signals to clarify situations such as performing an action, waiting for the other agent's necessary actions, or indicating a desire to remain passive.

In this chapter, we discuss relevant related work before describing the joint action model of execution that is central to our approach. We then describe the task planning problem and then introduce our novel framework. The following two sections to that, explain how the robot policy is generated by a three-step process: *exploration*, *characterization*, and *generation*. We empirically evaluated our approach in simulation. With a BlocksWorld scenario, we show how our approach can effectively produce a concurrent robot behavior that is compliant with human online decisions and preferences.

4.2 Related works

From HRI paper

There have been a few attempts to cater to concurrent execution, but they deal with explicit time to manage concurrency [Cirillo 2009b, Köckemann 2014]. In [Cirillo 2009a], the robot does not plan actions for humans but forecasts their actions/plans from their activities and bases its own decision on the distribution of possible human plans. Here robots can perform actions concurrently, estimating/managing the completion time of the agents' actions carefully. We can see the human activity recognition part as a form of our *identification (ID)* process of the automaton used. The robot needs such a plan/goal recognition technique to be compliant with the human's decisions. But, unlike ours, they do not consider an explicit shared goal among the agents, and hence humans are not concerned with stuff robots might be interested in during collaboration, e.g., giving signals to be passive. We believe that a shared goal creates a different context in HRC than the robot just being compliant with an estimated human's goals/plans. Moreover, we claim that

dealing with concurrent actions is inevitable in planning even if actions are instantaneous, to effectively deal with multiple agent systems [Crosby 2014, Shekhar 2020], especially if there is a human operator involved like in our case. We extend HATP/EHDA [Buisan 2022] to demonstrate that.

In another work, both *recognition* and *adaptation* take place simultaneously and comprehensively [Levine 2014]. It deals with action scheduling of an already generated contingent plan comprising human's and robot's actions. It outputs schedules for the robot actions that can execute concurrently but to do that explicit temporal constraints are considered.

Ramachandruni, Kent, and Chernova (2023) [Ramachandruni 2023] propose a communication free human-robot collaborative approach for an adaptive execution of multi-step tasks. In their approach, the robot observes and supports human decisions, actively selecting actions to optimize collaborative task efficiency. Unlike our approach, they introduce an extended collaborative HTN representation with role assignment for planning and state tracking during execution, which is more in line with [Roncone 2017]. In contrast, we employ two distinct HTNs for robot and human capabilities and use an AND/OR tree for exploration and execution tracking. While their online planning may enhance scalability, optimality is not guaranteed. Also, our scheme accommodates both verbal and non-verbal communication, allowing the human to express preferences that update the robot policy online.

4.3 Model of Execution

4.3.1 Rationale

Our task planning approach uses a model of execution to improve the fluency and amenability of HRC. This model is in the form of an execution controller as shown in Figure 4.1, and is based on several key notions and mechanisms borrowed from studies on joint actions [Michael 2016, Kourtis 2014], and adapted to Human-Robot Joint Action [Clodic 2017, Curioni 2019]. The key idea is that co-acting agents co-represent the shared task context and integrate task components of their co-actors into their own task representation [Schmitz 2017, Yamaguchi 2019]. Also, coordination and role distribution rely strongly on reciprocal information flow, e.g., social signals [Curioni 2019], prediction of other's next action [McEllin 2018].

Our proposed execution model is implemented on a robot that co-acts with a human, integrating explicit representation and exploration of the task representations for the robot and for the human. It also identifies precisely how reciprocal information flow is used in task execution (detecting and interpreting human actions, signals produced by the robot while acting, and also when the robot waits for human actions or their signals).

Another essential question is the criteria for choosing the next action, or more globally, how to share the load between the two co-actors. The choice depends on the context and actors' preferences [Gombolay 2015, Strachan 2020, Curioni 2022]. Concerning the case when one actor is a robot, we think it is important to provide a standard default behavior of the robot where the robot does its best to reduce human load but still leaves full latitude to act whenever humans want. Our scheme provides this ability and also allows humans to inform about their preferences at any moment.

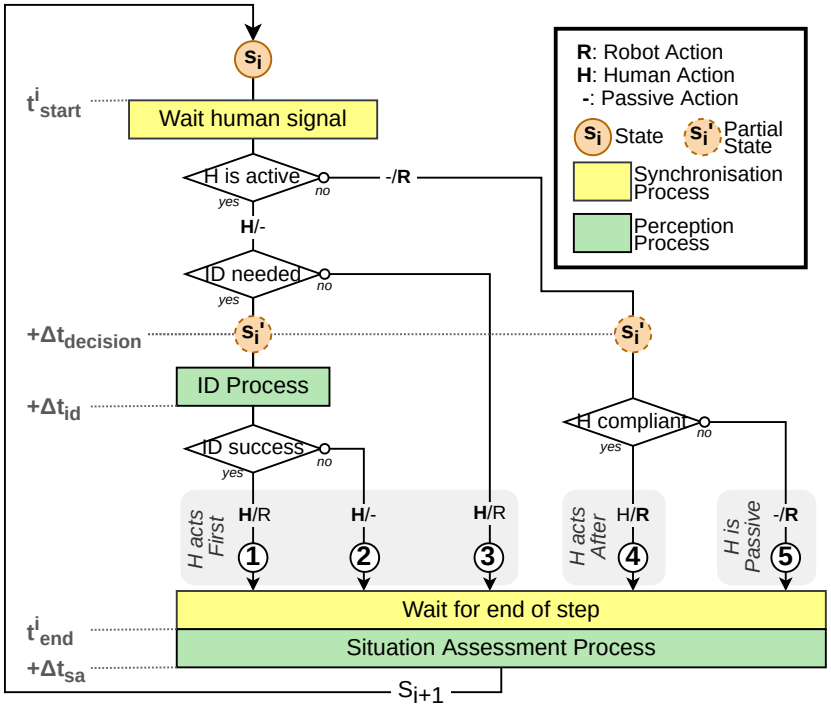


Figure 4.1: Simplified Model of Execution in the form of an automaton run by the robot. It captures the latitude of uncontrollable humans in their actions and guides our task-planning approach. In this paradigm, the two agents can act concurrently but one is always compliant with the other's decision to act. Here, the human is always free to decide whether to start acting first, or after the robot, or not to act at all. To be compliant, the robot attempts to identify human decisions using perception and situation assessment as well as possible collaborative human signaling acts (e.g., gestures or speech).

4.3.2 Simplified Model of Execution - Robot Automaton

Consider an example to clarify the execution automaton. Assume a human and a robot have to pick up two blocks, A and B, that both can reach. They can pick it up both at the same time unless they try to pick up the same block, which causes conflicts between their actions. As a result, despite being executable in parallel, the actions are interdependent, and in order to avoid conflicts, one agent must be compliant with the other. However, if we consider a third block C that only the robot can reach, it can always pick up this block without any risk of conflicts with the human's choice.

In a state, a human decision can result in one of three outcomes. First, the human can choose to act first (*left subtree*). If the robot's best action is not in conflict with the human action (e.g., *pick C*), the robot can safely perform this action concurrently with the human operator (*branch 3*). However, if the robot's best action is either *pick A* or *pick B*, the human action must be identified first with a subroutine in order to be compliant with it. If this subroutine is successful the robot can perform any action which is congruent with the identified human action (*branch 1*). This includes the robot's choice to be *passive* and let the human act alone. However, if the robot is unable to identify the human action, it must remain passive in order to avoid potential conflicts (*branch 2*). Then, the human can either decide to be *passive* or to act after the robot (*right subtree*). In both cases, the human is *passive* at the beginning, making the robot to start performing alone a feasible action. While the robot is acting, the human is free to remain *passive* until the next step (*branch 5*) or to choose a congruent action to act concurrently (*branch 4*). As a result, the human can always choose to 1) act first, 2) act after the robot, or 3) not act at all. The robot will always be compliant with these online human decisions.

When both agents finish their actions, the step is considered as *"over"*. Then, another subroutine assesses the new world state (s_{i+1}), which is the result of the concurrent actions being executed in the state s_i, before repeating the whole process until the task is solved.

Note that if both agents are passive (the human decides to be passive when the robot cannot act) then the step is repeated.

4.3.3 Complete Model of Execution - Dual Automaton

Now that the idea behind this model of execution has been explained we can present its complete version. Our model supposes that the robot is running an automaton to synchronize with the human, follow the generated policy, and execute actions. This automaton is a bit more complex than the simplified version presented above. Additionally, our model supposes that the human is also following an automaton when collaborating with the robot. The complete model specifies the human automaton and the visual signals exchanged between the agents to synchronize themselves.

4.4 Problem specification + solution

4.4.1 Problem

Belief divergences are out of the scope of this particular work. Hence, for simplicity reasons, we consider the two beliefs (robot and estimated human ones) as always aligned, and they are represented as a unique world state. However, we are convinced that this work could be adapted easily to consider the two distinct beliefs.

The problem is specified as follows. One initial world state, described using state variables. Distinct human and robot action models described with HTNs. Distinct human

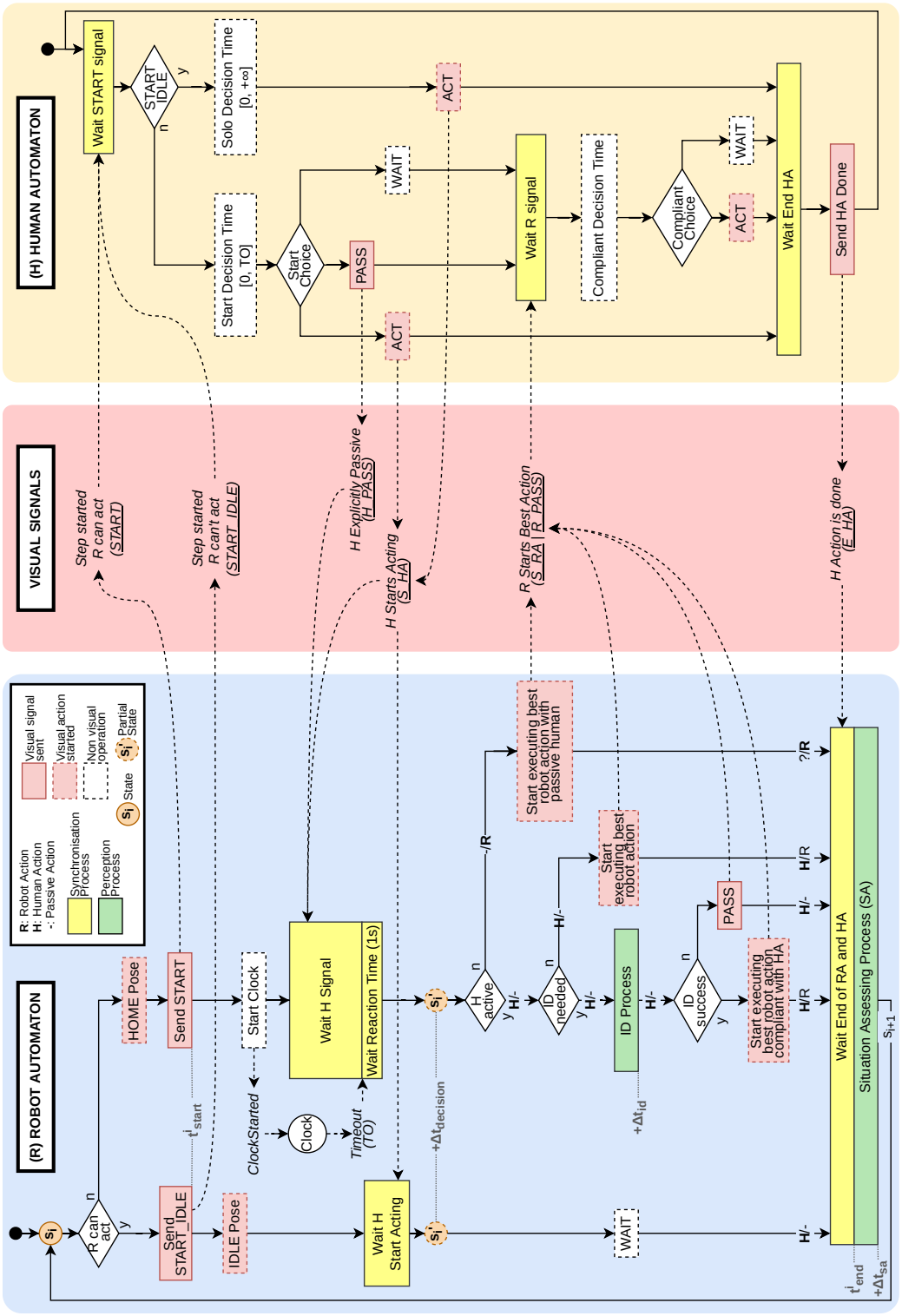


Figure 4.2: Complete Model of Execution - Dual Automaton. This version details the robot automaton and the assumed human automaton as well as the visual signals exchanges between the agents to synchronize themselves.

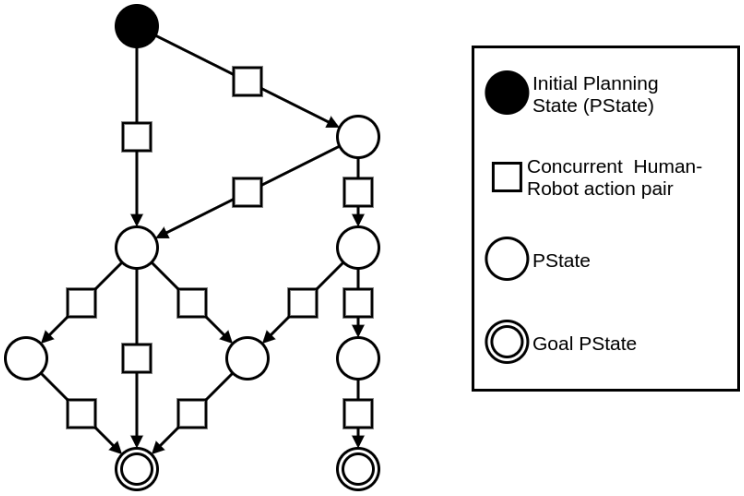


Figure 4.3: Directed acyclic solution graph

and robot initial agendas.

[More details about each]

4.4.2 Solution

A Planning state, referred as p-state, corresponds to a state in which the planning problem is while progressing toward a solution. A p-state contains the current world state (aligned beliefs of the agents) and the respective agenda of the human and the robot. Thus, keep in mind that p-states are very different from world states. The initial p-state is formed using the initial human-robot agendas and the initial world state given in the problem specification. P-states are connected with each other through concurrent pairs of human-robot actions. Note that we consider passive actions, hence, there may be only one active agent in a pair if concurrent human-robot actions. A goal p-state is characterized by a world state satisfying given goal conditions and by empty agendas. The exploration produces a Directed Acyclic Graph (DAG), referred as the search graph, from the initial p-state to several goal p-states through sequences of concurrent action pairs. Thus, any path from the root to a leaf is a possible plan. Once the exploration done, search graph computed, another process extract the optimal robot policy from the graph. In the manner of an AND-OR tree, this policy indicates for all p-states the best concurrent robot action (OR node) to execute to be compliant with any possible human action (AND node).

Clarification of the directed acyclic graph. Nodes represent p-states and are connected with each other through directed edges representing human-robot concurrent action pairs. However, for practical reasons action pairs are considered as nodes of the graph having only one parent p-state node and one child p-state node. Consider that each directed edge between two p-state have one unique intermediate action pair node. The graph has one root node (no parent) which is the initial p-state. All leaf nodes (no children) represent a goal p-state.

It is one of our design choice to do not consider explicit action costs and perform an exhaustive offline search to produce this search graph in order to solve a problem. Since the policy generation is very quick it allows generating and updating the robot policy online according to human feedbacks.

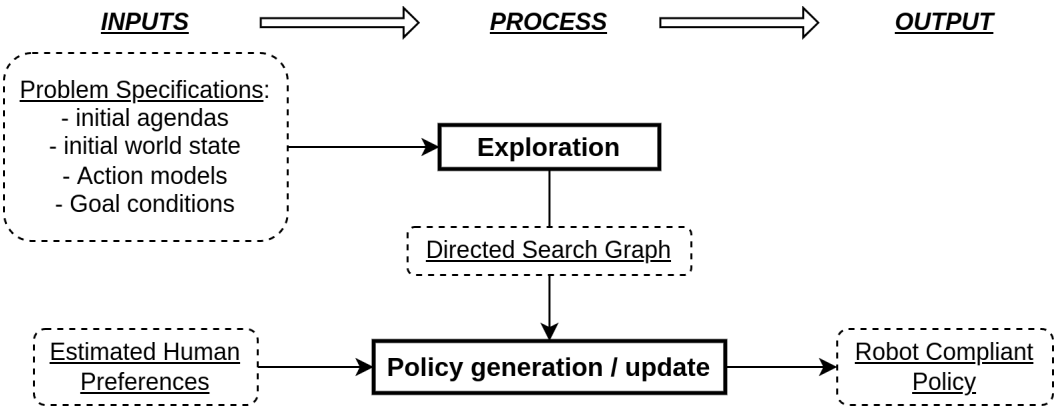


Figure 4.4: Overall planning process

4.5 Exploration

This section details how the exploration happens, and thus, how the search graph is generated. This requires several sub-process which are each detailed here. First, the overall exploration process is presented, and next subsections provide details on the sub-processes mentioned in the overall process.

4.5.1 Overall process

We keep track of the p-state to explore, this set being initialized with the initial p-state. Then, until the set is empty we select one and explore it. First, from this selected p-state, every possible concurrent human-robot action pairs are computed considering both agendas, the world state and reasoning on the compatibility of the actions in terms of preconditions and effects. This process requires several sub-steps and is detailed later. Thus, we obtain several action pairs leading to the same amount of new p-states (with updated world state and agendas). Second, we check if any of the newly created p-state are similar to any existing p-state. If so, we can "merge" them to avoid redundant computations. To do so, we basically keep track of the unique p-state already checked and for each new p-state we check if it is similar to one of the already checked one. If not, the new p-state is added to the set of already checked p-states. If a similar one is found, the new p-state is deleted and the action pair leading to it is connected to the existing p-state instead. Eventually, the remaining new p-states are added to the set of p-states to explore. When the set of p-states to explore is empty then the exploration is over and the graph obtained corresponds to the "search graph" where any path from the initial p-state to a leaf corresponds to a possible execution trace / plan.

4.5.2 Compute next agent actions

From a p-state, more precisely from a world state and an agenda, we can estimate what are the next action an agent is likely to perform. Doing so is referred as the refinement process. To do so we use the corresponding action model in the form of an HTN. Here the agendas are considered as list of abstract or primitive task, where primitive task can be executed (actions), and abstract ones can be decomposed into several others subtasks. The refinement process consists in applying applicable methods of the first task of the agenda until it is primitive. When several methods are applicable they are all applied in a distinct

refinement trace. Eventually, for each different sequence of applied method we obtain a new agenda starting with a primitive task. For each such primitive task we create a copy of the world state contained in the given p-state, and we applied the correspond action, updating the world state. Hence, for each possible action a new p-state is created with the refined agent agenda, the unchanged other's agenda and the updated world state. Note that this process can "generate" default passive actions in several cases. First, when an agent's agenda is empty an *IDLE* passive action is inserted as the first primitive task in the agenda. This means that the agent has nothing to do and thus is likely to remain passive. Either when there are no applicable methods or when the primitive task is not applicable then a *WAIT* passive action is inserted. This means that the agent has still something to do but cannot do it currently. Note that when computing the concurrent pairs of action we also add the *PASS* action which corresponds to the agent being voluntary passive, despite having something to do. Note also that these different passive action types just help to better understand the generated plans, but they are treated equally / similarly in the planning process.

4.5.3 Concurrent action pairs computation

This sub-process computes from a given p-state all possible concurrent action pairs that may be executed. It is based on the previously described refinement process. This main objective of this sub-process is to identify the next actions the agents are likely to perform to reach the goal and identify which of them can be executed in parallel. The classical way to do such reasoning is by analyzing the preconditions and effects of the two actions and determine if there are conflicts between them, e.g., the effects of the first action makes the preconditions of the other false. However, our current python implementation of the planner is convenient but has no "explicit" preconditions and effects. Everything is defined through python function. The effects of an action are a function with a world state as input and returns the updated world state. Action preconditions are functions with a world state as input and return a boolean. And methods (of an associated abstract task) are functions with world state as input and return a list of tasks to update the agenda with. Methods also have preconditions working similarly then action preconditions. Hence, it's challenging to extract the explicit effects and preconditions of an action. That is why we decided to rely on an assumption to check the compatibility of concurrent actions. We consider the following in a given state. If action 1 and action 2 can be sequentially performed in both orders (1→2 and 2→1). Then we can assume that there are no causal links between the two actions and that they can be performed concurrently. The only care to take are shared resources such as a tool that would only be used during the action, making it available before and after but not during the action. To tackle this issue, shared resources are explicitly declared in the world state and in the action models. Two actions requiring a same shared resource cannot be parallelized. One benefit the way we check if two actions are mutually exclusive is making the actions estimations a black box. Indeed, we can easily replace the way we estimate the next actions each agent is likely to perform. Especially for the human, we could use a pretrained human activity estimator neural network, or more classical planner like PDDL.

With the causal principle above in mind, we proceed as follows to compute the possible concurrent action pairs from a given p-state. We start by estimating all possible human actions by refining the human agenda, generating a new p-state for each possible action. For each such p-state we first create an action pair where the robot is passive by inserting a *PASS* robot action. This *PASS* pair is stored among all other "human pass pairs". Second, we refine the robot agenda to obtain all feasible sequences of human then robot actions and

their associated p-states. We refer to them as the sequential human starting pairs. In a symmetrical way we compute the sequential robot starting pairs and "robot pass pairs" by starting with the robot then refining the human agenda.

Eventually, every action pair which is present in both the human and robot starting pairs are extracted and added to a set of concurrent action pairs. Additionally, here passive action are always parallelizable. Indeed, the only case where this couldn't be the case is when considering "joint actions" requiring the two agents such as lifting a table/object together. For now, such actions are not considered. Thus, the two sets of *PASS* pairs are directly added to the set of concurrent action pairs. Lastly, a double passive pair with two PASS is generated and added to the concurrent set. This pair is special since it doesn't update the world nor the agendas, hence, it leads back to the previous p-state without progressing toward the goal. These pairs not need to be explored and help it different ways. First, it helps the execution for instance when only the human can act but decides anyway to pass voluntary. Then the policy will natively stay in the same p-state. Second, it is easy to detect dead-ends which correspond double *WAIT* pairs. In such case, both agent cannot act, due to the lack preconditions for action or decomposing an abstract task, and they remain stuck without solving the task. Last, double *IDLE* pairs indicates that both agendas are empty and thus that the task should have been solved.

The obtained set of concurrent action pairs corresponds to all possible concurrent action that the human and the robot can execute in parallel in the initially given p-state. Each possible pair leads to a new p-state with updated agendas and world state, creating a tree structure.

4.5.4 Merging p-states

Although the tree structure produced by the process described above is complete and sound, it is inefficient and scales badly. Indeed, during the exploration we are likely to encounter several times similar p-states. For instance, consider an action pair where both the human and robot are active leading to a new p-state, one branch. Now, consider a first action pair where only the human is active, and a second where only the robot is active. Performing those two last pairs in both orders creates two other branches. However, even though the trace is different, the three p-states are identical (same world state and agendas) and it's highly redundant to explore independently each of them. That is why after each computation of the concurrent action pairs we check if any of the newly generated p-state is similar to an existing one, and if so, we connect the corresponding pair to the existing p-state to avoid redundant explorations. Doing so we transform the tree structure into a directed acyclic graph (on the condition that we don't consider double *PASS* cycles). In the manner of a tree, we will refer to nodes without child as leaves nodes, which are goal p-states. Hence, now, each leaf can be reached with several paths. When looking for similar existing p-state, we assume that p-states which are parent with the new p-state, directly of not, will necessarily be different, and thus they are excluded. This speeds up the comparison process.

4.6 The Robot Policy

First of all, as depicted in figure 4.5, all children concurrent action pairs of a p-state (PS/ps_i) can actually be seen as an AND-OR graph. Since we want to preserve the latitude of choice the human have at execution, each possible human choice of action is considered as an AND

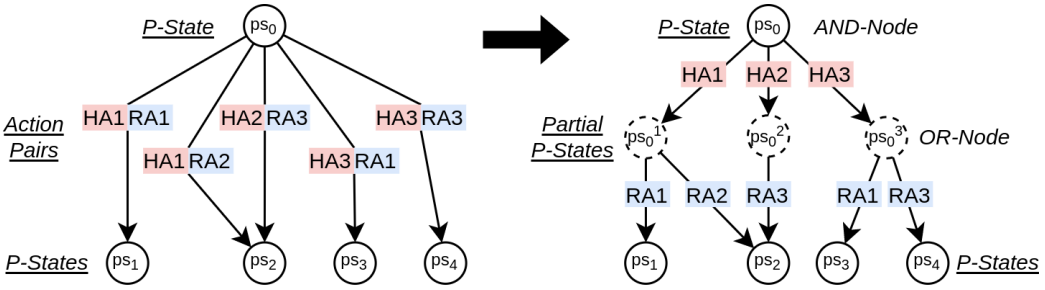


Figure 4.5: AND-OR graph representation of the action pairs. The robot policy must be compliant to any decision of the human. Hence, the problem can be seen as an AND-OR graph where for each possible human choice of action (AND node), we must determine the best concurrent robot action among the possible robot actions (OR node).

edge and leads to a partial p-state (PS'/ps_i^j). And from a partial p-state, each compliant concurrent robot action is considered as an OR edge and leads to another p-state.

Notations: A p-state is referred to as ps, and ps_i is the i-th p-state. A partial p-state is referred to as ps' and ps_i^j is the j-th partial p-state of the i-th p-state.

Thus, generating the robot policy Π consists in identifying the best concurrent compliant robot action RA^* for each possible human action, or partial p-state ps_i^j.

These best concurrent robot actions are determined by aiming to optimally satisfy an estimation of the human preferences regarding the task. Eventually, at execution, the human is free to perform any of the explored action and the robot will accordingly perform an optimal concurrent action to both solve the task and satisfy their estimated preferences.

Hence, before giving more details about the policy itself, we first describe the format of human preferences, how we could estimate them, and what it allows us to do. After, we describe the actual process to generate the robot policy from the search graph using the estimated human preferences

The robot policy consist in performing the best robot action according to the node and the human action, i.e., robot action stored in the "best compliant" pair corresponding to the human action. If all "best compliant" pair have the same robot action, the robot can perform as soon as the step stars, otherwise the human action must be identified.

4.6.1 Estimated Human preferences

estimations, format, Discussion(often inaccurate, hence our Approach), consider metrics, is a prioritized sequence of metrics to max or min, allow comparison of metrics

In this approach, instead of trying to minimize action and social costs, which are challenging to estimate accurately and quantitatively, we decided to aim at satisfying an estimation of the human preferences. In a way, the costs are included / covered / reflected in the human preferences. Our approach is to characterize each possible trace with a set of various metrics such as follows:

- **Time of Task Completion**: Time step at which the task if achieved.
- **Time of End of Human Duty**: Time step after which the human can remain passive.
- **Human Effort**: Number of non-passive human action.

- **Global Effort**: Number of non-passive human and robot action.
- ***Passive While Holding**: Number of steps where an agent is passive while holding a cube.
- ***Number of Drop**: Number of times an agent drops cube (place back a cube on the table, not in the stack).

Note that general metrics are complemented with additional domain specific metrics (marked with a star *), given in the problem specification. The set of metrics helps to characterize and evaluate each possible plan / trace. However, even if the possible plans are characterized we so far have no way to compare them and find the best one. Doing so requires additional criteria indicating how to prioritize and compare the different metrics.

That's where we consider an estimation of the human preferences to allow us to the plans and which are given by any mean external to our system. The preferences can be estimated or also given, verbally or not. Here, we consider the human preferences in the following form. These preferences are an ordered list of the metrics characterizing the traces. Note that it's not necessary for all metrics in the list above to be present, but only them can appear in the preferences. This ordered list indicates if each metric should be maximized or minimized, and in which order / with which priority. For instance, assuming the preferences aim to minimize all metrics, the best of two given plans is the one with the lowest first metric of the list. If the two plans have equal first metric then we use the second metric, and so on. Here are two arbitrary examples of human preferences trying respectively to finish the task as fast as possible and to minimize the human effort (all metrics are minimized):

TTC - GE - HE - TEH - PWH - ND

HE - TEH - TTC - GE - PWH - ND

Note that estimating either human preferences or explicit action and social costs is challenging and is hardly accurate, those are very context dependent and can even vary over time. Being aware of this, we use the estimated human preferences as a guide for the robot behavior, but we also make sure to be compliant to the human activity to lessen the impact of a wrong estimation.

