TOWARD HUMAN-MULTIAGENT TEAMS

by

Nathan Schurr

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

December 2007

Acknowledgments

I cannot take all of the credit for this. There are many people without which this would not have been possible. First and foremost, I would like to thank my advisor and mentor Milind Tambe. He helped my find my way along the treacherous path of the PhD. I am proud to see how much we both have grown in the process.

In addition, I would like to thank the members of my thesis committee for following my research, honing my direction and being patient with me: Anne Balsamo, Karen Myers, Michael Van Lent, Michael Zyda, and Randall Hill. I would also like to thank Isaac Maya and all of the members of the CREATE research center at USC for supporting my research and allowing me to form valuable contacts for such an important research application.

I hold a tremendous amount of passion for the application of these ideas, and I would like to thank the Los Angeles Fire Department for their time and information that helped keep my research grounded. In particular, I thank Fire Captain Ronald Roemer for his consistent support in this thesis and valuable input to the DEFACTO system. I would like to thank all of the people that have contributed to the Omni-Viewer portion of DEFACTO, including but not limited to Pratik Patil and Ankit Modi. I thank all of the members of staff that helped me wade through the paperwork and logistics of graduate school at USC.

I would like to thank all of my family for being there for me, for checking up on me and raising me to be able to handle something like this. I would like to thank Janusz Marecki because I have never been so prolific as I was with him. I would like to thank Paul Scerri for paving the way for my research and reaffirming my desire for the tangible. I would like to thank all of my friends, who are too numerous to name here, but please know that you have served to help me keep my sanity and make life worth living. And I would especially like to thank all of the members of the Teamcore Family, both immediate and extended, of which I am proud to belong.

Table of Contents

List of Tables

List of Figures

Abstract

One of the most fundamental challenges of building a human-multiagent team is adjustable autonomy, a process in which the control over team decisions is dynamically transferred between humans and agents. This thesis studies adjustable autonomy in the context of a human interacting with a team of agents and focuses on four issues that arise when addressing this team-level adjustable autonomy problem in real-time uncertain domains. Firstly, the humans and agents may differ significantly in their worldviews and their capabilities. This difference leads to inconsistencies in how humans and agents solve problems. Despite such inconsistencies, previous work has rigidly assumed the infallibility of human decisions. However, in some cases following the human's decisions lead to worse human-multiagent team performance. Secondly, it is desirable for the team to manage the uncertainty of action durations and plan for the optimal action at any point in time. This is a crucial challenge to address given that these human-multiagent teams are working in real-time with strict deadlines combined with the particularly uncertain duration of actions that involve a human. Thirdly, the team needs to be able to plan for the optimal time to interrupt certain actions. This is due to the fact that actions may take an uncertain amount of time and the deadline is approaching. The human-multiagent team may benefit from attempting an action for a given amount of time and interrupting the action if it does not finish in order to try another action that has a higher expected reward. Fourthly, team-level adjustable autonomy is an inherently distributed and complex problem that cannot be solved optimally and completely online.

My thesis makes four contributions to the field in order to address these challenges. First, I have included, in the adjustable autonomy framework, the modeling of the resolution of inconsistencies between human and agent view. This diverges from previous work on adjustable autonomy that traditionally assumes the human is infallible and decisions as rigid, but instead puts the humans and agents on an equal footing, allowing each to identify possible team performance problems. I have developed new resolution adjustable autonomy strategies that recognize inconsistencies and provide a framework to decide if a resolution is beneficial. Second, in order to address the challenges brought about by dealing with time, I have modeled these new adjustable autonomy strategies using TMDPs (Time dependent Markov Decision Problems). This allows for an improvement over previous approaches, which used a discretized time model and less efficient solutions. Third, I have introduced a new model for Interruptible TMDPs (ITMDPs) that allows for an action to be interrupted at any point in continuous time. This results in a more accurate modeling of actions and produces additional time-dependent policies that guide interruption during the execution of an action. Fourth, I have created a hybrid approach that decomposes the team level adjustable autonomy problem in a separate ITMDP for each team decision. In addition, team-based logics are used to coordinate and execute the team actions that are present in the ITMDP.

In addition to developing these techniques, I have conducted experimental evaluations that demonstrate the contributions of this approach. This has been realized in a system that I have constructed, DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams via Omnipresence), that incorporates this approach to team-level adjustable autonomy, along with agent coordination reasoning and a multi-perspective view of the team for the human. DEFACTO has been applied to an urban disaster response domain and used for incident commander training. The Los Angeles Fire Department has been supportive and has given valuable feedback that has shaped the system.

Chapter 1

Introduction

For roughly a decade, research in the area of multiagent systems has been focusing on enabling decentralized agents to coordinate together as a team [CAL, 2003; Goodrich et al., 2001; Collins et al., 1998; Ferguson and Allen, 1998; Fong et al., 2002; Wagner et al., 2004]. Recent advances have allowed humans to be integrated with the multiagent team in real-time while tackling more dynamic, unpredictable environments [Clancey et al., 2003; Sellner et al., 2006; Owens et al., 2006; Goodrich et al., 2007]. In addition, these teams must take into account that, since the world is so dynamic, there are deadlines for their task completion and decisions must be made quickly. However, the advent of these new human-multiagent teams introduces many new challenges concerning how the human and agents will interact.

One of the most fundamental challenges of building a human-multiagent team is that of adjustable autonomy [Sierhuis et al., 2003; Sellner et al., 2006; Reitsema et al., 2005; Scerri et al., 2002; Varakantham et al., 2005]. Conventionally, adjustable autonomy refers to the ability of an agent to dynamically adjust its own autonomy thereby adjusting the amount of control a human can have over that agent. Given the changing state of the environment and the team, it is beneficial for an agent to be flexible in its autonomy and not be forced to act either with full autonomy or with zero autonomy. The key to adjustable autonomy is when and where to transfer control of a decision.

As it is described in this thesis, adjustable autonomy is a process in which the control over a decision[1] is dynamically transferred between a human and an agent in order to find the teammate (agent or human) that is best fit for making the decision. The choice of the best fit teammate depends on the state of the world, the locality of information among team members, the priority of the decision, and the availability of the teammate. In addition, all of these factors change continuously over time and consequently, the best teammate to make a decision may change over time as well.

This thesis focuses on applying adjustable autonomy to complex, dynamic domains where a single human must interact with an entire team of agents. Given that human decisions in this context may be imperfect, adjustable autonomy raises a range of novel challenges and contributions that are outlined below.

1.1 Problem Definition

Given the setting of a single human coordinating with multiple agents, this thesis makes the following assumptions about the environment in which they act: (i) The team has limited communication channels due in part to the abilities of the human. (ii) Both the agents and the human have uncertain and incomplete information about the state of the world. (iii) These teams are acting in dynamic, unpredictable domains. (iv) Team decisions have deadlines beyond which they become irrelevant or unachievable. These assumptions imply a more realistic team setting than

[1] In the context of adjustable autonomy, a decision refers to any team decision whether it be a decision to perform an action or make a high-level team strategy decision.

what has been traditionally assumed in human-multiagent teams. Given these assumptions, the problem addressed in the thesis can be stated as follows:

Problem: *Team-level adjustable autonomy: Have team decisions be made by the best decision maker at a given time in order to improve the performance of these new human-multiagent teams.*

In this thesis, I will focus on four primary challenges that intensify for human-multiagent teams when addressing the problem above:

Firstly, the humans and agents may have inconsistent decisions about what the team should do, due to their distinct world-views. Their distinct world-views are a result of the limited communication that prevents all information from being communicated to all team members. Despite such inconsistencies, previous work has rigidly assumed the infallibility of human decisions. However, in some cases following the human's decisions lead to worse human-multiagent team performance. An inconsistency does not necessarily mean that the human is incorrect or that the agent is incorrect. An agent's model of the world is uncertain or incomplete and may not contain some insight that the human has from previous experience. In contrast, the human may not know the local details that could lead to degradation in team performance. Consequently, although resolving may take a long time, the human-multiagent team still desires to resolve these inconsistencies.

Secondly, the human-multiagent team must address the time limits that decisions have in these domains. After a deadline, the decision has either evolved or become irrelevant, so the team wants the decision to be made and executed before the opportunity is lost. The team needs to manage an increasing uncertainty over action duration coupled with the need for precisely timed actions.

Uncertainty over time is a crucial factor to consider given that the actions may take more time than is allowable in the deadlines. This is particularly difficult due to the increase in uncertainty over the duration of team actions, especially those including the human. The human-multiagent team must reason about time not only to complete the task, but also the time necessary to resolve these aforementioned inconsistencies. This challenge suggests that agents carefully plan their actions taking into account deadlines and uncertain action duration.

Thirdly, because actions make take a long time or have an uncertain duration, actions may be started and then interrupted. Current methods for planning with deadlines and uncertain action durations do not allow actions to be interrupted at any point in continuous time. However, given that each decision has a deadline, a plan (policy) may benefit from being able to leave the current action and pursue an action that takes less time or even less uncertainty in duration. This challenge of interruptible actions exists not only in adjustable autonomy, but in many other real world domains.

Fourthly, it is not feasible to solve team-level adjustable autonomy optimally in real-time. The planning problem is inherently distributed and incorporates a lot of uncertainty. Thus, this planning problem cannot be solved both quickly and optimally by any one solution technique. For example, multiple agents may each have adjustable autonomy processes with the human occurring simultaneously. This issue arises because of the potential for many transfers of control or resolving of inconsistencies to arise at once. The human may be overwhelmed by so many requests at the same time. Also, there may be multiple inconsistencies that pertain to the same group of decisions and thus must be condensed into one common resolution. In addition, there may be multiple distinct inconsistencies that are not directly related to each other, yet all of them need to be resolved. The team must be able to determine which inconsistency to attempt to resolve with the human and at which times. Finally, there is the goal of making the rest of the team aware of the outcomes of the attempted resolution so that inconsistencies are detected accurately.

These challenges arise, in part, due to the fact that the thesis is making novel assumptions and consequently solving a novel problem. Previous work in the areas of both mixed initiative and adjustable autonomy have addressed this general problem of how to have a human and an agent collaborate. Mixed initiative work has chosen to focus on the human-agent collaboration during the synthesis of a plan [Myers and Morley, 2003; Ferguson et al., 1996; Horvitz and Apacible, 2003]. These mixed initiative approaches consequently do not have to address issues such as an approaching deadline or the human not being available. However, this thesis focuses on human-agent collaboration during the execution of a plan and hence these challenges arise. Adjustable autonomy tries to address some of these challenges posed by factoring time into the state, but does so at the price of a huge explosion in state space [Varakantham et al., 2005; Scerri et al., 2002]. Another drawback to the adjustable autonomy work is that whenever the human or agent has the autonomy and makes a decision, that decision is final. This limits the team's ability to avoid a decision that could degrade performance even if other teammates have an inconsistent view of the decision.

1.2 Solution Approach

Thesis: *Performance of the human multiagent team can be improved by choosing the decision maker using the following techniques: (i) modeling of inconsistencies and the modeling of their resolution, (ii) developing plans that are continuous time dependent and factor in uncertain action durations, (iii) introducing a new model for planning which enables* *actions to be interrupted at any point during execution, and (iv) decomposing the team-level adjustable autonomy problem and leveraging the use of a hybrid approach to address coordination.*

My thesis will present an approach for Resolving Inconsistencies with Adjustable Autonomy in Continuous Time (RIAACT). RIAACT makes four contributions to the field of human-multiagent teams in order to address the challenges stated in Section 1.1.

Firstly, RIAACT allows for the recognizing and potential resolving of inconsistencies between human and agent view. This puts the humans and agents on an equal footing, allowing each to identify possible team performance problems in the decisions. A key idea from adjustable autonomy is the use of adjustable autonomy strategies: plans to flexibly transfer control of a decision between an agent and a human until a decision is made [Scerri et al., 2002]. I have developed resolution adjustable autonomy strategies that, once an inconsistency is detected, provide a framework to decide if a resolution is beneficial. This thesis does not focus on which method or process of resolution should be implemented. Instead, I focus on the planning problem given that there is some idea of the uncertain action duration of the chosen resolution method.

Secondly, in order to address the challenges brought about by dealing with time, RIAACT models team level adjustable autonomy in continuous time. Each adjustable autonomy strategy can depend on the state and current absolute time by modeling it as a Time-Dependent Markov Decision Problem (TMDP) [Boyan and Littman, 2000; Li and Littman, 2005; Marecki et al., 2007]. This TMDP formulation allows for a continuous time policy (space of optimal strategies) that allows for optimal actions to be prescribed at arbitrary points in time, without the state space explosion that results from factoring time into the state at fixed discrete intervals. In addition, a

TMDP models the uncertainty of action durations (for example the resolving of an inconsistency) as continuous distributions. This results in a fine-grained model representation and policy, which enables the team to develop more accurate strategies that are better designed to handle deadlines.

Thirdly, I have developed a new model for Interruptible Time-Dependent Markov Decision Problems (ITMDPs). This new model allows certain actions to be interrupted at any point in continuous time. In previous models, an action was either not interruptible or only interruptible at discrete points in time. As in TMDPs, the interruption can have a duration that follows an arbitrary distribution. Once the interruption has finished, the destination state is the original state that the interrupted action was taken from. These ITMDPs allow for more accurate estimation of action rewards by modeling them as interruptible. In addition, new policies are created that can determine if interrupting is optimal, given the amount of time that has elapsed since the start of the action. An example of an interruptible action is the above-mentioned resolving of inconsistencies, which are beneficial, yet also can take an uncertain amount of time to be completed.

Fourthly, in order to allow the team as a whole to coordinate when giving up or taking of autonomy over decisions, the team would need to use a distributed MDP, which is prohibitively expensive computationally (has been shown to be NEXP-complete in the general case [Bernstein et al., 2000]). RIAACT instead decomposes the team-level adjustable autonomy problem into three subproblems and solves them each with separate techniques to create a hybrid approach [Tambe et al., 2005; Nair and Tambe, 2005; Schut et al., 2001; Scerri et al., 2002; Boutilier et al., 2000]. (i) Developing the space of strategies is accomplished by solving an abstracted single agent MDP on behalf of the team, using the continuous time methods described previously. (ii) The communication and coordination is executed using team logic based reasoning. (iii) In addition, the agent team when trying to allocate a role among the team will employ distributed constraint reasoning. Combining these three techniques into a hybrid approach allows for the complex team-level adjustable autonomy to occur in real time while aiming to optimize coordinated team performance.

These contributions have been realized in a system that I have constructed, DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence) that incorporates these adjustable autonomy strategies, along with agent coordination reasoning and a multi-perspective view of the team for the human. This large-scale research prototype has served to explore these concepts in order to allow humans and agents to work together. DEFACTO has been shaped as a training simulation tool for incident commanders and demonstrated to the Los Angeles Fire Department (LAFD) with positive and helpful feedback. A novel aspect of this thesis is its ability to point to the lessons learned from the LAFD feedback and show how they have influenced the design of DEFACTO and consequently of the human-multiagent team.

1.3 Overview of Thesis

The rest of this thesis will be divided as follows: First, in Chapter 2, the human-multiagent teams of interest are explained and applied to the disaster response domain, followed by background information on relevant concepts. Related work is covered in Chapter 3. Chapter 4 describes the research prototype system, DEFACTO, which has been applied to this domain. Then, in Chapter 5, I will show how adjustable autonomy applies to human multiagent teams and present experimental results from testing different team level strategies. Chapter 6 explains RIAACT and shows how it can enable the human multiagent team to address the challenges of team-level adjustable autonomy. Chapter 7 points out some of the lessons learned through feedback from the

Los Angeles Fire Department. This is followed by a conclusion and a vision of how this thesis is working towards the future of human multiagent teams (Chapter 8). Next, in Chapter 9, there is a related discussion of how Isaac Asimov's Laws of Robotics [Asimov, 1990] can influence the design of real-world human-multiagent teams.

Chapter 2

Domain and Background

In this chapter, I will give details on the exact domain that this thesis is addressing. Also, I will provide background on some of the existing techniques utilized in this thesis. First, I will explain the types of human-multiagent teams that this thesis focuses on. Second, I will go more in depth into disaster response, which is a domain that is of particular interest. Thirdly, I give a brief background on adjustable autonomy. Fourthly, I provide an explanation of some of the MDP-based techniques that are used later in this thesis in addition to presenting a leading solution technique.

2.1 Human-Multiagent Teams

There are many domains where human-multiagent teams are valuable. For example, as illustrated in [Schurr et al., 2005], multiagent teams can help coordinate teams of fire fighters in rescue operations during disaster response. Furthermore, they are also being used as helping hand to humans in an office setting for assisting in various activities like scheduling meetings, collecting information, managing projects etc. [CAL, 2003; Scerri et al., 2002; Wagner et al., 2004]. In each of these domains, agents can relieve humans of the routine and mundane tasks and allows them

- **Environment Characteristics**

  - *Uncertain action durations*
  - *Non-deterministic sequence of world events*
  - *Unable to plan for all possible outcomes*
  - *Decisions have deadlines beyond which they become irrelevant or unachievable*
  - *A potential increase or decrease in team performance will have a strong impact*

- **Team Characteristics**

  - *Communication restrictions*
    - *Restriction on agent to agent communication due to lack of time*
    - *Restrictions on agent to human communication due to both a lack of time and limitations of human cognition*
  - *Information gap*
    - *Human has an abstract global view*
    - *Agents have a detailed local view*

Figure 2.1: Overview of the characteristics of human-multiagent teams on which this thesis is focused.

to be more productive and/or successful. Also, agents can replace the need for large amounts of personnel for activities like training exercises.

While there are a variety of different types of domains that multiagents can be applied to, I am interested in domains with particular characteristics and consequently particular challenges that the team must overcome. In addition, these characteristics also shape our solution methods. Figure 2.1 shows some of the key characteristics of the domains of interest.

This thesis looks at real-world domains where there is not enough time available for the multiagent team to compute pre-planned optimal decisions. This allows me to focus on some of the key challenges that humans and agents will face while working together online. Indeed, this research applies to similar human-multiagent teams that have been developed in projects like

COORDINATORS [Wagner et al., 2004] where agents are assisting a military team and CALO [CAL, 2003] where agents are facilitating a group of coworkers in an office environment.

The area of disaster response is particularly well suited for human-multiagent teams that will encounter these characteristics. There is a great potential, during a disaster crisis, for agents to take on some team responsibilities in order to free the humans involved from being unnecessarily overburdened. Disaster response is a dynamic scenario in which decisions must be made correctly and quickly because human safety is at risk (large penalties for poor team performance). It is likely that future disaster response systems will utilize such technology for coordination among different rescue vehicles.

While human-multiagent teams hold a lot of potential for the future of disaster response, in the near future, they will greatly benefit the training of fire fighters for the current disaster response methods by providing a simulated exercise. The simulation is a stand-in for the actual environmental observations received in the future. The methods mentioned in this thesis for allowing the human-multiagent team to coordinate will be helpful to both training and deployed use. Below, I will introduce an application of human-multiagent teams to training of disaster response personnel.

This thesis is focused on a collection of distributed agents that are collaborating with a single human towards a common goal(s). In order to accomplish these goals, there are numerous team decisions that must be made. These decisions must be made despite the challenging characteristics of both the human-multiagent team and the environment in which the team acts (as shown in Figure 2.1). The team has limited communication channels, due in part to the abilities of the human. Both the agents and the human have uncertain and incomplete information about the state of the world. These teams are acting in dynamic, unpredictable domains. Team decisions have deadlines beyond which they become irrelevant or unachievable. There are often team decisions that may be made by either a human or an agent, thus creating a need for adjustable autonomy. The human has the ability to monitor the progress of the team and can provide inputs to the agents in order to allow the human-multiagent team to perform better. However, the agents may receive human input that the agents are able to determine as harmful to the team's performance. The agents have their own perspective on the performance of the human-multiagent team and may hold an inconsistent viewpoint to that of the human.

The information gap exists because of a difference in the roles that the human and the agents have on the team. In addition, the gap exists because of the challenges of coordinating a team in a dynamic, unpredictable world. The human-multiagent team must coordinate across the restrictive human-agent communication channels. There is a limit to the amount, speed and kind of information that can be passed in either direction. Consequently all of the members of the human-multiagent team may not know the complete world state.

2.1.1 Inconsistency

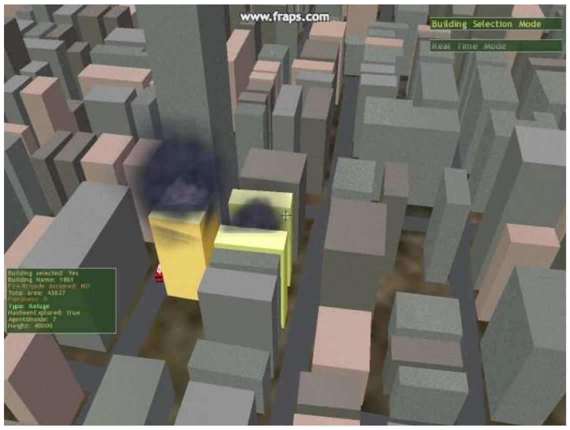
An important determinant of team performance in the domains of interest is often the consistency of team decisions. There are certain real-world assumptions in the domains of focus that cause consistency to be so important: decisions must be made under critical time pressure, while uncertainty and where there is interdependence between decisions. It becomes very important that the team resources are consistently focused toward complementary objectives rather than have them split their resources on conflicting, but near-optimal decisions.

This inconsistency exists as a result of the information gap described previously. This gap can exist because of a difference in the roles that the human and the agents have on the team. In addition, the gap exists because of the challenges of coordinating a team in a dynamic, unpredictable world. The human-multiagent team must coordinate across the restrictive human-agent communication channels. There is a limit to the amount, speed and kind of information that can be passed in either direction. Consequently all of the members of the human-multiagent team may not know the complete world state.

The human and agents must coordinate and divide team decisions among each of the teammates. However, if a human makes a decision that the agent team finds to be inconsistent, the agent team blindly following that decision can lead to poor team performance. The human-multiagent team would benefit from resolving the inconsistencies, yet the team cannot commit to doing so without addressing two challenges. First, because the world is dynamic and opportunities may change or may even be lost if they are not acted upon before a deadline. Second, action durations are uncertain, which make them harder to reason about. This new team task of resolving the inconsistency between a human and an agent is a prime example of a team task that takes an uncertain amount of time.

2.2 Disaster Response

Techniques for augmenting the automation of routine coordination are rapidly reaching a level of effectiveness where they can simulate realistic coordination on the ground for large numbers of emergency response entities (e.g. fire engines, police cars) for the sake of training. We have constructed DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence) as a high fidelity system for training and simulating of future disaster response. DEFACTO allows for a human user (fire fighter) to observe a number of fires burning in buildings



```
Building selected: Yes
Building Name: 1861
FireBrigade Assigned: NO
Total area: 43827
Fieryness: 0
Type: Refuge
HasBeenExplored: true
AgentsInside: 7
Height: 40000
```

Figure 2.2: The DEFACTO system displays multiple fires in an urban environment.

in an urban environment, and the human user is also allowed to help assign available fire engines to the fires (see Figure 2.2).

The DEFACTO system is focused on illustrating the potential of future disaster-response to large-scale disasters, for example ones that may arise as a result of large-scale terrorist attacks. DEFACTO is motivated by a scenario of great concern to first responders within Los Angeles and other metropolitan areas. In particular, it explores a natural or manmade disaster in an urban area that results in a large-scale disaster on the ground. This scenario leads to multiple fires in multiple locations with potentially many critically injured civilians. While there are many longer-term implications of such an attack, we focus on assisting first responders, and, in particular, the fire fighter "incident commander." At every disaster response, the incident commander is the single person that is in charge of monitoring and managing the entire situation. This incident commander helps facilitate the allocation of resources and helps inform team members of relevant information.

The incident commander's main duty during a fire is to shoulder all responsibility for both the outcome of the response and for the safety of the firefighters. In order to do this, the incident

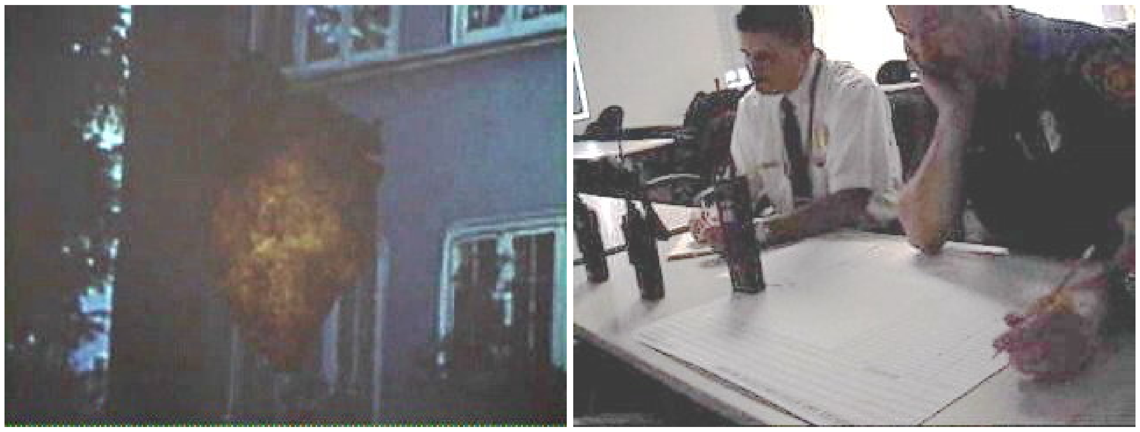

(a) (b)

Figure 2.3: Current Training Methods: (a) projected photo and (b) incident commanders at a table

commander must have constant contact with the firefighters and have a complete picture of the entire situation. The incident commander must make certain that dangerous choices are avoided and the firefighters are informed and directed as needed.

Although, the focus of this thesis is on techniques for deployed systems, in the near future a subset of these techniques can be applied to disaster response training. We had the opportunity to study the current methods that the Los Angeles Fire Department (LAFD) implements to train incident commanders. The LAFD uses a projection screen to simulate the disaster (Figure 2.3-(a)). In addition, the participating incident commander is seated at a desk, directing an assistant to take notes (Figure 2.3-(b)). Other firefighters remain in the back of the room and communicate to the incident commander via radios. Firefighters are taken temporarily off duty in order to help act out these pre-determined scenarios in order to test the incident commander's abilities.

The LAFD's current training approach, however, has several limitations. First, it requires a number of officers to be taken off duty, which decreases the number of resources available to the city for a disaster during training. Second, the disaster conditions created are not accurate in the way that they appear or progress. Since the image that the incident commander is seeing is static,

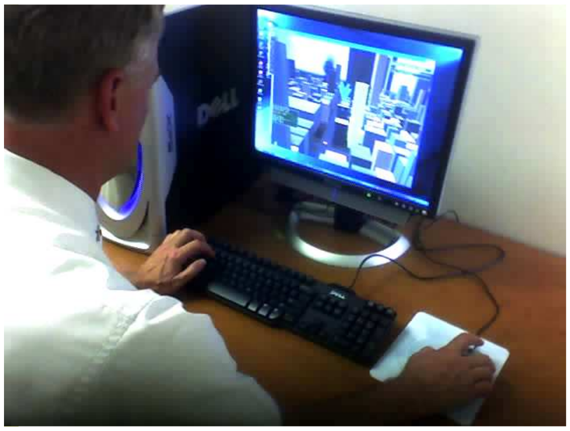


Figure 2.4: Fire Captain Roemer using the DEFACTO training system.

there is no information about state or conditions of the fire that can be ascertained from watching it, which is contrary to the actual scene of a disaster response. Furthermore, the fire's behavior is determined by the reports of the acting fire fighters over the walkie-talkie, which at times might not be a plausible progression of fire in reality. Third, this method of training restricts it to a smaller scale of fire because of the limited personnel and rigid fire representation.

Therefore, applying DEFACTO to disaster response holds a great potential benefit to the training of incident commanders in the fire department. DEFACTO would provide intelligent software agents that simulate first responder tactics, decisions, and behaviors in simulated urban areas and allow the incident commander (human) to interact. These agents form teams, where each agent simulates a fire engine, which plans and acts autonomously in a simulated environment. When using DEFACTO, incident commander trainees have the opportunity to see the disaster and the coordination/resource constraints unfold so that they can be better prepared when commanding over an actual disaster. Through interactions with these software agents, an incident commander can evaluate tactics and realize the consequences of key decisions, while responding to such disasters.

Thus, our system aims to enhance the training of the incident commanders (see Figure 2.4). Our approach allows for training to not be so personnel heavy, because fire fighter actors will be replaced by agents. By doing this we can start to train incident commanders with a larger team. Through our simulation, we can also start to simulate larger events in order to push the greater number of available resources to their limit. Also, by simulating the fire progression, we can place the Incident commander in a more realistic situation and force them to react to realistic challenges that arise. However, DEFACTO can also be seen as simulating a futuristic disaster response where humans are collaborating with a team of autonomous fire engines, each controlled by an agent.

2.3 Background on Adjustable Autonomy

In this thesis, I focus on a key aspect of the human-multiagent team coordination: Adjustable Autonomy. Adjustable autonomy refers to an agent's ability to dynamically change its own control, possibly to transfer this control over a decision to a human [Scerri et al., 2002; Goodrich et al., 2001; Dorais et al., 1998]. Previous work on adjustable autonomy could be categorized as either involving a single person interacting with a single agent (the agent itself may interact with others) or a single person directly interacting with a team [Goodrich et al., 2007; Reitsema et al., 2005; Owens et al., 2006; Sellner et al., 2006]. In the single-agent single-human category, the concept of flexible transfer-of-control strategy has shown promise [Scerri et al., 2002]. A transfer-of-control strategy is a preplanned sequence of actions to transfer control over a decision among multiple entities. For example, an AH_1H_2 strategy implies that an agent (A) attempts a decision and if the agent fails in the decision then the control over the decision is passed to a human H_1, and then if H_1 cannot reach a decision, then the control is passed to H_2. Since previous work focused on single-agent single-human interaction, strategies were individual agent strategies where only a single agent acted at a time.

An optimal transfer-of-control strategy optimally balances the risks of not getting a high quality decision against the risk of costs incurred due to a delay in getting that decision. Flexibility in such strategies implies that an agent dynamically chooses the one that is optimal, based on the situation, among multiple such strategies (H_1A, AH_1, AH_1A, etc.) rather than always rigidly choosing one strategy. The notion of flexible strategies, however, has not been applied in the context of humans interacting with agent-teams. Thus, a key question is whether such flexible transfer of control strategies are relevant in agent-teams, particularly in a large-scale application such as ours.

Throughout this thesis, adjustable autonomy will be referring to the autonomy over a *decision*. The general term "decision" will represent any decision that the human-multiagent team must make in order coordinate and achieve its goals. For example, the decision could be either to execute an action, reorganize the team structure, or allocate a team plan's role.

2.4 Modeling Continuous Time

2.4.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a mathematical model for decision making where the outcomes of actions are stochastic [Puterman, 1994]. A MDP assumes that there is a synchronous dialog between some world and some decision making entity. At each step of this dialog, the entity makes a decision which affects the world and then receives an observation which uniquely determines the new state of the world. I shall from now on refer to this entity as an *agent*. Formally MDP is a tuple $\langle S, A, P, R\rangle$ where S is a set of states of the world, A is a set of agent actions, P is a transition function and R is a reward function. At each step of the dialog, an agent that is in some state $s \in S$ chooses some action $a \in A$, transitions to a new state $s' \in S$ with probability $P(s, a, s')$ and receives a reward of $R(s, a, s')$.

2.4.2 Time-dependent MDPs

The standard MDP model as described above has a significant drawback when factoring time into the state s: the state space explodes. In addition, the standard model, each action must take one time step. Very often in realistic domains there are varying durations for actions and the action durations are uncertain. The semi-markovian decision model does allow for action durations to be sampled from a given distribution. However, the policy of a semi-markovian decision model is not dependent on time, but only state. Consequently, using a semi-markovian approach would result in an extremely large state space if it were to reason about the policy for each state s over continuous time.

There has been initial work done on an augmented model of MDPs that are time dependent [Boyan and Littman, 2000]. This approach deals with only discrete distributions that are slow to compute, however it produces time-dependent policies. This work has been extended in [Li and Littman, 2005] to have the model include action durations that are sampled from continuous distributions and also provided speedups to the model that is referred to as a TMDP (Time-dependent Markov Decision Process).

The TMDP model addresses a continuous state space that is continuous in one dimension: time. It does this by allowing the state-space to remain with all discrete variables and have a finite amount of discrete states. However the policy is a continuous time policy that maps each of the states over absolute time to the optimal action at that time. A more detailed explanation of the TMDP model appears later in Section 6.2.1.1, where the model is augmented.

Currently, TMDP is the best model to approach planning problems with uncertainty and continuous time. The TMDP model augments a general MDP with two sources of uncertainty: action durations and action outcomes. Execution of the policy is considering a deadline, beyond which reward is no longer able to be earned. The objective of the agent is to maximize its expected total reward until execution stops at the deadline.

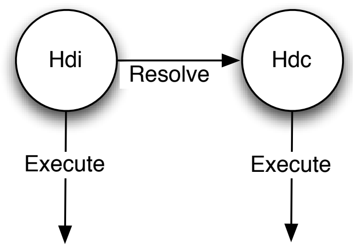
2.4.3 CPH Solver

There has been a significant amount of research recently on methods for solving a TMDP [Feng et al., 2004; Guestrin et al., 2004; Li and Littman, 2005; Petrik, 2007]. The primary challenge that any TMDP solver must address is how to perform value iteration over an infinite number of states because the time dimension is continuous. Each of the techniques leverage the tradeoff between the algorithm run time and the quality of the solution. They do so by making approximations of the action duration distributions.

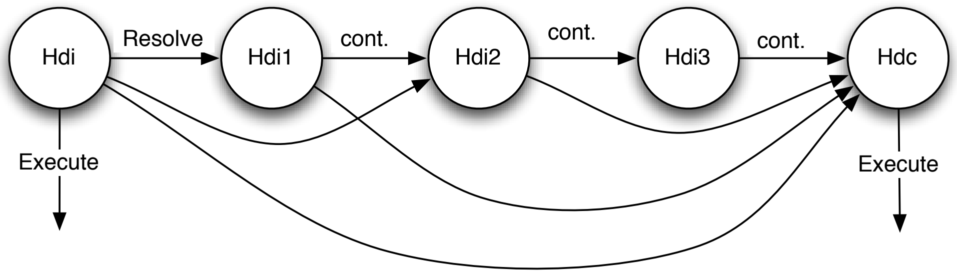
I have chosen to utilize the Continuous Phase (CPH) solver [Marecki et al., 2007] which has been shown to be the fastest of the TMDP solvers. CPH approximates each action duration distribution with a phase type distribution, which is a directed graph whose transitions are sampled from an exponential distribution (for example $e^{-\lambda_1 t}$, $e^{-\lambda_2 t}$, $e^{-\lambda_3 t}$ ...). Often, in order to increase the accuracy of the approximation of a non-exponential distribution, the process adds extra intermediate states, each with exponential transitions. This is the case with a common approximation technique known as a Coxian. After each action distribution is approximated as one or more exponential distributions, the result is a new model where every transition is an exponential distribution, albeit with possibly more states.

Figure 2.5-a shows an example action, *resolve*, which has a non-exponential action duration. Figure 2.5-b shows the extra states that are added according to a Coxian approximation. However in this new larger diagram, each transition is now an exponential. As seen in Figure 2.5-b, a phase type distribution can be modeled by having the action *cont*. (continue) result in either proceeding to a future state or self transitioning with a certain probability. Even though extra states have been created, these intermediate states ($Adi1$, $Adi2$, and $Adi3$) exist to provide an accurate action duration. They do not represent states in the real world, but instead represent being somewhere in the middle of the *resolve* action. In addition, the model undergoes an uniformization process [Puterman, 1994] that converts all of the transitions (arrows in the diagram) to each have the same λ, where the transition duration is sampled from $e^{-\lambda t}$. This results in the altering of the transition probabilities and adding self-transitions (see Figure 2.5-c). This figure will appear again to be expanded upon in Figure 6.8.

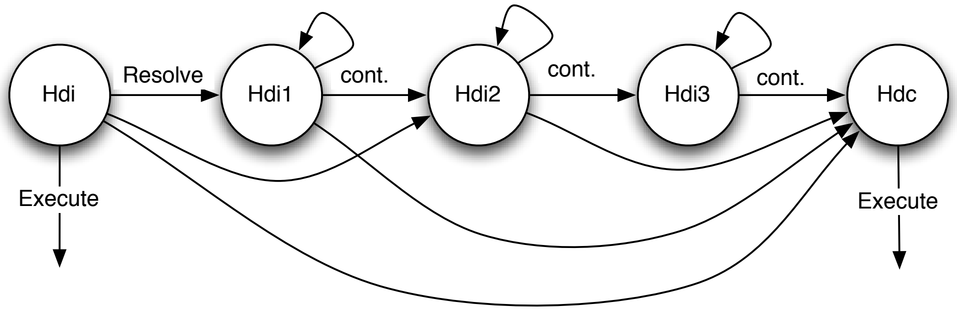
Once all of the model's transitions have been converted into uniform phase type distributions, CPH can achieve a great speedup by using numerical analysis methods to compute its TMDP policy. Although CPH must often handle more states than other solvers, it achieves faster runtimes by solving this larger state space analytically.



(a) Original resolve action model



(b) With additional states for approximations



(c) CPH converted model after uniformization

Figure 2.5: Figure (a) shows an example action, *resolve* which has a non-exponential action duration when transferring from *Adi* to *Adc*.. Figure (b) shows the model after the arbitrary action duration distribution is converted to an approximation as a sum of exponential distributions. Figure (c) shows the final CPH model after all transitions have undergone uniformization and self-transitions have been added.

Chapter 3

Related Work

In this chapter, I will cover three areas of work that are related to this thesis: (i) training systems, (ii) mixed initiative planning, and (iii) humans augmenting existing teams.