4.6.2 Generation

To generate the robot policy Π from the search graph we proceed from the leaves to the root. Overall, we progressively compute the set of metrics for each possible trace. When reaching a p-state with several children we compare the metrics of the different traces leading to this node. We are then able to identify the best trace leading to each partial p-state, and the best trace leading the p-state. Each best trace to a partial p-state is used to update the robot policy, and the overall best trace is used to continue propagation the best reachable set of metrics. This process is achieved by repeating two sub-routines detailed just below, namely: Propagation and Selection.

4.6.2.1 Format and Initialization

During this process we compute and store the best reachable set of metrics alternatively in the p-states and in the action pairs. Initially, we store default/null metrics in every leaf p-state. Then we keep track of two types of nodes. First, keep track of the nodes which metrics should be propagated in the next step, those are stored in the set $\{psToPropagate\}$ which is initialized with all leaf p-state nodes. This set is later populated by the Selection

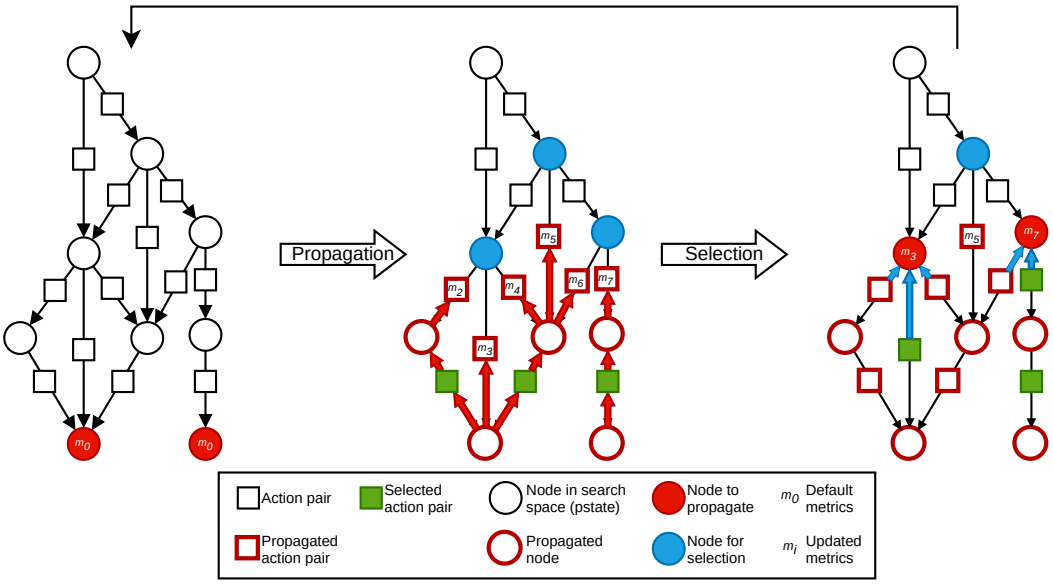


Figure 4.6: **TODO: show metrics in branches? just small m_i in each branch?** Policy generation process illustration on an arbitrary search graph. Propagation and Merge processes repeat until there are no more p-state/node to propagate nor for selection.

Algorithm 1 Policy Generation

input: $leafNodes$ ▷ Set of all leaf p-states from the search graph
$psToPropagate \leftarrow \emptyset$
$psForSelection \leftarrow \emptyset$
$Initialization(psToPropagate, leafNodes)$
while $psToPropagate \neq \emptyset$ and $psForSelection \neq \emptyset$ **do**
    $Propagation(psToPropagate, psForSelection)$
    $Selection(psToPropagate, psForSelection)$
end while

sub-routine. Secondly, we keep track of the nodes to that should select among several the best set of metrics, and thus, the best actions to perform. This set is $\{psForSelection\}$ and later is populated by the propagation sub-routine while being emptied by the selection sub-routine. The process is over when the two sets are empty, thus, when they are no more nodes either to propagate nor for selection.

4.6.2.2 Propagation

The propagation sub-routine is depicted in algorithm 2 and described here. It consists in picking a node to propagate from the set $\{psToPropagate\}$. For each parent action pair of that node we create a copy of the set of metrics of the propagated node and update them according the parent action pair and propagated node. The metrics must be cumulative.

Rules to update the standard metrics are the following **TODO: TO UPDATE, THERE IS AN OFFSET OF 1 FOR TIME STEP METRICS => should be done by giving the default metrics**:

- If the pair is not *IDLE-IDLE*, then increment (by 1) *Time of Task Completion*
- If the pair is not *IDLE-IDLE*, if the human action is passive, and if the current *Human Effort* is zero, then the temporary metric *Number Last Passive Human Action* is incremented.
- *Time of End of Human Duty = Time of Task Completion - Number Last Passive Human Action.*
- If human action is not passive, then *Human Effort* and *Global Effort* are incremented.
- If robot is not passive, then *Global Effort* is incremented.

Rules to updates domain specific metrics, must be provided:

- If human action is passive and in child p-state the human is holding a cube, then *Passive While Holding* is incremented.
- *(Similarly with the robot)*
- If the human action is to drop a cube back on the table, then *Number of Drop* is incremented.
- *(Similarly with the robot)*

The updated metrics are stored in their corresponding action pair. Then, two cases can occur for each parent action pair with stored metrics, referred as propagated pairs. First, if the parent node of the propagated pair has more than one child then we add this parent node to the set $\{psForSelection\}$. Otherwise, the metrics of the action pair are stored in the parent node which is also added in the set $\{psToPropagate\}$. The sub-routine repeats until the set $\{psToPropagate\}$ is empty.

4.6.2.3 Selection

Introduce policy notation, p-state and partial p-state (AND-OR tree)

When a node has several children, possible action pairs, then we must evaluate and compare them in order to make the best robot choices and update the policy with them. The evaluation is part is done by the propagation sub-routine. The Selection one checks when robot choices are ready to be made, update the robot policy and prepare the next propagation phase.

The selection sub-routine is depicted in algorithm 3 and described here. It checks every node in the $\{psForSelection\}$ set to know if we are ready to make a choice, i.e., if every child pair of that node has metrics stored in it, has been propagated. If not, nothing happens and the node remains in the set. If so, we are ready to compare the pairs and update the

Algorithm 2 Propagation Sub-Routine

```
1: while psToPropagate ≠ ∅ do
2:     N ∈ psToPropagate
3:     psToPropagate ← psToPropagate \ {N}
4:     for each P in N.parents do
5:         P.metrics ← CopyAndUpdateMetrics(N.metrics, N, P)
6:         if HasOneChild(P.parent) then
7:             P.parent.metrics ← P.metrics
8:             psToPropagate ← psToPropagate ∪ {P.parent}
9:         else
10:            psForSelection ← psForSelection ∪ {P.parent}
11:        end if
12:    end for
13: end while
```

policy. Since we want the human to be free to perform any action, even suboptimal, we must find the best concurrent robot action for each possible human action. The first step is to group/sort/organize the children pairs by similar human action. Then, for each group of pairs, we compare their metrics using the estimated human preferences and identify the best pair of the group, which is marked a "best compliant pair". After, all marked pairs are compared the overall best pair is identified and marked as "best pair". Eventually, the metrics of the "best pair" are stored in the node from $\{psForSelection\}$, the node is removed from the set and added to the other set $\{psToPropagate\}$.

Algorithm 3 Selection Sub-Routine

```
1: for each PS in psForSelection do
2:     if ReadyForSelection(PS) then
3:         bestPairs ← ∅
4:         for each partialPS in GetPartialPStates(PS) do
5:             pairs ← GetCorrespondingPairs(partialPS)
6:             P ← IdentifyBestPair(pairs) ▷ Compare metrics and identify best
7:             Π(partialPS) ← P.robotAction
8:             bestPairs ← bestPairs ∪ {P}
9:         end for
10:
11:        bestPair ← IdentifyBestPair(bestPairs)
12:
13:        PS.metrics ← bestPair.metrics
14:        psForSelection ← psForSelection \ {PS}
15:        psToPropagate ← psToPropagate ∪ {PS}
16:    end if
17: end for
```

4.6.2.4 Additional policy updates

After executing the main process to generate the robot policy Π, each partial p-state ps' is mapped to a robot action to execute. So, for the robot policy to be executed the partial p-state must be identified at runtime. Since, a partial p-state is characterized by the choice of action the human made, this identification is done through the ID process mentioned in the Model of Execution which briefly tries to identify which action the human is starting to execute during a step. However, such identification can be challenging, so we try to avoid it when possible.

Indeed, for any p-state ps_i from the search graph, if $\forall j, \Pi(ps_i^j) = RA$ with RA being one unique/same robot action, then the best robot action doesn't depend on the human choice. Thus, in such case, the ID process can be avoided, and the policy is complemented as follows $\Pi(ps_i) \leftarrow RA$.

Moreover, we consider cases where the ID process failed to identify the human action, and is noted as λ. To prevent potential conflicts due to identification failure the policy is updated s.t. $\Pi(\lambda) = PASS$. Note that in this work we only consider identification failures and not wrong identifications.

If human is passive ? dedicated partial p-state identified after WaitHumanDecision.

4.6.3 Execution

The execution of the policy stems from the Model of Execution automaton and is depicted in Algorithm 4.

Algorithm 4 Execution of the Robot Policy

```
ps ← ps_0                                                    ▷ Initial state
while ps.children ≠ ∅ do
    IndicateStepStarted()                                    ▷ Inform the human
    WaitHumanDecision()
    if ps ∈ Domain(Π) then                                   ▷ If ID not needed
        Execute(Π(ps))
    else
        if HumanIsPassive() then                 ▷ Detected by WaitHumanDecision
            Execute(Π(ps'))
        else
            idPartialPS ← IDProcess()                        ▷ ∈ {λ} ∪ {ps'}
            Execute(Π(idPartialPS))
        end if
    end if
    WaitEndStep()                          ▷ Human and Robot actions are done
    ps ← AssessementProcess()              ▷ Identify executed pair, and next ps
end while
```

4.7 Empirical Results

simulation of execution, without durative action

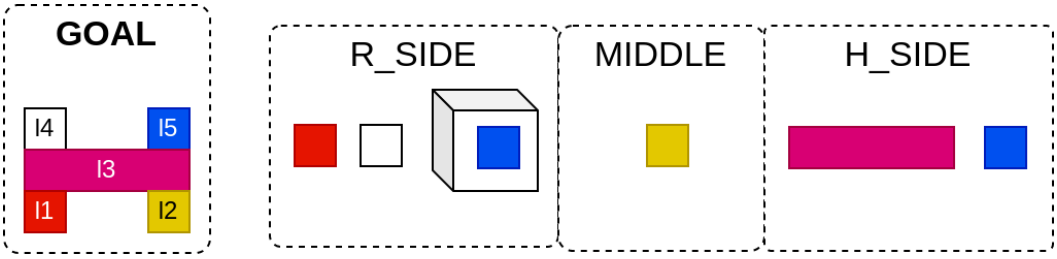


Figure 4.7: An instance of the BlocksWorld domain. The ideal plan is strongly influenced by the human desired preferences. For the earliest end of the task, the human prevents using the box. A lazy human will only place the required pink bar from their side. And a human in a hurry will place concurrently the yellow cube to place the pink bar at the earliest and be able to leave.

We provide results obtained after simulating symbolically the execution of robot policies produced with our approach, thus without durative actions.

4.7.1 Simulated Experiment

The execution is symbolically simulated by running an implemented version of the automaton described in the *model of execution*. This implementation is close to the presented algorithm 4 where action execution are mocked and replaced by symbolic and instantaneous actions. [Info on other mocked processes ? ID ? Wait?] Thus, the current state progresses in the search graph based on the human decision and the produced robot policy before eventually reaching a leaf node indicating that the goal is satisfied. We then retrieve which course of action has been executed to then analyze it.

We evaluated our approach in the BlocksWorld domain. Figure 4.7 shows one problem instance. The human and the robot are on two sides of a big table and their shared task is to stack colored cubes as shown in the given goal pattern. Initially, all colored cubes are arranged on the table that is divided into three zones: Each agent has a dedicated zone (RZ & HZ) and a common zone (CZ) is in the middle and accessible for both. Each agent can only pick cubes from either their own zone or from CZ. There is a box in RZ in which cubes can be inserted. To pick such cubes the robot must first perform a dedicated action to open the box before being able to use the cubes inside it like the regular ones.

4.7.2 Human behavior and erroneous estimated preferences

To simulate the human behavior, we consider and define human preferences that produce a human policy in the same manner as for the robot. The produced human policy makes the human always perform the best action regarding their defined preferences. The robot/-planner doesn't have access to the human preferences but only to an estimation of them.

In order to evaluate the quality of the executed trace regarding the actual human preferences we compare and rank every possible trace from the search graph, from best to worst. For legibility purposes, we normalize the ranks to obtain a score (H-score) s.t. the trace/plan with the lowest rank has a score of 0.0, while the highest rank corresponds to a score of 1.0. This score represents a quality indicator independent of the instance's size. Similarly, we can do the same and acquire the score regarding the robot's estimation of the preferences (R-score). Keep in mind that the R-score is an estimation of the actual

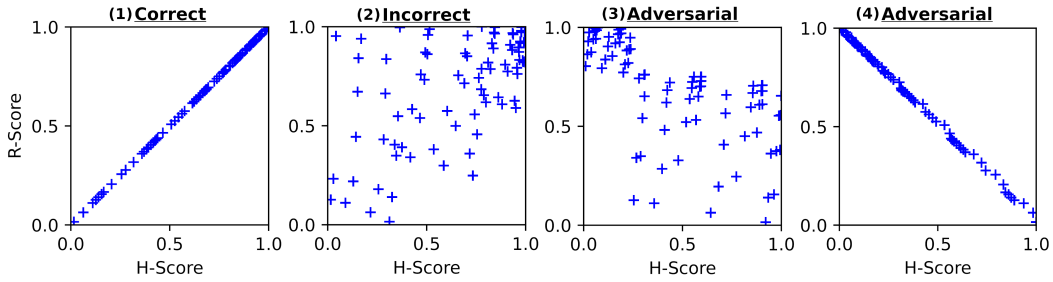


Figure 4.8: Correlation between the H-score and R-score according to different robot estimations. Four pairs of human preferences and their estimation are considered. For each pair, all possible traces are plotted as blue crosses according to their H-score and R-score. *(1)* show a correct estimation while the others show incorrect estimations. *(3)* and *(4)* depict *adversarial* estimations.

H-score, and the robot acts in order to maximize its R-score, hoping to maximize as well the H-score.

However, the estimation of the human preferences can be more or less accurate, causing the robot's decisions to differ from what humans would have preferred. Once again, that's why making the robot compliant with human online decisions and actions gives the human more influence over the execution and helps to reach high H-score even when the robot's estimation is incorrect. Despite the robot trying to maximize its R-score, it's important to note that reaching a low R-score is fine as long as a high H-score is attained.

In practice, a pair of human preferences and their estimation creates a correlation between the possibly obtained H-scores and R-scores when solving the task. Figure 4.8 depicts several possible correlations for a same task and same search graph but different pairs of human preferences and their estimation. On each sub figure are shown all possible traces as blue crosses according to their H-score and R-score. Let's consider the first case where the estimation is perfectly accurate (correct). Here, the choices of the robot which are maximizing the R-score will necessarily maximize the H-score. Indeed, when considering the few top possible robot plans (crosses with near 1.0 R-score), the human preferences are always well satisfied (near 1.0). Let's now consider the second case where the robot estimation is incorrect. Here when considering again the few top possible robot plans a wide range of H-score can be reached (near 1.0 as well as close to 0.0). Thus, an incorrect estimation can satisfy the human preferences but not necessarily, and so, can also fail to comply with them. Eventually, consider the last two cases. Here, the lack of blue crosses in the top-right corners means that the H-score cannot be near 1.0 while also having a high R-score. As a consequence, when maximizing the R-score the robot will necessarily deteriorate the quality of the plan w.r.t. the H-score. We refer to these cases as *adversarial* estimations since the robot, involuntary, goes against the human will. Such cases occur when the robot estimation is far from the actual preferences and intuitively going against the latter, for instance, the robot trying to minimize the human effort while the human is actually trying to do as much as possible.

4.7.3 Results

For the simulations, we first generated three problems of the BlocksWorld domain with different initial states and shared tasks, and we produced for each their corresponding search graph. After, we generated numerous pairs of human preferences and associated

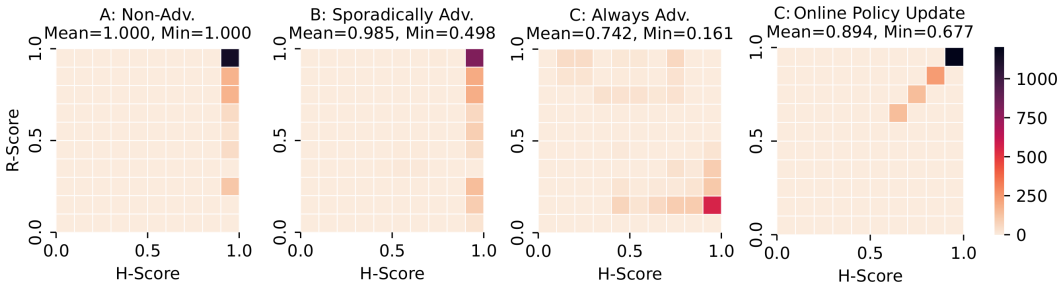


Figure 4.9: R-scores and H-scores of the obtained executed plans after simulating the execution of the robot and the human policy generated by considering three problems and three sets of pairs of preferences/estimations. The estimations in each set are (A) Never, (B) Sporadically, and (C) Always adversarial. On the right, it is shown the scores obtained using an enhanced human policy that can correct online the robot's estimation while using the set (C). **TODO: [show H mean+min and R mean+min ?]**

robot estimation, all those pairs are categorized in three distinct sets.

In Set A, the estimations are mostly correct and close to the human preferences (case *(1)* in figure 4.8). Set B includes incorrect estimations (case *(2)* in figure 4.8). And Set C contains only adversarial estimations (cases *(3)* and *(4)* in figure 4.8). Then, for each preference-estimation pair and each problem we generated the associated human and robot policies. Their execution was simulated symbolically using an execution automaton directly shaped upon the presented Model of Execution, and the obtained executed trace were retrieved. The R-score and H-score of every obtained executed trace are shown as heatmaps for each distinct set of pairs in Figure 4.9. This will help us to highlight the benefits of using such execution scheme.

In Set A, the estimation of the robot is close to the real human preferences and is never adversarial. So, the robot policies (maximizing the R-score) should naturally lead to high H-scores, and this is what we observed. Some plans had an R-score lower than 1.0 showing that the estimation was not perfect. Yet, the compliance to the human actions and the non-adversarial choices of the robot allow to always satisfy the maximal H-score of 1.0, and thus, to always satisfy the human preferences. With Set B, the incorrect estimations induced some detrimental robot choices, preventing the human from always reaching a score of 1.0. This is depicted by the minimal H-score of 0.498 obtained. Nonetheless, the average H-score of 0.985 indicates that the human preferences were overall largely met. Set C captures the worst possible estimations inducing the robot to always make adversarial choices. This is depicted by the lower average H-score (0.742) and the very low minimal H-score obtained (0.161). Yet, we can notice that the average H-score is still high and that the R-score drops significantly. A low R-score means that the robot wasn't able to follow its policy correctly. Indeed, thanks to our model of execution, the robot comply to the human online decisions and purposely deviate from its "optimal" policy to let the human follow their own optimal policy. Eventually, the relatively high H-score obtained shows that the compliance is effective and compensates significantly (of course not totally) for a very poor estimation of human preferences.

Additionally, we can reasonably complement the human policy, which is so far only based on preferences, with a *rule*. Whenever the robot performs an action that degrades significantly the best reachable H-Score, then the human reacts by correcting online the robot estimation. The rightmost sub-figure in Figure 4.9 shows the new scores obtained

using the Set C and the complemented human policy. We notice that correcting online the estimation avoids very low human scores (minimum of 0.677), and increases significantly the average H-score as compared with the original Set C results (from 0.742 to 0.894). Hence, making the robot compliant with online preferences is very effective in improving the quality of the joint plan executed.

TODO: [Comparison with baseline ??? ... this lack was criticized... A simple baseline with Robot first? Should show that on correct is fine but then H-score is highly degraded... Should be a mirror case of our results However, must replicate results and create more... currently unclear if can replicate.]

Overall, we can see that the compliant robot behavior regarding both online human actions and preferences benefits the collaboration thanks to the high human scores obtained.

Consider some counter-cases from social robotics (or HA collaborative planning). Assume a robot not giving the initiative to humans always executing the best action it found. It is less acceptable and restricting for humans this way, even if the robot computed its best action by taking into account some social rules and estimated preferences. Unlike here, humans would appear compliant with the robots. In those cases, as it is evident from our simulation results in adversarial setups, the robot strongly impacts the solution H-score. Thus wrong robot choices can significantly degrade the human scores. In some sense, being compliant and adjusting to online preferences can be seen as some social factor that robots should maximize, and our framework helps achieve that.

4.8 Performances

TODO: To develop...

In this section, we provide some details about the computational performances of our approach.

- Aware that doesn't scale well
- HRC scenario never requires long plans (10-15 actions), restricted horizon
- Here clearly enough for our scenarios
- This already allows to explore millions of possible plans
- Give some numbers: Exploration time (tens of seconds, clearly ok but should be done offline) + policy generation (less than a second, can be done online)

4.9 Discussion and Limitations

This section discusses a few limitations of the proposed approach and possible future works to overcome them.

First, in order to explore relevant courses of action, we assume a step-based progression toward the goal. Hence, we assume that the human and the robot must synchronize together after every action. This can work efficiently as long as we assume that all actions have roughly the same durations. However, in practice, this is never exactly the case and one agent must wait for the other at every step. Since the robot tends to be slower, the human might have to often wait for the robot during the collaboration with these assumptions. To be implemented on a real robot, this approach requires an additional execution scheme supervising the plan execution. In specific situations, this scheme could skip one synchronization and synchronize after the next step. Such a scheme could allow the human

to perform more actions than the robot before synchronizing and thus promote a smooth and flexible execution.

Speaking of execution, the results presented in this chapter have only been simulated symbolically, without durative actions nor real human decisions. The next two chapters (5 and 6) present a user study conducted on an interactive simulator where participants collaborated with a simulated robot executing the produced policy. To do so, we created an execution scheme able to supervise the execution of the robot policy with agent synchronizations. This scheme is directly based on the model of execution presented in fig. 4.2. Thus, it relies on the steps and doesn't provide the flexible execution mentioned just above.

Additionally, in the proposed model of execution, the robot always gives the initiative to the human. We show that this decision makes the collaboration robust to erroneous human preference estimations, and thus, is beneficial. However, it could be relevant for the robot to sometimes intelligently switch from follower to leader. Indeed, currently, even when there are no possible conflicts between agents' actions, the robot waits for the human to start acting to begin. Such synchronization isn't really necessary, so the robot could start acting directly to solve the task faster.

Eventually, the plan evaluation is a bit limited. For now, plan selection relies on the estimated human preferences which are a list of metrics to maximize or minimize in the priority order given by the list. This means that there is no balance between the metrics so, depending on the ordering, a plan of length N where the robot is never intrusive and always compliant could be rejected against a plan of length $N-1$ where the robot is intrusive twice and never compliant. Yet, in our examples, we tend to explore numerous very similar possible plans. Thus, there are always several possible plans with a same top-priority metric value, e.g. plan length, and the ordering will help select the best plan satisfying at best the other listed metrics.

4.10 Conclusion

We addressed the complex challenge of concurrent task planning for a shared goal in the context of human-robot collaboration, acknowledging the inherent need for autonomy in humans' choices of 'what' and 'how' aspects during task execution.

Based on studies about joint action, we formulate an execution model, and we present a new human-aware task planner designed to accommodate the uncontrollability factor inherent in human agents while employing this execution model leveraging social signals to facilitate the exploration of human-robot joint actions and a smooth execution. We also propose a plan evaluation and selection based on estimations of the human inner preferences. As a result, the planner produces the behavioral policy for a robot that complies with online human decisions and their (online) provided estimated preferences, ensuring it solves the task, satisfies at best the estimated human preferences, and allows smooth and sound execution of concurrent joint action.

We provide a detailed account of the novel planning process and joint action model. We demonstrated its effectiveness through symbolically simulated BlocksWorld scenarios and how our model makes the robot robust to erroneous preference estimations.

Additionally, as mentioned in the previous section, we implemented an interactive simulator in which a real human can collaborate with a simulated robot running the generated policies. We used this simulator to conduct a user study to validate our approach with real humans. This simulator and study are described in the next two chapters.

Chapter 5

Interactive Simulator

Contents

5.1 Introduction

In order to validate the approach presented in the previous chapter (4), we would like to be able to perform collaboratively a task with a simulated robot executing the produced policy. This allowed us to conduct a user study validating the approach which is described in the next chapter (chap. 6).

This chapter is a technical description of the interactive simulator I developed. As depicted in fig. 5.1, it consists of several components which will be detailed throughout this chapter.

It consists of an execution and supervision module directly inspired by the model of execution which takes as input a robot policy produced by the concurrent and compliant approach. A simul

5.2 Simulated scene

This interactive simulator is based on the Gazebo Simulator. It is an open-source 3D simulator able to simulate articulated robots in a dynamic and complex environment. This section describes all aspects directly linked to this existing 3D simulator our framework has been developed upon.

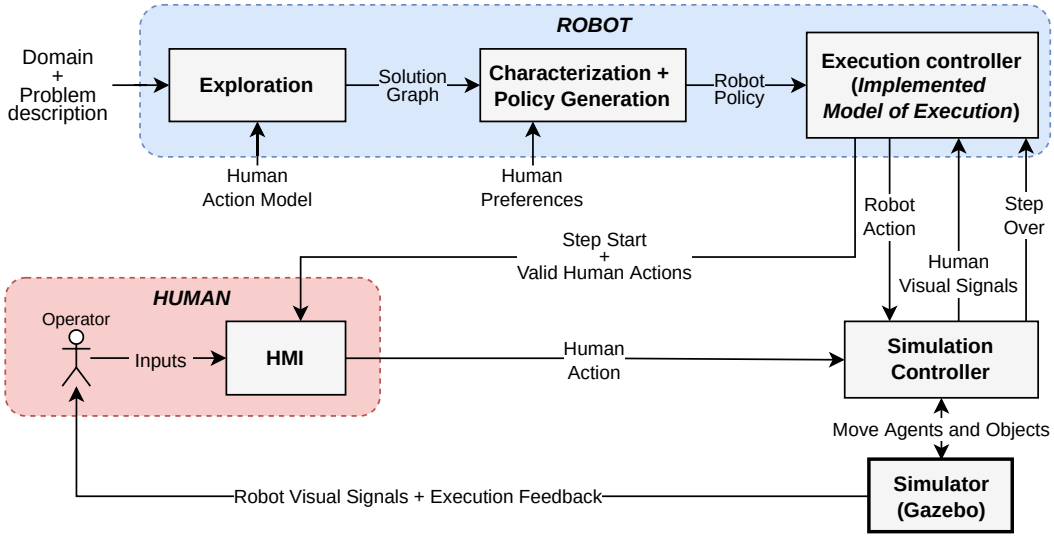


Figure 5.1: Overview of the Interactive Simulator's Architecture

The static scene consists of a large room, with a ground and 4 walls, and a table placed in the center with two marks on its right side. The marks represent the locations where the cubes must be stacked. The agents are represented as follows. First, the human is represented only with a floating hand as the camera simulates a first-person point-of-view. Then we used a Tiago robot from PAL Robotics because it has an arm to manipulate its environment, a head to send gaze information and signals and because many related resources are available for it (3D models, controllers, tutorials). The human and the robot are facing each other, each from one opposite side of the table. Finally, colored cubes were disposed of on the table in three distinct zones. Cubes are either close to the edge of one of the agents, or they are in the middle of the table. This disposition influences the reachability of the agents as one agent can only reach cubes from their side (just in front of them) or the middle, but never from the other agent's side. This scene corresponds to one specific collaborative task, but another setup could be easily implemented.

The Gazebo simulator can be integrated with the Robot Operating System (ROS) which is a set of software libraries and tools that help to build robot applications. The main pros of ROS are its ability to run several sub-programs in parallel and make them communicate with each other. Hence, the other components of the interactive simulator have been developed with ROS.

The overall scene is depicted in the figure 5.2. The stacking goal is displayed in the top left corner and a text prompt is shown in the top right corner for the robot to communicate with the human.

5.3 Simulation controllers

To control the simulated agents and the simulation, 4 distinct controllers have been developed and handle dedicated jobs.