3.1 Training Systems

In the area of military training, there are products like JCATS [Joint Warfighting Center, 2005] and EPICS [Advanced Systems Technology, 2005]. JCATS represents a self-contained, high-resolution joint simulation in use for entity-level training in open, urban and subterranean environments. Developed by Lawrence Livermore National Laboratory, JCATS gives users the capability to detail the replication of small group and individual activities during a simulated operation. At this point however, JCATS does not incorporate simulated agent teammates. Finally, EPICS is a computer-based, scenario-driven, high-resolution simulation. It is used by emergency response agencies to train for emergency situations that require multi-echelon and/or inter-agency communication and coordination. Developed by the U.S. Army Training and Doctrine Command Analysis Center, EPICS is also used for exercising communications and command and control procedures at multiple levels. Similar to JCATS however, EPICS does not allow agents to participate in the simulation. More recently multiagents have been successfully applied to training navy tactics [van Doesburg et al., 2005] and teams of Uninhabited Air Vehicles [Baxter and Horn, 2005; Karim and Heinze, 2005]. Our work is similar to these in spirit, however our focus and lessons learned are based on the training of Incident Commanders in disaster rescue environments. More importantly, the issues of adjustable autonomy and performance guarantees are emphasized in our work, since our ultimate long-term goals go beyond training to potential deployment in the real world.

Another major area is arising by leveraging the strengths of the video game industry and applying them to training to create a more immersive and enjoyable experience. This area is beginning to be referred to as “serious games” [Zyda, 2005], but has predecessors in military training such as the Mission Rehearsal Exercise system [Hill et al., 2001] and battlefield simulations for helicopter pilots [Hill et al., 1997]. More recently full computer and consoles have had video games released that have very strong training applications such as America’s Army [Zyda et al., 2005] and Full Spectrum Command [Swartout and van Lent, 2003]. Also, in a very related project at Carnegie Mellon University, a video game is being used to train fire fighters to handle hazardous materials [Schell, Spring 2007]. Our work is distinguished from current serious games because of our emphasis on adjustable autonomy for the interaction.

3.2 Mixed Initiative Planning

Mixed Initiative can be broadly defined as studying the collaboration of a human and a system, where either can drive the process or take the lead in the partnership. This being the case, adjustable autonomy can be considered as a specific subproblem of this general research area. However, mixed initiative systems, as they are approached in current practice, focus on collaboration during the synthesis of plans, whereas adjustable autonomy focuses on collaboration during the execution of a plan. Consequently mixed initiative work often makes the following assumptions: (i) focus on the collaborative process and don't worry about the human being there, (ii) the collaboration is occurring in an offline situation with the planning being done by a centralized system. This thesis in the context of adjustable autonomy makes very different assumptions about the collaborative process, which alters what kinds of techniques can be use in the collaboration process. Some very relevant work has been done in allowing a human to give offline guidance for adjustable autonomy and strategy preference to a team of agents [Myers and Morley, 2003]. Another example, in the area of applied trip planning, the TRAINS project [Ferguson et al., 1996], the later TRIPS [Ferguson and Allen, 1998], and the Collagen system [Rich and Sidner, 1998] are all instances of this work that fall under the area of mixed initiative planning and implement a variety of different ways for the human planner to interact. More recently, the PLOW [Allen et al., 2007] software assistant collaborates with a human to learn a task model.

Because the mixed initiative work has been focused on static generation offline, it could benefit from the work presented as they move towards online planning. I believe that there is even further synergy possible with the new RIAACT model and the mixed initiative work. Adjustable autonomy has previously focused on the dynamic transfer of control between a human-agent team. With the addition of the resolve action in the RIAACT model, a human and an agent must resolve an inconsistency in a coordinated fashion. This resolve action can be modeled as a mixed initiative dialogue where each party is attempting to bridge this information gap. The RIAACT model could reason over time distribution of how long the mixed initiative dialogue will take to reach a modified plan.

This idea of an ”information gap” is implicitly one of the focuses of mixed initiative systems. Some areas of mixed initiative research attempt to allow systems that do not have enough domain knowledge to work alongside domain-expert humans. These humans help to fill in any gaps through a mixed-initiative dialogue. However, in order for the human to help, the human must be interrupted by the system. There has been work done that tries to compute the expected cost of interruption for a user in [Horvitz and Apacible, 2003]. This work focuses on the modeling of the human in order to guide the interaction with a centralized system and a human. My thesis tries to solve a related problem. For one, the teams that I am looking at are inherently decentralized and must collaborate as a cohesive team. In addition, adjustable autonomy is geared toward plan execution, rather than mixed initiative which primarily concentrates on the challenges of the synthesis problems such as planning and scheduling. Because of this, mixed initiative is not concerned with how long the dialogue may take and consequently deadlines. However the work in this thesis could be enhanced by having a more accurate model of the human.

3.3 Humans Augmenting Agent/Robot Teams

There has also been a lot of work where humans are beginning to interact with agent/robot teams [Scerri et al., 2003; Crandall et al., 2003; Fong et al., 2002; Goodrich et al., 2001; Varakantham et al., 2005]. These efforts recognize that humans may not provide a timely response. In part to alleviate the lack of such timely response, Scerri et al [Scerri et al., 2002] leveraged the notion of *adjustable autonomy strategies*. Our work already incorporates such strategies, but recognizes that human users may still provide such imperfect or incorrect inputs. There has also been some work that involves humans interacting with multiagent teams actually leading to performance degradation due to imperfect input: In past work [Schurr et al., 2003] illustrated that an autonomous team of simulated robots performed better than when aided with human inputs (although in some situations the humans were able to improve the simulated robot performance). However, this work did not address the question of how the simulated robots would recover from such setbacks.

Some projects that deal with agent interactions with humans have started to focus on the safety of those humans. Consequently, they have started to look to Asimov's laws of robotics for some crucial guarantees. This past work ([Weld and Etzioni, 1994; Gordon, 2000; Pynadath and Tambe, 2001]) has focused on Asimovian agents but dealt with only the first law (solely against human harm). More specifically, this previous work only focused on a single law (the first law of safety) and, in the process, addressed two of the issues I mentioned earlier in Section 9.2: defining the notion of harm to humans and applying the laws to teams rather than individual agents. The key novelty of our work is going beyond previous work to consider the second of Asimov's laws, and more importantly in recognizing the fundamental role that uncertainty plays in any faithful implementation of such a law. In particular, Asimov's second law addresses situations where an agent or agent team may or may not obey human orders it specifies that in situations where (inadvertent) harm may come to other humans, agents may disobey an order. However, in the presence of uncertainty faced either by the agents or the human user about each other's state or state of the world, either the set of agents or the human may not be completely certain of their inferences regarding potential harm to humans. This thesis illustrates that in the presence of such uncertainty, agents must strive to gather additional information or provide additional information. Given that the information reduces the uncertainty, agents may only then disobey human orders to avoid harm.

Chapter 4

DEFACTO System

This thesis is presented in the context of a disaster response system called DEFACTO. While this thesis will focus on DEFACTO as a domain, the approach is more generally applicable to human-multiagent teams, as described in Section 2.1.

In this chapter, I will describe DEFACTO's architecture in order to better explain the domain in which the experiments in Chapters 5 and 6 have been run. This system provides a unique testbed for human-multiagent teams and has potential as both a training tool and a decision support system. In addition, this architecture will be useful in understanding the hybrid approach mentioned later in Chapter 6.

4.1 Aims of DEFACTO

I have constructed a new system, DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence). DEFACTO incorporates multiagent teams, 3D visualization and human-interaction reasoning into a unique high fidelity system for training incident commanders. By providing the incident commander interaction with the coordinating agent team in a complex environment, the commander can gain experience and draw valuable lessons that

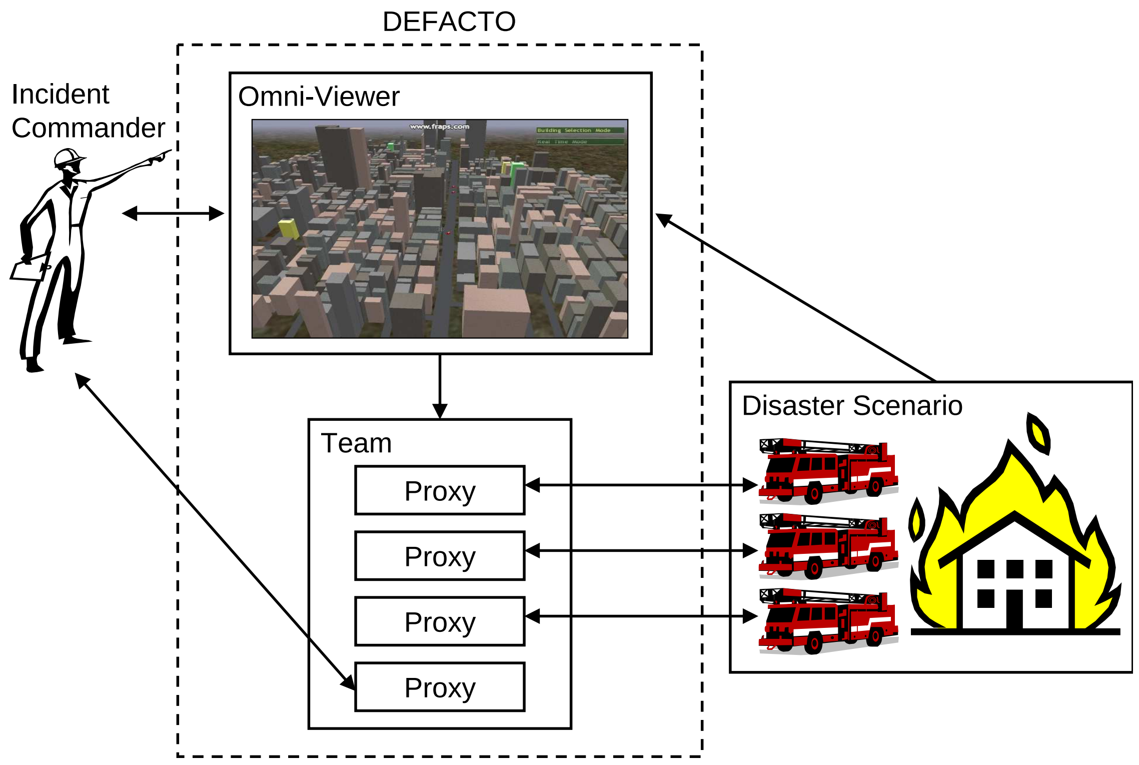


Figure 4.1: DEFACTO system architecture.

will be applicable in the real world. The DEFACTO system is comprised of three main components: (i) Omnipresent Viewer - intuitive interface, (ii) Proxy Framework - for team coordination, and (iii) Flexible Interaction - between the incident commander and the team.

4.2 System Overview

First, DEFACTO incorporates a viewer that allows for the human to have an *omnipresent* interaction with remote agent teams. We refer to this as the Omni-Viewer (see Figure 4.1), and it combines two styles of operation: (i) a navigable, high quality 3D visualization of the world and (ii) a traditional 2D view with a list of possible task allocations that the human may perform.

With the Omni-Viewer, human experts can quickly observe on-going agent and world activity, taking advantage of both the brain's favored visual object processing skills (relative to textual search, [Paivio, 1974]), and the fact that 3D representations can be innately recognizable, without the layer of interpretation required of map-like displays or raw computer logs. The 3D mode enables the human to understand the local perspectives of each agent in conjunction with the global, system-wide perspective that is obtained in the 2D mode.

Second, DEFACTO utilizes a proxy framework that handles both the coordination and communication for the human-multiagent team (see Figure 4.1). We use coordination algorithms inspired by theories of teamwork to manage the distributed response [Scerri et al., 2003]. The general coordination algorithms are encapsulated in *proxies*, with each team member having its own proxy and representing it in the team.

Third, we also use the above mentioned proxy framework to implement adjustable autonomy (AA) between each member of the human-multiagent team. The novel aspect of DEFACTO's flexible AA is that we generalize the notion of *strategies* from single-agent single-human context to agent-team single-human. In our work, agents may flexibly choose among team strategies for adjustable autonomy instead of only individual strategies; thus, depending on the situation, the agent team has the flexibility to limit human interaction, and may in extreme cases exclude humans from the loop.

4.2.1 Omni-Viewer

Our goal of allowing fluid human interaction with agents requires a visualization system that provides the human with a global view of agent activity as well as shows the local view of a particular

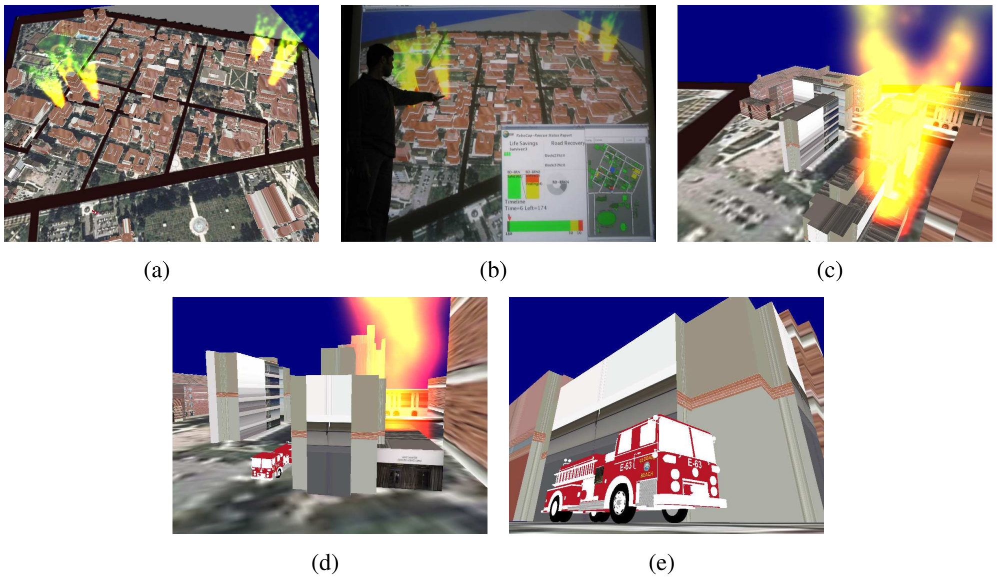


Figure 4.2: Omni-Viewer during a scenario: (a) Multiple fires start across the campus, (b) The Incident Commander navigates to quickly grasp the situation, (c) Views a closer look at one of the fires and assigns a fire engine, (d) The fire engine has arrived at the fire, (e) The fire has been extinguished.

agent when needed. Hence, we have developed an omnipresent viewer, or Omni-Viewer [1], which will allow the human user diverse interaction with remote agent teams. While a global view is obtainable from a two-dimensional map, a local perspective is best obtained from a 3D viewer, since the 3D view incorporates the perspective and occlusion effects generated by a particular viewpoint. The literature on 2D- versus 3D-viewers is ambiguous. For example, spatial learning of environments from virtual navigation has been found to be impaired relative to studying simple maps of the same environments [Richardson et al., 1999]. On the other hand, the problem may be that many virtual environments are relatively bland and featureless. Ruddle points out that navigating virtual environments can be successful if rich, distinguishable landmarks are present [Ruddle et al., 1997].

[1]The Omni-Viewer was developed with the help of the following collaborators: Nikhil Kasinadhuni, Pratik Patil, Ankit Modi, J. P. Lewis, and Fred Pighin.

To address our discrepant goals, the Omni-Viewer incorporates both a conventional map-like 2D view and a detailed 3D-perspective viewer (Figure 4.2). The global overview shows events as they are progressing and provides a list of tasks that the agents have transferred to the human. The user can navigate the same dynamic world view, while allowing for more freedom to move to desired locations and views. In particular, the user can drop to the virtual ground level, thereby obtaining the world view (local perspective) of a particular agent. At this level, the user can "walk" freely around the scene, observing the local logistics involved as various entities are performing their duties. This can be helpful in evaluating the physical ground circumstances and altering the team's behavior accordingly. It also allows the user to feel immersed in the scene where various factors (psychological, etc.) may come into effect.

4.2.2 Proxy: Team Coordination

A key premise in this work is that intelligent distributed agents will be a key element of a future disaster response. Taking advantage of emerging robust, high bandwidth communication infrastructure, we believe that a critical role of these intelligent agents will be to manage coordination between all members of the response team. Specifically, we are using coordination algorithms inspired by theories of teamwork to manage the distributed response [Scerri et al., 2003]. The general coordination algorithms are encapsulated in *proxies*, with each team member having its own proxy and representing it in the team. The current version of the proxies is called *Machinetta* [Scerri et al., 2003] and extends the successful Teamcore proxies [Pynadath and Tambe, 2003]. Machinetta leverages BDI (Belief Desire and Intentions) based logic to guide communication and establish joint commitments to serve the team plans. Machinetta is implemented in Java and is freely available on the web. Notice that the concept of a reusable proxy differs from many

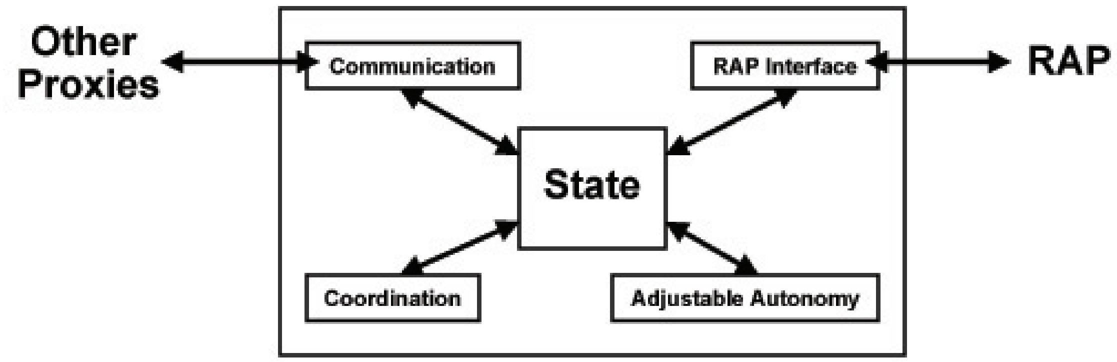


Figure 4.3: Proxy Architecture.

Communication: communication with other proxies

Coordination: reasoning about team plans and communication

State: the working memory of the proxy

Local Adjustable Autonomy: reasoning about whether to act autonomously or pass control to the team member the proxy represents

RAP Interface: communication with the team member

Figure 4.4: Proxy Modules.

other "multiagent toolkits" in that it provides the coordination *algorithms*, e.g., algorithms for allocating tasks, as opposed to the *infrastructure*, e.g., APIs for reliable communication.

The Machinetta software consists of five main modules, as seen in architecture diagram in Figure 4.3. Figure 4.4 provides a brief overview of each of the modules. Three of the modules are domain independent, while the other two are tailored for specific domains. The three domain independent modules are for coordination reasoning, maintaining local beliefs (state) and local adjustable autonomy. The domain specific modules are for communication between proxies and communication between a proxy and a team member. The modules interact with each other only via the local state with a blackboard design and are designed to be "plug and play." Thus new adjustable autonomy algorithms can be used with existing coordination algorithms. The coordination reasoning is responsible for reasoning about interactions with other proxies, thereby implementing the coordination algorithms. The local adjustable autonomy algorithms reason about the interaction with the team member, providing the possibility for the team member to make any coordination decision instead of the proxy. It should be noted that the local adjustable autonomy described here focuses on the delegation of control between a team member and its representative proxy. This is distinct from the adjustable autonomy that occurs between agent and human team members, which is the focus of this thesis. It should be noted that this local adjustable autonomy module is augmented to allow for team-level adjustable autonomy in Chapter 5. This extends the reasoning about the transfer of control to not only its attached teammate, but other teammates as well.