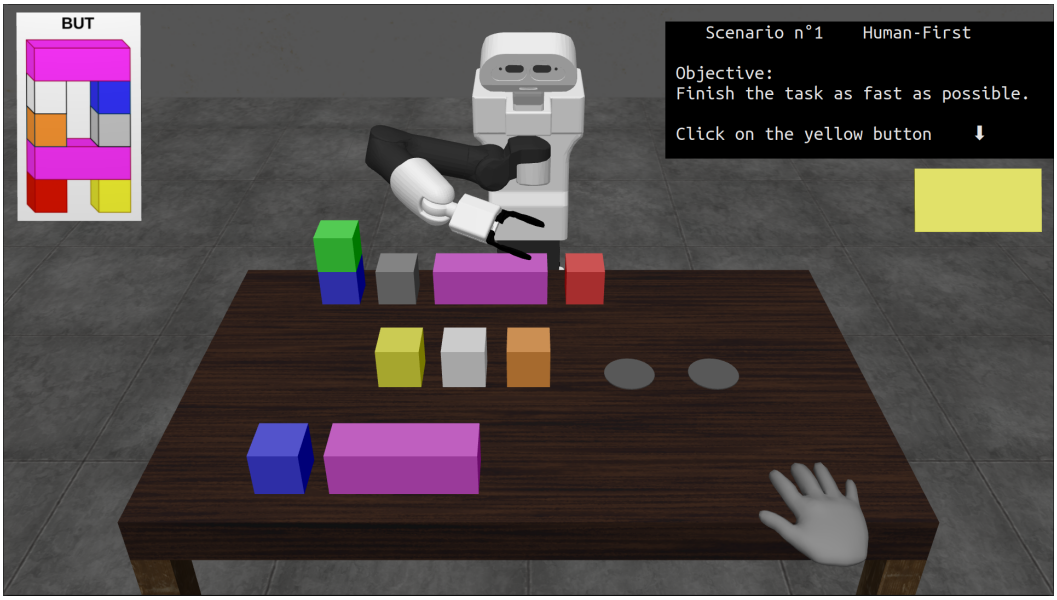


Figure 5.2: Participant view of the interactive simulator. Text prompt. Goal pattern. robot. table. cubes. stacking area. human hand.

5.3.1 Robot arm motion controller

First, there is one controller to move the robot arm. This controller is called with either a 3D position to reach (pose target) or a predefined configuration (named target). In both cases, when called, this controller uses the MoveIt framework. MoveIt creates links between several libraries in order to have access to a unified interface for Motion Planning algorithms, Inverse Kinematics, Control, or Collision Checking. This way, our controller can "simply" request MoveIt to find a trajectory to a given position and then make the robot follow this trajectory while taking into account the geometry of the robot arm.

In fact, this controller is slightly more sophisticated. Indeed, in our collaborative context, we prefer the robot to be reactive rather than to have optimal motions. This is why, through the MoveIt interface, we use two different motion planners. The first one is RRT^* which is an optimal planner that stops only when the optimal motion plan has been found. This is desirable to have the robot exhibit consistent and efficient motions. However, this process sometimes takes too much time (4-5s), which breaks the rhythm of the interaction. As a consequence, a short timeout (0.6s) has been set for this optimal motion planner. If the optimal plan isn't found within the defined timeout, we use the second motion planner. The second planner is SBL which isn't an optimal planner. That is, the planner stops when a solution is found, but there is no guarantee of optionality. The major pro of this planner is its speed. Overall, for every robot arm motion, we first try to find an optimal motion within a short amount of time. If the optimal motion isn't found, we quickly find a suboptimal solution to prevent the robot from being passive for too long.

Additionally, to reduce motion planning time, a few simplifications have been done. First, collisions are only considered with the table and the robot's body. Hence, the robot arm sometimes go through the other cubes. Also, the pick and place orientation are ignored, for instance, when picking a cube the robot gripper just reaches the cube's center from any angle and then the cube is attached to the gripper. When placing a cube, the robot moves the cube to the target position and when dropping it, the cube is detached and both its

orientation and position are overwritten to match the actual target location. This way, the robot always performs perfect place action, improving both the reactivity of the robot and its robustness, but the robot's motions are less realistic.

5.3.2 Robot head motions

To exhibit a more intuitive and collaborative behavior, the robot head is controlled to look at various elements during the interaction. Indeed, when waiting for the human the robot looks at the camera, and as soon as the human hand moves to either perform an action or indicate its passivity, the robot starts following the hand to show it is observing the human motions to understand their intentions. Then, the robot looks at a cube to indicate its intention to pick it since this is faster than the arm motion. Also, the robot looks at the target position when placing a cube to indicate its intention to place the cube there.

To perform those head motions a dedicated controller has been developed based on an existing controller provided in the Tiago robot modules. The existing controller could only change the robot's gaze through a visual and clickable window allowing an operator to manually click on the scene to change the robot's gaze. This controller has been modified to allow additional features such as directly requesting to look at 3D point or an object. But also, we can now ask to follow an object. That's how we are able to exhibit the head behavior described just above.

5.3.3 Human hand motions

Moving the human hand requires a dedicated controller but is much simpler than the robot arm motion one. Here, the hand must either move to a position or perform a PASS signal. The PASS signal is a hand gesture indicating to the robot that the human desires to be passive for the current step. This motion is performed by simply rotating the hand back and forth at a constant speed. The PASS motion has a duration of 0.7s. On the other hand, to move the hand to a target position we do not use a real motion planner. We simply compute a straight line between the current hand pose and the target pose, and update the hand position at $50Hz$ to move at a constant speed set to $0.25m/s$. The planning time is thus negligible compared to the robot motion planning times.

5.3.4 Simulation controller

5.3.4.1 Action decomposition

Last but not least, I developed a fourth controller whose main job is to translate the actions from the plan produced by our task planner to low-level motions that can be executed by the controllers listed above. Hence, some parts of this controller are domain-specific, but a major part is generic for manipulation tasks. This controller received high-level agent actions to execute such as "Robot pick(b1)" for the robot picking the cube b1. The first step is to decompose this action into low-level generic actions which are themselves made of low-level motions. This controller also keeps track of a few low-level facts to check low-level preconditions, such as, what are each agent holding or which object is still on the table. Hence, when an agent tries to pick a cube while already holding a cube the controller throws an error.

Let's first comment the low-level generic actions available. This controller is given a list of the object names (cubes) and the names of some predefined locations in the scene (placing locations in the stack). Note also that many of the following actions can be performed either

by the human or the robot, this is defined by a parameter given when calling the action. The following low-level actions are available:

- **move_pose_target:** moves either the hand/robot arm to a given position.
- **move_location_target:** moves either the hand/robot arm to the position of a given location.
- **move_obj_target:** moves the hand/robot arm to the position of a given object.
- **move_home:** puts the agent in its "home" (default) configuration.
- **move_named_target:** moves the robot arm to the given configuration.
- **grab_obj:** attaches the given object to the hand/robot gripper.
- **drop_obj:** detaches the given object to the hand/robot gripper.
- **set_obj_rpy:** sets the orientation of a given object.
- **set_obj_pose:** sets the position of a given object.
- **delta_move_obj:** sets the position of a given relative to its current position.
- **human_hand_gesture:** makes a hand gesture to indicate passivity.
- **robot_head_look_pose:** makes the robot look at a given position.
- **robot_head_look_human:** makes the robot look at the human (camera).
- **robot_head_follow_obj:** makes the robot follow a given object.
- **robot_head_follow_hand:** makes the robot follow the human hand.

Then the high-level actions, from the plan produced, are the following with their respective simplified low-level decomposition:

- **PickCube:** Starts by retrieving the cube's current position by sending a request to the Gazebo simulator. If it is the robot, call robot_head_look_obj to look at the cube. Then, call move_pose_target with the retrieved cube position. Once over, grab the cube with grab_obj. After, move_home is called to retract the robot arm or bring the hand to its initial position. Finally, if it is the robot, it looks to the human with robot_head_look_human.
- **PlaceCube:** Starts by retrieving the position of the target location to stack the cube. If it is the robot, call robot_head_look_pose with this position. Then, we move either the arm or the hand to that position with move_pose_target before dropping the cube in the stack with drop_obj and adjusting its position. After, we go back to the home configuration with move_home. And for the robot, we look at the human with robot_head_look_human.
- **BePassive:** The human waves their hand to explicitly be passive, thus, human_hand_gesture is called. The robot doesn't move and only display some texts saying the robot wants to be passive. The text prompt mechanism will be described later.
- **DropCubeTable:** When an agent cannot stack a cube they are holding, they can place it back on the table. First, the drop position is defined. It corresponds to the initial position of the cube being held, except for the green one which is initially on top of another cube and has a dedicated drop position. The robot looks at the drop position with robot_head_follow_pose. The hand or the robot arm is moved to the position with move_pose_target. The cube is dropped with drop_obj and its position is adjusted. Then, the agent is put in its home configuration with move_home, and the robot looks at the human with robot_head_look_human.

5.3.4.2 Manage steps

One last thing, the simulator controller is in charge of indicating when a step is over. Basically, since we do not consider steps where both agents are passive, a step begins when an agent starts an action. And it is over when both agent actions are done. These rules cover many situations such as if the human initially wants to be passive and wave their hand, as a result, the robot starts performing an action which starts the step. Currently, the step would be over as soon as the robot action is over since the human is passive. However, if the human decides anyway to start performing an action concurrently then the step will be over when both the robot and human actions are over. This may imply that if the human starts acting right before the end of the robot action then the robot will just wait for the end of the human action, and thus, the end of the step before the next step begins.

5.3.4.3 Send visual signals

The model of execution synchronizes the agents based on explicit visual signals such as the start of a step, the start of an action, the end of an action or hand gesture (PASS). Those signals are modeled explicitly inside the system and managed by the simulator controller.

Indeed, when starting an action, the associated visual signal is sent only when the agent starts to move. Hence, we track the arm and hand motions to know when the action is visible to the other agent. Then, when an action is over, the associated visual signal is sent directly. It is worth mentioning that the human will naturally have a reaction time when seeing the robot visual signals (start/end of action, step start). However, so far, the robot doesn't have any reaction time to such symbolic visual signals. Thus, to simulate the delay introduced by a real perception module (here perfectly simulated) the human visual signals are delayed to the robot by reaction time set to 0.3s. This is still quite fast but at least the robot doesn't interpret instantly the human motions.

5.4 Robot execute and supervision

TODO: to develop

model of execution, progress in the solution graph of the robot policy, sound for synchronization step start

takes as input the solution graph produced by chap. 4 appraoch. Then, it is basically an implementation of the model of execution, sending and synchronizing on the visual signals produced and sent my the simulator. Hence, we have a durative execution of the plan with an effective effect in the simulated environment.

ID process, best robot concurrent action, TO

Note that during the planning process described in the previous chapter, the model of execution has been abstracted and taken into account to explore relevant courses of actions by anticipating possible executions. Here, the model of execution is rather implemented, matching the abstraction made, to be able to execute the produced robot policy and supervise its execution by synchronizing with the human.

5.5 Human Machine Interface (HMI)

TODO: to develop

text prompt, clickable zones, signals

Human operator or participant can interact with the simulation by performing any feasible action during the process. The model of execution is still followed, thus, every step starts by the robot sound and a dedicated process receives internally the list of the feasible human actions for the current step. This list is sent by the robot execution scheme which is reading the solution graph. Yet, the feasible actions aren't shown nor known by the participants. Each possible human action is associated to a zone (rectangle) on the screen. When the human mouse click in a zone associated to a currently feasible action, then the process request the simulation controller to perform the corresponding human action. delayed visual signal for robot.

5.6 Logs and timeline

There is also a dedicated process to record the different events and signals during one execution. All other components can send events to log to this process to save them for later post-processing. A log event is defined with a name and a time stamp. Additionally, visual signals are also captured by this logging process and saved them both as a signal and creates an associated event. Logged events mostly help to trace the internal states and processes ran by the robot during the execution. The human ones mostly only serve to identify when the human is acting or not since we don't have access to the internal reasoning process of the human participant.

5.6.1 Activities extraction

All logs are saved after each execution and can be loaded later for post-processing. The post-processing involve three steps. First, the agent activities are extracted from the event list. The human and the robot have different possible activities, but again, since we don't have access to the internal human reasoning there are more robot activities than human ones. The extracted activities are shown in the following table 5.1. The activities orders try to match what can happen during the execution but not all activities are present in every step. Indeed, on the robot side, the ID process isn't always necessary and thus not always executed, and the robot either perform an action or is passive for diverse reasons. But the robot always end up waiting for the end of the step, perform the SA process and get ready for the next step. On the human side, if the human performs an action or make a hand gesture, there will be a decision time than respectively the human action or the explicit passive activity. If the human remains passive without signaling the robot then the only activity is passive without signaling. Only if the human is active, the waiting next step to start activity is added, otherwise being passive for a step is implicitly equivalent to waiting for the next step.

5.6.2 Metrics computation

When loading the execution logs we extract a set of metrics from the events and extracted activities. Currently, 27 metrics are extracted, but the following table 5.2 only shows 11 because several sub-metrics are hidden. Indeed, for four of the listed metrics, we compute the sum, average, standard deviation, min, and max values of the corresponding metric.

5.6.3 Execution example with timeline

A complete scenario is presented and commented on in this section. Using the previously extracted agent activities, we can draw a visual timeline depicting the different steps, the

Robot Activities	Human Activities
Waiting for human decision	Decision Time
Identification Process (ID process)	Human Action
Planning arm motion	
Robot Action	
Waiting for human action when itself cannot act	Being passive with signaling (after PASS hand gesture)
Being passive	Being passive without signaling (after TimeOut is reached)
Waiting for the end of the step	Waiting for the next step to start
Situation Assessment Process (SA process)	
Getting ready for the next step (GRNS)	

Table 5.1: Agent activities extracted from logs. The activities orders and relative positions try to match the execution but not all activities are present in every step.

Metric	Description
Task Completion Time	Time for the task to eh completed.
Number of steps	Number of steps executed.
Number of optimal human actions	Number of human action being optimal regarding a given objective.
Ratio of optimal human actions	Number of optimal human action divided by the number of steps.
Human decision time*	Duration for the human to make a visible decision in a step.
Waiting next step*	Duration during which the human waits for the next step to start.
Number of human action	Number of non-passive human actions executed in the scenario.
Human action duration*	Duration of the human actions.
Number of robot action	Number of non-passive robot actions executed in the scenario.
Robot action duration*	Duration of the robot actions.
Time human free	Time after which the human is free, i.e., as soon as the robot can finish alone.

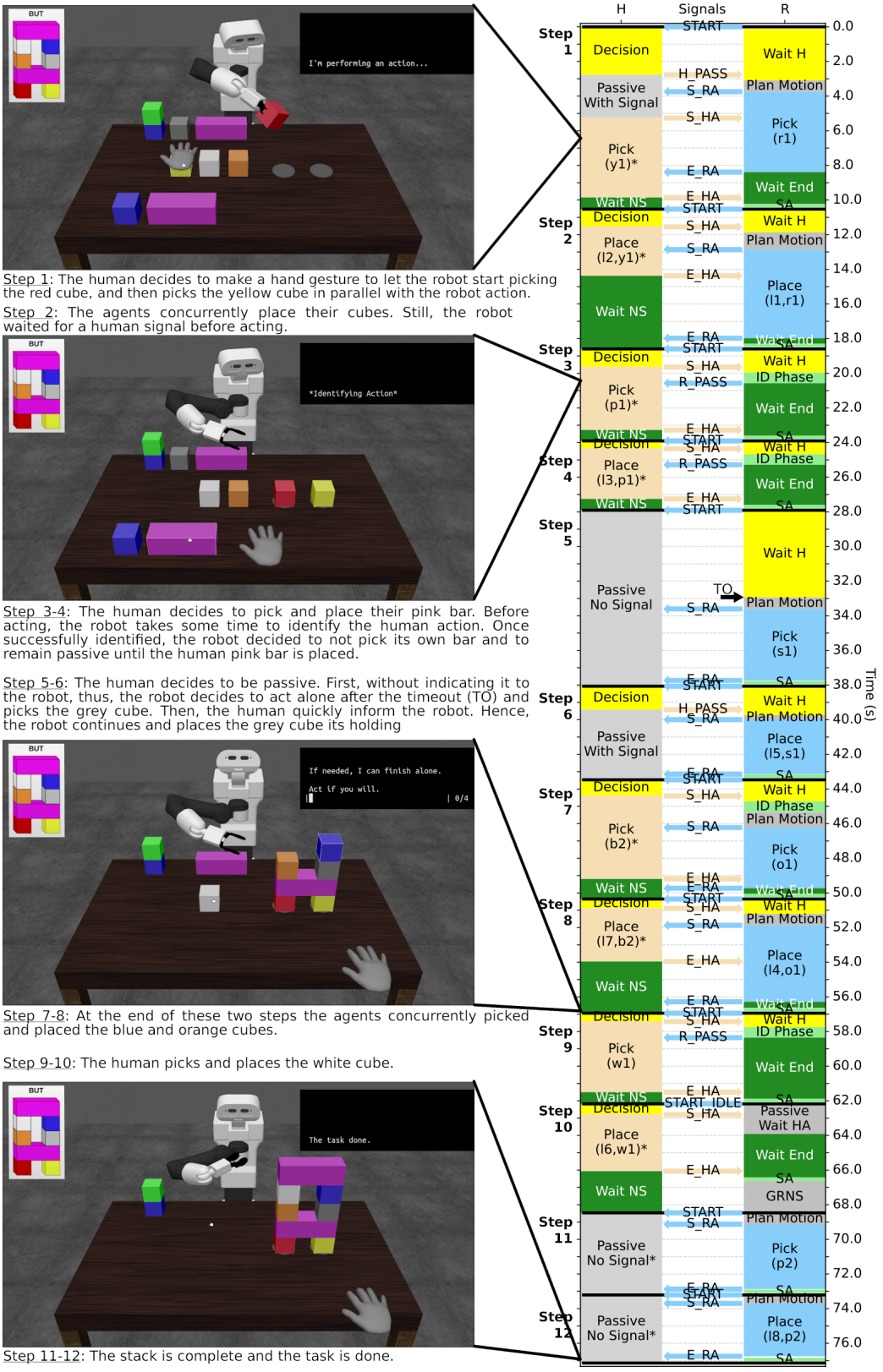
Table 5.2: Metrics extracted from the execution logs for each scenario. Marked items with a star (*) correspond to five sub-metrics computed over all steps: sum, average, standard deviation, min, and max values.

extracted activities, and the visual signals exchanged between the agents. Figure 5.3 depicts the complete commented timeline, including agents' activities, exchanged signals, and snapshots from the simulator. The table 5.3 describes the different signals exchanged during the execution and shown on the timeline.

Signal name	Description
START	Visual and audible signal from the robot indicating that the current step has started.
START_IDLE	Visual and audible signal indicating the start of a step where the robot cannot act.
S_RA / E_RA	Signals sent respectively from the start (S) and the end (E) of a robot action (RA).
S_HA / E_HA	Signals sent respectively from the start (S) and the end (E) of a human action (HA).
H_PASS	Explicit hand gesture from the human informing their desire the be passive.
TO	The TimeOut is an internal robot signal indicating that the absence of human signal.

Table 5.3: Signals exchanged between the agents during the execution and used for synchronization and coordination.

In this execution example, according to the extracted metrics, the task has been completed in $77.14s$ in 12 steps. The average human decision time is $1.00s \pm 0.69s$ with $max = 2.77s$ and $min = 0.42s$. The human waited on average for the start of the next step during $1.68s \pm 1.28s$. There were 8 human actions with an average duration of $3.67s \pm 0.72s$. According to the objective/preferences to finish the task as fast as possible, the human acted 9 times optimally out of 12 steps (optimal ratio= 75%). Indeed, the human agent purposely decided to be passive during steps 5 and 6, but they could have picked the orange cube in parallel to complete the stack faster. In this example, the task description forbids agents from picking cubes in advance. They must be able to immediately place a cube in the stack to pick it. This rule is detailed and justified later in the next chapter 6.3. Hence, in step 9, the human picked the white cube, but the robot could not pick the bar until the white cube was placed. Thus, the human could have let the robot place the white cube to reduce their effort without slowing the task. There were 8 robot actions with an average duration of $3.97s \pm 0.69s$. Overall, the robot took $5.25s$ to plan its arm motions.



Step 1: The human decides to make a hand gesture to let the robot start picking the red cube, and then picks the yellow cube in parallel with the robot action.

Step 2: The agents concurrently place their cubes. Still, the robot waited for a human signal before acting.

Step 3-4: The human decides to pick and place their pink bar. Before acting, the robot takes some time to identify the human action. Once successfully identified, the robot decided to not pick its own bar and to remain passive until the human pink bar is placed.

Step 5-6: The human decides to be passive. First, without indicating it to the robot, thus, the robot decides to act alone after the timeout (TO) and picks the grey cube. Then, the human quickly inform the robot. Hence, the robot continues and places the grey cube its holding

Step 7-8: At the end of these two steps the agents concurrently picked and placed the blue and orange cubes.

Step 9-10: The human picks and places the white cube.

Step 11-12: The stack is complete and the task is done.

Figure 5.3: Complete execution timeline commented.

Chapter 6

User Study to evaluate an integrated plan and execution scheme in simulation

Contents

6.1 Introduction

To validate the approach presented in the previous chapter we conducted a user study on more than twenty participants. The purpose of this study is two-sided. First, we want to validate our overall planning approaches based on the proposed Model of Execution, and thus, show how it allows successful collaboration with a human. Secondly, we want to validate the Model of Execution itself, that is, showing how it allows the human to always be the leader and able to decide while the robot follows concurrently. We use a simple baseline to compare the Model of Execution with, and we show how our model allows to better satisfy human preferences and thus is preferred.

We decided to conduct this study in simulation for various reasons. First, one of our assumptions is that all actions should roughly have the same duration. However, real-life robots are slow and not very reactive. Those aspects may bias the results of our study which is focused on decision-making. Secondly, simulation allows several simplifications that are acceptable for study. Collision with the cubes has been disabled to make the robot faster

both in planning and executing its arm movements. In addition, simulation allows having a perfect perception of the environment. In a real-life experiment, perception errors may occur leading to replan, and thus slower execution, or even wrong decisions. Moreover, our model assumes that both agents synchronize after each step. Hence, it was easy in simulation to prevent the human from acting too soon and synchronize automatically their actions. In a real-life scenario, we couldn't physically prevent the participant from acting. This would imply a heavier training process for the participants to avoid desynchronizing with the robot. In practice, an additional execution supervisor should be developed to permit desynchronizing as long as they are not too big, and hence, prevent the system from crashing. This would require a significant technical effort to implement.

To conduct this study I developed a dedicated interactive simulator using a Tiago robot. In addition, the automaton described by the MoE has been implemented and integrated with the simulator to provide a proper execution and supervision scheme. Eventually, through carefully designed scenarios and using a shortened version of the PeRDITA questionnaire [Devin 2018] we gathered the feelings and impressions of the participants regarding the different robot behaviors. We also recorded logs from each executed scenario allowing us to draw a timeline of the execution and compute objective metrics for each scenario among which can be found: the time to complete the task, the human decision time, or the time for the human to be free. Several relevant facts and conclusions can be extracted from the collected results and are discussed in this chapter.

This chapter is organized as follows. First, the interactive simulator functionalities and operations are described. Then the methodology of the user study is provided along with anonymous information on the participants. After, the results obtained are presented and then discussed, validating the proposed hypothesis.

****

why simulation: HRI rebuttal: we rely on a reactive execution, real life robot are slow and not very reactive, may have perception errors, thus may bias our results focused on decision making.

Thus, we developed an interactive simulator running robot policies generated as explained the Chapter 4. Then, we conducted a user study using this simulator.

The purpose of this study is two-sided. First, we want to validate our overall planning approaches based on the proposed Model of Execution, and thus, show how it allows successful collaboration with a human. Secondly, we want to validate the Model of Execution itself, that is, showing how it allows the human to always be the leader and able to decide while the robot follows concurrently. Compared to a simple baseline, we also want to show how this model allows to better satisfy human preferences and thus is preferred.

This chapter is organized as follows. First, interactive simulator functionalities and operations are described. After, the study protocol is presented as well as some analysis of the participants. Eventually, the obtained results are presented and discussed.

6.2 Related work

6.2.1 Questionnaires

Many questionnaires are used in the field of HRI. The main ones are listed here: GodSpeed, HRIES, PeRDITA, RoSAS, and Trust Perception Scale-HRI.

Each questionnaire has specificities and helps to measure certain aspects of the robot. Many include appearance items to evaluate the look of the robot. Since our focus is on robot decision-making, we decided to base our questionnaire PeRDITA. Indeed, this questionnaire

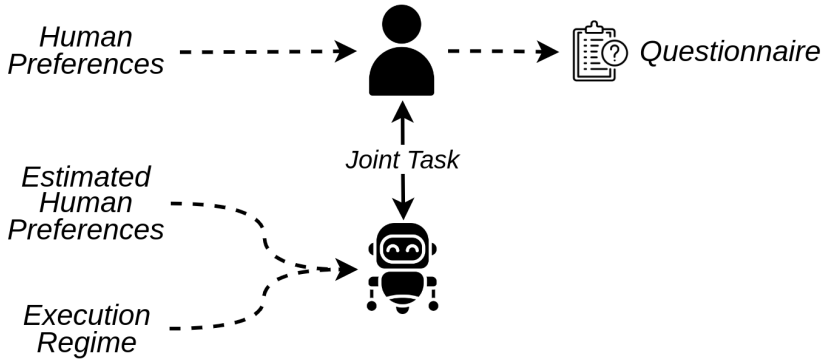


Figure 6.1: A scenario of the User Study Protocol. Each participant goes through 6 scenarios and answers 6 questionnaires to evaluate each different robot's behavior

has been designed to evaluate the pertinence of robot decisions in a Human-Robot Joint Action Context, which is exactly our case. Yet, the full questionnaire is a bit heavy and also covers communication which isn't our topic in this work. That is why we decided to shorten the questionnaire by removing the section on communication and a few redundant items. Redundant items are helpful to evaluate the consistency of a questionnaire and this has already been done in [Devin 2018]. Hence, to avoid participants getting bored and lost by filling out the whole questionnaire after every scenario we kept 12 items covering the following dimensions: robot perception, interaction, collaboration, and acting.

6.3 Study protocol

Methodology: Initial demographics questionnaire to identify potential individual difference ratings. Then, presentation instructions/text. After, familiarized with the simulator through an interactive tutorial. Eventually, the six scenarios in randomized order are experienced by the participant. After each scenario, the participant answers a shortened version of the PeRDITA questionnaire and logs about the execution are saved. Eventually, the participant is asked about their overall impressions regarding the interaction they had with the simulated robot, and they are asked which scenario they preferred the most and the least. The overall experiment takes about 35-40 min per participant.

objective, participants, material, experiment design, procedure, measures

This section describes the objectives and protocol of this study.

Through this study we want to demonstrate the benefits of using the model of execution described in the previous chapter 4 in a collaborative context. We believe this model of execution is pertinent to be taken into account when executing and supervising a robot's plan. For the same reasons we based the policy generation of this model and aim to justify our choice and validate our approach.

In this study, each participant is made to collaborate six times with a simulated robot to achieve a shared task, each time is referred to as a scenario. The robot exhibits a different behavior in each scenario. After each scenario, the participant evaluate the robot's behavior through the PeRDITA questionnaire [Devin 2018].

Beforehand, every participant answers a few general/demographic questions and is familiarized with the simulator functionalities through an integrated tutorial. Only then they

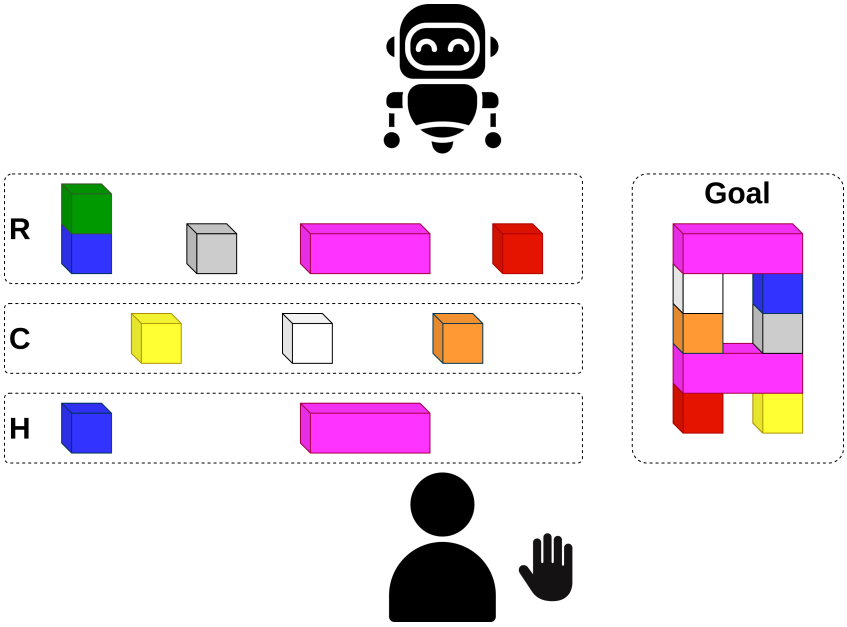


Figure 6.2: Description of the shared task to achieve in the study.

start the six consecutive collaborative scenarios, answering every time a questionnaire to describe the interaction. Eventually, every participant is asked to share their general feelings and impressions about the overall interaction with the simulated robot, and they are asked to tell which scenario they preferred the most and the least.