Teams of proxies implement *team oriented plans* (TOPs) [Pynadath et al., 1999] which describe joint activities to be performed in terms of the individual *roles* to be performed and any constraints between those roles. The TOP serves as a reactive plan for the team. Generally, TOPs are instantiated dynamically from TOP templates at runtime when preconditions associated with the templates are filled. Typically, a large team will be simultaneously executing many TOPs. For example, in my disaster response domain, a team would execute a TOP similar to that shown in Figure 4.5 each time a fire is reported.

The TOPs can also contain constraints between these roles that specify interactions such as required execution ordering and whether one role can be performed if another is not currently being performed. For example, given our TOP example above, the roles for Fight Back and Fight Front can have an AND constraint. This would be very helpful for a large-scale fire that would be too risky for one fire engine company to fight alone from the front. The AND would force both

Simple Team Oriented Program (TOP) Example

Plan Precondition: Fire exists in building X.

Plan Body: Assign fire engines (and personnel) to Fight Fire Roles (move to building and extinguish fire).

Role: Fight Front of Fire in building X.

Role: Fight Back of Fire in building X.

Role: Fight Side of Fire in building X.

Plan Postcondition: (Succeed) Fire in building X has been extinguished.

Plan Postcondition: (Fail) Fire in building X has burned building completely down.

Figure 4.5: TOP example for fighting of a fire that is found in building X.

the front and back to have someone assigned and fighting, before a fire engine would act on the role. Notice that TOPs do not specify the coordination or communication required to execute a plan; the proxy determines the coordination that should be performed.

Current versions of Machinetta cast the role allocation problem as a Distributed Constraint Optimization Problem (DCOP) and solve it via the LA-DCOP role allocation algorithm [Scerri et al., 2005]. LA-DCOP exploits tokens to solve the DCOP. The decision for the agent becomes whether to assign values represented by tokens it currently has to its variable or to pass the tokens on. Algorithm 1 shows a portion of the LA-DCOP algorithm that is used for token passing (Token Monitor). First, the team member can choose the minimum capability the agent should have in order to assign the value. This minimum capability is referred to as the *threshold*. The threshold is calculated once (Algorithm 1, line 6), and attached to the token as it moves around the team.

Second, the agent must check whether the value can be assigned while respecting its local resource constraints (Algorithm 1, line 9). If the value cannot be assigned within the resource constraints of the team member, it must choose a value(s) to reject and pass on to other teammates in the form of a token(s) (Algorithm 1, line 12). The agent keeps values that maximize the use of its capabilities (performed in the MaxCap function, Algorithm 1, line 10).

Algorithm 1 TokenMonitor($Cap, Resources$)

```
1:  $V \leftarrow \emptyset$
2:  **while** true **do**
3:      $msg \leftarrow getMsg()$
4:      $token \leftarrow msg$
5:      **if** $token.threshold = NULL$ **then**
6:          $token.threshold \leftarrow$ ComputeThreshold($token$)
7:      **if** $token.threshold \leq Cap(token.value)$ **then**
8:          $V \leftarrow V \cup token.value$
9:          **if** $\sum_{v \in V} Resources(v) \geq agent.resources$ **then**
10:             $out \leftarrow V -$ MaxCap($Values$)
11:             **for all** $v \in out$ **do**
12:                 PassOn($newtoken(v)$)
13:             $Values \leftarrow Values - out$
14:     **else**
15:         PassOn($token$) /* $threshold > Cap(token.value)$ */
```

Chapter 5

Applying Adjustable Autonomy to Teams

This chapter reports on experiments with humans in order to explore the effects of applying adjustable autonomy to the team-level. This will demonstrate some of the new key challenges that arise and will be addressed later in Chapter 6. An important difference that influences these new challenges is that autonomy can now be adjusted between human and multiple entities. In addition, the goal is to improve the human-multiagent team performance, rather than merely to serve the human.

DEFACTO is an experimental test bed to test transfer-of-control strategies in the context of human-multiagent teams. One key advance in DEFACTO is that the strategies are not limited to individual agent strategies, but also enables team-level strategies. For example, rather than transferring control from a human to a single agent, a team-level strategy could transfer control from a human to an agent-team. DEFACTO leverages teamwork proxies and consequently each team member (agent or human) has a representative proxy (for more details see Section 4.2.2). Each team member combined with its representative proxy can be considered a full-fledged teammate. In the context of this thesis, adjustable autonomy strategies are implemented by the proxy on behalf of its full-fledged teammate. The techniques presented in this thesis do not require the use of a proxy architecture, however it facilitates the design and modification of DEFACTO.

Concretely, each proxy is provided with four defined strategy options; the key is to select the right strategy given the situation. Previously, the adjustable autonomy module would only reason about strategies that adjust between a team mate and its proxy, however team level adjustable autonomy focuses on strategies that transfer control to any other teammates. For example, if a team level strategy would want to leverage strengths of both the agents and the human on the team, it would combine A_T Strategy and H Strategy in order to make the A_TH Strategy. The agent team strategy, A_T, keeps control over a decision with the agent team for the entire duration of the decision. The H strategy always immediately transfers control to the human. A_TH strategy is the conjunction of team level A_T strategy with H strategy. This strategy aims to significantly reduce the burden on the user by allowing the decision to first pass through all agents before finally going to the user, if the agent team fails to reach a decision. In these strategies, the transfer occurs after the single agent or the agent team has finished attempting a decision.

There is a large space of possible strategies that can be applied to a human-multiagent team. In order to try and find the best strategies to use, the autonomy of team members can be reasoned over using a decision theoretic model. Consequently, the policy that results determines the transfer of control and can be considered the adjustable autonomy strategy. In the next section, I will present some initial models that are used to reason about team-level adjustable autonomy. At the end of this chapter, some of the drawbacks to this approach are highlighted. Later, in Chapter 6, I will present a new technique that addresses these challenges.

5.1 Mathematical Model of Strategy Selection

Whereas strategies in Scerri's work [Scerri et al., 2002] are based on a single decision that is sequentially passed from agent to agent, here it is assumed that there are multiple homogeneous agents concurrently working on multiple tasks interacting with a single human user. These assumptions (which fit the domain) are exploited to obtain a reduced version of the model and simplify the computation in selecting strategies.

5.1.1 Background on individual strategies

A decision, d, needs to be made. There are n entities, $e_1 \ldots e_n$, who can potentially make the decision. These entities can be human users or agents. The expected quality of decisions made by each of the entities, $\mathbf{EQ} = \{EQ_{e_i,d}(t) : \mathcal{R} \rightarrow \mathcal{R}\}_{i=1}^{n}$, is known, though perhaps not exactly. $\mathbf{P} = \{P_{\top}(t) : \mathcal{R} \rightarrow \mathcal{R}\}$ represents continuous probability distributions over the time that the entity in control will respond (with a decision of quality $EQ_{e,d}(t)$). The cost of delaying a decision until time t is denoted as $\{\mathcal{W} : t \rightarrow R\}$. The set of possible wait-cost functions is $\mathbf{W}$. $\mathcal{W}(t)$ is non-decreasing and at some point in time, Γ, when the costs of waiting stop accumulating (i.e., $\forall t \geq \Gamma, \forall \mathcal{W} \in \mathbf{W}, \mathcal{W}(t) = \mathcal{W}(\Gamma)$).

To calculate the expected utility, EU, of an arbitrary strategy, the model multiplies the probability of response at each instant of time with the expected utility of receiving a response at that instant, and then sum the products. Hence, for an arbitrary continuous probability distribution if e_c represents the entity currently in decision-making control:

$$EU = \int_0^\infty P_\top(t) EU_{e_c,d}(t)\ .dt \qquad (5.1)$$

Since the primary focus of this thesis is in the effects of delay caused by transfer of control, the expected utility of a decision at a certain instant, $EU_{e_c,d}(t)$, is decomposed into two terms. The first term captures the quality of the decision, independent of delay costs, and the second captures the costs of delay: $EU_{e_c,d}t = EQ_{e,d}(t) - \mathcal{W}(t)$. To calculate the EU of a strategy, the probability of response function and the wait-cost calculation must reflect the control situation at that point in the strategy. If a human, H_1 has control at time t, $P_\top(t)$ reflects H_1's probability of responding at t.

5.1.2 Introduction of team level strategies

A_T **Strategy:** Starting from the individual model, team level A_T strategy, where the agent team is denoted as A_T, are introduced in the following way: Start with Equation 5.2 for single agent A and single task d. Obtain Equation 5.3 by discretizing time, $t = 1, ..., T$ and introducing set Δ of tasks. Probability of agent A performing a task d at time t is denoted as $P_{a,d}(t)$. Equation 5.4 is a result of the introduction of the set of agents $AG = a_1, a_2, ..., a_k$. Assume the same quality of decision for each task performed by an agent and that each agent A has the same quality so that $EQ_{a,d}(t)$ reduces to $EQ(t)$. Given the assumption that each agent A at time step t performs one task, $\sum_{d\in\Delta} P_{a,d}(t) = 1$ which is depicted in Equation 5.5. Then express $\sum_{a=a_1}^{a_k} \sum_{d\in\Delta} P_{a,d}(t) \times W_{a,d}(t)$ as the total team penalty for time slice t, i.e, at time slice t, subtract one penalty unit for each not completed task as seen in Equation 5.6. Assuming penalty unit $PU = 1$ finally resulting Equation 5.7.

$$EU_{a,d} = \int_0^{\infty} P_{\top a}(t) \times (EQ_{a,d}(t) - \mathcal{W}(t)).dt \tag{5.2}$$

$$EU_{a,\Delta} = \sum_{t=1}^{T} \sum_{d \in \Delta} P_{a,d}(t) \times (EQ_{a,d}(t) - \mathcal{W}(t)) \tag{5.3}$$

$$EU_{A_T,\Delta} = \sum_{t=1}^{T} \sum_{a=a_1}^{a_k} \sum_{d \in \Delta} P_{a,d}(t) \times (EQ_{a,d}(t) - W_{a,d}(t)) \tag{5.4}$$

$$EU_{A_T,\Delta,AG} = \sum_{t=1}^{T} (\sum_{a=a_1}^{a_k} EQ(t) - \sum_{a=a_1}^{a_k} \sum_{d \in \Delta} P_{a,d}(t) \times W_{a,d}(t)) \tag{5.5}$$

$$EU_{A_T,\Delta,AG} = \sum_{t=1}^{T} (|AG| \times EQ(t) - (|\Delta| - |AG| \times t) \times PU) \tag{5.6}$$

$$EU_{A_T,\Delta,AG} = |AG| \times \sum_{t=1}^{T} (EQ(t) - (\frac{|\Delta|}{AG} - t)) \tag{5.7}$$

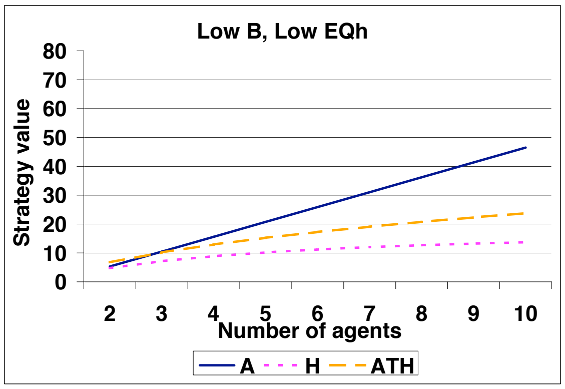
H **Strategy:** The difference between $EU_{H,\Delta,AG}$ and $EU_{A_T,\Delta,AG}$ results from three key observations: First, the human is able to choose strategic decisions with higher probability, therefore his $EQ_H(t)$ is greater than $EQ(t)$ for both individual and team level A_T strategies. Second, I hypothesize that a human cannot control all the agents AG at disposal, but due to cognitive limits will focus on a smaller subset, AG_H of agents (evidence of limits on AG_H appears later in Figure 5.4-a). $|AG_H|$ should slowly converge to B, which denotes the upper limit of the number of agents the human can focus on ($|AG_H|$), but never exceeds the total number of agents, AG. Each function $f(AG)$ that models AG_H should be consistent with three properties: i) if $B \to \infty$ then $f(AG) \to AG$; ii) $f(AG) < B$; iii) $f(AG) < AG$. Third, there is a delay in human decision making compared to agent decisions. This phenomenon is modeled by shifting H to start at time slice t_H. For $t_H - 1$ time slices the team incurs a cost $|\Delta| \times (t_H - 1)$ for all incomplete tasks. By inserting $EQ_H(t)$ and AG_H into the time shifted utility equation for A_T strategy to obtain the H strategy (Equation 5.8).

$A_T H$ **Strategy:** The $A_T H$ strategy is a composition of H and A_T strategies (see Equation 5.9). Essentially, this strategy the whole team of agents will attempt to allocate a particular task and if not able to, will ask the human for help.

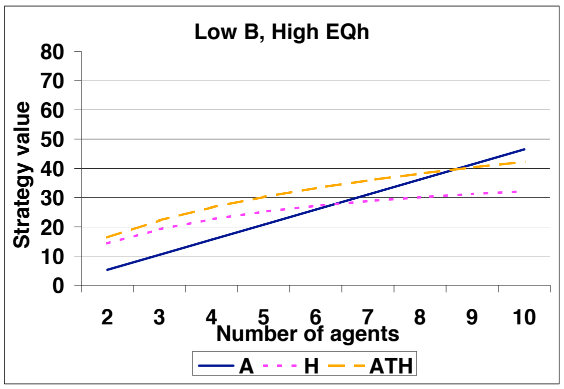
$$EU_{H,\Delta,AG} = |AG_H| \times \sum_{t=t_H}^{T}(EQ_H(t)$$
$$-(\frac{|\Delta|}{AG_H} - (t - t_H))) - |\Delta| \times (t_H - 1) \tag{5.8}$$

$$EU_{A_TH,\Delta,AG} = |AG| \times \sum_{t=1}^{t_H-1}(EQ(t) - (\frac{|\Delta|}{|AG|} - t))$$
$$+|AG_H| \times \sum_{t=t_H}^{T}(EQ_H(t) - (\frac{|\Delta| - |AG|}{|AG_H|} - (t - t_H)))) \tag{5.9}$$

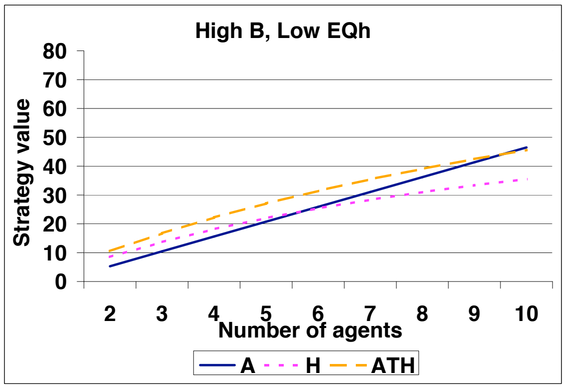
Strategy utility prediction: Given the strategy equations and the assumption that $EQ_{H,\Delta,AG}$ is constant and independent of the number of agents the graphs representing strategy utilities are plotted (Figure 5.1). Figure 5.1 shows the number of agents on the x-axis and the expected utility of a strategy on the y-axis. The focus is on humans with different skills: (a) low EQ_H, low B (b) high EQ_H, low B (c) low EQ_H, high B (d) high EQ_H, high B. The last graph representing a human with high EQ_H and high B. The curve of AH and A_TH appears to be flattening out to eventually cross the line of A_T. Moreover, observe that the increase in EQ_H increases the slope for AH and A_TH for small number of agents, whereas the increase of B causes the curve to maintain a slope for larger number of agents, before eventually flattening out and crossing the A_T line.



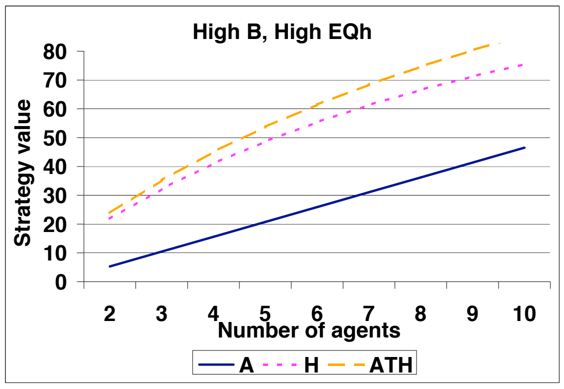
(a)



(b)



(c)



(d)

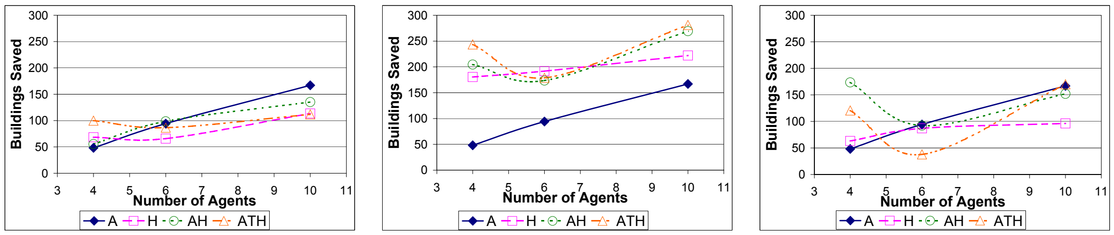
Figure 5.1: Model predictions for various users.

Team	4,6,10 fire engine agents and 1 real human
Scenario	Fires have been ignited at 3 buildings and will start increasing in intensity and possibly spread. The human-multiagent team must coordinate to assign resources and extinguish as many fires as possible.
Decision	To determine which fire engine, if any, should be allocated to a firefighting role associated with the team plan to fight a fire in a particular building.
Information	Human - Has a top down view of the whole map with intensity of fire in each building. Knows task allocation and geographic location of all agents. Agents - Each agent is aware of the size, height, and fire intensity of buildings within a limited radius, how much water is in its tank, status of only agents that are fighting the same fire, allocation of all teammates.

Figure 5.2: Initial team level AA DEFACTO experimental details.

5.2 Experiments and Evaluation

Our DEFACTO system was evaluated through experiments comparing the effectiveness of Adjustable Autonomy (AA) strategies over multiple users. In order to provide DEFACTO with a dynamic rescue domain, we connected it to the previously developed RoboCup Rescue simulation environment [Kitano et al., 1999]. Figure 5.2 explains the details of this set of experiments



(a) Subject 1 (b) Subject 2 (c) Subject 3

Figure 5.3: Performance of human test subjects in a human-multiagent team.

that were conducted. The results of our experiments are shown in Figure 5.3, which shows the results of subjects 1, 2, and 3. Each subject was confronted with the task of aiding fire engines in saving a city hit by a disaster. For each subject, we tested three strategies, specifically, H, AH and A_TH; their performance was compared with the completely autonomous A_T strategy. AH is an individual agent strategy, tested for comparison with A_TH, where agents act individually, and pass those tasks to a human user that they cannot immediately perform. Each experiment was conducted with the same initial locations of fires and building damage. For each strategy we tested, we varied the number of fire engines between 4, 6 and 10. Each chart in Figure 5.3 shows the varying number of fire engines on the x-axis, and the team performance in terms of numbers of building saved on the y-axis. For instance, strategy A_T saves 50 building with 4 agents. Each data point on the graph is an average of three runs. Each run itself took 15 minutes, and each user was required to participate in 27 experiments, which together with 2 hours of getting oriented with the system, equates to about 9 hours of experiments per volunteer.