We now provide details about the task, the scenarios and how the different robot behavior are generated. The shared goal, which is stacking the cubes to match the given pattern, remains the same in all scenarios. The cube disposition on the table also doesn't change either. The task description is depicted in fig 6.2. For this problem, our planning approach generated a solution graph with 700 PStates/nodes leading to 6 different final states/leaves. This solution graph comprises 6839430 different possible courses of action. The length of the plans is about 19.77 ± 1.59 steps, with a minimal length of 11 steps and a maximal length of 23 steps.

To progress in the task, the agents can perform three different primitive actions which are the following: *pick* a cube, *place* a cube in the stack, or, *drop* a cube back on the table. These actions have a few preconditions, more or less intuitive, that are communicated and experienced by the participant during the integrated tutorial. First, one can *place* a cube if they hold the cube, and if the targeted location is free and supported. That is, the cubes directly below the targeted location must be placed before being able to place a cube in the targeted location. Secondly, one can only *pick* a cube from their respective reachable zones of the table, i.e., Human and Center zones for the human, and Robot and Center zones for the robot. Also, one can only pick a cube if it can be placed immediately. Thus, one cannot pick a cube "in advance" and has to wait to its placement condition to be true before picking it up. For instance, both pick bars can only be picked up after the yellow and red cubes have been placed. This rule helps to create interaction conflicts serving the purpose of this study. Moreover, although the participants find this not intuitive they get used to it really fast and this feeling seems to be significantly reduced along the experiment. Third, one can *drop* a cube back on the table only if they hold it and if it cannot be placed.

For each scenario the participant is given instructions regarding how to solve the task.

The participants are asked to consider these instructions as their own choice and preferences regarding the task resolution, and thus, to act accordingly while collaborating. The instructions at each scenario are one of the two following. On the first hand, the participant shall act in a way to finish the task as soon as possible. Here, it consists in trying to perform as many actions in parallel as possible to progress faster. These preferences are latter referred to as Task End Early (TEE). On the other hand, the participant shall act in a way to be freed as soon as possible. That is, they should finish their mandatory part of the task as soon as possible, so they could leave and let the robot finish alone. Here it consists in placing the pink bar from the Human zone as soon as possible. These preferences are latter referred to as Human Free Early (HFE). On its side, the robot doesn't directly have access to these instructions/preferences, they are only estimated. Hence, for each scenario, the robot is given a more or less accurate estimation of the human preferences that are communicated to the participant. Note that the participants aren't aware that the robot has an estimation of their preferences, neither that this estimation can be inaccurate. This way, we created three scenarios with different pairs of human preferences and associated estimation. In the first pair, the human shall finish the task early and the robot has a correct estimation, i.e., the robot's policy is helping the human to finish the collaborative task early. In the second pair, the human preferences remain the same, but the robot estimation is incorrect. The robot is trying, mistakenly, to minimize the human effort. As a consequence, the robot tends to pick cubes that the human could pick, preventing the human from acting and making the task completion longer. In the third pair, the human shall free themselves early, but the robot estimation is again erroneous. The robot will try to finish the task early while its priority is to place the first pink bar, which is conflicting with the given human preferences.

Additionally, in each scenario, the robot follow one of the two following execution regimes:

- **Robot-First (RF)**: the robot always initiates actions first, and the participant take action afterwards.
- **Human-First (HF)**: the robot always lets the participant take the initiative, and then acts.

The *Human-First* execution regime corresponds to the Model of Execution described in the previous chapter. At each step, the robot waits for the human's decision and will execute the best action that complies with it. The human always start acting first and the robot follows. On the other hand, the *Robot-First* regime corresponds to a naive and straightforward policy execution where, at each step, the robot directly starts executing the overall best robot action given by the policy. The robot always starts acting forcing the human to comply. The *Robot-First* regime serves as a baseline to evaluate the proposed *Human-First* regime, described by our Model of Execution and used in the policy generation. Eventually, we associate each of the three previous pairs of preferences and estimation with one of the two different execution regime. As a result, we obtain six different scenarios with six different robot behaviors named in table 6.1.

Note that our goal is to evaluate and compare the different robot behaviors. However, at the beginning, the participants don't have any references to compare with which can influence their answers in the very first scenarios. One solution is to ask the participants to answer all six questionnaires at the end, after being familiar with the six scenarios. We consider that this option demands a too heavy mental workload to recall accurately each specific scenario, and may bias the answers. As a consequence, we decided to ask the participants to answer the questionnaire after each scenario as a draft. During the

Table 6.1: Name of the six scenarios. Columns represent the preferences/estimation pairs and the rows correspond to the execution regimes.

	Pair A TEE: correct	Pair B TEE: incorrect	Pair C HFE: incorrect
Human-First	S1	S3	S5
Robot-First	S2	S4	S6

experiment, they can rectify their answers to match more accurately their feelings. At the end, using the drafts, they share their final answers for each scenario. We believe this process allows to more accurately gather the feelings of the participants. Moreover, the ordering in which the participants encounter the scenarios is uniformly randomized to prevent any order effect.

Dimension	Question	Item
Robot perception	In your opinion, the robot is rather:	Apathetic/Responsive Incompetent/Competent Unintelligent/Intelligent
Interaction	In your opinion, the interaction with the robot was:	Negative/Positive Complicated/Simple Ambiguous/Clear
Collaboration	In your opinion, the collaboration with the robot to perform the task was:	Restrictive/Adaptive Useless/Useful Inefficient/Efficient
Acting	In your opinion, the robot choices of action were:	Inappropriate/Appropriate Annoying/Accommodating Unpredictable/Predictable

Table 6.2: PeRDITA Questionnaire: Participants have to place themselves between the two antonym items in a scale of 7.

The questionnaire filled by the participants after each scenario is a shorten version of the PeRDITA questionnaire, and its items are gathered in table 6.2. In addition to the questionnaires, for each scenario, the interactive simulator produces logs from which we extract several metrics and an overall timeline of the execution. The timeline depicts the activities and actions of each agent along the task progression. The subjective measures done through the questionnaire are complemented with the objective metrics extracted such as the duration to complete the task, the number of human actions, the total duration of human inactivity, and more.

There are a few restrictions on the actions that can be performed. First, an agent can only pick cubes that can be placed immediately. This means that agents cannot pick cube in advance to anticipate each other's actions. Allowing such behavior could generate a very interesting scenario. However, here we want to purposely generate some conflicts to evaluate the robot's behavior and reactions. Without this restriction, the agents would have too much flexibility in their actions and decisions which makes it harder for the conflicts to happen. Additionally, when holding a cube, the agents can only place the cube in the stack on back to its original place. As a result, the agents cannot displace the cube on the table to make them reachable to the other agent. This restriction has been added for the same reasons as the first one and simplifies the conflicts generation.

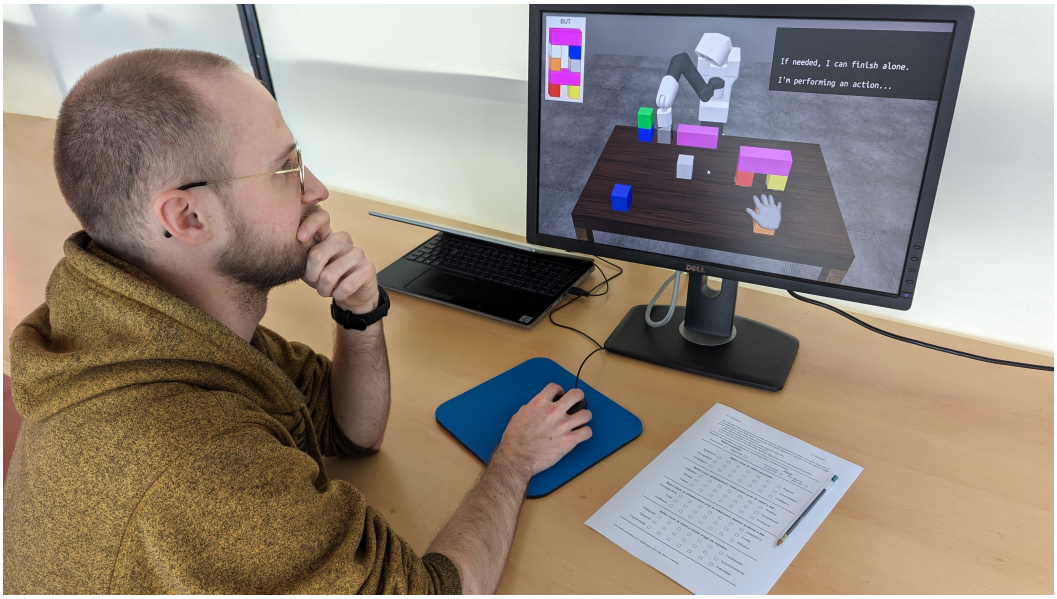


Figure 6.3: One scenario execution where a participant is collaborating with the simulated robot using the mouse. Once the task is completed, the participant is asked to fill the questionnaire on the desk with a pencil to transcribe their impressions.

The participants were collaborating with the robot using a mouse. The simulation was run on a laptop connected to a bigger screen allowing participants to see clearly the simulated environment. After each scenario, participants answered the printed questionnaire using a pencil. Figure 6.3 depicts the execution of a scenario where the participant collaborates with the simulated robot using the mouse. Once the task is completed and the scenario is over, the participant is asked to fill out the printed questionnaire on the desk with a pencil. For each scenario, a new paper sheet is provided to the participant.

6.4 Participants

This section shares and analyzes some information on the participants.

TODO: provide info number, age, ext, familiar with R tech, vision of robotics

6.5 Study results

In this section, we analyze the results of the study with first some technical comments regarding the experiment. After, we analyze the results obtained from the execution logs. Then, we discuss the questionnaire's answers. Finally, we discuss the participant's comments regarding the experiment. Note that all the numeric results of the study are given in appendix A, including questionnaire answers, execution metrics, comments, and scenario preferences.

6.5.1 Technical comments

Numerous scenarios were executed in the simulator to conduct this study. More precisely, 150 scenarios were executed and a total of 1914 steps were executed. It is interesting to share a few technical comments about how those executions.

To begin with, very few technical issues or crashes occurred during the study. About 2 or 3 crashes were due to a failure in the HMI that had happened when participants clicked at a very specific instant. I wasn't able to identify the origin of the issue, but this only happened a few times considering that 1048 human actions were performed during the whole study. This means that 0.29% of the human action failed. Then, about 4 to 6 crashes occurred due to a failure of the robot arm motion controller. The arm motions were planned successfully but the controller failed to execute the planned trajectory in the simulator which led to a crash of the controller, freezing the robot. This kind of failure was specific to the MoveIt framework and I couldn't find a solution to them. But again, those issues were quite rare considering that the robot performed a total of 1586 actions during the study. This means that 0.38% of the robot action failed. Overall, less than 10 scenarios crashed during the study which is less than 6% of failures. In practice, recovering from a crash was quite easy and fast. After a brief intervention of less than $30s$, the participants were able to start again from the scenario that crashed. Sometimes, this implied the participants to repeat a large part of the crashed scenario which affects the participant impression (less novelty effect) but none significantly changed their actions when repeating such a scenario.

On the other hand, it is worth mentioning and discussing the durations of the different processes run by the robot. At every step, the robot has to decide which action to perform, move its head, plan its arm motion, and move its arm. First, the decision time of the robot is negligible because it is given by the policy computed previously by our planning approach. Before every step, the robot identifies the current state. Given a state, the policy dictates the robot which action to perform, including if the human action must be identified first. Since all this is precomputed, the decision time is negligible. Head motions are also not demanding, and their execution occurs in parallel with the other robot processes, hence they can be neglected. However, planning the robot's arm motions is heavy computing and takes about $0.56s \pm 0.28s$. Notice that the standard deviation is quite high, about half of the average value. Indeed, the motion planning is based on algorithms using randomized exploration which makes the solving time random, sometimes begin very fast ($min \cong 0.001s$) and sometimes quite slow ($max = 5.37s$). Yet, we were able to plan online the arm motion. Eventually, the robot arm motion durations are about $4.09s$ (max=$9.09s$, min=$1.46s$) and will be discussed more precisely below

Additionally, since the task is quite simplistic, repetitive and deterministic, one could say that we could have pre-computed the robot arm motions in order to lighten the execution and avoid technical problems linked to motion planners. First, we insist on the fact that the arm motions failures were due to execution failure, not planning, thus it is unclear if pre-computed the trajectory would have helped regarding those failures. Doing so would certainly make the simulator less demanding in terms of computation power. But here we wanted to keep a generic simulator able to able any other task, thus, movements couldn't be pre-computed.

6.5.2 Statistical assumptions

Our data are close to following a normal distribution (checked using Kolmogorov-Smirnov, Shapiro-Wilk and Anderson-Darling tests). Thus, parametric tests can be applied, and we used both paired t-tests and Analysis Of the VAriance (ANOVA) with repeated measures

to analyze the collected data, more precisely, to identify significant differences between different group of measures. In the last case, Bonferroni Post-hoc-Tests are performed to identify exactly which groups are significantly different from others.

It has been commonly assumed that a statistical test demonstrates a significant difference if a p-value lower than 0.05 is obtained. However, obtaining a value lower than 0.001 is desired. To make the p-values more legible the following standard notation is commonly used and will be used below:

$$
\begin{aligned}
p > 0.05 &\Rightarrow ns \text{ (non significant)} \\
p \leq 0.05 &\Rightarrow * \text{ (significant)} \\
p \leq 0.01 &\Rightarrow ** \text{ (very significant)} \\
p \leq 0.001 &\Rightarrow *** \text{ (highly significant)}
\end{aligned}
$$

Additionally, the value a metric x will often be given in the following format including the average value M and the associated standard deviation σ: $x = M \pm \sigma$.

6.5.3 From execution logs

This section is focused on analyzing the results obtained through the execution logs saved after each scenario.

Preferences satisfaction (task completion time + time to be freed)

In this study, the human preferences consist of either finishing the collaborative task as soon as possible or being free as soon as possible while letting the robot finish alone. Thus, to evaluate how the human preferences were satisfied, we can measure in the first case the time to complete the task and in the second case the time after which the human can leave.

Figure 6.4 depicts through box plots for each pair of scenarios the corresponding relevant metric to evaluate the human preferences' satisfaction. We used t-tests for paired samples for pairwise comparison. For each pair, the tests for normal distribution suggest that the data does not significantly deviate from normality, and thus parametric tests such as t-tests can be conducted.

In Pair A, in addition to completing the stack the human wants it to be completed as soon as possible. The robot has a correct estimation of human preferences. The completion time using HF and RF are shown. The completion times in S1 and S2 are roughly similar with the respective values: $59.84s \pm 5.83s$ and $56.64s \pm 6.46s$. The completion time of Scenario 1 is higher than Scenario 2. However, a t-test for paired samples showed that this difference was not statistically significant ($p = 0.055$) and there was a small effect ($d = 0.4$) according to Cohen's d [Cohen 1988] (small effect = 0.2, medium effect = 0.5, large effect = 0.8). Thus, the RF regime allowed the human to solve the task slightly faster than the HF regime, and thus, satisfy their preferences slightly better. In both scenarios, the collaboration goes smoothly, and the task is achieved without trouble.

In Pair B, the human still wants the stack to be completed as fast as possible, however, the robot has an erroneous and adversarial estimation of their preferences. This time the HF regime in S3 had lower values ($66.62s \pm 14.87s$) than the RF regime in S4 ($82.82s \pm 4.42s$). This difference is statistically significant ($p < 0.001$) and there was a large effect ($d = 1.07$). This indicates that in S4 the participants' preferences were significantly less satisfied than

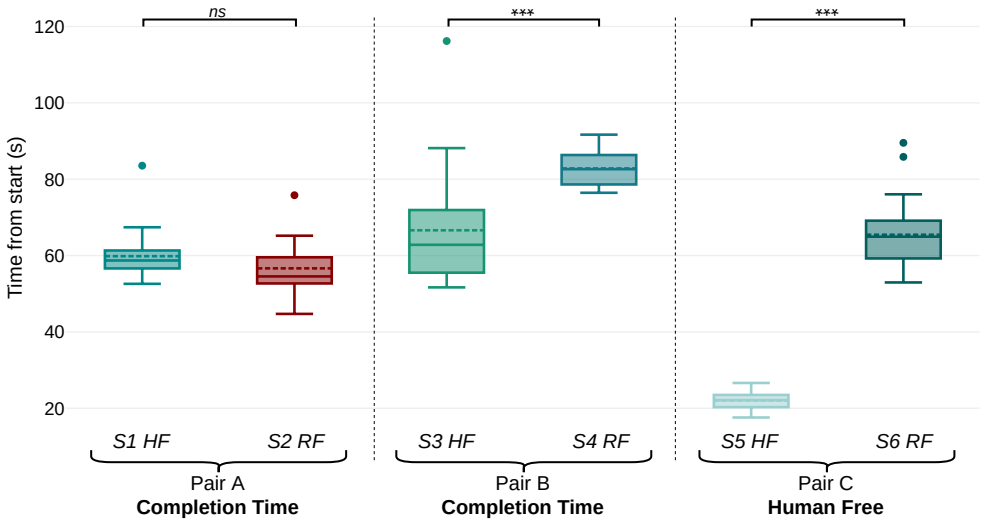


Figure 6.4: Human preference satisfaction. According to the scenarios, it corresponds either to completing the task as fast as possible (Pair A and B) or being free as soon as possible (Pair C). Using t-tests for paired samples, we can identify in pair B and C the criteria of preferences is significantly better satisfied. In pair A, the difference isn't significant but the completion time is slightly shorter when using RF.

in S3. Indeed, in this pair, the robot erroneously thinks that the human wants to minimize their effort. Thus, the robot ends up trying to "*steal*" cubes from the human to prevent them from acting, thus, minimizing their effort. With RF the human has no choice and cannot act most of the time, leading to a high completion time with a low standard deviation due to the restricted human choices leading to very similar executions. With HF the robot always acts compliantly in parallel right after the human. Hence, the human is able to pick the cubes they want and that the robot wants to pick, *forcing* the robot to adapt and pick other cubes. This eventually leads to executions close to S1. However, if for any reason the human doesn't pick the cubes then the robot picks them preventing again the human from acting, explaining the high standard deviation. Comments about the feelings of the participants in each of these scenarios are given in the next subsection using the answers to the questionnaire. Overall, S4 was perceived as frustrating and S3 was perceived similarly to S1 and S2.

In Pair C, the human prefers to be freed as soon as possible. Hence, we measured the time after which the human isn't required to finish the task, i.e., the time after which the robot can finish the task alone. Scenario 5 (HF) allowed the human to be free earlier ($22s \pm 2.35s$) than Scenario 6 (RF) ($65.45s \pm 9.08s$). This difference is statistically significant ($p < 0.001$) and there was a very large effect ($d = 4.43$). This indicates that the HF regime allowed the participants to satisfy their preferences significantly better than the RF regime. Here, the erroneous estimation of human preferences makes the robot try to place its pink bar first, which implies that the human should place their own at the top of the stack as the last cube. Such a plan forces the human to stay until the end of the task which is in direct contradiction with the actual human preferences. Hence, after placing the yellow and

red cubes concurrently, both agents tend to pick their pink bar. At this point, in S5 (HF) the robot waits for the human decision, the human can place their bar and free themselves from the task, and the robot compliantly drops its pink bar before finishing the stack alone. However, in S6 (RF), the robot doesn't wait for the human decision and places its pink bar before the human can do anything, forcing them to stay until the end to place the pink bar. As a result, the S6 values are significantly higher than S5. Moreover, the participants had various reactions to the frustrating robot action of placing the pink bar before them. Some remained passive until the end while holding their bar, others would drop it to help the robot aiming to place the bar as fast as possible anyway. These various reactions led to various executions explaining the high SD of S6.

Overall, RF tends to slightly better satisfy human preferences only when the estimation is correct (Pair A), yet, the difference wasn't significant compared the HF. On the other hand, when the estimation is erroneous, HF satisfies human preferences significantly better than RF due to how compliant the robot is when using HF. This indicates that using our model of execution instead of a simplistic baseline (RF) is beneficial for collaboration in terms of satisfaction of human preferences.

****

We should be able to show that in scenario with similar execution, i.e. in pair A with S1 and S2, RF allows to solve the task faster, thus, human preferences are better satisfied with RF. Indeed, the S1 M1 group had higher values (M = 59,9, SD = 5,95) than the S2 M1 group (M = 56,29, SD = 6,34). A t-test for paired samples showed that this difference was statistically significant, t(23) = 2,27, p = ,033, (Small Effect) In addition, since agents have to synchronize together at each step, we are able to measure the amount of time the human has to wait for the robot. This amount should be significantly lower when using RF than HF. The S1 M11 (Wait ns total) group had higher values (M = 2,04, SD = 0,32) than the S2 M11 group (M = 1,29, SD = 0,36). A t-test for paired samples showed that this difference was statistically significant, t(23) = 7,4, p = <.001 (Large effect)

Ratio human optimally

The participants were given in every scenario an objective to satisfy, to consider as their own preferences regarding the task, and that should guide their behavior. However, in practice, the explicit actions to conduct were not given, and the participants were free to act as they would. Naturally, not all participants behaved in the same way. There were differences in the decision time of each, but also in the action decisions, leading to different execution traces. Since different execution traces influence significantly the metrics of the timeline, it is worth discussing how the participants behaved.

First, table 6.3 depicts the number of different execution traces per scenario and overall. There were 45 different execution when considering all scenarios, which can appear quite low compared to the 6839430 possible plans. This also means that our exploration covers enough possibilities for this task. Additionally, it worth noticing the high number of different plans in S6 and the low number in S1. In S6, the robot acts in a quite frustrating manner which lead to various reactions of the participants. On the other hand, it seems that S1 was quite clear since participants performed only 4 different sequences of actions. It's also worth to mention that since there are only two different human objective or preferences, we would expect only 2 different optimal traces, one satisfying each objective. All other 43 other obtained traces are due either to "wrong" robot decision or suboptimal human decisions.

In the same manner as for the robot, an optimal human policy is generated for each scenario (considering the actual preferences given to the participant). Hence, it is possible

	Total	S1	S2	S3	S4	S5	S6
Number of different executed plan	45	4	9	10	6	7	16

Table 6.3: Number of different plan executed in each scenario and overall.

to check at each step if the participant performed the optimal action or not, and thus, compute an optimal ratio which is the number of optimal human actions performed divided by the total number of human actions performed. This can help us to analyze the results and can explain some outlier values.

Though there are no significant differences between the different scenarios, some scenarios still have a lower average optimal ratio and high standard deviation, meaning that participants tend to have more varied behaviors in these specific scenarios. The average number of human actions per scenario is about 7, from 2 to 10.

	S1	S2	S3	S4	S5	S6
Mean	*95.52*	*95.4*	*92.04*	*98.03*	*96.74*	*87.09*
Std. Deviation	*7.56*	*7.78*	*8.28*	*4.58*	*7.65*	*13.48*
Minimum	*71.43*	*71.43*	*78.57*	*81.25*	*75*	*61.54*
Maximum	*100*	*100*	*100*	*100*	*100*	*100*

Table 6.4: Optimal human action ratio per scenario

As depicted in the table 6.4, S6 has the lowest average optimal ratio and the highest std. deviation. In this scenario the robot places its own pink bar even though the human holds one already, preventing the human from placing it and forcing them to drop it back on the table. This surprising behavior seems to cause frustration and confusion which led to various human decisions and actions, and more likely to deviate from the optimal course of action. In practice, a significant amount of participants get confused and are passive during several steps after the frustrating robot action. Some even remain passive almost for the whole task, waiting for the robot to stack the cubes alone until the human pink bar has to be placed. This diversity in the participants' reaction is reflected in the high sd of S6.

The low SD of S4 is also noticeable. Indeed, here the robot tends to steal the cube from the reach of the human. This behavior prevents the human from acting, and thus, from making decisions. As a result, fewer decisions are taken by the human in this scenario which results in less possible deviation from the optimal course of action.

Decision time

Participants' decision time fluctuates a lot. Especially with HF. Indeed, at every step, the HF robot waits for a defined amount of time to observe the human decision and acts accordingly. Any human visual signal received interrupts this timer. This amount of time will be referred to as the HF Timeout because after it is reached the robot considers the human as passive. This timeout was initially set to $3s$ with the hypothesis that it should be quite small to allow fluent interaction. With a precise action in mind, the human is able to act first, otherwise, the robot fluently takes the lead and acts first. However, during the preliminary tests, the participants felt in a rush and oppressed by this relatively low timeout. Indeed, when they didn't have a precise action to perform when the step started, they didn't have the time to think properly and tended to be rushed by the timer progressing

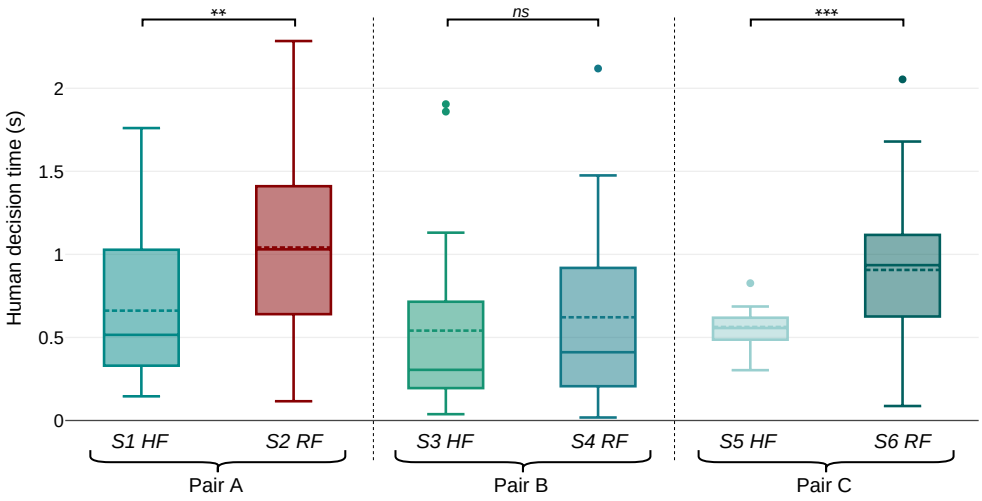


Figure 6.5: Average human decision time in the six scenarios. This decision time tends to be lower when using HF than with RF.

towards the timeout. Hence, we decided to increase the timeout from $3s$ to $4s$ which was felt way more comfortable.

One could think about comparing the total (sum, cumulative) decision time over each scenario. However, since different human actions can lead to various number of steps, this isn't representative. (except in scenario with same execution, only use 100% opti ratio?)

We compare the average human decision times which are measured similarly with HF and RF and as follows. After one participant finished one scenario, we measured their decision time on each step. To do so, we first consider for each step the time when the step begins, which is signaled with text, a gaze and a sound from the robot. Then we consider the time when the human sends a signal by either starting an action or by waving their hand. The duration between these two times is considered as the decision time of the human. Note that if the human remains passive (no signal until) for a step, no decision time is computed for this specific step. Then, we extract the average decision time of the participant on the scenario from all the computed ones, compute the standard deviation and get the maximum and minimum values.

A one-factor analysis of variance with repeated measures showed that there was a significant difference between the variables, F = 5.99, p = <.001 with an effect size Eta squared $\eta^2 = 0.2$ which corresponds to a large effect. When doing pairwise comparisons with t-tests obtain the following results.

In pair A, S1 (HF) had lower values (0.66 ± 0.41) than S2 (RF) (1.04 ± 0.58). This difference is statistically significant ($p = 0.002$) with a medium effect ($d = 0.68$).

In pair B, S3 (HF) had lower values (0.54 ± 0.51) than S4 (RF) (0.62 ± 0.55). This difference is not statistically significant ($p = 0.551$) with a very small effect ($d = 0.12$).

In pair C, S5 (HF) had lower values (0.56 ± 0.11) than S6 (RF) (0.91 ± 0.46). This difference is statistically significant ($p = 0.001$) with a medium effect ($d = 0.76$).

Considering the defined scenario pairs, the decision time with RF tends to be longer. However, this difference is statically significant only for pair C (S5-S6). This is expected

because when the robot places the first pink bar the human gets confused and takes time to adapt to the situation. On the other hand, this is not reflected in S4 despite the similar confusing robot actions. Indeed, in S4 the robot "steals" cubes from the human reach which is confusing, but this prevents the human from acting and thus no decision time can be computed.

I think the overall slower human decision time in the RF scenarios is due to the fact that the human acts after the robot. This way, the human has to pay attention to the scene and to the robot action/intentions, which is longer than only looking at the scene like in HF scenarios.

Overall the decision time of the human is an average of about $0.72s$.

Agent actions' duration

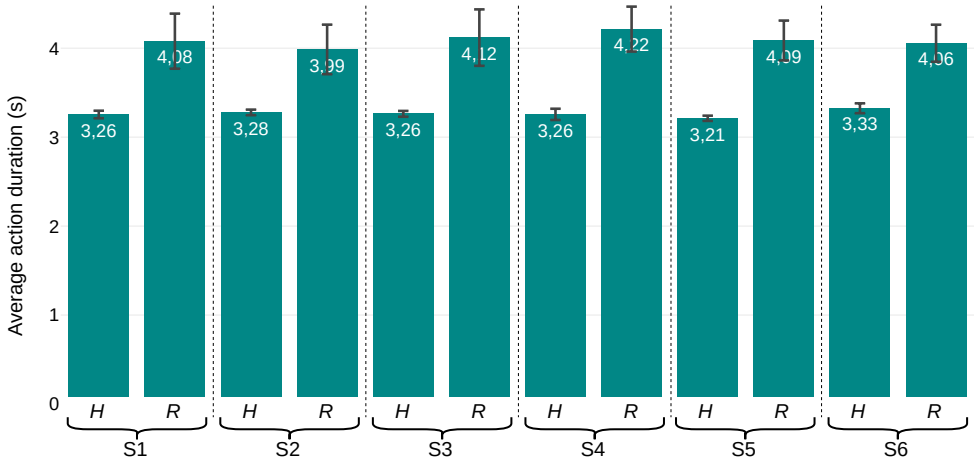


Figure 6.6: Average action duration of the human and robot agents over the 6 scenarios, with standard deviations. Compared to the human action durations, the robot ones tend to be longer and have various durations.

As depicted in fig. 6.6, the human actions are in average significantly faster than the robot ones. In addition, the robot action durations tend to fluctuate more than the human one. This can be explained by the difference in motion execution between the avatar and the robot. The human has a simplified motion planner that simply moves the hand at a constant speed and in straight lines to the cubes or the stack. However, the robot uses a real motion planner to move its arm which is longer than the human motions. The motion planning process doesn't always found the same solutions, nor in the same amount of time. Meaning that both the motion planning duration and the motion execution duration can fluctuate. Here only the motion execution duration is considered in this metric. Note that to avoid having too much difference between the human and robot action durations, collisions with the dynamic objects are not considered in the robot motion planner, nor the objects' orientation. Hence, the robot can pick or place cubes from any angle and pass through the other cubes. When placing a cube its orientation is corrected. Collisions with the table where kept preventing the robot from picking cubes from below.

Overall scenarios and steps, mean = 3.27s the maximum human action duration is 4.63s and the minimum is 2.55s. For the robot, mean=4.09s, the maximum action duration is 9.09 and the minimum duration is 1.46.

6.5.4 From questionnaires

This section is focused on providing the results obtained by analyzing the answers to the questionnaires filled by the participants after each scenario.

To help the reader understand the following plots we list here the items of the questionnaire from the table 6.6 with their associated numeric ID in table 6.5. These IDs will be used in many plots in the x-axis to analyze the questionnaire's answers.

Robot perception	Interaction	Collaboration	Acting
1 Responsive	4 Positive	7 Adaptive	10 Appropriate
2 Competent	5 Simple	8 Useful	11 Accommodating
3 Intelligent	6 Clear	9 Efficient	12 Predictable

Table 6.5: Questionnaire items with their associated IDs.

6.5.4.1 Overall analysis

We start by commenting overall questionnaire's results using some relevant average values and standard deviations before having a deeper statistical analysis in the next subsection.

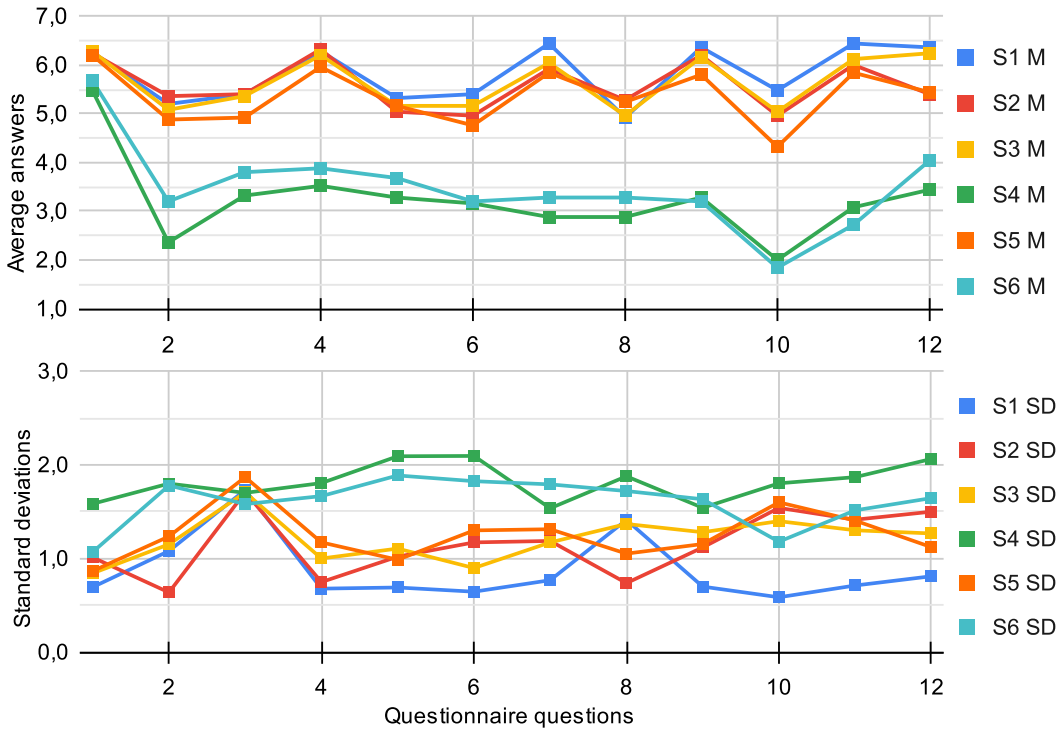


Figure 6.7: W.r.t. each scenario, average answers (M, top) and standard deviations (SD, bottom) obtained for each question of the questionnaire.

Figure 6.7 depicts the answers obtained for each question of the questionnaire w.r.t. each scenario. This figure provides a very visual overall summary of the study. On the top part, for each of the 12 questions on the x-axis, the average answers obtained are plotted for each scenario, 7 being the maximal or best value and 1 being the minimal or worst value. We can see that four scenarios obtained quite similar high answers whereas scenarios S4 and S6 have noticeably worse answers. Those scenarios are the Robot-First scenarios of pairs B and C where the robot has an erroneous, and even adversarial, estimation of human preferences. Note that question 1 (Q1) which evaluates the reactivity of the robot is the only question which answers are relatively high in every scenario. Considering the answers to all other questions than Q1, S4 and S6 seem to deviate significantly from the other scenarios which can be analyzed as follows. First, it means that with a correct estimation both HF and RF regimes are roughly perceived similarly. Second, an erroneous estimation doesn't seem to induce lower answers, and thus, despite the wrong estimation the robot in S3 and S5 is roughly perceived similarly to the one in S1 with a correct estimation. On the other hand, when using RF, an erroneous estimation seems to have a significant detrimental impact on how the robot is perceived by the participants. All these preliminary conclusions will be confirmed in the statistical analysis below.

It is also worth commenting on the standard deviations obtained, depicted in the bottom part of figure 6.7. The standard deviations depend a lot on the scenarios and go from 0.6 up to 2.1. There are two noticeable facts to comment on. First, question 3 evaluating how intelligent the robot is perceived is the only question with a relatively high SD for every scenario. Participants had various definitions of "intelligence" which led to a wide range of answers. Some participants evaluated the intelligence of the robot on its choices of actions, and thus, fluctuated depending on the scenario. Some others evaluated the intelligence of the robot on other criteria independent of the robot's decisions. Thus, they would rather indicate that the robot was always intelligent (or not) over all scenarios. Additionally, we can see that the SD of S4 and S6 seem to be higher than the other scenarios.

In contrast with the previous figure, Figure 6.8 shows the answers obtained for each question w.r.t. to each execution regime. Indeed, the HF average answers and standard deviations, shown in blue, correspond to the union of the scenario's answers using the Human-First regime, i.e., $S1 \cup S3 \cup S5$. Similarly, the RF values, shown in red, correspond to $S2 \cup S4 \cup S6$ where the Robot-First regime is used. Additionally, the results for all scenarios combined are shown in light yellow. The results shown here can be deduced from the previous figure since we already commented on all scenarios. Yet, this new figure highlights more visually the difference between the HF and RF regimes.

The first noticeable fact is that when using the HF regime the average answers for each question are better than when using the RF regime. Again, the only question where the answers are roughly the same regardless of the regime is when questioning the reactivity of the robot. HF's average answers are relatively high for all questions which indicates that the collaborations with HF seem to be appreciated. On the other hand, RF's answers are average (around 4) which indicates that collaborating with RF seems to be less appreciated.

Concerning the standard deviations, it is also noticeable that the answers concerning the RF regime have higher SD. This indicates that there was a wider range of answers from the participants when collaborating with the RF regime. This means that participants tend to be less certain about their answers when evaluating RF than with HF. This can be expected because when facing the HF regime the collaboration is overall quite positive, thus, participants concentrated their answers on the higher part of the scales. However, the frustrating robot actions due to the RF regime degraded the collaboration and participants had to evaluate how bad this degradation is. Some participants were more emotionally

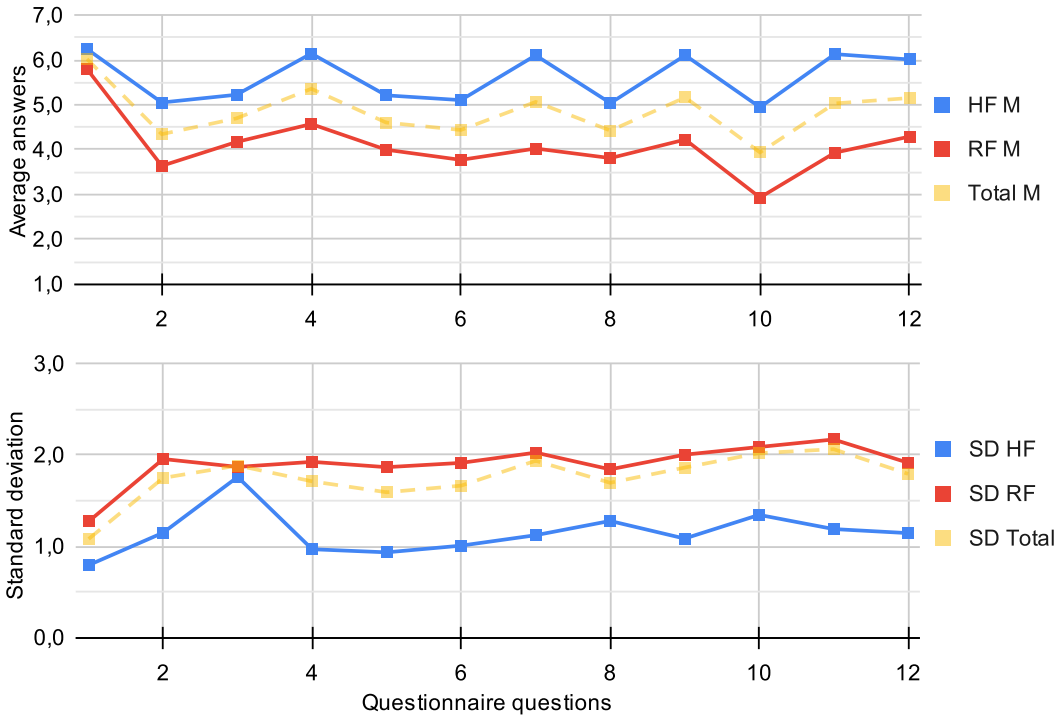


Figure 6.8: W.r.t. each regime of execution, average answers (M, top) and standard deviations (SD, bottom) obtained for each question of the questionnaire.

affected than others by the robot's actions, hence, it led to a wide range of answers. Another interesting fact is that regardless of the regime, question 3 has a high standard deviation. This question evaluates the *intelligence* of the robot. Participants had various definitions of intelligence which is reflected in this high SD. Indeed, some participants evaluated the robot as unintelligent because of its nature and thus regardless of its actions or the scenario. Others perceived less intelligence when the robot performed frustrating actions. Also, we can see that the SD regarding the reactivity of the robot is quite close and low for both regimes. This is a consequence of the very high average values regarding Q1 for both regimes.

Eventually, I would like to insist on the average answers concerning the RF regime. Even if these answers are mediocre, it is important to note that they are not very low. In comparison, consider another robot behaving completely erratically. This robot would randomly pick and place cubes around itself. As a consequence, the robot would neither help the human nor solve the task, on the contrary, it might only disturb the human trying to achieve the task. The robot could make the task impossible to solve for the human by picking a relevant cube and never placing it, or by removing already well-placed cubes from the stack. Here, during one step, the RF regime forces the human to comply with the robot's decisions which can be frustrating. However, the robot takes into account the human action anyway and adapts accordingly its actions for the next step. This is thanks to our planning approach, common to both regimes, which explores every possible human action to generate the robot policy. Thus, we can say that our planning approach seems to benefit the collaboration and interaction between the human and the robot.

6.5.4.2 ANOVA Analysis

So far, we only conducted a preliminary analysis of the questionnaire's answers using only the average values and standard deviations. Now, a statistical analysis must be conducted to confirm the preliminary comments of the previous subsection. For each question, and thus each item of the questionnaire, we performed an analysis of the variance with repeated measures. Each analysis led to p-values $<= 0.001$ indicating that there is a significant difference between the 6 scenarios. To evaluate the strength of this significant difference the effect size Eta squared η^2 has been calculated where the limits are .01 (small effect), .06 (medium effect), and .14 (large effect). However, ANOVA tests can only indicate if there is a significant difference between N-samples, but it is only of interest to identify between which exact group that difference exists. In the Bonferroni post-hoc test in a repeated measures ANOVA, multiple t-tests are calculated for dependent samples. However, the problem with multiple testing is that the so-called alpha error (the false rejection of the null hypothesis) increases with the number of tests. To counteract this, the Bonferroni post-hoc test calculates the obtained p-values times the number of tests. The obtained p-values indicate in a pairwise manner between which samples the significant difference exists. The results from the ANOVA and Bonferroni post-hoc test are shown in table 6.6.

As suggested by the preliminary analysis, the Reactivity of the robot is the item with the lowest difference. The ANOVA indicates that answers regarding the reactivity are significantly different with a large effect over the 6 scenarios. However, compared to other items, the effect size η^2 is quite low indicating that this difference is less significant than for other items. Also, the Bonferroni post-hoc test wasn't able to identify where exactly the difference exists, which means again that this difference isn't very significant in the end.

Besides reactivity, all other items have large significant differences according to the scenario which can be exploited using the Bonferroni post-hoc test. Indeed, the pairwise comparisons indicate the existence of major significant differences for every question in the following pairs: S1-S4, S1-S6, S2-S4, S2-S6, S3-S4, S3-S6, S4-S5, and S5-S6 (in bold in the table). Having in mind that S4 and S6 are the two scenarios using Robot-First with erroneous estimations, we can see that erroneous estimation with RF systematically leads to significant differences compared to all other scenarios using HF or RF with a correct estimation (S4-S1, S4-S2, S4-S3, S4-S5, and S6-S1, S6-S2, S6-S3, S6-S5). This is a clear indicator that the RF regime is very sensitive to the estimation of the human preferences and thus an erroneous estimation is significantly detrimental to the collaboration and the overall interaction.

Only a few other significant differences exist and are in S1-S5 and S3-S5. Indeed, in S5, the robot's actions were perceived as more or less significantly less appropriate ($p = 0.004$) and predictable ($p = 0.004$) than in S1, slightly less predictable ($p = 0.017$) than in S3. Indeed, in S5 the HF robot surprisingly picks up its pink bar while the human picks up its own whereas the human wants to place their bar to be freed from the task. Using HF allows the human to place their bar anyway, but the robot actions were therefore perceived as less predictable and appropriate than in S1 or S3. We can also notice that despite S3 having an erroneous estimation in contrast to S1, there is no significant difference in answers for each question between S1 and S3. This indicates that using the HF regime allows the robot to be way more robust to erroneous estimation than RF. However, the few existing significant differences among the HF scenarios indicate that an erroneous estimation is still noticeable and can still have a detrimental influence. Thus, the robot cannot fully rely on being reactive and compliant to human actions. Estimating the human preferences accurately to plan appropriately the robot's actions is mandatory for optimal collaboration.

Looking at the eta squared η^2 of every question, one can notice that we can group the

	ANOVA		Bonferroni Post hoc tests													
	p	η^2	1-2	1-3	**1-4**	1-5	**1-6**	2-3	**2-4**	2-5	**2-6**	**3-4**	3-5	**3-6**	**4-5**	4-6	**5-6**
Responsive	***	*0.16*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*
Competent	***	*0.49*	*ns*	*ns*	***	*ns*	***	*ns*	***	*ns*	***	***	*ns*	**	***	*ns*	*
Intelligent	***	*0.33*	*ns*	*ns*	**	*ns*	*	*ns*	**	*ns*	**	**	*ns*	***	*	*ns*	*
Positive	***	***0.65***	*ns*	*ns*	***	*ns*	***	*ns*	***	*ns*	***	***	*ns*	***	***	*ns*	***
Simple	***	*0.34*	*ns*	*ns*	**	*ns*	**	*ns*	**	*ns*	*ns*	*	*ns*	*	**	*ns*	*
Clear	***	*0.41*	*ns*	*ns*	***	*ns*	***	*ns*	**	*ns*	***	***	*ns*	***	*	*ns*	**
Adaptive	***	***0.62***	*ns*	*ns*	***	*ns*	***	*ns*	***	*ns*	***	***	*ns*	***	***	*ns*	***
Useful	***	*0.43*	*ns*	*ns*	***	*ns*	*	*ns*	***	*ns*	***	***	*ns*	**	***	*ns*	***
Efficient	***	***0.62***	*ns*	*ns*	***	*ns*	***	*ns*	***	*ns*	***	***	*ns*	***	***	*ns*	***
Appropriate	***	***0.62***	*ns*	*ns*	***	**	***	*ns*	***	*ns*	***	***	*ns*	***	***	*ns*	***
Accommodating	***	***0.68***	*ns*	*ns*	***	*ns*	***	*ns*	***	*ns*	***	***	*ns*	***	***	*ns*	***
Predictable	***	*0.45*	*ns*	*ns*	***	**	***	*ns*	*	*ns*	*	***	*	***	**	*ns*	**

Table 6.6: Significant differences in the questionnaire answers between the different scenarios. For each item of the questionnaire are shown the overall p-value and η^2 (effect size) obtained after an ANOVA. Additionally, the p values obtained after conducting Bonferroni Post-hoc-Tests are shown to identify in a pair-wise manner which scenarios were significantly different from others. As depicted, scenarios 4 and 6 are distinguishable from the others and their evaluation are significantly different on all the measured aspect (expect reactivity).

items into 4 groups.

1. $\eta^2 = 0.16$: First, the Reactivity item is alone with the lowest effect size. This group doesn't help to distinguish the regimes.

2. $\eta^2 \simeq 0.33$: Secondly, we can state that when using RF the robot was perceived as slightly less intelligent and the interaction as slightly less simple than when using HF.

3. $\eta^2 \simeq 0.45$: Then, due to the moderate effect size, we can state that when using RF the robot was perceived as moderately less competent, the interaction as moderately less clear, the collaboration as moderately less useful, and the robot actions as moderately less predictable.

4. $\eta^2 \simeq 0.64$: Eventually, with a high effect size, when using the RF regime the interaction was perceived as significantly less positive, the collaboration as significantly less adaptive and efficient, and the robot actions as significantly less appropriate and less accommodation. Since the five items from the last group are the ones with the highest effect size, they can be seen as the main characteristics differentiating the HF from the RF regimes and thus are highlighted in the table.

6.5.5 From comments

At the end of the experiment, every participant was asked two questions gathering their general impressions. First, they were asked to comment on the overall experiment and robot interaction they just had. Secondly, participants were asked to indicate which scenario they preferred the most and the least.

Participants' comments concern several aspects of the experiment and are worth discussing. Overall, the comments confirm the outcome of the statistical analysis, and they also provide some feedback about the overall experiment protocol and conditions, especially regarding the simulator itself. The comments are discussed per categories below.

TODO: Try to add more argumentation and links to previous results, be less descriptive...

6.5.5.1 Simulation

First, most of the participants found the experiment and the simulation as a good experience. They felt committed and active during the different scenarios. The simulation has been described several times as clear, simple, pleasant, intuitive, captivating, and funny. Some participants mentioned that it was like a video game and enjoyed it. These comments suggest that collaborating with a simulated robot has been appreciated, and they raise the question if the participant had felt the same way in an experiment with a real robot.

6.5.5.2 Display

Some participants think that the simulation display had too much information (goal + scene + text prompt), and a few had trouble reading the text prompt. Yet, the robot can be seen well. Indeed, the text prompts could be a bit fast and written white on black, which can be disturbing when not used to. However, I believe this didn't affect much the executions, maybe slightly the decision times.

6.5.5.3 General

A few participants would have appreciated the robot giving instructions and guidance regarding the actions to perform. This is linked to another comment saying that using the Robot-Fist regime with correct estimation felt better because makes the task simpler. If the human trusts the robot, it can be appreciated to let the robot compute the optimal plan and just follow the robot's instructions, lowering the cognitive load of the human. Indeed, some participants consider that they made mistakes and that they could have acted in a better way in some scenarios.

6.5.5.4 Steps

The step synchronization wasn't appreciated by everyone. Some participants found this kind of synchronization useful as it structured the collaboration. However, many others found this a bit confusing at first and frustrating because they had to wait for the robot's actions to be done before being able to act again.

6.5.5.5 Task

The task was found clear and quite simple. One participant said that they felt significant emotions such as satisfaction and frustration and that if the task was less abstract and more real, these emotions would have been enhanced. Moreover, another participant said that in such simple tasks, humans tend to think they know better how to solve the task than the robot. Thus, the robot should follow human decisions in such cases, with a hierarchy relation. A few participants also mentioned that performing the last action, i.e. placing the last cube, is very satisfying. These people liked the robot adapting to allow them to do so. The fact that both agents must perform actions to solve the task makes the collaboration relevant and useful.

6.5.5.6 Action

Many participants said that not being able to pick cubes in advance isn't natural and, at first, it is confusing, frustrating, and a bit complicated. Yet, they also said that they got used to it quite fast. One also said that they felt obliged to act at every step. Indeed, a majority of the participants were signaling their passivity to the robot even when they were not able to act. About the movements, one participant stated that the actions were stiff and rigid, in contrast to being able to drag and drop the cubes thanks to the physics simulation. Additionally, the lack of collision with the cubes felt a bit unrealistic but not very confusing. On the other hand, another participant said that the robot's movements seem real. This is probably due to the fact we used an online motion planner to move the robot arm, thus, the movements were not always optimal.

6.5.5.7 Objective

A few participants said that the objective of "trying to be free early" is a bit frustrating since they would like to keep acting, even if not necessary. It was hard for them to consider this objective as their personal preference, and thus, to act accordingly. Additionally, one participant mentioned that Scenario 5 creates double satisfaction: being free early (preferences) and the task is fulfilled.

6.5.5.8 Regimes

One participant said that they didn't see much difference between the two execution regimes HF and RF. The same participant couldn't indicate which scenario they preferred at the end. Moreover, some participants also indicated that the difference between HF and RF was unclear at first. However, once used to the task and the scenarios, the difference becomes clearer and before the end of the experiment, most of the participants had a clear idea of each regime, and even apprehended the RF one.

6.5.5.9 Being in control

Participants indicated that when using HF they felt in control, free to decide which action they performed, and that the robot was adapting to their decisions and actions, which was appreciated. In contrast, when using RF, participants didn't feel in control and were forced to adapt to the robot's decisions. Even when the robot's decisions are good, the lack of control is uncomfortable. One participant stated that they disliked when the robot took initiative because the robot could be wrong.

6.5.5.10 Human-First (HF) regime

Most of the participants enjoyed the HF regime and stated that they were able to fulfill their objective with it. Some comments qualify the HF regime as slower than RF and sometimes inconsistent. The latter is mostly referring to the robot picking up the pink bar in S5. However, especially when used to it, HF has been qualified as smooth, efficient, interesting, predictable, and less frustrating than RF. Several participants mentioned that they enjoyed being able to predict the robot's behavior, proving that having predictable behavior is crucial for a seamless collaboration. It has been mentioned that HF makes less wrong choices than RF. Moreover, some participants said that compared to the RF regime with a correct estimation HF is less efficient, however, in pairs B and C HF is more efficient than RF.

6.5.5.11 Robot-First (RF) regime

RF, bad, not in control, bad choices: having to drop bar + using common resources first

Every participant had an overall negative opinion regarding the RF regime. The latter has been qualified as very frustrating, confusing, constraining, unpredictable, inefficient, and even adversarial. A significant number of participants stated that to finish the task quickly RF could be great, fast, efficient, and less cognitively demanding, despite the lack of control. Also, a few participants noticed that even if during a specific step the human is forced to comply with the robot's actions, in the next step the robot takes into account the human action and adapts its behavior. However, it has been said that RF doesn't consider the human's objective or preferences. Participants really disliked when the robot forced them to drop a cube back on the table (pink bar in S6) and when the robot picked cubes in the middle zone instead of its own zone. The latter was perceived as the robot stealing the cubes from the human. Due to those frustrating robot actions, the RF regime was putting the participants in an adversarial setup and the robots were explicitly qualified as "enemies". Some participants said that they were more focused on preventing the robot's mistakes than on the actual task.

6.6 Discussion

TODO: to develop

There are several elements to discuss in this study, including several participants' comments.

- simulation to real life ? what would it take ? different results ? One in between solution is using VR. Requires flexible execution scheme like described in limitations in chapter 5.
- Simu not realistic enough (pass trough cubes, no orientation, etc..): since focused on decisions, was enough but we could have done better.
- Text prompt: Which info to prompt ? when ? and how (sometimes hard to read). maybe vocal would have been better ? Here no specific study has been conducted, we simply show to the participants some inner state of the robot (waiting for human, acting, identifying human action, ...)
- A different task ? only manipulation ? no navigation ? Here the task carefully design to not be too long and still be significantly influenced by pairs of pref-esti
- Using an additional baseline with an erratic robot could have helped showing RF not so bad.

6.7 Conclusion

TODO: be more specific which conclusion come from which data? preferences satisfaction from log, adjectives from questionnaire, from comments ? confirmation + discussion materials.

Thanks to this study, we aimed to validate the overall planning approach and the model of execution Human-First, which is critical to our approach.

After statistically analyzing the execution log data as objective metrics and the questionnaire answers and participants' comments as subjective metrics, we can confidently state that this study successfully validates both our planning approach and our model of execution.

Indeed, we have solid proof that the HF regime gives humans control over the execution, which was significantly appreciated. The participants perceived the robot as accommodating, adaptive, and acting appropriately while being predictable. HF also helps to satisfy better human inner preferences, which makes it more robust to erroneous estimations and thus more enjoyable. On the other hand, we show how the RF regime can be greatly appreciated when estimating human preferences correctly. However, we demonstrate how erroneous estimations strongly harm collaboration and interaction using the RF regime. Hence, the Human-First regime is preferred and allows for achieving smooth, efficient, and positive collaborations. Nevertheless, thanks to our planning approach, we also show that the RF regime always solves the task with humans, and thus, it is always helpful. Additionally, it also always adapts to human action in the next step.