Figure 5.3 enables us to conclude the following:

- *Human involvement with agent teams does not necessarily lead to improvement in team performance.* Contrary to expectations and prior results, human involvement does not uniformly improve team performance, as seen by human-involving strategies performing worse than the A_T strategy in some cases. For instance, for subject 3 AH strategy provides higher team performance than A_T for 4 agents, yet at 10 agents human influence is clearly not beneficial.

- *Providing more agents at a human's command does not necessarily improve the agent team performance.* As seen for subject 2 and subject 3, increasing agents from 4 to 6 given AH

and $A_T H$ strategies is seen to degrade performance. In contrast, for the A_T strategy, the performance of the fully autonomous agent team continues to improve with additions of agents, thus indicating that the reduction in AH and $A_T H$ performance is due to human involvement. As the number of agents increase to 10, the agent team does recover.

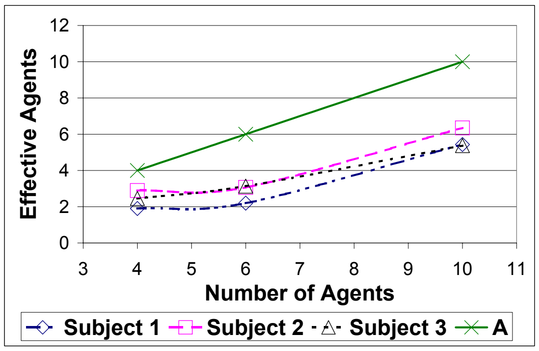
- *No strategy dominates through all the experiments given varying numbers of agents.* For instance, at 4 agents, human-involving strategies dominate the A_T strategy. However, at 10 agents, the A_T strategy outperforms all possible strategies for subjects 1 and 3.
- *Complex team-level strategies are helpful in practice.* $A_T H$ leads to improvement over H with 4 agents for all subjects, although surprising domination of AH over $A_T H$ in some cases indicates that AH may also need a useful strategy to have available in a team setting.

5.2.1 Analysis

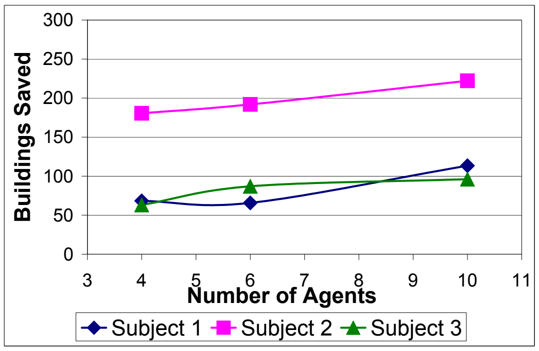
Since no single strategy dominated and, in fact, performance decreased in certain cases, this subsection looks further into this phenomenon. In particular, this section focuses on finding the causes of this dip in performance. Note that this ranges over multiple users, multiple runs, and multiple strategies. The most important conclusion from these figures is that flexibility is necessary to allow for the optimal AA strategy to be applied. The key question, then, is then how to select the appropriate strategy for a team involving a human whose expected decision quality is EQ_H. In fact, by estimating the EQ_H of a subject by checking the "H" strategy for small number of agents (say 4), and comparing to A_T strategy, we may begin to select the appropriate strategy

Strategy	H			AH			A_TH		
# of agents	4	6	10	4	6	10	4	6	10
Subject 1	91	92	154	118	128	132	104	83	64
Subject 2	138	129	180	146	144	72	109	120	38
Subject 3	117	132	152	133	136	97	116	58	57

Table 5.1: Total number of allocations given.



(a) Agents effectively used (AG_H) by the subjects

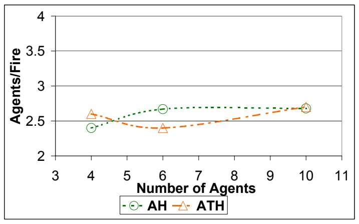


(b) Performance of strategy H

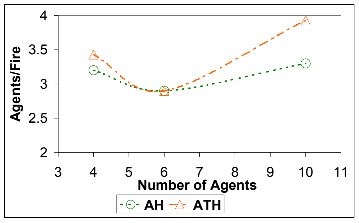
Figure 5.4: Analysis of the effects of the Human on the team.

for teams involving more agents. In general, higher EQ_H lets us still choose strategies involving humans for a larger team. For large teams however, the number of agents AG_H effectively controlled by the human does not grow linearly, and thus A_T strategy becomes dominant.

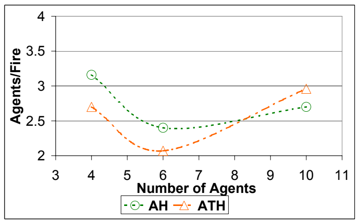
Unfortunately, the strategies including the humans and agents (AH and A_TH) for 6 agents show a noticeable decrease in performance for subjects 2 and 3 (see Figure 5.3). It would be useful to understand which factors contributed to this phenomenon.



(a) Subject 1



(b) Subject 2



(c) Subject 3

Figure 5.5: Number of agents assigned per fire.

Our crucial predictions were that while numbers of agents increase, AG_H steadily increases and EQ_H remains constant. We believed the dip at 6 agents to be due to either a lower than expected AG_H (agents in focus of human) or EQ_H (quality of human decision). We first tested AG_H in our domain. The number of effective agents, AG_H, is calculated by dividing how many total allocations each subject made by how many the A_T strategy made per agent, assuming A_T strategy effectively uses all agents. Figure 5.4-(a) shows the number of agents on the x-axis and the number of agents effective used, AG_H, on the y-axis; the A_T strategy, which is using all available agents, is also shown as a reference. However, the number of effective agents is actually about the same in 4 and 6 agents. This would not account for the sharp drop we see in the performance. We then shifted our attention to the EQ_H of each subject. One reduction in EQ_H could be because subjects simply did not send as many allocations totally over the course of the experiments. This, however is not the case as can be seen in Table 5.1 where for 6 agents, as the total number of allocations given is comparable to that of 4 fire engine agents. To investigate further, we checked if the quality of human allocation had degraded. For our domain, the more fire engine agents that fight the same fire, the more likely it is to be extinguished and in less time, which directly relates to a higher EQ_H in this domain. For this reason, the number of agents that were tasked to each fire is a good indicator of the quality of allocations that the subject makes 5.4-(b). Figure 5.5 shows the number of agents on the x-axis and the average number of fire engines allocated to each fire on the y-axis. Thus, we found the cause of the dip in performance with strategies AH and A_TH for 6 agents to be due to a significantly less average fire engines per task (fire) and therefore lower average EQ_H, and not due to limits of human cognitive load (AG_H).

5.3 New Challenges

This lower average fire engines per task that lead to lower team performance was a result of inconsistency between the agents and the human. The agent team's perspective and the human's view differed, which resulted in the prioritization of different team tasks. Since both the human and the agent team were able to make decisions, this resulted in a lower average number of fire engines assigned to any one fire. This leads to the worse human-multiagent team performance. The agent team may have been able to detect this potential problem, but current adjustable autonomy strategies do not model inconsistency or its resolution. Abstractly, inconsistency is when a decision that is made by the agent or human is thought to be detrimental for the team by the other party. The aim of the adjustable autonomy policies is to have the best available decision maker make a given decision for the team. However, it may be the case that the decision, once made, is determined to be a poor team decision by the other teammate. Each teammate has its own view of the world. This inconsistent decision may occur because of the gap in information between the different teammates. While the distributed team is trying to handle the dynamic world in real-time, the team cannot transfer all information to all teammates. This is due to total information sharing being too time consuming and also to the limited attention span of the human.

The inconsistency can serve as an indicator that the human may be making bad decisions for the team. It cannot be known for sure, since there is uncertainty concerning whether the human or the agent team has the better decision. Although this did not occur for every possible human-multiagent team configuration, it is important to avoid this inconsistency if possible. However determining if the resolution is possible needs to take into account exactly how much time is available to execute a decision. Chapter 6 will address these new challenges that team-level adjustable autonomy brings.

Chapter 6

RIAACT: Resolving Inconsistencies with Adjustable Autonomy in Continuous Time

A key research area that arises is how to best enable the human-multiagent team to transfer control of team decisions between human and the agent team, namely team-level adjustable autonomy. Previously, in Chapter 5, I described experiments where a group of humans and a team of fire fighter agents collaborated in order to extinguish fires that are quickly spreading through a dense urban environment. Figure 5.3 presents experiments where 3 human subjects were coordinating with varying the size of the team and varying the team-level adjustable autonomy strategies (policies). What these experiments showed was that with 4 and 10 fire engines on the team, humans were able to provide input that improved the performance of the team. However, when the human was put on a team with 6 agents, the strategies that included both agents and humans ended up performing worse than if the agent team were to act without incorporating the human's input. Thus, always following a static policy of either always following (in the case of 6 agents) or always ignoring human advice (in the case of 4 or 10 agents) can lead to worse performance for the human-multiagent team.

A primary challenge is how to not blindly listen or blindly ignore the human's input while leveraging the capabilities of both the human and the agents on the team. In the above experiments, there was an inherent difference in the information available between the human and the agents. The human incident commander has a global perspective on the situation but may be unaware of local details such as the specific intensity of a particular fire, which the agents may be well aware of. This led to inconsistencies between the human and the agent decisions. The inconsistent human decisions were accepted blindly by the agents. Thus team resources were split that would have been more effective if leveraged in a coordinated manner. Although the agents are able to detect these inconsistencies, the solution is not to automatically reject inconsistent human input. The agents may be incorrect in their assessment due to uncertainty and lack of information.

Consequently, the problem becomes that of how to modify team-level adjustable autonomy to prevent this degradation in team performance from happening. However there are four key challenges that arise: First, these aforementioned inconsistencies can occur in many human-multiagent team domains. The challenge is how to have the human and agents resolve these possible inconsistencies as they arise. Second is the challenge of factoring a deadline into the team decisions, given that actions may take an uncertain amount of time. There is a deadline that, once reached, the state of the world changes and the opportunities that were available evolve or disappear. This may limit or even prevent the ability for the team to work on the time consuming task of resolving inconsistencies that exist between humans and agents. Also, affecting the reasoning about the deadline is the uncertainty in both the state of the world and in the duration of team tasks. Specifically, this new team task of resolving the inconsistency between a human and an agent takes an uncertain amount of time. Thus, whereas previously we had mentioned the planning (policy generation) problem for transferring control between agents and humans, we must now extend the planning to include reasoning about potential inconsistency resolution. Furthermore this planning must take into account deadlines and uncertainty of the duration of resolution. Third, the team must consider, in its planning, the ability for certain actions to be interrupted. Actions in many domains are often started and then interrupted if they do not finish on time or if another action starts to have a higher estimated reward. This allows for uncertain actions to be attempted for a limited time, while still being able to recover and make a decision before the deadline. Interruptible actions not only have direct implications for team-level adjustable autonomy, but for any general domain where an action can be interrupted. However, as discussed later, actions that are interruptible at many different points in time create novel challenges in planning. Fourth, in order to be solved optimally for a coordinating team, the team-level adjustable autonomy problem would be too computationally expensive to solve online. Furthermore, these four challenges must be addressed repeatedly for each team decision in a large-scale domain.

Previous work has not fully addressed these challenges in adjustable autonomy. Given the challenge of planning under uncertainty, previous work has often used decision theoretic techniques to solve the adjustable autonomy problem. In particular, these techniques include Markov Decision Problem (MDP) and Partially Observable MDP (POMDP) based approaches [Scerri et al., 2002; Varakantham et al., 2005; Marecki et al., 2006]. However, in order to have adjustable autonomy be flexible enough to change its strategy at different points in time, the state space is often discretized based on time intervals. This has two primary problems. First, the policy may only execute actions at a specific discrete interval. If there is a different action that results in higher expected reward, the policy must wait until the next discrete interval to change its prescribed action and take advantage of this opportunity. Thus if the intervals are longer, then there is a greater chance of loss of potential reward. Second, in order to have a policy that can choose the best action at many different points in time, there must be a large amount of states to be reasoned over. This is due to the creation of a large amount of short discrete intervals so that a policy may react in this short amount of time and a new state is created at each interval. For example, in the Electric Elves domain, there were thousands of states in order to reason about the scheduling of meetings [Scerri et al., 2002]. This same state space explosion is encountered once again when trying to reason about the interrupting of an action at many different points in time. Such state space explosion would create significant inefficiency. Another difference is that these previous approaches assumed that once a decision is made, it should be executed. This does not allow the team to resolve potential inconsistencies when they occur. Finally, in previous work, the problem was confined to a single agent and a single human. Consequently, the challenge of adjustable autonomy involving a team of agents did not arise.

I have developed a new approach that addresses these challenges called RIAACT (Resolving Inconsistencies with Adjustable Autonomy in Continuous Time). RIAACT provides four contributions to the field of human-multiagent teams (i) modeling of inconsistencies and the modeling of their resolution, (ii) developing policies (strategies) that are continuous time dependent and factor in uncertain action durations, (iii) extending current continuous time planning models to allow for the interrupting of actions, and (iv) decomposing the team-level adjustable autonomy problem and leveraging the use of a hybrid approach to address coordination.

Firstly, RIAACT extends existing adjustable autonomy policies (strategies introduced in Chapter 5) beyond the initial team member decision and overcome inconsistencies between the human and the agents by leveraging resolution policies. This allows the agents to avoid a potentially poor input from the human. Not only does this allow for the agents and human to model explicitly the duration of resolving their inconsistency, but through the modeling, they can determine for how long to attempt to do so or even whether it is even worth it to attempt. The aim of this thesis is an overarching framework that must stand above any resolution method that is chosen. I do not model the resolution process or determine what method is optimal for reaching this resolution, but instead I focus on planning given the uncertain duration of the resolution.

Secondly, this approach leverages recent work in Time-Dependent Markov Decision Problems (TMDPs) [Li and Littman, 2005; Marecki et al., 2007; Mausam et al., 2005]. Thus, by exploiting the latest advances in TMDPs, I have illustrated the feasibility of applying this TMDP methodology to the adjustable autonomy problem. Also, I have shown concretely how to model the team-level adjustable autonomy problem within the TMDP framework. This results in both a feasible and an efficient way to reason about continuous time and uncertainty over time. This new TMDP formulation allows for a continuous time policy that allows for actions to be prescribed at arbitrary points in time, without the state space explosion that results from solving with fixed discrete intervals. In addition, uncertain action durations (especially the *resolve* action) are modeled by continuous distributions. In order to solve TMDPs, I have leveraged the fastest current TMDP solution technique [Marecki et al., 2007], which utilizes phase-type approximations of the action duration distribution in order to use numerical analysis methods for speed ups. This results in a fine grained model representation and policy, which enables the team to develop more accurate policies that are better designed to handle deadlines.

Thirdly, a major contribution of this work is to introduce a new model that allows for actions to be interruptible, referred to as an Interruptible TMDP (ITMDP). Prior work that used discretized states had the ability to interrupt an action at one of the discrete intervals, but as I discussed earlier, it paid the price of a massive state space explosion. TMDPs do not have this state space explosion, yet if an action is able to be interrupted at any point in time, then this can lead to the same state space explosion that TMDP was designed to avoid. In particular, current TMDP techniques allow for the starting of an action at any point in continuous time, however, if started, that action must be pursued until completion or else a new state must be created for each interval in time at which the action may be interrupted. However, this thesis aims to have actions able to be interrupted at any point in continuous time. This allows for a more accurate reward estimation and representation of actions as they are in the real world. In an uncertain world with deadlines, actions are often attempted and then abandoned if they do not complete with enough time to finish before the deadline. For example, in adjustable autonomy, a human may be given the autonomy over a decision and starts to make that decision but takes too long. Then the agent may wish to interrupt that human decision action and make the decision itself. RIAACT introduces a model in which an interruptible action may be stopped and return to the previous state, where the interruption follows a given duration distribution. This results not only in more accurate ITMDP policies for the given states, but also an additional time-dependent policy for each interruptible action. This new policy prescribes whether to continue or interrupt each action while it has not completed after a certain amount of elapsed time. The real difficulty in implementing interruptible actions is to have the interruption be able to occur at any point in continuous time without an explosion in the state space. At any point in time an interrupt would be possible, a new state would be needed. RIAACT avoids this state space explosion in ITMDPs by leveraging properties of existing TMDP solution techniques, which reason over a finite set of states that represent a continuous state space.

Fourthly, current techniques are not capable of implementing the complete solution to the team level adjustable autonomy problem in real world scenarios for use during execution. In order to allow the team as a whole to coordinate when giving up or taking of autonomy over decisions, the team would need to use a distributed MDP, which is prohibitively expensive and has been shown to be NEXP-complete in the general case [Bernstein et al., 2000]. RIAACT instead decomposes the team-level adjustable autonomy problem into three subproblems and solves them each with different, but interacting, techniques that are combined to create a hybrid approach [Tambe et al., 2005; Nair and Tambe, 2005; Schut et al., 2001; Scerri et al., 2002; Boutilier et al., 2000]. This approach is called a hybrid because it combines decision theoretic reasoning (ITMDP) with distributed constraint reasoning and BDI logic-based teamwork approaches that existed in the Machinetta architecture (as explained in Section 4.2.2). Developing the space of policies is accomplished by having an agent solving an abstracted single agent ITMDP on behalf of the team, using the continuous time methods described previously. The actions within the ITMDP are, in fact, team actions and are executed using these hybrids. For example, if an inconsistency is detected, then a ITMDP policy determines if a resolve action is beneficial and if so, how long before the resolution should be interrupted. If the resolve action is attempted, then the team-communication regarding this resolution is implemented using team logic based reasoning (see Section 4.2.2). The team then knows about the resolve team action and knows when it should be interrupted. This hybrid approach allows for the complex team-level adjustable autonomy to occur in real time due while aiming to optimize coordinated team performance.

In the rest of this chapter, firstly, I will present an illustrative scenario in order to demonstrate the use of the model. I have leveraged the fastest continuous time solver with quality guarantees for TMDPs (CPH) by approximating duration distributions with Phase-type distributions made up of exponential distributions [Marecki et al., 2007]. This has been shown to be solved in a very short amount of time. Secondly, I have applied RIAACT to the disaster response domain using the DEFACTO incident commander training system. In DEFACTO these new RIAACT adjustable autonomy policies have been shown to improve performance of the team.