Human-Aware Task Planning Conclusions

TODO: TASK PLANNING IS CONCLUDED

Part II

Social Navigating Agents Simulation

Chapter 7

Challenging Robot Navigation Systems by Simulating Intelligent Human: InHuS

Contents

7.1 Introduction

TODO: update, currently from copy/paste from SSC

Significant efforts are being dedicated today toward the development of robots that interact, assist or work side-by-side with humans. However, people working in the field of human-robot interactions (HRI) face constraining issues while testing and evaluating their systems. Apart from being mandatory to validate mature systems, experimenting using real humans and robots is burdensome: they are slow, hardly repeatable, expensive, etc. Moreover, the system needs to be run extensively for debugging and tuning before it reaches maturity. Doing so with real-life experiments is generally a long and tiresome process where colleagues in the lab and volunteers spend unproductive hours, if not days, interacting with a robot running a system under debugging. Moreover, such methods require exclusive physical access to the robot, a place to run the tests, and cannot run faster than real-time or be parallelized.

Simulations are well suited for such tasks as they allow working without a real robot or a physical space. Further, they allow multiple tests to run simultaneously and faster than in real life. The simulated environment for the tests can be changed very easily in contrast to real-life tests. However, simulating realistic human behaviors and interactions is tough, which could make simulations unreliable. Consequently, HRI researchers face some difficulties such as: "How to test repeatedly and intensively their systems even when they are not sufficiently robust?" and "How to challenge their systems in a large variety of environments and situations?". Therefore, there is a need for an "intelligent artificial human" that would help challenge the robot's interactive and decision-making abilities.

7.1.1 Human simulations in human-aware robot navigation

Being a part of HRI, the field of human-aware social robot navigation inherits all these limitations. One way that is employed for simulating an intelligent avatar in this field is to manually control the human avatar, in real-time [Echeverria 2012]. This can be done using a variety of devices like a gaming controller, keyboard, or motion capture. Such approaches require a real human operator only focused on controlling the avatar, which brings back some mentioned limitations like human fatigue. Autonomous human avatars on the other hand seem to offer an adequate solution to this, but they often lack intelligence and rationality.

Most of the current autonomous avatars available are either scripted or reactive. A scripted avatar executes a series of predefined actions, like following a fixed path, without being reactive to its environment which makes interactions very limited. Reactive agents use models like social force [Helbing 1995] or optimal reciprocal collision avoidance (ORCA) [Van Den Berg 2011]. These systems are highly scalable and can simulate groups or even crowds composed of numerous agents. MengeROS [Aroor 2018] and PedSim_ROS[1] are some examples. Despite their number, the generated agents usually fail in intricate social scenarios. Some recent works like VirtualHome [Puig 2018] and SEAN [Tsoi 2020, Tsoi 2022] discuss simulating human agents to challenge robot systems, but the navigation of the agents in these systems is still based on reactive-only models. The work presented in [Yige 2021] proposes a learning-based method to generate more realistic pedestrian navigation. This ongoing work shows an interesting navigation behavior like waiting and letting the other agent pass embedded in iGibson [Shen 2020] simulator. However, this work is more focused on motion generation than decision-making to solve conflicts.

We propose the InHuS System to contribute to the lack of intelligent and rational human agents with conflict-resolution skills to challenge the human-aware robot navigation systems. Our contribution includes 1) an intelligent human agent controller, 2) a high-level interface to control the simulated agents, and 3) a GUI to plot execution data and metrics for evaluating the interaction. Such a system could help people working in the field of human-aware robot navigation to test and debug their schemes. Our system is designed to run, analyze and evaluate repeatable and long navigation scenarios involving a robot and an autonomous reactive and rational avatar. This work focuses on intricate and narrow scenarios where, in addition to being reactive, rational decisions should be taken in order to solve the conflicts occurring. Note that our contribution is focused on navigation decision-making and not the motion generation part. Throughout this paper, we use the term 'rational' in a meaning close to Goal Reasoning [Vattam 2013, Johnson 2018], i.e., the ability of autonomous agents which can dynamically reason about and adjust their goals.

[1] https://github.com/srl-freiburg/pedsim_ros

It enables the agents to adapt intelligently to changing conditions and unexpected events, allowing them to address a wide variety of complex situations.

7.2 Description

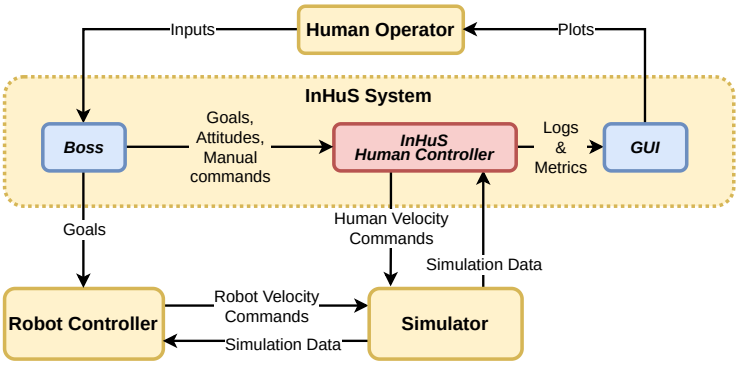


Figure 7.1: The InHuS System interacts with three external systems: the simulator, the robot controller, and a human operator. Our system is separated into three parts: the Boss high-level interface gathering inputs from the human operator, InHuS which is the actual human controller, and a GUI to plot the metrics and other data produced.

The InHuS System[2] works along with a human operator, a chosen simulator, and the challenged robot controller as depicted in Fig 7.1. The system is mainly implemented using ROS. The InHuS System is three-sided. First, the system comes with a high-level interface called Boss that helps to manage the simulated agents. Secondly, there is the main part which is the intelligent human avatar controller itself, called InHuS. Finally, a GUI provides an interactive visualization of the data and metrics computed by InHuS during execution that can help to evaluate interactions. We present below some details for each component.

7.2.1 Boss

For the human operator to easily control the simulated agents and run repeatable scenarios, we provide a simple graphical user interface component called Boss. Predefined or manually entered goals can be sent to the human, the robot, or both. Goals are by default considered as "Pose goals" that only require one navigation action to be achieved. However, the human agent (only) can handle "Compound goals" that need a specified sequence of navigation and waiting for actions to be achieved. This type of compound goal is useful to emulate more complex activities. For example, "Make coffee" could be described as a sequence of three actions: nav(coffeeMachine), wait($15s$), nav(myOffice).

The Boss allows defining scenarios with start positions and goals for each agent to repeatedly generate the same situation. Running a scenario consists of first sending each agent to their respective starting position. Then, the corresponding goals are sent to the human and the robot. A delay can be specified while starting the scenario to delay either the robot's or the human's goal. This is very useful to adjust the timing of a specific situation or conflict. The Boss can also put an agent in "endless" mode where the agent continuously gets a new goal from a given list after completing one.

[2]https://github.com/AnthonyFavier/InHuS_Social_Navigation

Each navigation action can specify a radius for the "Pose goal", within which a new "Pose goal" is randomly sampled. This mechanism adds randomness to the execution and diversifies the situations encountered, especially in the "endless" mode. Setting the radius to zero disables the randomization and selects the given goal.

All the goals, scenarios, and endless goal sequences are defined using an XML format. Hence, defining new goals or scenarios is straightforward. There is an XML goal file associated with each map/environment. Thus, it is easy to switch between environments since the corresponding goal file is automatically loaded.

7.2.2 InHuS

The macro component InHuS is mainly in charge of controlling the avatar and generating rational behaviors. InHuS itself is made of several components as depicted in Fig. 7.2. However, three components, namely HumanBehaviorModel, Supervisor, and GeometricPlanner, constitute the major functional part of InHuS. We discuss each of these major components in detail.

7.2.2.1 HumanBehaviorModel:

The HumanBehaviorModel is responsible for most of the rational behavior of the agent. The first role of this component is to manage the goals. Goals can either be received from the Boss component or generated by the HumanBehaviorModel using the same XML file as the Boss. When a goal is selected, it is sent to the Supervisor for execution.

This component is also responsible for detecting and handling navigation conflicts. Currently, the kind of navigation conflict handled by InHuS is path blockage (e.g. another agent standing in a doorway). While the human agent is navigating, a path to the goal is calculated at regular intervals using Dijkstra's algorithm, and its length is tracked to detect such conflicts. If the tracked path length increases significantly or the path ceases to exist, it could mean that another agent is blocking either the only possible way or the shortest way. When such situations are detected, the plan execution is temporarily suspended, and the agent performs an approach action to get close to the blocking location. This shows the agent's intention to move in a specific direction and might induce the blocking agent to react and clear the way. Eventually, once the avatar is at a specified distance of the blocking location, here set to 1.5 m, the agent stops its approach and actively waits for the path to be cleared.

To generate a lot of different and specific situations, we created what we call *Attitudes*. They are operating modes affecting both goal decisions and reactions toward the other agents. One can activate them through the Boss to generate diversified behaviors of the agent. Some of the *Attitudes* currently implemented in InHuS consist of: 1) randomly

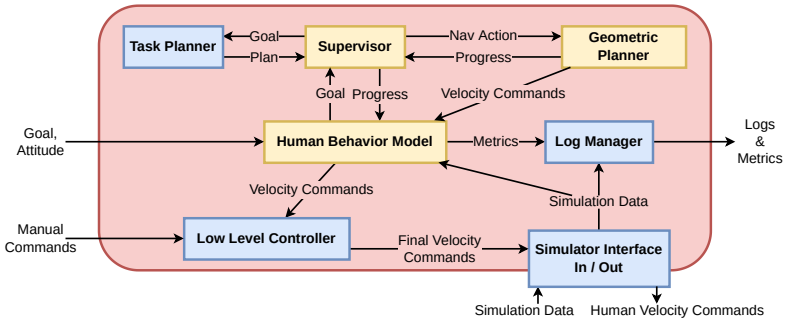


Figure 7.2: The human controller InHuS with its components and subsystems.

picking a new goal, like someone suddenly changing their mind, 2) harassing the robot by constantly going in front of it, like a child would do [Nomura 2016], and 3) stopping close to the robot and looking at it for a few seconds before resuming its goal which emulates a curious behavior.

The final purpose of this component is to build the perception of the human agent based on the map and information about the other agents from the simulator. We build the perception by directly accessing the simulation data rather than adding simulated sensors to the human avatar. Using this perception, we compute the visibility of the human agent and then update the human's knowledge about the robot's position and speed.

7.2.2.2 Supervisor:

The Supervisor is a central component as it coordinates different components to execute the plan and achieve the current goal. When the Supervisor receives a goal from the HumanBehaviorModel, it requests the TaskPlanner component a plan to achieve the goal. For now, the plan generation is quite simplistic. For a "Pose goal", a plan filled with a single navigation action is generated. For a "Compound goal", the sequence of navigation and waiting actions is extracted from the XML goal file and the plan is populated. Despite the simplistic plan generation, this architecture handles complex goals that require several steps to be achieved and emulate human activities.

The execution of each action of the plan is then supervised by the Supervisor by sending requests to other components. When a navigation action needs to be performed, the Supervisor starts by sampling a random position if the given action radius is not zero. Then, it requests the GeometricPlanner to plan for the target position without considering other agents initially. This way, the avatar starts following the shortest path, and we initialize the conflict detection. After this, the system starts to consider the other agents, and the Supervisor periodically requests the HumanBehaviorModel component to check for potential navigation conflicts. The Supervisor can suspend and resume the plan execution at any time, which can be used to resolve the detected conflicts or to generate specific reactions like the *Attitudes*.

7.2.2.3 GeometricPlanner:

The last major component is the GeometricPlanner. This motion planner component receives a target position to reach from the Supervisor and generates velocity commands to make the avatar move. This component defines how the agent moves around and adapts its velocity to the other agents in the scene. Since the system is implemented in ROS, we use the standard ROS navigation stack for the GeometricPlanner.

The planner used in InHuS is a publicly available human-aware navigation planner called CoHAN [Singamaneni 2021]. It is built over the ROS navigation stack and uses a local planner based on a modified version of the timed elastic band with human-aware properties. We benefit from the high-level decision-making of InHuS and the enhanced local navigation of CoHAN with trajectory predictions. Moreover, CoHAN is highly tunable which helps to generate different agent behaviors.

7.2.3 Logs, metrics and GUI

The InHuS system logs the execution data like the positions and speeds of the agents along with some computed metrics. All the logged data is sent to the GUI component, which generates interactive plots. These plots can help evaluate the interaction and thus the

performance of the given robot controller. The snapshot of the GUI shown in Fig. 7.3 shows two kinds of visualizations. On the right side, there is a colored visualization of the paths taken by each agent. These paths are colored over time according to a corresponding legend that helps estimate an agent's position at a specific moment. The left side is composed of several plots showing some computed metrics over time. The first plot is about conflict detection and solving. It shows the length of the path to the goal computed when checking for conflicts. Without any conflict, the path length should decrease linearly over time. If it's not the case, the avatar has been disturbed during the navigation. This plot also shows the conflict state of the agent: Nominal (no conflict), Approach (conflict detected), Blocked (stopped and waiting). The subsequent plots show over time the speeds of each agent, their relative speed, the distance separating them, and a metric called time to collision (TTC). This metric estimates the time remaining before the agents collide with their current velocities. We can argue that TTC corresponds to a "threat feeling" since a low TTC value corresponds to a high threat of collision. Hence, social robots should be tuned to not exceed a minimum TTC value to make humans more comfortable.

7.3 Main results

able to numerically identify the HA behavior of CoHAN w/r SMB)

In this section, we show some results through a set of experiments to highlight how our system can help challenge human-aware robot navigation systems. First, we discuss the limits of reactive-only systems to strengthen the need for rational avatars. Then, we present how our system effectively challenges robot navigation systems and we interpret the corresponding plots. Next, we show how the InHuS System can compare the human-aware performances of two different robot controllers. Finally, we present additional experiments showing the diverse behaviors that can be produced using the *Attitudes*, and how "long runs" can benefit the development of a robot controller.

7.3.1 Limits of reactive-only agents

Most of the current human agent simulations used by the social navigation community rely either on the social force model or ORCA. In order to highlight the limitations of such

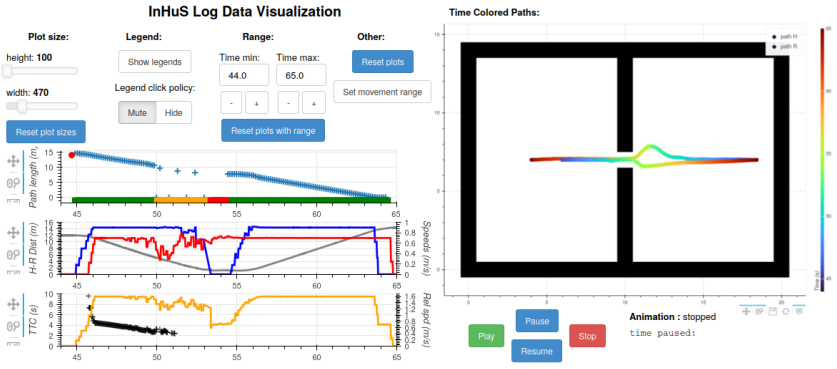


Figure 7.3: Overview of the GUI interface which is organized as follows. On the right side are shown the paths taken by the agents and colored over time. On the left side, several metrics and data produced by InHuS are plotted over time on graphs. Additional widgets help to configure the plots.

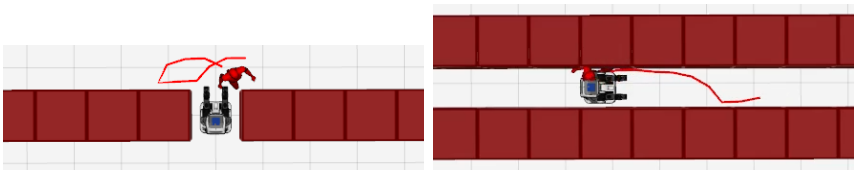


Figure 7.4: In the doorway scenario (left), the reactive-only (Pedsim) agent never stops moving while trying to go through the robot even though its path is blocked. In the narrow corridor scenario (right), the agent squeezes itself between the wall and the robot colliding with both.

approaches, we present results obtained with a PedSim_ROS (or simply PedSim) agent. PedSim is a pedestrian simulator that uses the social force model. It is very efficient for generating crowds to test robot navigation. However, at the individual level, the simulated agents are purely reactive and have no decisional abilities like most pedestrian simulators.

Consider the doorway scene shown in the left part of Fig. 7.4. Both agents have to cross a narrow opening. Here, the robot is blocking the way that the human agent intends to cross. The PedSim agent approaches the robot and tries to push itself through, but it fails due to a very high value of social force. The agent never stops moving and tends to go right or left along the wall before wiggling again just in front of the robot. This confusing behavior can make the agent's intentions unclear to the robot planner. The narrow corridor scenario, shown in the right part of Fig. 7.4, also exposes some limits. In this scene, there is not enough space for the agents to cross each other, and the only solution is for one of them to back off. Here the path is blocked by the static robot. The PedSim agent slowly gets closer and closer to the robot before squeezing itself between the wall and the robot. For some reason here the social forces allowed the agent to pass, unlike the previous example. It highlights that the PedSim agent doesn't use a defined hitbox or footprint for the agent and relies only on repulsive social forces to prevent collisions. This lack of defined collision shapes makes the agent temporarily pass through the walls and other agents. As a consequence, it breaks many intricate scenarios where a rational decision should be taken and results in unrealistic situations. Despite being efficient for large spaces or crowds, based on the above observation, we can state that in intricate scenarios such approaches can lead to confusing and even unrealistic behaviors.

7.3.2 Interpretation of plots with human-aware planner

The InHuS System is able to generate challenging situations and associated logs to allow further evaluation. Here, we present one such conflict and a detailed interpretation of the corresponding plots. The plots were produced while challenging the CoHAN system in the doorway scenario.

The robot starts closer to the opening and enters the doorway first. The execution can be analyzed with the metric plots and the time-colored paths of the agents in Fig. 7.5. We notice that the robot's speed (red line on the second graph) goes down around 50 s as it is entering the doorway and creating a conflict. The conflict is detected by InHuS (zero path length = no path), and the agent switches to the approach state (green to the yellow line on the first graph). The non-zero path length in the approach state corresponds to how the approach is performed. In order to keep moving despite the blocked path, the GeometricPlanner is requested at a defined frequency to plan without considering the robot (all non-zero path length). In between these requests, to check if the path is still blocked, the conflict detection plans while considering the robot (zero path length). When the avatar

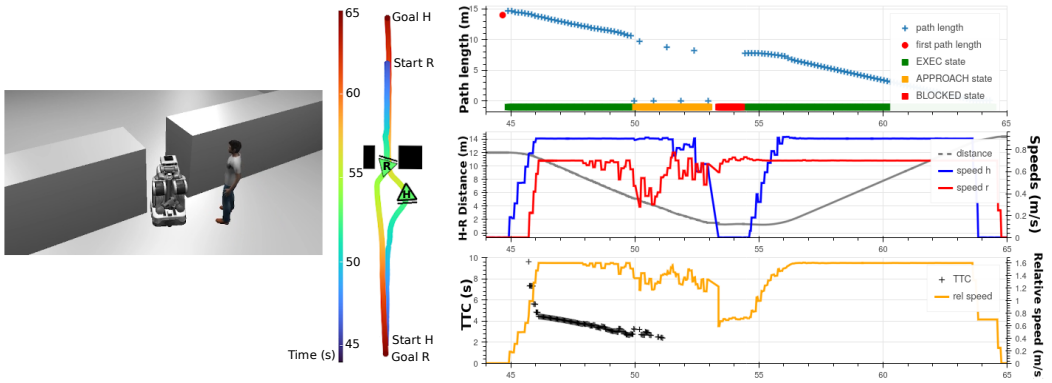


Figure 7.5: A condensed view of the InHuS GUI and MORSE simulator for the doorway scenario with a robot running the CoHAN planner. Several plots depict the detection and resolution of the conflict created.

is at a predefined distance from the blocking robot around 53 s, it switches to the blocked state (red line) to stop and wait for the path to be cleared. Further, the time-colored paths show that the GeometricPlanner made the avatar move aside while approaching to avoid blocking the robot. As a result, the agents were no longer moving towards each other, and thus, there was no longer any collision threat (no TTC values). When there is no more collision threat, around 51 s, the robot's speed starts to increase again. Such behavior is a good sign of human-aware properties and might increase human comfort.

From the plots produced by our system, a lot of useful information can be extracted for improving or evaluating the social robot planner's performance like a) finding ways to decrease the blocked state time for the human, b) maintaining a particular threshold for TTC, c) slowing down near the human, or waiting for the human to cross the door without blocking.

7.3.3 Quantitative comparison between two robot controllers

Our system can be used to run similar scenarios repetitively to produce robust metric values. These values can help to evaluate the human-aware performances of a given robot controller. To show this, we present a comparison between two different robot controllers. The first one is again the CoHAN system, and the second one is the Simple Move Base (or SMB). It uses the *teb_local_planner* and the ROS navigation stack with default parameters. We just add an additional process to consider the human agent as a static obstacle to avoid it, so it is not human-aware. Therefore, we should be able to notice a clear difference through the metrics computed by our system. For this comparison, we used three different scenarios: 1) The doorway scenario where the agents have to cross a narrow opening, 2) the corridor scenario where the agents cross each other with just enough space, and 3) the open space where they cross each other without any environmental constraints. We performed 10 repetitions of each scenario for each robot controller. For each set of 10 repetitions, we extracted the mean values of three different metrics and presented them in Table 7.1. The metrics are the following. First, the time to goal (TTG) is the time taken by the avatar to reach its goal. Second, the minimum distance between the robot and the human (Min HRDist). And, the minimum time to collision (TTC). Intuitively, we want the TTG to be as small as possible, the minimum HRDist to be as high as possible, and since a low TTC value represents a collision threat, we want the min TTC to be as high as possible.

At first glance, we see in Table 7.1 that almost all CoHAN values are better than

Scenario	CoHAN			SMB		
	TTG (s)	*min Dist (m)*	*min TTC (s)*	*TTG (s)*	*min Dist (m)*	*min TTC (s)*
Doorway	18.38	**2.32**	**1.33**	**18.26**	2.23	1.16
Corridor	**16.34**	**2.06**	**1.03**	17.05	1.59	0.81
Open Space	**9.55**	**2.52**	**1.61**	11.01	2.34	1.18

Table 7.1: Mean values of three InHuS metrics over 10 repetitions in three different scenarios and with two different robot controllers. Bold values indicate when the corresponding robot controller has better performance than the other.

SMB values. Due to the nature of the doorway environment, the execution of the scenario is quite constrained which explains why the values are not too different between the two controllers. However, we notice anyway that, compared to SMB, the CoHAN planner tends to keep a greater distance between the agents and a greater TTC (lower threat of collision). The time to goal of CoHAN is slightly higher because the robot slows down when crossing and moving in the direction of the human. Thus, it is the price to pay in this scenario to maintain adequate TTC values.

In the corridor scenario, The SMB robot tends to wait until the last moment to move aside, which is threatening. On the other hand, the CoHAN robot proactively moves to one side of the corridor. As a consequence, it leaves more space for humans and reduces the threat of collision, which is visible in the obtained values. Also, this pro-activity has the effect of smoothing the trajectory of the avatar, which makes this last one reach its goal faster.

Finally, the open space scenario is a bit similar to the previous one. The SMB robot waits until the last moment to avoid the human, which puts the load of the avoidance maneuver on the human. As a result, the human has to move aside which extends the duration to reach the goal. Also, due to the same behavior, the SMB robot is on average closer to the avatar and more threatening. Since the CoHAN robot moved again aside early, its metric values are noticeably better than SMB.

In summary, the human-aware behavior of the CoHAN controller was captured through significant value differences in the computed metrics compared to a non-human-aware robot controller. This implies that our system can help evaluate and compare human-aware robot controllers.

7.3.4 Generating different behaviors with Attitudes

By activating *Attitudes*, InHuS is capable of producing more complex behaviors to diversify the conflicts and challenges imposed on the robot. We present the time-colored paths for the execution of two *Attitudes* : *Harass* and *StopAndLook* in Fig. 7.6. Concerning the *Harass Attitude*, by paying attention to the colors, we see that the human is always in front of the robot that continuously tries to avoid the harassing agent causing erratic movements. The robot should be able to detect such non-cooperative behavior from humans and act accordingly. On the same figure we see the execution of the *StopAndLook Attitude*. The color discontinuity behind the human marker shows how the human suspended its goal to stop and briefly stare at the robot before moving again. A robot not pro-active enough could be disturbed by the sudden stop of the human, which could be a situation of interest to handle.

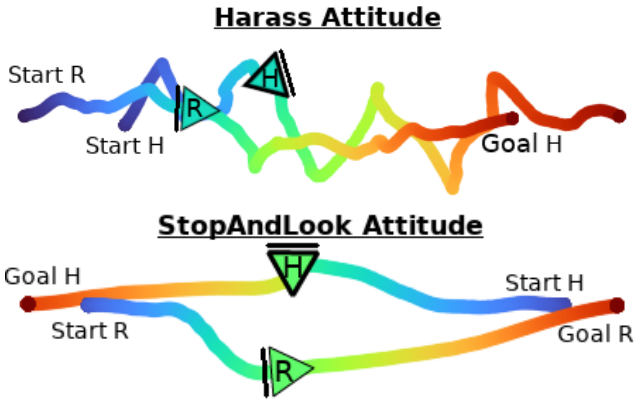


Figure 7.6: Behaviors obtained by activating the *Harass* and *StopAndLook Attitudes*. With *Harass*, the human is always in front of the robot. With *StopAndLook*, when close to the robot, the human stops to look at it for a few seconds.

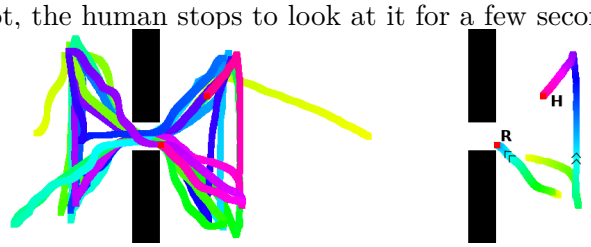


Figure 7.7: Execution of the long run scenario using the TDP robot planner and InHuS. We see the complete set of time-colored paths on the left. On the right, the same path is cut, around the moment when the robot got stuck in the wall.

7.3.5 Long run scenarios

The proposed system can help test the stability and robustness of the robot planner by conducting long randomized runs. Indeed, thanks to the Boss component, possibly randomized goals can be sent autonomously to the agents. This can generate unexpected situations and conflicts that can be of interest. Fig. 7.7 depicts such a test conducted with InHuS and a human-aware robot planner from Kollmitz et. al. [Kollmitz 2015] here referred to as TDP. The agents were made to endlessly loop over four goal-positions (each with a 1 m radius) in reverse order to create as many conflicts as possible. After 3 minutes, the robot got stuck in the wall of the doorway, indefinitely blocking the path for the human. In addition to highlighting problematic situations where the robot doesn't act as expected, long runs can expose low-level issues like unexpected crashes or memory leaks.

7.4 Discussion and Limitations

Although InHuS provides an autonomous human agent, if needed, the agent can be controlled manually. We do not yet provide a handy controller, but velocity commands generated by any means can be sent to the Boss component to control the human. This extends the usability of InHuS as one can use scripted trajectories or motion capture to control the human agent in the simulator.

The proposed system interacts with an external simulator and robot controller. Since the system is mainly implemented using ROS, switching from one simulator to another is straightforward if it has a ROS interface. InHuS has specific components to abstract the

simulation data format. Thus, just by slightly editing these components we were already able to run InHuS on three different simulators: MORSE [Echeverria 2011], Stage[3] and Gazebo [Koenig 2004]. Furthermore, any robot controller using the ROS Navigation Stack can be directly used with InHuS.

Simulating intelligent human avatars is a novel field and only a few works apart from ours try to address this limitation. A similar work in ROS2 was recently presented in [Pérez-Higueras 2023]. It is clear that the idea of intelligent human agents is of interest to the community, and it is a necessity to test social navigation effectively. Like any other system, InHuS has limitations too. We claim to generate only reactive and somewhat rational behavior, which is still far from natural or realistic human behavior. We currently handle scenarios with two agents only, the human and the robot. We can run scenarios with other human agents, but they will be treated like robots.

7.5 Conclusion

TODO: update, copy/pasted from SSC paper

Human-aware social robot navigation is rapidly growing, but the community lacks good human agent simulations to test and debug their systems. The existing reactive approaches offer only limited testing. Through the InHuS system, we proposed a pertinent approach to address this issue. We showed that our system could generate conflicting situations that need resolution by making rational choices. Moreover, all the metrics and data recorded during execution and their visual plots allow us to evaluate the interaction and behavior of the robot. With such evaluation, we showed that we could compare different robot controllers. InHuS can also generate various tunable behaviors that can diversify the situations and conflicts imposed on the robot, and thus, it helps to debug and tune the system. Long runs provide additional potential ways to improve the system.

We already use this system to test our human-aware motion planners and refine them over time using the tests conducted. In the future, we plan to integrate situation detection and diagnosis in the long run to catch the problematic situations that need to be analyzed afterward to tune, refine or extend a given planner. We also plan to handle scenarios with more human agents, like groups and crowds, using a combination of intelligent and reactive agents.

[3] https://github.com/ros-simulation/stage_ros

Chapter 8

Interactive Social Multi-Agents Simulation for Robot Navigation: IMHuS

Contents

This short chapter introduces an additional work called Intelligent Multi-Human Simulator (IMHuS) [Hauterville 2022b, Hauterville 2022a] which is strongly inspired by the InHuS framework presented in chapter 7. Together with two researchers from the University of Leon in Spain and an intern, we designed this framework based on InHuS. The major part of the implementation has been done by the intern. This system has been evaluated in the elevator scenario, as defined for the SciRoc competition 2019. The main purpose of IMHuS is to address one of the major limitations of InHuS which is not being able to simulate several human agents.

We begin with a comparison between InHuS and this additional work while briefly describing it. After, a more formalized presentation of IMHuS is provided detailing the information given in the prior comparison.

8.1 Comparison InHuS vs. IMHuS

8.1.1 Similarities

Let's first mention the similarities between InHuS and IMHuS work. Like InHuS, this work aims at replicating scenarios involving humans in order to help social robotics research. Similarly, the simulated interactive agents can be choreographed in a step-based manner to create high-level social behaviors such as waiting for an elevator, getting in and out of it, or standing in front of a store window and moving to the next one. The agents navigate in

the environment while avoiding static obstacles and being reactive to moving obstacles to prevent collisions. The agents can also wait for a defined amount of time or turn to look in a direction. Like in InHuS, *Attitudes* can be activated to generate specific behavior such as harassing the robot. Additionally, this work also analyzes the execution to compute metrics evaluating the robot's performance in defined social situations. The framework architecture is close to the InHuS one and can also be used with different robotic simulators, here it has been implemented with Gazebo.

8.1.2 Differences

However, IMHuS differs from InHuS in several ways. The major reason is that it manages several interactive agents instead of a unique one in InHuS. Moreover, the agents can exhibit social behavior using explicit social groups, for instance, they can move together to another location or talk in pairs. This requires the new *grouping* actions which create or dismantle groups of humans. An interesting addition is the implementation of asynchronous actions. They differ from synchronous actions, such as navigation/turning/grouping actions, which are accomplished by the agents during one specific step. Asynchronous actions are not associated with a specific step and correspond to a system of *request* and *response*. Thanks to them, one agent can request another to perform a specific task. For instance, the robot can request a human agent to call the elevator, or if supported, a human agent can request the robot to go somewhere. Not every agent can respond to the robot's requests. For instance, if the robot asks for someone to press the button and no one is around, the request will be dropped, counting as a failure.

In order to handle several human agents, some simplifications were made compared to the InHuS framework. First, when navigating, IMHuS agents are reactive to other agents. However, their movements are less smooth than the InHuS agent. This is because InHuS couples a frequent global path replanning taking into account static and moving obstacles and an elaborated local planner to follow the planned path. The local planner used is nothing else than the Human-Aware robot navigation planner CoHAN presented in this chapter. It also allows the agent to have proactive avoidance movements. Also, InHuS uses frequent global path replanning to be even more reactive and to identify sudden path blockage due to other agents. Hence, instead of blindly following the global path it can identify when its optimal path is blocked and switch into a conflicted mode where it reasons on its goal to potentially adapt it. Currently, the agent approaches the blocked spot before stopping to wait for the path to be cleared. On the other hand, IMHuS agents don't use any local planner and simply move at a defined speed along the updated global path. This induces human agents to sometimes move abruptly and avoid obstacles at the last moment. In addition, IMHuS agents' actions are dictated by the given choreography and don't have individual goal reasoning processes like in InHuS. Hence, IMHuS agents are currently not capable of identifying path blockage situations and may behave erratically in such cases.

Yet, this new scheme is still very interesting because of the multitude of human agents generated and the social behaviors that can be choreographed. It can generate relevant challenging situations for social robotics to handle.

8.2 Rational

TODO to check

The goal of the tool described in this paper is to provide a means to generate scenarios in which predefined social interactions of groups of reactive humans can be used to test the

social performance of a robot behavior under evaluation. In the rest of the paper, we will refer to it as *tested robot*. The aim is to use the system for benchmarking human-robot interaction behaviors.

The goal of IMHuS (Intelligent Multi Human Simulator), the tool described in this paper, is to provide an open-source toolkit for defining high-level reactive simulated humans with the ability to show the behavior of social groups but using realistic standard robotics simulators that allow researchers to use models of their real robots, both for debugging their algorithms and for benchmarking and repeatability.

8.3 Design of a step-based social simulation

The goal of IMHuS is to standardize the validation of the autonomous robot behavior in the presence of people, allowing researchers to design repeatable human social situations. For example, defining a set of waypoints where different simulated people arrive, meet and move as a group to a different position, or two people facing each other and then moving together to a different location.

Our proposal considers that both individuals and groups have to be included in the simulated environment and that in both cases, each simulated person should exhibit adaptive behavior. To achieve this goal, we assume that global path planning for the whole set of agents is more suitable for defining fixed social behaviors than the individual path planning approach. This assumption also serves the purpose that humans exhibit a common social behavior that the robot must be able to detect in order not to cross, for example, through the middle of a social group.

A central process can have perfect knowledge of the simulated environment, accessing the real and accurate positions of all the elements of the simulations (obstacles, robot, etc.). It is not designed to emulate an embedded agent, since the robot simulation process has to obtain the information through the noise-simulated sensor readings provided by the simulator. This process can use the API of the simulator to get all the information directly, without noise and identification problems (it will know which types of elements are in the simulation and their state). For instance, it does not need to identify if something is “a door" or if it is opened. The door state is obtained directly from the simulator.

8.3.1 Choreography oriented simulation

The constraints and assumptions made for IMHuS are:

1. The set of persons $P = \{p_1, p_2, ...p_n\}$ in the environment is defined *a priori*, and each person will be unequivocally identified by its ID (p_i) during each execution.
2. The number of locations $L = \{l_1, l_2..., l_m\}$ for these people is also known *a priori*.
3. The number of locations will be greater than the number of people $|L| \geq |P|$.
4. In a given time-step, only one person can stay at an individual location l_i.
5. Group locations L_i^n will have a maximum number n of individual positions $l_1, l_2, ..., l_n$. For instance, an elevator with four positions will be named L^4.
6. The number of actions is also finite and known.

The goal of the tool is to allow researchers to have a high-level definition of a “choreography” of people moving in a social way. For instance, let’s consider an example, first a set of people (p_1 to p_4) is defined. In a first step, p_1, p_2 and p_3 have to move from their initial

positions to a "group location" (L^3). This joint navigation is represented in figure 8.1 as a continuous top-opening box grouping the three people at time-step t_0 and a double arrow labelled with the goal destination. In the same way, p_4 will remain in its position but will face a particular direction (30 degrees), indicated by the circle with an arrow.

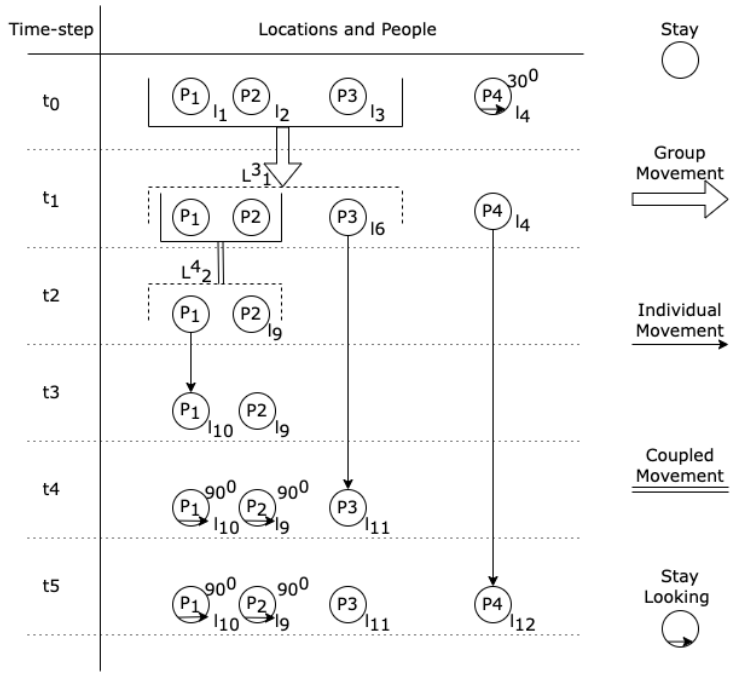


Figure 8.1: Schematic representation of the definition of a social navigation choreography

At t_1, p_1 and p_2 engage in a pair-move to L^4, while p_3 and p_4 initiate individual moves, indicated by a single arrow. A pair-move is a specific type of group move included in the design because it is usually the most common one and because it can be considered as the smallest group unit. At t_2, the two persons moving as a pair would have reached their destination and p_1 will initiate its movement towards l_{10}, while p_2 remains in its current position with no particular orientation (indicated by a circle wit no arrow), and p_3 and p_4 continue to navigate to their targets.

p_1 at t_2 begins to move individually towards l_{10}, which reaches at t_3 while p_3 and p_4 continue to execute their solo movement. p_3 reaches l_{11} at t_4, when p_1 and p_2 begin to face in the same direction. Finally, at t_5, p_4 reaches its destination (l_{12}) and the choreography ends.

Time-step (t_i) in figure 8.1 means "*choreography steps*", that is, significant moments for the definition of the simulation, it does not refer to a magnitude measured by a clock. We will refer to them as *steps*, which is an increased ordered sequence of discrete points in time at which a given set of events has to occur. For example, $step_0$ usually specifies the initial state of all the elements of the simulation, i.e., the position of the robot, human', and the other elements. A typical step specifies a set of actions that are initiated at that instant, a navigation task for one of the humans, for instance, at $step_1$ (t_1 in figure 8.1).

This is the type of simulation that the tool should be able to generate. Figure 8.2 shows the general architectural framework. This tool receives the definitions of the social scenarios in a definition file (an XML in its current version). It then manages the simulation,

obtaining the simulation data, using existing libraries and other tools (such as *move_base* for calculation of trajectories in ROS), and finally generates the log data for evaluation.

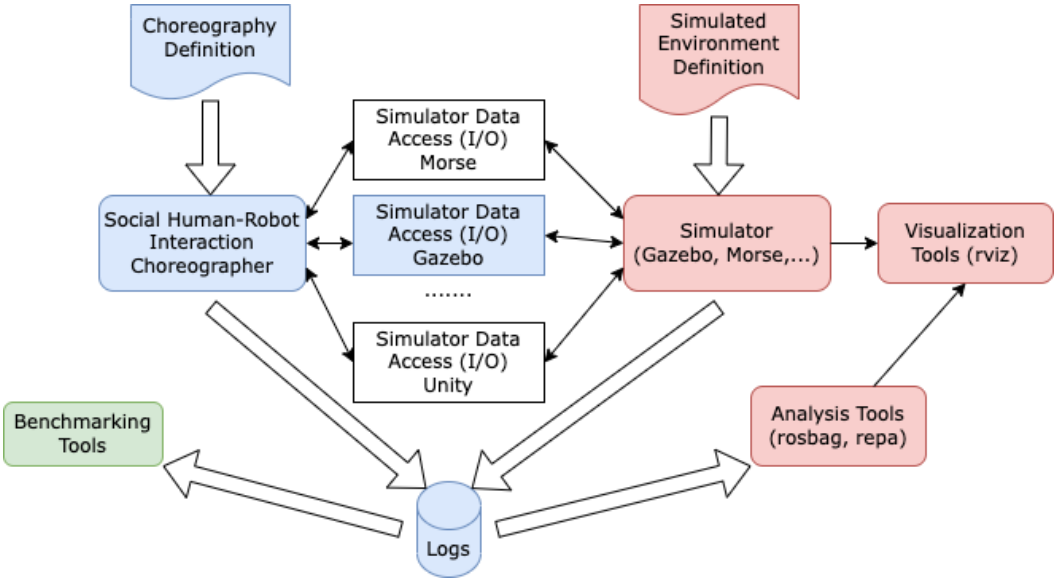


Figure 8.2: Global simulation architecture. Red signifies existing tools, blue components are those described in this paper, green signifies on-going development, and uncolored boxes alternative simulators under consideration.

The definition of the choreography shown in the upper left of the figure 8.2 is defined in XML. The design allows five types of *basic actions* that the *agents* (simulated robots or simulated people) can perform:

Navigation actions : Those actions modify the global pose of the agents in the environment. Typical actions in this group are $GoToPose$ and $Wait$.

Turning actions : Actions related to the facing of the agents, such as $LookAt$ and $Turn$ depending on the way the action is specified.

Grouping actions : These actions manage the creation and dismantling of groups.

Attitude actions : Actions modelling the high level behavior of the agent. For instance, a simulated human can be ordered to *harass* the robot.

Synchronous actions : Actions related to the environment to be accomplished by an agent during one specific step of the simulation. They have been standardized as *publish* and *subscribe*.

Asynchronous actions : Actions related to the environment and not associated with a specific step in the timeline of the simulation. They have been standardized as *request* and *respond*.

These actions can be applied to a single agent or a group. Basic actions can be integrated into a *compound_task*. For defining these actions, the following components are used:

- *map*: corresponds to the “world" where the simulation will happen. Inside the map a set of *objects* can be defined specifying their individual ID.

- *poses*: correspond to the “location" used in figure 8.1. They are made up by the x, y position and orientation θ, and a *radius* of tolerance for the motion planner to consider that the goal has been reached.
- *agents*: that can be either *robots* or *humans*, each of them identified by an unique ID. They are given an initial pose where they appear in the world.
- *groups*: they can be created and dismantled during the simulation

The evolution of the simulation is based on steps, as described in the previous section. The regular steps are synchronous, but there is an asynchronous step for interacting with the tested robot:

- A scenario is composed of a set of regular steps and a singular asynchronous step.
- Steps are made up of different actions like navigation, grouping, etc.
- Every action of a step is executed *at the same time*, meaning that the actions are executed in simulated parallelism.
- One step ends when all its actions have finished.
- The asynchronous step runs in parallel to the execution of the synchronous steps.
- The scenario ends when the last regular step ends.

8.4 Implementation

In the current version choreographies are defined in an XML file that includes four main sections:

- Map = poses and objects. Example: poses definition.

```
<poses> ::= <pose> (<pose>)* ;
    <pose> ::= <poseID> <x> <y> <theta> <radius> ;
```

- Agents = humans/choreographed robots with their initial poses, and groups with their composition. Example: groups definition.

```
<groups> ::= (<group>)* ;
    <group> ::= <groupID> <pair> (<human> (<human>)*) ;
        <pair> ::= <pairID> <human> <human> ;
```

- Tasks = generic actions. Example: compound tasks and look-at action definition.

```
<compound_tasks> ::= <compound_task>* ;
    <compound_task> ::= <compound_taskID> <action> (<action>)*;
<lookAt_action> ::= <actionID> (<humanID>|<robotID>|<objectID>);
```

- Scenarios = step elements of synchronous steps and asynchronous step. Example: scenario definition.

```
<scenario> ::= <name> <step> (<step>)* (async_step);
    <step> ::= <stepID> <stepElement>;
        <stepElement> ::= <agentsID> <pose>|<compound_task>;
        <agentsID> ::= <humanID>|<robotID>|<pairID>|<groupID>;
    <async_step> ::= <respond_event_action>;
```

The prototype has been implemented as a new version of the InHus tool ([Favier 2023]) which connects to the Gazebo simulator to obtain the information about the world. It uses the *move_base* ROS to generate the global plan for each agent and updates the positions of each agent in the next step of the simulation accordingly. The new version is capable of handling the navigation of multiple agents (humans) in parallel while managing conflicts and completing individual tasks as in the previous version. It also provides an updated graphical user interface for repeatedly selecting and executing scenarios defined in the XML file.

The IMHuS tool shown in Figure 8.3 has been structured in three layers: the IMHuS layer (in blue), its configuration in the application layer (in yellow), its communication with the simulator and ROS (in red). The robot whose behavior would be tested in the tool has also been included (in violet).

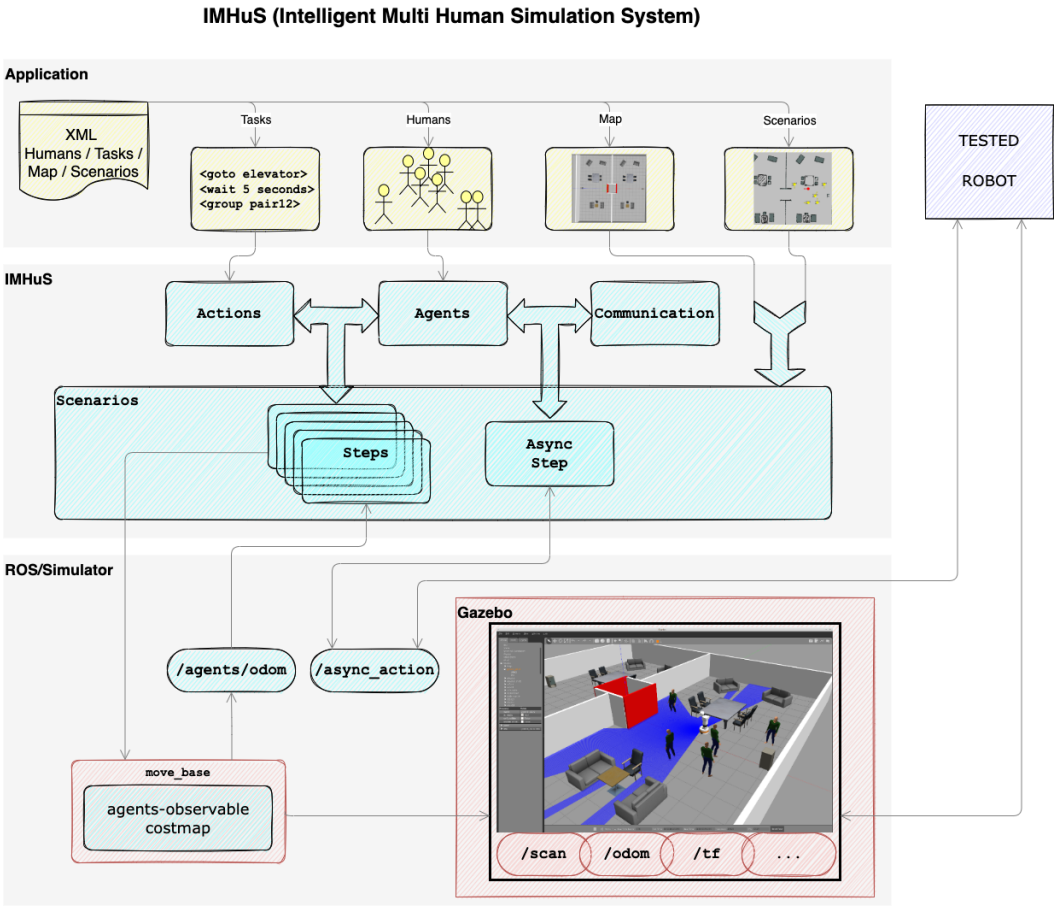


Figure 8.3: Structure of the new IMHuS tool.

Application layer This layer includes the XML configuration file needed to use IMHuS by defining its main components: map, agents, tasks and scenarios. The map includes all the locations of the agents to be used during the choreography, as well as those of static objects. These locations are represented as poses and a name is assigned to each of them. The description of the agents includes the name and initial position of all humans and robots choreographed, and also the name and composition of the groups that will appear in any step. Generic tasks are described with no specific subject to perform them, so that they can be reused by several agents. Last, scenarios

describe the steps of the choreography. Each step includes a set of tasks assigned to a particular agent or to a groups. They are the step elements.

IMHuS layer The tool interprets the information contained in the XML configuration file to represent the tasks as actions and the humans/robots as agents. When the scenarios are run, a command combines actions with agents for every step element. The commands concerning each agent are executed in a separate thread inside a step. The step finishes when all the threads are done. This is the way simultaneous movement of agents is achieved. This is the behavior of the tool for the choreography, but taking into account that the tested robot will be present, there must be a way for it to communicate with the agents of the simulation, just as it would happen in the real world. The asynchronous step is in charge of this task. At any point in time, the tested robot can ask something, for example, to press the elevator button. This action would be done through the communication module and would be answered by the agents in the simulation in this asynchronous step by means of a respond action. Not every agent can respond to a request from the tested robot. For instance, if the robot asks for someone to press the button and no one is around, the request will be dropped, counting as a failure.

ROS/Simulator layer This layer is used by IMHuS to place the agents in the simulation and to ask for the trajectories to move them around, as well as to communicate with the tested robot. The agents are placed in the world as obstacles so that when `move_base` is asked for a new path, they are avoided. The costmap obstacle layer has been modified to obtain what we call the *agents-observable costmap* that IMHuS needs.

Interface of the tested robot The way to include the software of a robot in the simulator for its social behavior to be tested would be through the simulator, Gazebo for now. Outside the simulator, the only communication would be through the asynchronous action thanks to which the robot can send a request action to be answered by a respond action of one agent of the simulation.

8.5 Use cases

This module is available as Open Source and it has been evaluated in the elevator scenario, as defined for SciRoc competition (IMHuS repository link).

An elevator scenario will be used to show a running example of the behavior of the IMHuS system with a tested robot, and also to explain the communication between the *tested robot* and the agents of IMHuS (video available at this link). The proposed scenario includes five human agents and the *tested robot*, whose goal is to go to the second floor. In order to do that, it has to ask a human agent to push the elevator button. Figure 8.4 shows the initial situation where the *tested robot* is approaching the elevator, human 2 is walking and the rest of humans are idle.

Once the *tested robot* gets to the elevator door, it requests an action to be performed by one of the humans. In order for a *tested robot* to trigger an action in IMHuS, it has to publish a specific message type on the `async_action` topic (see Figure 8.3). By doing so, it triggers an asynchronous action response from IMHuS. This message should contain the time of emission, the name of the agent requesting, the pose of this agent and the name of the task it is requesting. If the IMHuS' scenario includes in its configuration an asynchronous action response corresponding to the requested action, that action is triggered on the humans' side.

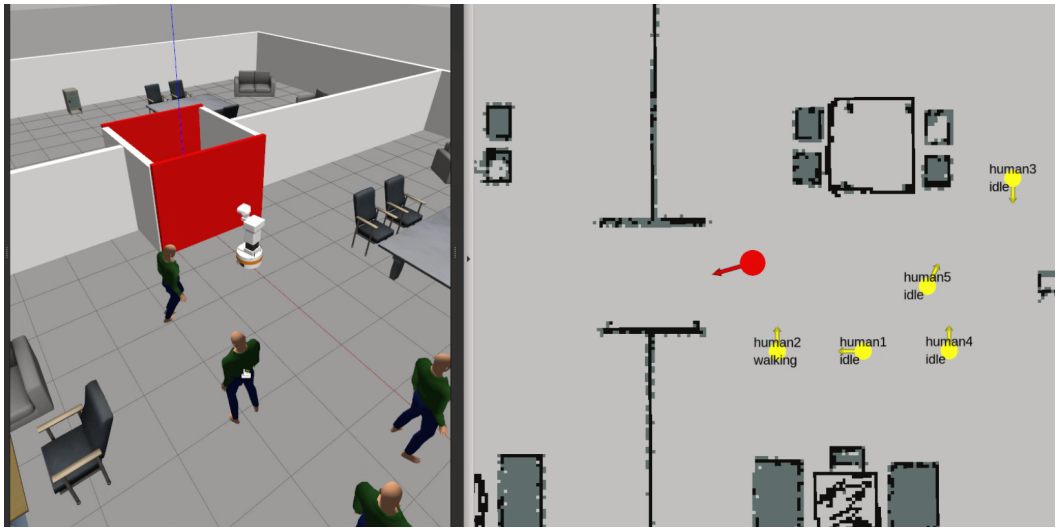


Figure 8.4: Initial situation.

At this point, two different things can happen. If there is no human close enough to the *tested robot*, or there is a human but it is not in an idle state, no one will respond to the request and it will be dropped, as Figure 8.5 shows.

Figure 8.5: Request dropped.

The *tested robot* repeats the request every five seconds until human 2 enters an idle state and responds to the request. Figure 8.6 shows this moment of the simulation. When an asynchronous action request is triggered, IMHuS periodically checks if one human is able to respond. The requirements for a human agent to respond are to be in the idle state and within a three meters radius from the *tested robot* position when it requested the action. If several human agents are able to accept the task, the task will be assigned to the closest human.

Figure 8.7 shows the final situation where the *tested robot* has reached the second floor, simulated in the scenario as the room on the other side of the elevator.

Figure 8.6: Request accepted by human2.

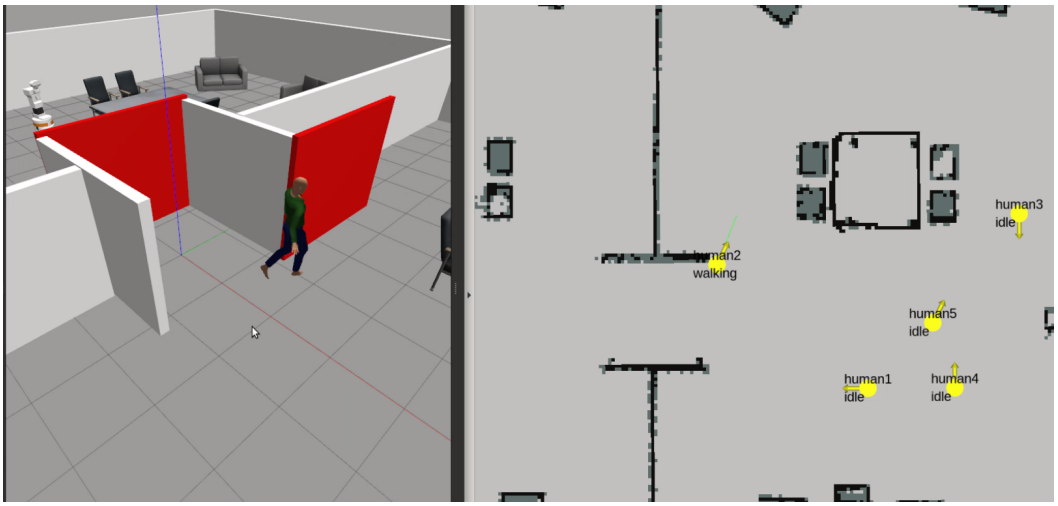


Figure 8.7: Final situation. *Tested robot* in second floor.

8.6 Conclusions and Future Work

The proposed tool, IMHuS, offers the possibility to create a realistic and challenging simulated environment in which groups of humans can be choreographed to evaluate the behavior of a *tested robot*. Different scenarios can be easily created using an XML configuration file in which social situations can be defined to measure the behavior of *tested robots* in a replicable environment. Furthermore, human agents are programmed to respond to interactions related to the particular situation of each scenario and their communication with the *tested robot*.

IMHuS's code is available as Open Source in the IMHuS repository. The current version has been implemented for Gazebo simulator, but the design presented in section 8.3 can be easily migrated to other simulators, such as MORSE or Unity. The tool could be used for benchmarking of competitions such as SciRock or RoboCup as a previous step for the teams before getting to the physical robot challenges. To create a new scenario with IMHuS, all

that is needed is a map and the configuration file to choreograph the human agents. The software has been designed to easily support the addition of both new tasks to be performed by the humans, and new interactions between them and the *tested robot*.

One of the ongoing developments is the automatic generation of simulation metrics such as the distance between the robot and the agents, time-to-collision, etc. Another line of work is the extension of basic actions, especially towards the social behavior of groups of agents. Last, IMHuS is being tested with CoHAN [Singamaneni 2021], a human-aware robot navigation planner to challenge CoHAN under several human-robot interaction settings. From this, we plan to identify the areas of improvement for human-aware navigation planners and provide a benchmark for testing these planners.

Social Agents Simulation Conclusions

TODO: NAVIGATION IS CONCLUDED

Part III

Conclusions

Conclusions

TODO: THESIS IS CONCLUDED

Part IV

Appendix

Appendix A

User Study results

This appendix provides more details and numbers about the User Study described in chapter 6.

A.1 Participants information

TODO: give information gathered, remove names to anonymize, only participant id.

A.2 Questionnaire answers

TODO: Add questionnaire answers (12 answer X 6 scenarios)

A.3 Execution metrics extracted

TODO: Add Execution metrics. Will be huge, specific format or just several pages?

A.4 Participants comments

This section gathers the participants' comments about the experiment. Since these comments were given verbally they are here described as note-taking, in French which is the language spoken by a large majority of participants, and using as much as possible the participant's own words.