6.1 Disaster Rescue Scenario

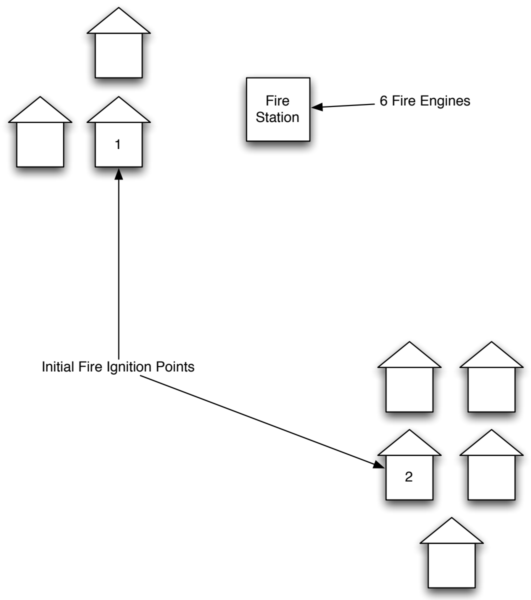


Figure 6.1: Example scenario map with 6 fire engines (agents) to be dispatched to initial fires located in buildings 1 and 2. Buildings 1 and 2 are seen here beside adjacent buildings that the fire could potentially spread to.

Here, I will instantiate a specific disaster response scenario that fits the general problem definition described earlier in Section 2.1. This will serve as an illustrative scenario for motivation and will be used again in the experimental results later seen in Section 6.3. This specific scenario has been chosen because it highlights the situation that led to the dip in performance that was observed in the experiments of Section 5.2. From these experiments, it became apparent that disagreement between humans and agents on which tasks to prioritize could provide a hint toward a dip in team performance.

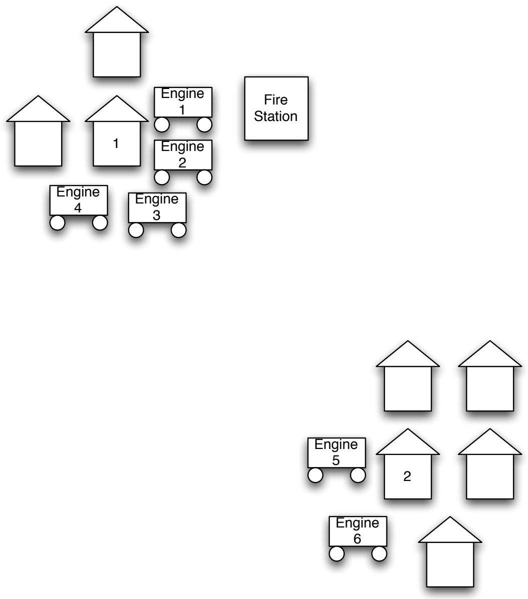


Figure 6.2: Initial allocation given by the incident commander.

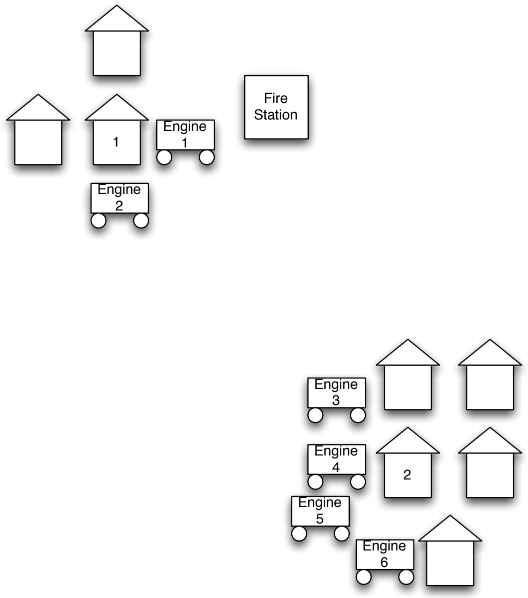


Figure 6.3: Alternate allocation determined by the agent team.

The scenario includes 6 fire engines that must address a large scale disaster in which 2 high-rise buildings in an urban area have caught on fire (see Figure 6.1). These fires are engulfing the buildings quickly and each have the chance of spreading to adjacent buildings. A decision must be made very quickly about how the team is to divide their limited resources (fire engines) among the fires.

There are many possible ways to allocate these six fire engine agents to the two fires. In this scenario, a human has decided to send 4 agents to the fire in building 1, and let the remaining 2 fire engines go and fight the fire in building 2 (see Figure 6.2). This seems logical to the human because building 1 is closer to the fire station and can be put out sooner so that resources may later be concentrated on building 2. However, once the agents receive this order from the incident commander, they determine that team's performance will degrade. They calculate this based on building size and fire intensity. Consequently the agent team believes that fire in building 2 is much more likely to spread than the fire in building 1. The agents believe that more priority should be given to building 2 (see Figure 6.3).

I will now explain how the challenges mentioned earlier arise in this scenario. First, there is inconsistency. In particular, Engines 3 and 4 believe that the decision made by the human is inconsistent with what they believe the team should be doing. Other engines may be aware of the inconsistency, but the resolution is left to the engines that are the objects of these inconsistent decisions because they are most aware of their local state. Engines 3 and 4 cannot just reject the human's orders, because of their incomplete world state information and their uncertainty of how the world will progress. Secondly, there is a need to reason over continuous time with uncertain action durations. These fire engines would like to resolve this inconsistency, however, the resolution may take a long time and the team is acting in a disaster response situation where time is short. There is a deadline that occurs when the fire begins to spread to adjacent buildings. Thirdly, interruptions are desired for some actions. It would be beneficial to reason about attempting an inconsistency resolution, but interrupt that action, if the deadline approaches to near. Fourthly, a hybrid approach is needed. Having both engine 3 and engine 4 attempt to resolve the same inconsistency is redundant and results in wasted time; but coordinated distributed planning under uncertainty is extremely computationally expensive. The resolving of the inconsistency may find that the agents were correct or that the agents were incorrect and should have listened to the human. In addition, trying to factor in all of the possible outcomes and modeling it as a distributed MDP would take too long to compute online, even in this small example. Some type of hybrid approach is necessary to reason about which actions are optimal in continuous time and execute these team actions in real-time.

6.2 RIAACT

RIAACT is a system that has been designed to address the challenges that arise when addressing such a scenario as in the previous section. RIAACT extends and improves the previous implementation of team-level adjustable autonomy policies in Chapter 5. As mentioned earlier, the team-level adjustable autonomy problem cannot be mapped over to a distributed MDP and solved optimally because it would be computationally infeasible. In particular, distributed MDPs have been shown to be doubly exponential (NEXP-complete) in the general case [Bernstein et al., 2000]. Consequently, RIAACT is implemented as a hybrid system [Tambe et al., 2005; Nair and Tambe, 2005; Schut et al., 2001; Scerri et al., 2002; Boutilier et al., 2000] where the team-level adjustable autonomy problem is decomposed into three subproblems and solves them each with separate techniques. Within this hybrid, developing the space of policies is accomplished by solving an abstracted single agent ITMDP on behalf of the team. In the rest of this section, I first describe the TMDP-based approach for adjustable autonomy. Next, I describe how the model is extended to allow for some actions to be interruptible and present the ITMDP model. Finally, I return to the hybrid and show how the communication and coordination is executed using team logic based reasoning as described in Section 4.2.2. In addition, the agent team when trying to allocate a role among the team will employ distributed constraint reasoning as also described in Section 4.2.2. This hybrid approach allows for the complex team-level adjustable autonomy to occur in real time due while aiming to optimize coordinated team performance.

6.2.1 ITMDP Model for Adjustable Autonomy

The RIAACT Model has the overall goal of improving the team performance by allowing either the human or agent to make team-level decisions, depending on which is best, given the circumstances. In the context of this work, team level decisions can range anywhere from the decision to make an action to the decision to allocate a role to another team member or even to execute a team plan. From this point forward, I will refer to all of these possible team-level decisions as simply "decisions." The focus of RIAACT is a more overarching framework that must stand above such a resolutions in a time-constrained (deadline) environment where actions, including resolve actions, take an uncertain amount of time to execute. The work does not focus on the details of a resolution itself, but should be viewed as planning above the level of any specific resolution approach that can be adopted. The planner provides a policy for when an inconsistency is detected then under what conditions to engage in resolution, how long to do it, when is it appropriate to interrupt the resolution and just act, all while in a distributed team setting.

Recent work in adjustable autonomy transfers the control over a decision back and forth between agent and human [Goodrich et al., 2007; Reitsema et al., 2005; Owens et al., 2006;

Sellner et al., 2006]. However in that work, once the decision was made by any party, that decision was final. That decision would then be executed to its completion. This was also the case with the team-level adjustable autonomy policies that were introduced earlier in Chapter 5. Having these decisions be immediately executed, caused the team to have a dip in performance seen in Figure 5.3.

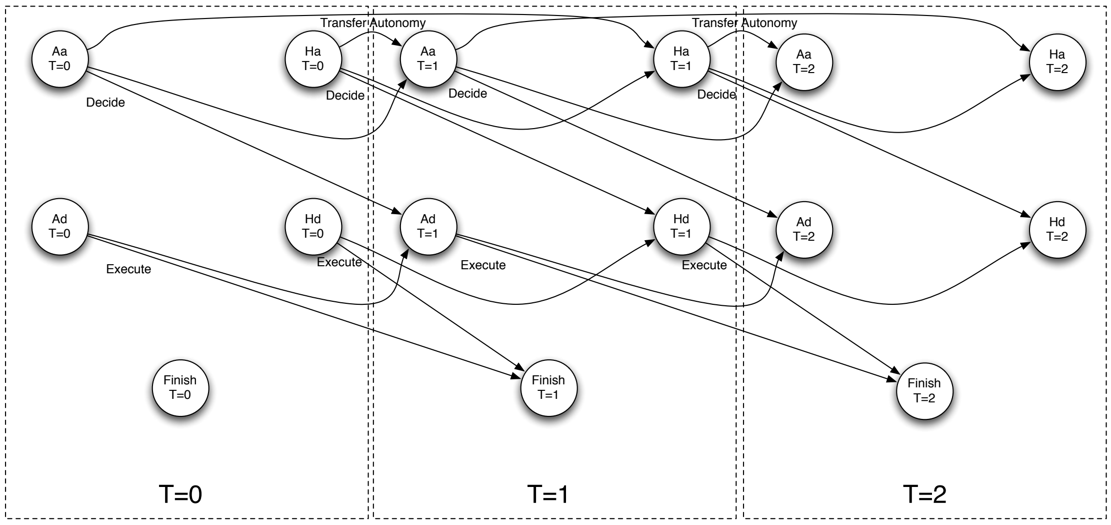


Figure 6.4: Previous model of Adjustable Autonomy for 3 time steps (T=0,1,2): Agent has autonomy Aa, Human has autonomy Ha, Agent decision Ad, Human decision Hd, and task finished $Finish$. Three actions are shown: $Transfer$ Autonomy, $Decide$, and $Execute$.

Figure 6.4 shows the abstracted states of the MDP for solving the adjustable autonomy problem as it was implemented in previous work [Scerri et al., 2002] and the prior team-level policies (strategies) that I implemented in Chapter 5. Each dashed-line box represents a discretized time slice at T=0,1 and 2 respectively. As time is broken into more and more intervals, more states and transitions will be added. This figure represents a single decision and execution toward a $finish$ state, where a reward is earned. Each circle represents a state that the decision can be in: Agent has autonomy Aa, Human has autonomy Ha, Agent decision Ad, Human decision Hd or task finished $Finish$ (for more explanation of the states, refer to Section 6.2.2.1). Notice that these states abstract out other features of the world for simplicity of presentation (for example in a personal assistant meeting scheduling domain, the location of a meeting, number of attendees etc. would be a part of the state). Also, Figure 6.4 contains the human action *decision*, since this policy must be thought of from the perspective of the agent team, that action should also be considered the agent action $wait\ for\ human\ decision$.

Each arrow in the diagram represents a possible action (state transition) and is labeled with the names of their respective actions: $Transfer$ Autonomy, *Decide*, and $Execute$. In this previous model, actions take a fixed amount of time that modeled their average duration. However, since actions can take longer than a single time step interval, some action transitions result in the same state at the subsequent interval. In the Electric-Elves project and other applications, variable decision granularity and other techniques were used to attack such a state space explosion, but they had to fundamentally grapple with this explosion [Scerri et al., 2002].

Previous work has indeed used many more states than those shown in Figure 6.5 to construct MDPs for adjustable autonomy. For instance, for each time slice t only one Ad_t state is shown, however, there can be arbitrarily many different states per time slice (Ad_t^1, Ad_t^2... Ad_t^n) each with their own reward for execution (where each state differs in other features of the world as mentioned above). Figure 6.4 represents the policy space over 3 time intervals. One of RIAACT's contributions is its abstraction of previous work into general categories of adjustable autonomy based states. This allows for MDP to retain a manageable number of states while extending the model to reason about inconsistencies. Further explanation of this abstraction appears later in Section 6.2.2.1.

Figure 6.5 represents how the previous work shown in Figure 6.4 can be modeled as a Time-dependent Markov Decision Problem (TMDP). In this new figure, instead of the number of states

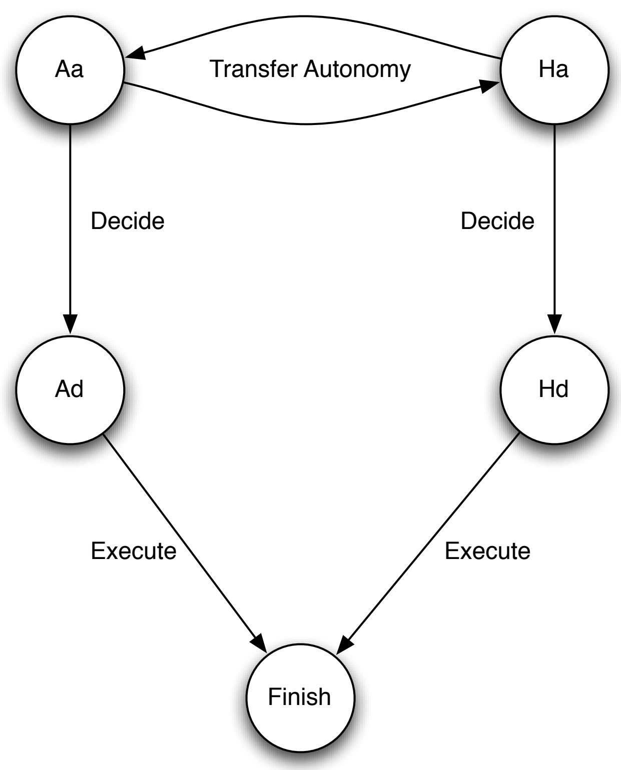


Figure 6.5: Previous model of Adjustable Autonomy updated as a TMDP.

increasing to model the same states over time, single states now have policies that are functions over time (see Section 2.4.2). In addition, each arrow now represents not a single duration, but an entire action duration distribution that can be any arbitrary distribution. While I represent *Ad* and *Hd* as individual states, again there may be multiple such states (Ad^1, Ad^2 ... Ad^n), but it is important to note that there are no extra states to represent each time slice as above.

RIAACT wishes to address the inconsistency mentioned in Section 2.1.1. In order to do this, I assume that there is a certain likelihood, for a given domain, that the local perspective of the agent will coincide with the global perspective of the human. This is mapped as discrete probability of consistency $P(c, x)$ and determines the transition probability into the consistent and inconsistent states when teammate x is making a decision. The probability of consistency, $P(c, x)$, may depend on whether the decision maker is human $P(c, H)$ or an agent $P(c, A)$.

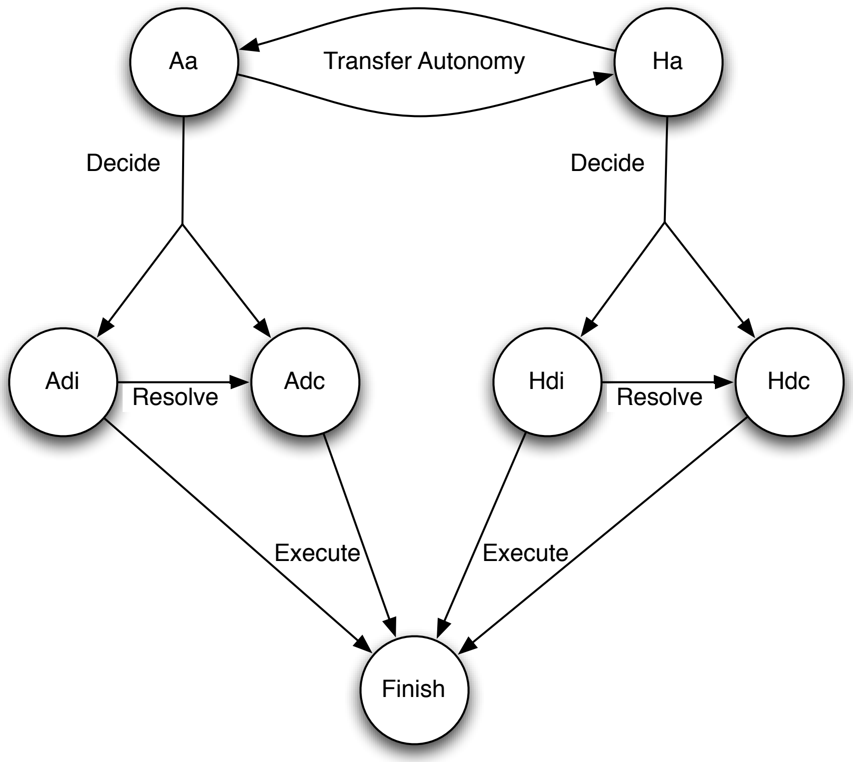


Figure 6.6: Adjustable Autonomy model that has been augmented to reason about inconsistency (*Adi*,*Adc*,*Hdi*,*Hdc*) and has also added the *resolve* action.

Figure 6.6 is a diagram showing the addition of reasoning about inconsistency. Now the state Human Decision (*Hd*) has been replaced with Human decision consistent (*Hdc*) and Human decision inconsistent (*Hdi*). There is a transition probability associated with consistency of a human decision, $P(c, H)$ and thus inconsistency as well, $1 - P(c, H)$. The same changes also apply to the agent *decision* action. In addition, there is an action (*resolve*) that transitions from an inconsistent state to a consistent state. This action is also modeled as having a certain duration distribution. As mentioned earlier, I will not focus on the details of the *resolve* action itself, but rather focus on the overarching planning problem of determining whether to and for how long to *resolve*.