A.5 Scenario preference

This section provides the results obtained after asking participants which scenario they preferred the most and the least.

ID	Comments
1	Quand robot défie notre logique perturbant et frustant. Globalement bien aimé. Notion de hierarchie, en fonction de la tâche (notamment simple) on se supérieur (on sait mieux faire) et donc le robot doit nous suivre. (2) bien pour tâche au + vite car va vite mais moins bien pour être lib, dépend de la tâche, pire (4)
2	RF est bien mais les mauvais choix sont pénible, frustrant, il ne concidère pas mon objectif/preferences. Cependant il s'adapte quand même une fois l'action faite. (2) RF trop cool, va + vite pour faire la tâche mais marche pas bien pour lib. HF est plus efficace en fonction de l'objectif. pire (4)
3	Chrono TO stressant. ne pas pouvoir prendre les cubes à l'avance n'est pas naturel, pertubant et rend un peu compliqué mais devient simple une fois habitué. Son de fin de tâche. Se sent obligé de faire quelque chose à chaque étape, et donc meme en RF clique sur la main pour confirmer que je serai passif. Rappeller à la fin de tache le regime et obj. (5) et (1) HF pour finir la tâche une fois habitué va relativement vite et fluide. HF pour etre libéré aussi, apres rien a faire.
4	Bon moment, Clair preference pour HF. RF est une catastrophe. En RF on n'a pas vraiment son mot à dire. En plus, quand RF commence à faire une erreur on est plus concentré à l'empecher de continuer à faire des erreurs plutot que sur la tâche, très frustant. Meme quand RF fait bon choix on se sent obligé de l'ecouter et impuissant.(5) HF et lib, car fluide en controle et j'ai pu atteindre mon objectif. Pire (6)
5	Intuitif, marche bien, efficace. Etre libere un peu frustant car on a envie d'agir, regarder le robot faire etre penible.. Sol: Donner une tache auxiliaire à l'humain a faire uniquement quand se désengage de la tache principale ? (3) HF et tache wrong. Car H agit le plus, R agit seulement quand necessaire. De plus, le robot s'adapte. Il a anticipé si jamais je ne prend pas le bleu en prenant le vert, mais une fois qu'il m'a vu prendre le bleu il n'a pas pris le sien. pire (4)
6	Globalement positif. Aurai aimé que le robot donne plus d'indication, guide plus les actions. (2) vif, mieux pour la tache, mais moins prévisible. (1) bien aussi mais plus lent/passif. pire le (6)
7	Certain scenario vraiment contraignant et frustant. Uiliser ressource commune en 1er est vraiment un mauvaix choix. Globalement ok. 2 efficace une fois compris. Faire la dernière action est gratifiant, donc que le robot s'adapte pour le permettre s'est positif.Preféré (2), rapide et bon choix, pire 4

Table A.1: Comments from participants given after the experiment. Part 1

ID	Comments
8	Parfois frustant a cause des mauvais choix du robot. Les actions sont "rigide", pas naturel (attraper cube bleu sous vert, le faisant "tomber" le vert). Bien dans l'ensemble, le fait que chacun ai sa part rend la collaboration pertinente et utile. Je prefère que la priorité des choix et actions soit à l'humain. (3) car s'adapte à ce que je fais et c'est clair. Pire (4) un mauvais choix du robot a impliqué un mauvais choix de ma part, frustrant (1) simplement frustrant, le robot semble contre moi.
9	RF pas efficace, mais follow est plus simple, moins compliqué. HF mieux mais parfois incohérent (barre rose). (1) est le plus satisfaisant. (4) est le pire
10	ne pas pouvoir attraper en avance un peu agaçant. Le fait de devoir regarder le but à gauche + la scène + lire le prompt text un peu complex =>mieux si robot dit quoi faire.Prefère quand le robot est "passif", dans le sens follower. Que le robot attende qu'on decide puis agisses. N'aime pas quand le robot "prend des initiatives" car possible mauvais choix: vole les cubes très agaçant.. + prend cube commun en 1er. Pref 1 Pire 4
11	Bonne simu, claire. Mvt ont l'air réel tache claire et globalement se passe bien. Les steps cadense bien, pratique Meilleur (1) pire (4)
12	Temps d'attente (step) frustrant. Serait bien de pouvoir prendre avant pour indiquer intention, donner info. HF + interessant + de control, - efficace mais - frustrant, - imprevu et donc de mauvais choix.(+) 1 (-) 6 car on est obligé de reposer le cube.
13	Interaction simple, comme un jeu video. Les mauvais choix du robot sont assez frustrant. Simple et plaisant.(+) 3 car s'est bien adapté malgré erreur humaine (-) 4 car vole les cubes
14	Tres bien dans l'ensemble. Géné par l'affichage, du mal à lire. Certain scenario efficace d'autre non. Simple, sauf lecture/texte. (+) 5 fini rapidement, double satisfaction de finir sa part vitre puis voir la pile fini par le robot (-) 6 degouté, frustrant
15	Bien, a volé 2 fois les cubes, pas agréable. HF + facile.(+) 2 (-) 4, 3
16	Simple, agréable. Le manque de collision avec cube réduit le réalisme mais ok. Un peu confu/pertubant et un peu frustrant de devoir attendre que robot finisse action avant d'agir à nouveau.(+) 2, RF car rapide (-) 6

Table A.2: Comments from participants given after the experiment. Part 2

ID	Comments
17	certain scenario ok =>c'est plutot H qui a mal agit, 2 sce avec mauvais choix de R. HF/RF pas tellement choix !=(-) 6 RF etre lib (+) 2
18	pas pouvoir prendre cube en avance frustrant. En RF, R devient prévisiblement génant, on prévoit et réfléchi pour s'adapter et anticiper mauvais choix (defensif). HF mieux.(+) HF tache plus vite, plus interessant que lib rapidement, ennuyeux. (-) 4, R imprévisible, devient ennemi
19	Globalement ok reagit bien, comprend bien ce que je faisait (pink), avant dernier tres frustrant, vole les cubes.. Sinon bien passé (+) 2 (-) 4
20	Simu claire, voit bien le R et quand il agit, interaction bonne et claire, 4 mauvais mais globalement benefique interaction. Pref au quotidien laisser R faire les tache chronophage, donc bien aimé obj lib au plus vite.(+) 5 puis 2 (-) 4
21	R prévisible cool, on peut preshot et anticiper. (+) 1 3 (-) 4
22	Sympa, simu bien faite. interactif, on est pris dedans et acteur. Beaucoup d'émotion déjà (satisfaction / frustration) donc si c'était une tache plus concrete ça serait encore plus frustrant.(+) 1 3 (-) 4
23	Simu bien faite, s'imagine bien interaction. Au debut un peu confu diff entre HF/RF puis ok. Confusion cube blanc et gris(+) aucun (-) 6
24	Interessant, jamais une gène, interaction amusante, sympa de prédire RA(+) 1, 3 (-) 6
25	HF ok, qd RF on peut rien faire. Simulation moins naturelle, pas parfaitement réaliste. Notion etape/synchronisation perturbant un peu (+) 5 HF lib (-) 4

Table A.3: Comments from participants given after the experiment. Part 3

	S1	S2	S3	S4	S5	S6
Times preferred the most	***9***	***7***	***3***	*0*	***5***	*0*
Times preferred the least	*0*	*0*	*0*	***16***	*0*	***9***

Table A.4: Number of times each scenario has been respectively preferred the most and the least. HF scenarios are never disliked and RF scenarios with erroneous estimations are never preferred.

Bibliography

[Alili 2009] Samir Alili, Rachid Alami and Vincent Montreuil. *A task planner for an autonomous social robot.* Distributed autonomous robotic systems 8, pages 335–344, 2009. (Cited in page 24.)

[Arntz 2022] Alexander Arntz, Carolin Straßmann, Stefanie Völker and Sabrina C. Eimler. *Collaborating eye to eye: Effects of workplace design on the perception of dominance of collaboration robots.* Frontiers in Robotics and AI, vol. 9, page 999308, September 2022. (Cited in page 44.)

[Aroor 2018] Anoop Aroor, Susan L. Epstein and Raj Korpan. *MengeROS: a Crowd Simulation Tool for Autonomous Robot Navigation.* arXiv:1801.08823 [cs], January 2018. (Cited in page 116.)

[Bartneck 2020] Christoph Bartneck, Tony Belpaeme, Friederike Eyssel, Takayuki Kanda, Merel Keijsers and Selma Šabanović. Human-robot interaction: An introduction. Cambridge University Press, 2020. (Cited in page 7.)

[Belle 2023] Vaishak Belle, Thomas Bolander, Andreas Herzig and Bernhard Nebel. *Epistemic planning: Perspectives on the special issue.* Artificial Intelligence, vol. 316, page 103842, March 2023. (Cited in page 43.)

[Bolander 2017] Thomas Bolander. *A Gentle Introduction to Epistemic Planning: The DEL Approach.* Electronic Proceedings in Theoretical Computer Science, vol. 243, pages 1–22, March 2017. (Cited in page 43.)

[Bolander 2021] Thomas Bolander, Lasse Dissing and Nicolai Herrmann. *DEL-based Epistemic Planning for Human-Robot Collaboration: Theory and Implementation.* In Proc. of KR, 2021. (Cited in page 48.)

[Bratman 1987] Michael Bratman. Intention, plans, and practical reason. Cambridge, MA: Harvard University Press, Cambridge, 1987. (Cited in page 10.)

[Buisan 2021] Guilhem Buisan. *Planning For Both Robot and Human: Anticipating and Accompanying Human Decisions.* PhD thesis, INSA Toulouse, France, 2021. (Cited in pages 13, 18, and 24.)

[Buisan 2022] Guilhem Buisan, Anthony Favier, Amandine Mayima and Rachid Alami. *HATP/EHDA: A Robot Task Planner Anticipating and Eliciting Human Decisions and Actions.* In Proc. of ICRA, 2022. (Cited in pages 2, 56, and 57.)

[Chakraborti 2015] Tathagata Chakraborti, Gordon Briggs, Kartik Talamadupula, Yu Zhang, Matthias Scheutz, David E. Smith and Subbarao Kambhampati. *Planning for serendipity.* In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015, pages 5300–5306. IEEE, 2015. (Cited in page 13.)

[Chen 2018] Xu Chen, Martin Treiber, Venkatesan Kanagaraj and Haiying Li. *Social force models for pedestrian traffic–state of the art.* Transport reviews, vol. 38, no. 5, pages 625–653, 2018. Publisher: Taylor & Francis. (Cited in page 19.)

[Cherry 1966] Colin Cherry. *On human communication: a review, a survey, and a criticism.* The Technology Press of MIT, 1966. (Cited in page 9.)

[Cirillo 2009a] Marcello Cirillo, Lars Karlsson and Alessandro Saffiotti. *A Human-Aware Robot Task Planner*. In Proc. of ICAPS 2009, 2009. (Cited in page 56.)

[Cirillo 2009b] Marcello Cirillo, Lars Karlsson and Alessandro Saffiotti. *Human-aware task planning for mobile robots*. In ICAR 2009, 2009. (Cited in page 56.)

[Clodic 2017] Aurélie Clodic, Elisabeth Pacherie, Rachid Alami and Raja Chatila. *Key Elements for Human Robot Joint Action*. In Sociality and Normativity for Robots-sPhilosophical Inquiries into Human-Robot Interactions, Studies in the Philosophy of Sociality, pages 159–177. Springer, 2017. This document is an extended version of the one published in the proceedings of RoboPhilosophy conference. (Cited in pages 56 and 57.)

[Cohen 1970] Philip Cohen and Hector Levesque. *On Team Formation*. February 1970. (Cited in page 9.)

[Cohen 1988] Jacob Cohen. *The concepts of power analysis. Statistical power analysis for the behavioral sciences*. Hillsdale: Elrbaum, 1988. (Cited in page 95.)

[Cohen 1991] Philip R Cohen and Hector J Levesque. *Teamwork*. Nous, vol. 25, no. 4, pages 487–512, 1991. Publisher: JSTOR. (Cited in pages 9 and 11.)

[Crosby 2014] Matthew Crosby, Anders Jonsson and Michael Rovatsos. *A Single-Agent Approach to Multiagent Planning*. In Proc. of ECAI, 2014. (Cited in pages 53 and 57.)

[Curioni 2019] Arianna Curioni, Cordula Vesper, Günther Knoblich and Natalie Sebanz. *Reciprocal information flow and role distribution support joint action coordination*. Cognition, vol. 187, pages 21–31, 2019. (Cited in page 57.)

[Curioni 2022] Arianna Curioni, Pavel Voinov, Matthias Allritz, Thomas Wolf, Josep Call and Günther Knoblich. *Human adults prefer to cooperate even when it is costly*. Proceedings of the Royal Society B: Biological Sciences, vol. 289, 04 2022. (Cited in page 57.)

[Darvish 2021] Kourosh Darvish, Enrico Simetti, Fulvio Mastrogiovanni and Giuseppe Casalino. *A Hierarchical Architecture for Human-Robot Cooperation Processes*. IEEE Trans. Robotics, vol. 37, no. 2, pages 567–586, 2021. (Cited in page 29.)

[De Carolis 2000] Berardina De Carolis, Catherine Pelachaud and Isabella Poggi. *Verbal and nonverbal discourse planning*. In Proc. AAMAS 2000 Workshop "Achieving Human-Like Behavior in Interactive Animated Agents, 2000. (Cited in page 18.)

[De Carolis 2001] Berardina De Carolis, Catherine Pelachaud, Isabella Poggi and Fiorella de Rosis. *Behavior planning for a reflexive agent*. In International Joint Conference on Artificial Intelligence, volume 17, pages 1059–1066. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001. Issue: 1. (Cited in page 18.)

[Devin 2018] Sandra Devin, Camille Vrignaud, Kathleen Belhassein, Aurelie Clodic, Ophelie Carreras and Rachid Alami. *Evaluating the Pertinence of Robot Decisions in a Human-Robot Joint Action Context: The PeRDITA Questionnaire*. In 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 144–151, Nanjing, August 2018. IEEE. (Cited in pages 88 and 89.)

[Echeverria 2011] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote and Séverin Lemaignan. *Modular open robots simulation engine: Morse*. In 2011 ieee international conference on robotics and automation, pages 46–51. IEEE, 2011. (Cited in page 125.)

[Echeverria 2012] G. Echeverria, S. Lemaignan and A. et Al. Degroote. *Simulating Complex Robotic Scenarios with MORSE.* In Simulation, Modeling, and Programming for Autonomous Robots, volume 7628, pages 197–208. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. (Cited in page 116.)

[Erdogan 2022] Emre Erdogan, Frank Dignum, Rineke Verbrugge and Pınar Yolum. *Abstracting Minds: Computational Theory of Mind for Human-Agent Collaboration.* In Stefan Schlobach, María Pérez-Ortiz and Myrthe Tielman, editors, Frontiers in Artificial Intelligence and Applications. IOS Press, September 2022. (Cited in page 44.)

[Erol 1996] Kutluhan Erol, James Hendler and Dana S Nau. *Complexity results for HTN planning.* Annals of Mathematics and Artificial Intelligence, vol. 18, no. 1, pages 69–93, 1996. Publisher: Springer. (Cited in page 14.)

[Favier 2023] Anthony Favier, Phani Teja Singamaneni and Rachid Alami. *Challenging Human-Aware Robot Navigation with an Intelligent Human Simulation System.* In Social Simulation Conference (SSC), Glasgow, France, September 2023. (Cited in page 133.)

[Ferrari 2022] Davide Ferrari, Federico Benzi and Cristian Secchi. *Bidirectional Communication Control for Human-Robot Collaboration*, June 2022. arXiv:2206.05202 [cs]. (Cited in page 44.)

[Ghallab 2016] Malik Ghallab, Dana Nau and Paolo Traverso. Automated planning and acting. Cambridge University Press, 2016. (Cited in pages 14, 16, and 31.)

[Gombolay 2015] Matthew Gombolay, Reymundo Gutierrez, Shanelle Clarke, Giancarlo Sturla and Julie Shah. *Decision-Making Authority, Team Efficiency and Human Worker Satisfaction in Mixed Human-Robot Teams.* Autonomous Robots, vol. 39, 07 2015. (Cited in page 57.)

[Gordon 2023] Jeremy Gordon, Guenther Knoblich and Giovanni Pezzulo. *Strategic Task Decomposition in Joint Action.* Cognitive Science, vol. 47, no. 7, page e13316, 2023. (Cited in page 56.)

[Grice 1975] HP Grice. *Logic and Conversation.* Syntax and Semantics, vol. 3, pages 43–58, 1975. (Cited in page 9.)

[Gurney 2022] Nikolos Gurney and David V. Pynadath. *Robots with Theory of Mind for Humans: A Survey.* In 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 993–1000, August 2022. ISSN: 1944-9437. (Cited in page 44.)

[Hauterville 2022a] Olivier Hauterville, Camino Fernández, Phani-Teja Singamaneni, Anthony Favier, Vicente Matellán and Rachid Alami. *IMHuS: Intelligent Multi-Human Simulator.* In IROS2022 Workshop: Artificial Intelligence for Social Robots Interacting with Humans in the Real World, Kyoto, Japan, October 2022. (Cited in page 127.)

[Hauterville 2022b] Olivier Hauterville, Camino Fernández, Phani-Teja Singamaneni, Anthony Favier, Vicente Matellán and Rachid Alami. *Interactive Social Agents Simulation Tool for Designing Choreographies for Human-Robot-Interaction Research.* In ROBOT2022: Fifth Iberian Robotics Conference, Zaragoza, Spain, November 2022. (Cited in page 127.)

[Helbing 1995] Dirk Helbing and Peter Molnar. *Social force model for pedestrian dynamics.* Physical review E, vol. 51, no. 5, page 4282, 1995. (Cited in pages 19 and 116.)

[Hoffman 2004] Guy Hoffman and Cynthia Breazeal. *Collaboration in human-robot teams.* In AIAA 1st intelligent systems technical conference, page 6434, 2004. (Cited in page 11.)

[Johannsmeier 2016] Lars Johannsmeier and Sami Haddadin. *A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes.* IEEE Robotics and Automation Letters, vol. 2, no. 1, pages 41–48, 2016. (Cited in page 29.)

[Johnson 2018] Benjamin Johnson, Michael W. Floyd, Alexandra Coman, Mark A. Wilson and David W. Aha. *Goal Reasoning and Trusted Autonomy.* In Foundations of Trusted Autonomy, volume 117. Springer, 2018. (Cited in page 116.)

[Köckemann 2014] Uwe Köckemann, Federico Pecora and Lars Karlsson. *Grandpa hates robots-interaction constraints for planning in inhabited environments.* In Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014. (Cited in page 56.)

[Koenig 2004] Nathan Koenig and Andrew Howard. *Design and use paradigms for gazebo, an open-source multi-robot simulator.* In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), volume 3, pages 2149–2154. IEEE, 2004. (Cited in page 125.)

[Kollmitz 2015] Marina Kollmitz, Kaijen Hsiao, Johannes Gaa and Wolfram Burgard. *Time dependent planning on a layered social cost map for human-aware robot navigation.* In 2015 European Conference on Mobile Robots (ECMR), pages 1–6, Lincoln, United Kingdom, September 2015. IEEE. (Cited in page 124.)

[Kourtis 2014] Dimitrios Kourtis, Günther Knoblich, Mateusz Woźniak and Natalie Sebanz. *Attention allocation and task representation during joint action planning.* Journal of Cognitive Neuroscience, vol. 26, no. 10, pages 2275–2286, 2014. (Cited in page 57.)

[Lallement 2014] Raphaël Lallement, Lavindra De Silva and Rachid Alami. *HATP: An HTN planner for robotics.* arXiv preprint arXiv:1405.5345, 2014. (Cited in page 24.)

[Levesque 1990] Hector J Levesque, Philip R Cohen and José HT Nunes. On acting together. SRI International Menlo Park, CA 94025-3493, 1990. (Cited in page 9.)

[Levine 2014] Steven James Levine and Brian Charles Williams. *Concurrent plan recognition & execution for human-robot teams.* In ICAPS, 2014. (Cited in page 57.)

[McEllin 2018] Luke McEllin, Natalie Sebanz and Günther Knoblich. *Identifying others' informative intentions from movement kinematics.* Cognition, vol. 180, pages 246–258, 08 2018. (Cited in page 57.)

[McMillan 2023] Donald McMillan, Razan Jaber, Benjamin R. Cowan, Joel E. Fischer, Bahar Irfan, Ronald Cumbal, Nima Zargham and Minha Lee. *Human-Robot Conversational Interaction (HRCI).* In Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction, HRI '23, pages 923–925, New York, NY, USA, March 2023. Association for Computing Machinery. (Cited in page 44.)

[Mehrabian 1967] Albert Mehrabian and Morton Wiener. *Decoding of inconsistent communications.* Journal of personality and social psychology, vol. 6, no. 1, page 109, 1967. (Cited in page 8.)

[Michael 2016] John Michael, Natalie Sebanz and Günther Knoblich. *Observing Joint Action: Coordination Creates Commitment.* Cognition, vol. 157, pages 106–113, 09 2016. (Cited in page 57.)

[Nomura 2016] Tatsuya Nomura, Takayuki Kanda, Hiroyoshi Kidokoro, Yoshitaka Suehiro and Sachie Yamada. *Why do children abuse robots?* Interaction Studies, vol. 17, no. 3, pages 347–369, 2016. (Cited in page 119.)

[Pérez-Higueras 2023] Noé Pérez-Higueras, Roberto Otero, Fernando Caballero and Luis Merino. *HuNavSim: A ROS2 Human Navigation Simulator for Benchmarking Human-Aware Robot Navigation.* arXiv preprint, 2023. (Cited in page 125.)

[Perille 2020] Daniel Perille, Abigail Truong, Xuesu Xiao and Peter Stone. *Benchmarking metric ground navigation.* In 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 116–121. IEEE, 2020. (Cited in page 19.)

[Puig 2018] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler and Antonio Torralba. *VirtualHome: Simulating Household Activities Via Programs.* In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8494–8502, Salt Lake City, UT, June 2018. IEEE. (Cited in page 116.)

[Ramachandruni 2023] Kartik Ramachandruni, Cassandra Kent and Sonia Chernova. *UHTP: A User-Aware Hierarchical Task Planning Framework for Communication-Free, Mutually-Adaptive Human-Robot Collaboration.* ACM Transactions on Human-Robot Interaction, 2023. (Cited in page 57.)

[Romeo 2022] Marta Romeo, Peter E. McKenna, David A. Robb, Gnanathusharan Rajendran, Birthe Nesset, Angelo Cangelosi and Helen Hastie. *Exploring Theory of Mind for Human-Robot Collaboration.* In 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 461–468, August 2022. ISSN: 1944-9437. (Cited in page 44.)

[Roncone 2017] Alessandro Roncone, Olivier Mangin and Brian Scassellati. *Transparent role assignment and task allocation in human robot collaboration.* In Proc. of ICRA, 2017. (Cited in pages 56 and 57.)

[Samarakoon 2022] SM Bhagya P Samarakoon, MA Viraj J Muthugala and AG Buddhika P Jayasekara. *A Review on Human–Robot Proxemics.* Electronics, vol. 11, no. 16, page 2490, 2022. (Cited in page 19.)

[Schmitz 2017] Laura Schmitz, Cordula Vesper, Natalie Sebanz and Günther Knoblich. *Co-Representation of Others' Task Constraints in Joint Action.* Journal of experimental psychology. Human perception and performance, vol. 43, 04 2017. (Cited in page 57.)

[Sebanz 2006a] Natalie Sebanz, Harold Bekkering and Günther Knoblich. *Joint action: bodies and minds moving together.* Trends in cognitive sciences, vol. 10, no. 2, pages 70–76, 2006. (Cited in page 56.)

[Sebanz 2006b] Natalie Sebanz, Harold Bekkering and Günther Knoblich. *Joint Action: Bodies and Minds Moving Together.* Trends in Cognitive Sciences, vol. 10, no. 2, pages 70–76, 2006. (Cited in page 1.)

[Sebanz 2009] Natalie Sebanz and Günther Knoblich. *Prediction in Joint Action: What, When, and Where.* Top. Cogn. Sci., vol. 1, no. 2, pages 353–367, 2009. (Cited in page 56.)

[Selvaggio 2021] Mario Selvaggio, Marco Cognetti, Stefanos Nikolaidis, Serena Ivaldi and Bruno Siciliano. *Autonomy in physical human-robot interaction: A brief survey.* IEEE Robotics Autom. Lett., 2021. (Cited in page 56.)

[Shekhar 2020] Shashank Shekhar and Ronen I. Brafman. *Representing and planning with interacting actions and privacy.* AIJ, vol. 278, 2020. (Cited in pages 53 and 57.)

[Shen 2020] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martın-Martın, Linxi Fan, Guanzhi Wang, Shyamal Buch, Claudia D'Arpino, Sanjana Srivastava, Lyne P Tchapmi, Kent Vainio, Li Fei-Fei and Silvio Savarese. *iGibson 1.0: a Simulation Environment for Interactive Tasks in Large Realistic Scenes.* arXiv preprint, 2020. (Cited in page 116.)

[Shvo 2022] Maayan Shvo, Ruthrash Hari, Ziggy O'Reilly, Sophia Abolore, Sze-Yuh Nina Wang and Sheila A. McIlraith. *Proactive Robotic Assistance via Theory of Mind.* In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 9148–9155, October 2022. ISSN: 2153-0866. (Cited in page 44.)

[Singamaneni 2021] Phani Teja Singamaneni, Anthony Favier and Rachid Alami. *Human-Aware Navigation Planner for Diverse Human-Robot Ineraction Contexts.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021. (Cited in pages 19, 119, and 137.)

[Smith 1998] I.A. Smith, P.R. Cohen, J.M. Bradshaw, M. Greaves and H. Holmback. *Designing conversation policies using joint intention theory.* In Proceedings International Conference on Multi Agent Systems (Cat. No.98EX160), pages 269–276, Paris, France, 1998. IEEE Comput. Soc. (Cited in page 9.)

[Strachan 2020] James Strachan and Georgina Török. *Efficiency is prioritised over fairness when distributing joint actions.* Acta Psychologica, vol. 210, page 103158, 10 2020. (Cited in page 57.)

[Thomaz 2016] Andrea Thomaz, Guy Hoffman and Maya Cakmak. *Computational Human-Robot Interaction.* Foundations and Trends in Robotics, vol. 4, no. 2-3, pages 104–223, 2016. (Cited in page 10.)

[Tsoi 2020] Nathan Tsoi, Mohamed Hussein, Jeacy Espinoza, Xavier Ruiz and Marynel Vázquez. *SEAN: Social Environment for Autonomous Navigation.* In Proceedings of the 8th International Conference on Human-Agent Interaction (HAI), November 2020. (Cited in page 116.)

[Tsoi 2022] Nathan Tsoi, Alec Xiang, Peter Yu, Samuel S Sohn, Greg Schwartz, Subashri Ramesh, Mohamed Hussein, Anjali W Gupta, Mubbasir Kapadia and Marynel Vázquez. *Sean 2.0: Formalizing and generating social situations for robot navigation.* IEEE Robotics and Automation Letters, vol. 7, pages 11047–11054, 2022. (Cited in page 116.)

[Van Den Berg 2011] Jur Van Den Berg, Stephen J Guy, Ming Lin and Dinesh Manocha. *Reciprocal n-body collision avoidance.* In Robotics research, pages 3–19. Springer, 2011. (Cited in page 116.)

[Vattam 2013] Swaroop Vattam, Matthew Evans Klenk, Matthew Molineaux and David W. Aha. *Breadth of Approaches to Goal Reasoning: A Research Survey.* In Annual Conference on Advances in Cognitive Systems: Workshop on Goal Reasoning, 2013. (Cited in page 116.)

[Verhagen 2022] Ruben S. Verhagen, Mark A. Neerincx and Myrthe L. Tielman. *The influence of interdependence and a transparent or explainable communication style on human-robot teamwork.* Frontiers in Robotics and AI, vol. 9, page 993997, September 2022. (Cited in page 44.)

[Yamaguchi 2019] Motonori Yamaguchi, Helen J. Wall and Bernhard Hommel. *The roles of action selection and actor selection in joint task settings.* Cognition, vol. 182, pages 184–192, 2019. (Cited in page 57.)

[Yige 2021] Liu Yige, Li Siyun, Li Chengshu, Perez-D'Arpino Claudia and Savarese Silvio. *Interactive Pedestrian Simulation in iGibson.* RSS Workshop on Social Robot Navigation, 2021. (Cited in page 116.)

[Yu 2023] Chuang Yu, Baris Serhan, Marta Romeo and Angelo Cangelosi. *Robot Theory of Mind with Reverse Psychology.* In Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction, pages 545–547, Stockholm Sweden, March 2023. ACM. (Cited in page 44.)