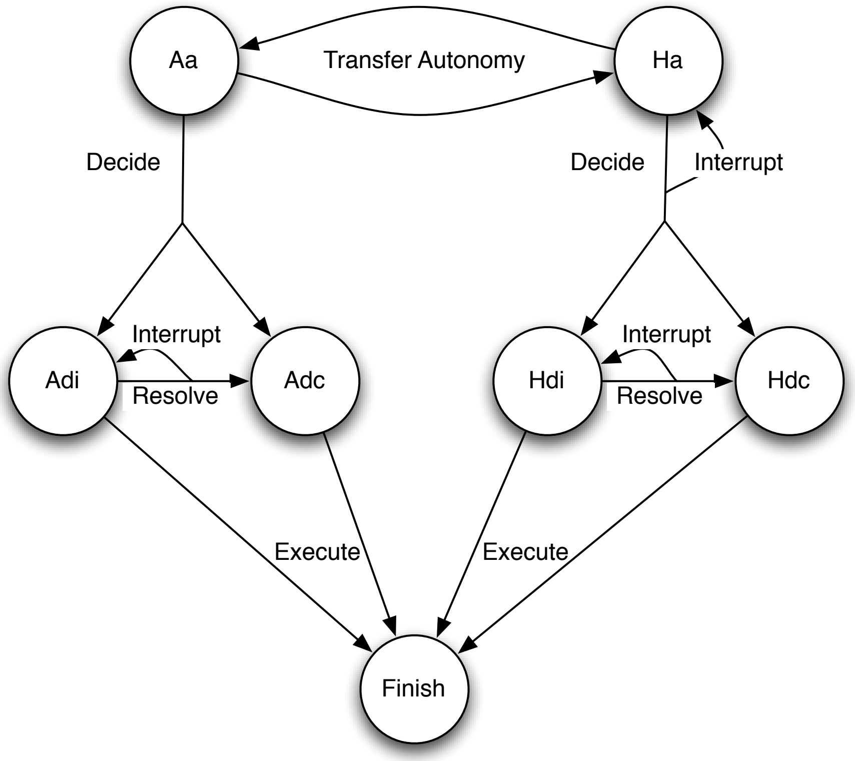


Figure 6.7: RIAACT ITMDP Model: Added the *interrupt* action.

The RIAACT ITMDP model diagram in Figure 6.7 adds the ability to interrupt the human *decision* action and both *resolve* actions. The *interrupt* action is drawn again with an arrow and is also modeled as having a certain action duration distribution. The ITMDP model will be explained in detail later in Section 6.2.1.1. The key feature of the ITMDP model is that an action can be interruptible at any point in continuous time, as shown in Figure 6.7 via the arrows labeled *Interrupt*.

The general model shown in Figure 6.7 improves on the previous model of adjustable autonomy by (Figure 6.4) in three important aspects: (i) the reasoning about and modeling of the resolving of inconsistencies, (ii) action durations (the arrows in the diagram) are now modeled by continuous distributions, and (iii) allowing for the interruption of actions in continuous time. Keep in mind that this model represents possible actions and states for a single team decision. One of these would be instantiated for each team decision.

6.2.1.1 ITMDP Model

In order to explain the ITMDP, I will first present the components that are leveraged from the existing TMDP model that it builds upon. The TMDP[1] model consists of a tuple of the elements $\langle S, A, P, D, R \rangle$ that are defined as follows:

- S - Finite set of discrete states.
- A - Finite set of discrete actions.
- P - Discrete transition table of probabilities for each tuple $\langle s, a, s' \rangle$ where executing action $a \in A$ transitions from the starting state $s \in S$ to ending state $s' \in S$
- D - Set of transition duration functions. For all $p \in P$ there exists a $d(t)$ where $d(t)$ is the duration probability distribution function for $p = \langle s, a, s' \rangle$. If action $a \in A$ is executed from state $s \in S$ at time t, then new state $s' \in S$ will be reached at time t' with probability $d(t' - t)$.
- R - Reward for executing an action $a \in A$ from discrete state $s \in S$ that results in a transition to a new discrete state $s' \in S$ at time t, shown as $\langle s, a, s', t \rangle$.

[1]It should be noted that the original TMDP formulation in [Boyan and Littman, 2000] allows for both absolute and relative formulations of both duration and reward functions. However the model shown here assumes relative durations and absolute reward functions.

The policy for a TMDP aims to find the expected utility for the optimal action $a^* \in A$ for each state $s \in S$ and absolute time t until deadline, given by $\langle s, t \rangle$. This creates a real-time value function that maps $\langle s, t \rangle$ to an expected utility[2] shown as $U(s, a^*, t)$ where a^* is the action with the maximum total expected utility when taken from discrete state s at time t maximized over all possible actions in set A.

The new ITMDP model consists of a larger tuple of the elements $\langle S, A, P, D, R, S_i, A_i, P_i, D_i, R_i \rangle$ with a subsection of elements that are defined as stated above. In addition new elements are defined as follows:

- S_i - Finite set of discrete transitory states. A transitory state $s_i \in S_i$ is derived for each possible tuple of state-action pairs $\langle s, a \rangle$ where $s \in S$ is the originating state and $a \in A$ is the action taken from that state.

- A_i - Finite set of interrupt actions. Interrupt actions transition from any transitory state $s_i \in S_i$ to that transitory state's originating state $s \in S$.

- P_i - Discrete interrupt transition table of probabilities (p_i) for each tuple $\langle s_i, a_i, s \rangle$ where executing interrupt action $a_i \in A_i$ transitions from the transitory state $s_i \in S_i$ to originating state $s \in S$. For an interruptible action $p_i = 1$, and conversely for an non-interruptible action $p_i = 0$.

- D_i - Set of interrupt transition duration functions. For all $p_i \in P_i$ where $p_i \neq 0$ there exists a $d_i(t)$ where $d_i(t)$ is the duration probability distribution function for $p_i = \langle s_i, a_i, s \rangle$. If interrupt action $a_i \in A_i$ is executed from transitory state $s_i \in S_i$ at time t, then originating state $s \in S$ will be reached at time t' with probability $d_i(t' - t)$.

[2]How to derive $U(s, a^*, t)$ is not shown here, but refer to [Boyan and Littman, 2000] to see more details.

- R_i - Reward for executing an interrupt action $a_i \in A_i$ from transitory state $s_i \in S_i$ that results in a transition to the originating discrete state $s \in S$ at time t, shown as $\langle s, a, s', t \rangle$. In addition the "continue" or *null* action can be represented by $< s_i, null, s', t >$ where $s' \in S$ represents the end state for the transitory state s_i.

The policy for an ITMDP results not only in the same mapping of $\langle s, t \rangle$ to a $U(s, a^*, t)$ for all $s \in S$ as shown above, but also with a mapping of $\langle s_i, t \rangle$ to a $U(s_i, a_i^*, t)$ for all $s_i \in S_i$. The aim of the ITMDP interrupt policy is to find the maximum expected utility between the interrupt action $a_i \in A_i$ versus continuing execution (*null*) for each state $s_i \in S_i$ and absolute time t until deadline, given by $\langle s_i, t \rangle$. This creates a real-time value function that maps $\langle s_i, t \rangle$ to a $U(s_i, a_i^*, t)$ where a_i^* is either the interrupt action a_i or the continuing execution *null*, whichever yields the maximum total expected utility when taken from transitory state s_i at time t.

6.2.1.2 States

Each circle in Figure 6.7 represents the state of the decision, with the additional state distinction of whether the current decision is consistent or not with the view of the other teammate. This results in the new states: Agent decision inconsistent *Adi*, Agent decision consistent *Adc*, Human decision inconsistent *Hdi* and Human decision consistent *Hdc*. Similar to the previous model (Figure 6.5), there are also the states where: Agent has autonomy *Aa*, Human has autonomy *Ha*, and the task finished *Finish* (for more explanation of the states, refer to Section 6.2.2.1). I assume that each state is fully observable.

6.2.1.3 Actions

Arrows again represent the actions that enable state transitions. In order to allow the resolving on inconsistencies, the new action "resolve" has been added. However, now in RIAACT, much like the real world, actions do not take a fixed amount of time. Instead, each arrow also has a corresponding function which maps time to probability of completion at that point in time. For further explanation of action duration distributions, please refer to Section 2.4.2.

There are four available actions: $Transfer$, $Decide$, $Resolve$, $Execute$ (as seen in Figure 6.7). $Transfer$ results in a shift of autonomy between a human and an agent. $Decide$ allows for a decision to be made and results in a consistent state with probability $P(c, A)$ for an agent or $P(c, H)$ for a human. Conversely an inconsistent state is reached with probability $1-P(c, A)$ by an agent and $1 - P(c, H)$ by a human. $Resolve$ is an action that attempts to resolve an inconsistency Adi or Hdi and leads to a consistent state Adc or Hdc, which yields higher rewards. To $Execute$ a particular decision results in the implementation of that decision toward the $finish$ state.

6.2.1.4 Rewards

The reward R for a state is only received if that state is reached before the deadline. In previous adjustable autonomy work, then the decision Ad or Hd would have been made by either party and assumed to have some average quality or reward $R(Ad)$ or $R(Hd)$. In our effort to try and model the diverse perspectives that the agents and humans can have, I extended the model to categorize the decision as either consistent Adc or Hdc or inconsistent Adi or Hdi. It is the case that there can be a wide variety of both consistent and inconsistent team decisions and the model allows for that. However, for illustration purposes I have assumed that all possible consistent decisions (for this particular team decision) made by the agent have an average reward of $R(Adc)$ and that all inconsistent decisions (for this particular team decision) on average provide a reward of $R(Adi)$ to the team.

6.2.1.5 Policy

The policy prescribes the best action for each state from now until the time of deadline td. The time of deadline for a particular role is the point at which that role becomes unachievable or irrelevant. Thus, reasoning about the role further would not provide any benefit.

For every team decision, there is an adjustable autonomy policy between the human and the agent team. The goal of this policy is to determine the critical point at which a decision should be delegated ($transfer$), when a decision should be made (*decide*), and when to undergo inconsistent decision resolution ($resolve$). Actions involving the human take more uncertain and varying amounts of time. Thus time becomes crucial for both decision for transfer of control but also for the duration and even attempt for the resolution of the inconsistencies. The RIAACT policies that I have developed address both of these.

6.2.2 Abstraction

To address the challenge of scale while trying to maintain an online solution, I have leveraged state abstractions. Initially, I have abstracted the world state and team state into general states that pertain to team decisions. More importantly, RIAACT develops a method for abstracting the states into categories that can be reasoned about online. The categories are divided based on salient characteristics that are pertinent to team-level adjustable autonomy. In addition, these categories can be broken into sub-categories to more accurately model the world and, given the extra time has the potential to give more accurate solutions for RIAACT.

6.2.2.1 Adjustable Autonomy State Abstraction

In order to address the size of the state space (even without time factored in), RIAACT models allows for the states to be abstracted into separate state categories. Each of the categories are divided based on characteristics that pertain to the general team-level adjustable autonomy problem. These categories can be reasoned about (and averaged) as a single state, or separated into sub-categories if more granularity is needed. For instance, the state below of a human decision that is inconsistent, *Hdi*, can be expanded out into several possible inconsistencies that may arise, each with their own average reward, or just be modeled as a single state with a single average reward. Below are the state categories that a single team decision could be in during a RIAACT policy:

- *Agent has autonomy (Aa)* - The agent team has autonomy over the decision. At this point, the agent team can either transfer control to the human or try to make a decision.
- *Human has autonomy (Ha)* - Human has the autonomy over the decision. At this point, the human can either transfer control to an agent or make a decision.
- *Agent decision inconsistent (Adi)* - This state represents any state in which the agent has made a decision and the human disagrees with that decision.
- *Agent decision consistent (Adc)* - This state represents any state in which the agent has made a decision and the human agrees with that decision.
- *Human decision inconsistent (Hdi)* - This state represents any state in which the human has made a decision and the agent believes that the decision will result in substantial decrease

in average reward for the team. The agent does this by factoring in its detailed local information into determining the results of implementing the human decision. There may be many possible inconsistent states, but for faster calculation, I have abstracted the states out as an average inconsistent state.

- *Human decision consistent (Hdc)* - This state represents any state in which the human has made a decision and the agent believes that the decision will either increase the reward for the team or does not have enough information to raise an inconsistency about the decision. There may be many possible consistent states, but for faster calculation, I have abstracted the states out as an average consistent state.

- *Task finished (Finish)* - This represents the state where the task has been completed and a reward has been earned. As stated before, the reward ($R(s,a,s')$) depends on the starting state s, the action a and the ending state s'. Thus, the reward depends on which decision (s) was executed (a) to get to the $Finish$ state (s'). There is a separate average reward earned for executing each decision type (Adi, Adc, Hdi, Hdc).

These state categories are used to construct the (I)TMDPs and are shown in Figures 6.4, 6.5, 6.6, and 6.7. By reducing the state space to this manageable state space, the ITMDP is capable of executing online. In addition the RIAACT model decomposes the distributed ITMDP to a single ITMDP for each role to be executed by the agent that has the autonomy over that role. Also, this abstract representation that is given by RIAACT reasons about the benefit of resolution policies for each role, thereby enhancing the team-level adjustable autonomy.

6.2.3 Inconsistency Resolution

The advantage of adding the ability to resolve is that the decision will not be implemented blindly. Depending on the potential loss of reward and the time available, the decision can be resolved by bringing new information to light or explaining the problem. This is the new resolve action that has been introduced in this thesis. Depending on the factors involved, the resolve action can be attempted for a limited time or even not attempted at all and a total rejection of the decision. An important aspect to note is that the resolve action is a team action, and the agent team must coordinate in order to prevent unnecessary or overwhelming resolves from occurring.

There are many different methods of interaction between a human and an agent that could be implemented in order to resolve. In addition, there are many different orderings of the information that is communicated in an effort to resolve an inconsistency. RIAACT does not focus on these issues, but rather focuses on the modeling of the action duration.

Inconsistent State - As mentioned earlier in Section 2.1.1, inconsistency occurs as a result of the information gap between the agent team and the human. For planning, the likelihood that this inconsistency will occur is modeled in RIAACT by $1 - P(c, H)$ for the human and $1 - P(c, A)$ for an agent. Inconsistency is detected during execution by evaluating a decision based on the abstract world states of the agent team's capability and the decision's priority.

Resolve Action - The resolve action is a team action that transitions from an inconsistent state (Adi or Hdi) to a consistent one (Adc or Hdc). The advantage of being in a consistent state is that the expected utility of executing a consistent decision is often higher than that of executing an inconsistent one. In order to be able to react to the approaching deadline, the resolve action is assumed to be interruptible and implemented as such in the ITMDP model.

Transition - RIAACT assumes that all inconsistencies can be resolved, yet it may take varying amounts of time to do so. This results from the abilities of the ITMDP model to have an action duration follow a distribution. If an inconsistency is especially difficult to resolve, this is captured in the action duration distribution being very long.

Reward - In addition, RIAACT reasons about the average reward for successfully completing (arriving at $Finish$) the *execute* action from an inconsistent state versus a consistent one. If the average rewards of these do not differ substantially, then it might not be beneficial to resolve an inconsistency when it arises.

One may propose that, instead of inconsistent decisions, the human-multiagent team should model them as suboptimal decisions. It is correct that the overall aim is to have the team perform better, however, because of differing perspectives, the human and multiagent team might have different ideas of which decisions are suboptimal (see Section 2.1.1). In addition, no single team member knows the complete state of the world and thus there it is difficult to determine with certainty if a decision is really suboptimal for the team. Furthermore, often the primary determinant of team performance in our domains of interest is consistency. There are real-world assumptions that we make about our domains that cause consistency to be so important: decisions must be made under critical time pressure, while uncertainty and where there is interdependence between decisions.

6.2.4 Time Dependent Models

In RIAACT, much like the real world, actions do not take a fixed amount of time. Instead, each action also has a corresponding function which maps time to probability of completion at that point in time. For further explanation, please see Section 2.4.2.

In order to address the challenges brought about by dealing with time, I have modeled these new adjustable autonomy policies using TMDPs (Time-dependent Markov Decision Problems). This allows for an improvement over previous approaches, which used a discretized time model which is either inaccurate or computationally too expensive to solve. The policies introduced in previous work have significant computational complexity from the immense amount of states created when factoring in time [Scerri et al., 2002; Varakantham et al., 2005]. Consequently, they are inappropriate in large-scale complex domains requiring real-time decisions. However, RIAACT uses continuous time to achieve precise timings for both the transfer of control and the ending of inconsistency resolution, thus resulting in higher quality solutions and better team performance.

The model that RIAACT utilizes for continuous time planning with MDPs was developed in [Boyan and Littman, 2000] and was later extended in [Li and Littman, 2005]. A more detailed explanation of the TMDP model can be found in Section 2.4.2.

6.2.5 ITMDP: Interruptible TMDPs

In this section, I will explain the details and inner-workings of the ITMDP (Interruptible Time-Dependent Markov Decision Problem) that I have developed. In addition, I have addressed some of the challenges that arise when implementing an ITMDP solver. This section will be divided into four parts: (i) Modeling an ITMDP, (ii) Efficient planning for ITMDPs by using CPH, (iii) Calculating belief distribution, and (iv) Converting the model.

6.2.5.1 Modeling an ITMDP

Although the model of a Time-dependent Markov Decision Problem (TMDP) has the advantage of outputting a continuous time-dependent policy for each state, there is a drawback to this model: actions are atomic and uninterruptible. In many domains, action durations may be uncertain. Consequently, the optimal policy may be to attempt an action for a limited time and then interrupt that particular action if the deadline is approaching and the action has not completed yet.

RIAACT extends the model to allow for actions to be interrupted at any point in time. The resulting model of an Interruptible Time-dependent Markov Decision Problem (ITMDP) can more accurately model the expected utility of taking an action by factoring if it can be interrupted at a later time. As with actions in the TMDP model, the interrupting of an action takes an uncertain amount of time that follows a given duration distribution. A potential penalty for interrupting an action is captured in the time that it takes, which may result in the finish state being unreachable before the deadline. In addition, a reward/cost can be associated with the interrupt action itself.

A formal description of an ITMDP can be seen in Section 6.2.1.1. This ITMDP model has much broader applications than just adjustable autonomy (and RIAACT). There are often planning problems in real-world domains where actions are interruptible and should be modeled as such. For example if there is a group of satellites that are coordinating to observe as many phenomena as possible before the opportunity passes by. There might be a situation where there is a very small chance that a beautiful meteor shower may be visible from the satellites or another option is that they take a daily picture of a planet. Now the reward for capturing the meteor shower might be very high, however the time that it takes for them to appear might last longer than time available until deadline. The optimal policy for a standard policy would suggest that the satellite should just focus on the planet, however if it reasoned about interruptible actions, then the meteor shower could be attempted for a limited time and if that did not succeed, then the satellite could reposition itself to view the planet.

In RIAACT, examples of actions that would benefit from being interrupted are the human *decide* action and the *resolve* that tries to eliminate an inconsistency. If the human has been given autonomy, but the human is taking too long to make a decision (*decide*), then the agent may want to interrupt that action and have autonomy transferred back to the agent so that it can quickly make a decision. Also, if the resolving of an inconsistency is taking too long, the agent may wish to end the *resolve* action and pursue the inconsistent human decision *Hdi* so that some reward may be attained before the deadline.

Existing TMDP models allow for uncertain action durations, but once an action has been taken, it must be maintained until that action has finished. Consequently, the policy that is produced guides only at what point in time to start a particular action. For example, suppose that the TMDP policy shows that if an inconsistent human decision *Hdi* is detected at a point in time before 2 minutes to deadline, then the optimal action is to *resolve* in an effort to reach *Hdc*, otherwise if less than 2 minutes are left until deadline at that point in time when inconsistency is detected, the optimal action from *Hdi* should be to *execute*. When an inconsistency is detected, there are 2 important questions that must be answered: (i) whether or not a resolve should be attempted and (ii) if it should be attempted, for how long. The policy currently only answers the first question, yet the challenge now is to understand when to stop resolving. The goal is to have a policy that, once an action is taken, recommends over continuous elapsed time either to continue with the current action or to abort (interrupt).

6.2.5.2 Efficient planning for ITMDPs by using CPH

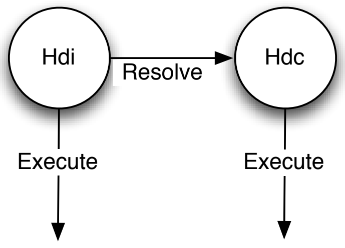
I have chosen to utilize the Continuous Phase (CPH) solver [Marecki et al., 2007] which has been shown to be the fastest of the time dependent MDP solvers. Background on the CPH solver has been provided in Section 2.4.3.

A few modifications are performed in order to allow for efficient execution. Figure 6.8 shows how the resolve action from the ITMDP is modified at first for the CPH solver in subfigure (b) and then for a policy that allows for the interrupting of actions in subfigure (c). Figure 6.8-a shows a subsection of the larger ITMDP seen in Figure 6.9. This highlights the *resolve* action which attempts to turn the inconsistent decision Hdi into a consistent human decision Hdc. Each of these diagrams assume that the uniformization process [Puterman, 1994] has already occurred and all exponential durations are the same. This results in a modification of transition probabilities, but no addition of extra states or transitions.

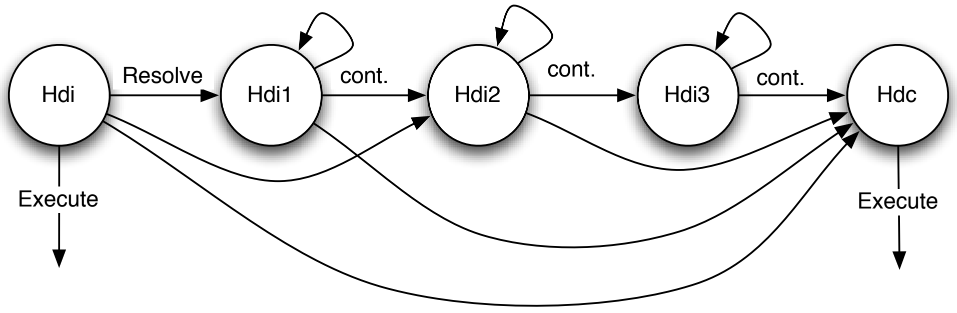
By using the model shown in Figure 6.8-c, we are able to use a traditional CPH solver that has been augmented to allow for an extra action in the middle of execution states ($Hdi1$, $Hdi2$, $Hdi3$) and compute a new expected utility over these states. Previously, these extra states existed only for planning about action durations and did not yield a policy, since they don't represent actual states in the ITMDP. Instead they represent being in the middle of an action (similar to the transitory states mentioned in Section 6.2.1.1).

6.2.5.3 Calculating belief distribution

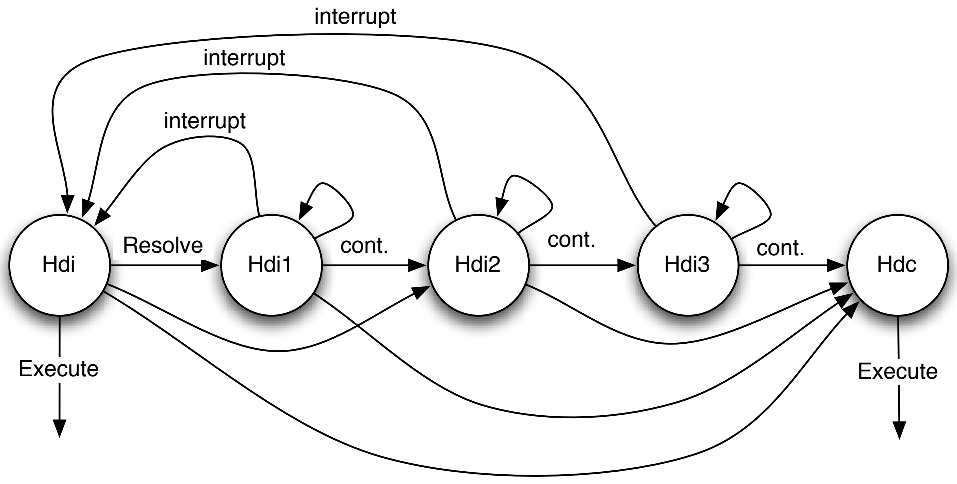
In order to translate this into a policy at execution of an interruptible action, the ITMDP must also reason about the likelihood of being in each of the intermediate states ($Hdi1$, $Hdi2$, or $Hdi3$). This is because during execution, the intermediate state cannot be observed. What can be observed



(a) Subsection of the original MDP model



(b) CPH converted model



(c) CPH with interruptible actions

Figure 6.8: This figure shows how the resolve action in (a), which is a subsection of the original RIAACT ITMDP model with an arbitrary action duration distribution is converted to an approximate sum of exponential distributions for CPH in figure (b) and how that model is extended to allow for interruptible actions in figure (c).

is that state is somewhere between *Hdi* and *Hdc* and in the middle of an action. However, the time elapsed since the start of an execution is observable and known. Consequently, based on transition properties of the phase type distributions, a belief distribution can be constructed that maps time elapsed since the start of the action to a belief state over all intermediate states.

I ran a Monte Carlo simulation in order to quickly obtain the belief distribution over the intermediate states. A histogram can be created by running the simulated action (*resolve*) from start (*Hdi*) to finish (*Hdc*) a number of times and recording for each run at what time each intermediate state is reached. Then a belief distribution is created by looking at a certain amount of elapsed since the action as started (this is earlier collectively referred to as a transitory state) and evaluating how often on average the agent is in a particular intermediate state. This does not factor in, the probability that the action has completed, because if the action has completed then the next state has been reached and the traditional TMDP policy resumes. The action (continue or interrupt) rewards of the intermediate phases are weighted with the belief distribution of being in those particular intermediate phases after a certain amount of elapsed time.

Adding interruptible actions to the model results in a time-dependent policy for each state, as before, but also results in a time dependent policy for what to do during the execution of an action. This policy has only two available actions: continue (*null*) and interrupt. The continue action is not an explicit action, but represents the continuing of the current action. The interrupt action represents stopping the current action and going back to the previous real world state. For each interruptible action, there is a new policy created that shows based on how much time has elapsed since the action was taken, what is the optimal choice: continue or interrupt.

6.2.5.4 Converting the model

As explained earlier, non-exponential distributions will be approximated as a sequence of phase-type exponential functions. In the example scenario shown earlier in Section 6.1, I categorize actions as either involving a human or being a machine-only action. It is customary to model the action duration involving a human as having a normal distribution and machine-only actions are modeled as having an exponential distribution [Younes and Simmons, 2004].

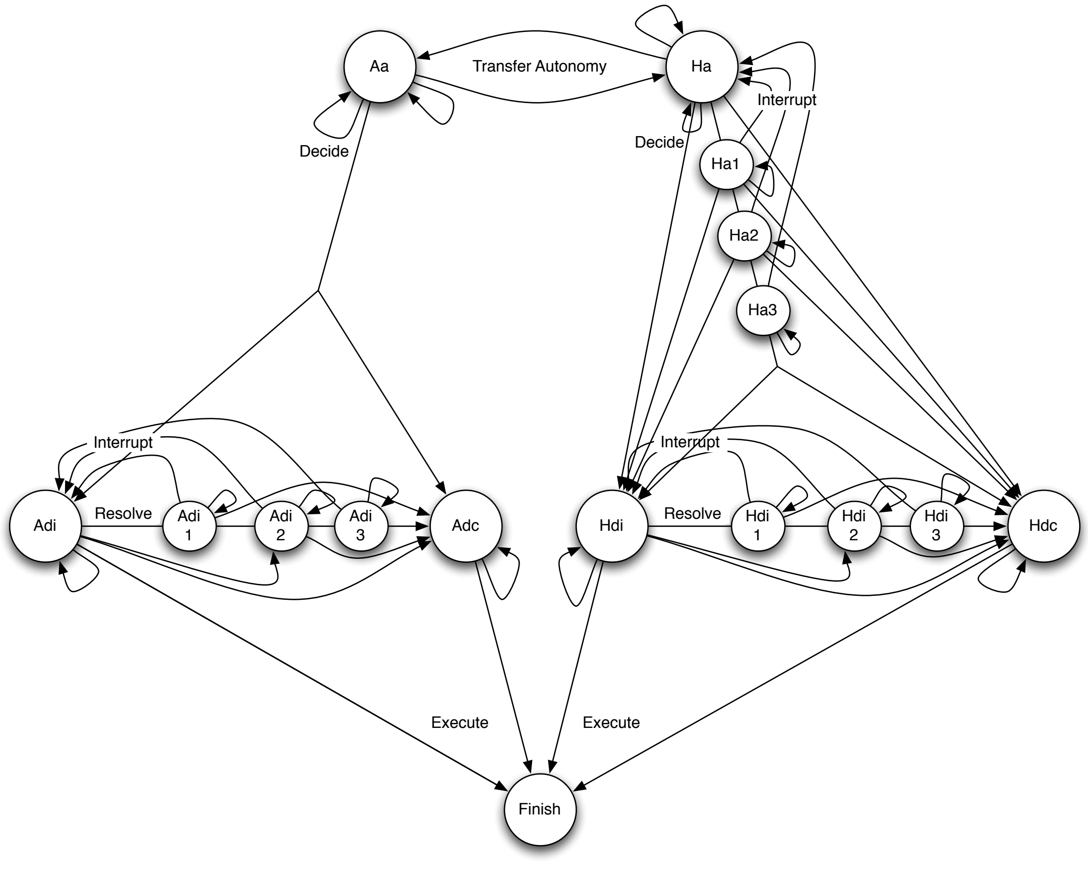


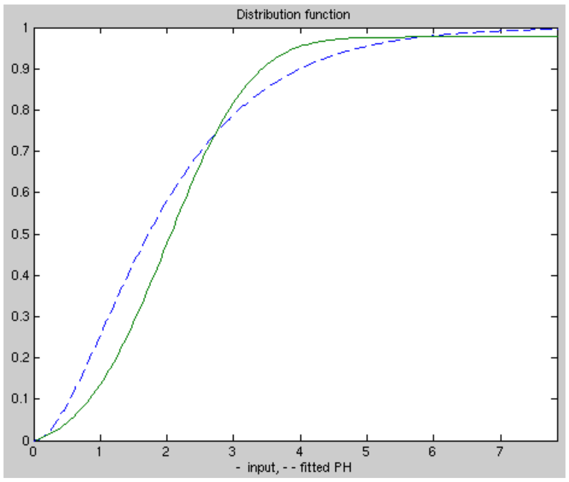
Figure 6.9: RIAACT modified to be input into CPH

Figure 6.9 shows how Figure 6.7 must be augmented to allow for extra states since CPH can only take exponential distributions or sums of exponential distributions as input. As mentioned earlier in Section 2.4.3, after all distributions are converted to exponentials, the uniformization process takes place [Puterman, 1994], which converts all the exponentials to have the same duration, but introduces self-transitions. In order to allow the ITMDP to work with CPH, I had to uniformize the ITMDP. No extra states are necessary for the process of $Transfer$ of control, *Execute*, and agent *Decision* because they are machine only (agent only) actions in our scenario. They only need extra self transitions due to the uniformization. Each of the other actions is approximated by a sequence of intermediate states with exponential distributions that also have self transitions. Any number of intermediate states (phases) can be used, although I have chosen to use 3 phases (as seen in Figure 6.9).

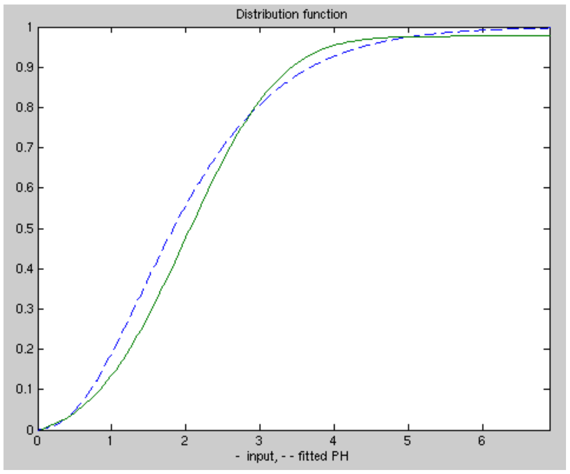
Although Figure 6.9 has added intermediate states that must be reasoned about, it has the advantage that each state transition (arrow) now represents an exponential function. As mentioned in Section 2.4.3, the fact that each of these are exponential distributions allow for the substantial speedups achieved by CPH. The intermediary states are not part of the eventual policy that is generated: they only represent models of time duration.

I have chosen to approximate the normal distributions with three exponential phases and this creates three extra states for each of the actions with normal distribution durations (as seen in Figure 6.9) [Marecki et al., 2007]. In addition, I have chosen to model the normal distribution with a Coxian approximation. The Coxian is a standard phase-type approximation that only allows transitions from the current state to either the next intermediate state, the final state, or a self transition. This allows for an accurate approximation without having exponential numbers of transitions as the number of phases increases.

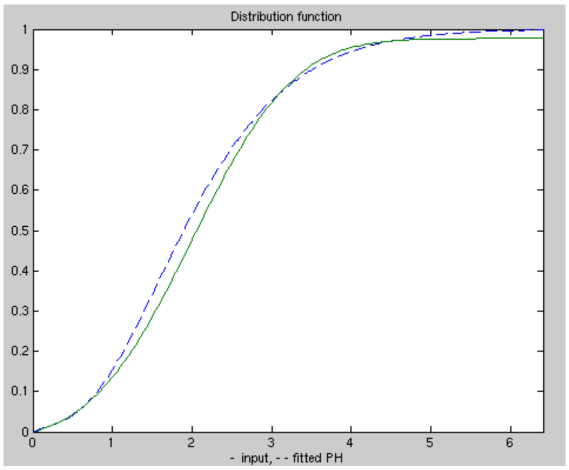
Figure 6.10 shows the cumulative distribution function of the Normal(2,1) as compared to the Coxian approximation with varying numbers of intermediate (extra) phases (2,3,4 and 5). On the



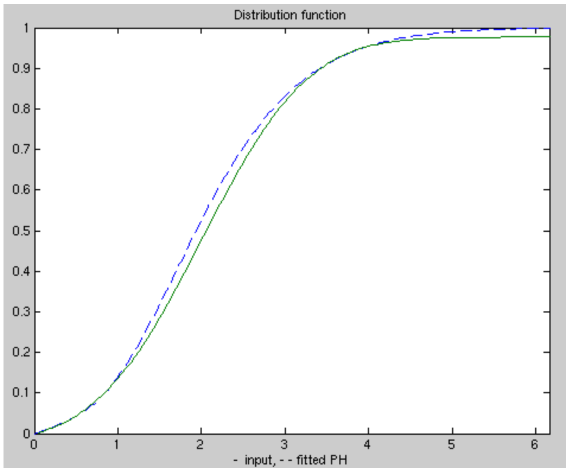
(a) 2 Phases



(b) 3 Phases



(c) 4 Phases



(d) 5 Phases

Figure 6.10: This figure compares each of the phase type approximations with the original Normal(2,1) distribution.

x-axis is the amount of time that has passed and on the y-axis is the cumulative probability that the execution has completed and the final state has been reached. The original normal distribution input is shown in the solid line and each of the dashed lines represents the fitted Coxian. RIAACT allows for any number of intermediate states to be used. Computing the approximation of the distribution can be done offline, however it is important to choose how many intermediate phases will be used to approximate. In particular, when choosing how many phases to approximate any distribution, both accuracy and runtime must be considered. Using more phases may lead to a closer approximation, however it will also lead to an increase in the amount of states that must be considered. I have chosen to use a 3 phase approximation as seen in Figure 6.10-b, where the difference between the input and the fitted distribution does not exceed .05. The improvements gained from using 4 and 5 phases are minimal, whereas the number of states are increasing and the runtime will suffer (this phenomenon can be seen in [Marecki et al., 2007]).

6.2.6 Hybrid Coordination

6.2.6.1 Agent Coordination

In order to improve performance and prevent miscoordination, each team member's proxy will communicate on the team member's behalf. There are two primary types of messages that get communicated between the proxies: (i) related to joint commitments and (ii) for role allocation. Both of these are in an effort to search the team's goals and depend on the state of both the world and the agent team.

If a change is observed in the world state that is significant or relevant to either a joint commitment that the agent is a part of, that new world state is communicated to the other proxies. This is based on joint intentions framework [Cohen and Levesque, 1990]. If the characteristics of a role has changed such as which team member is performing the role or which team member has the autonomy of the role, then that updated role is communicated to the other proxies. Agents communicate in order to jointly commit to a team plan. In addition, if anything occurs affecting the ending of a plan and it being achieved, irrelevant or unachievable, then that information about the plan is communicated to the other proxies.

The communication message types are used to help in the filtering or abstraction of messages so that unimportant messages do not crowd the communication channels. Changes in joint commitments and roles are communicated so that the agents may more accurately reason about the world and the state of the human-multiagent team.

By employing this agent coordination, the human-multiagent team can avoid having lower priority or redundant resolve actions from being attempted. For example, there is a team plan that is instantiated for a resolve action. Consequently the agents are jointly committed to this team plan and each know about the existing plan. This prevents the agents from unnecessarily executing their own resolve plan for the same reasons and sending redundant messages to the human. In addition, multiple agents may be able to detect the same inconsistency, however only one resolve is necessary. In addition, this prevents the team from over-correcting or thrashing back and forth with inconsistencies. Furthermore, if many different resolves plans executing in parallel, the team could overcorrect if they are not made aware of the result of other resolve plans.

6.2.6.2 Hybrid Algorithms

I have explained how an agent can compute an online abstract, decomposed ITMDP, however the team needs to decide when to compute these and how to coordinate as a team so that the human is not overwhelmed with resolve actions all at once.

Algorithms 2 and 3 represent a hybrid approach since different sections of the algorithms represent different components of the hybrid. For example, everywhere the *policy* is mentioned in these algorithms, this references either the computing or executing of the RIAACT ITMDP policy. In addition, functions like OfferToHuman and ResolveWithHuman represent team plans that will be executed in a coordinated fashion using joint commitments and team logics. Furthermore, the CommunicateToTeammates function represents a logic-based communication policy that will decide the method of communication and to whom to communicate.

Algorithm 2 is executed once a new, unallocated team decision is created. It begins by leveraging Algorithm 1 that I have previously shown which handles the passing of decision tokens through the team for role allocation. After, no more agents can consider the decision, help is asked for from the human and an adjustable autonomy/resolution policy should be computed online as seen in Line 3 of Algorithm 2.

Algorithm 2 TeamDecisionTask(*decision*)

```
 1: TokenMonitor($decision$) /* Algorithm 1 */
 2: if $decision.NextAgent = NULL$ then
 3:    $policy$ ←ComputeITMDP($decision$)
 4:    if $policy[AgentHasAutonomy, currentTime] = transfer$ then
 5:       OfferToHuman($decision$)
 6:       while $policy[HumanDecisionTransitoryState, currentTime] = continue$ do
 7:          $msg \leftarrow getMsg()$
 8:          $decisionMade \leftarrow msg$
 9:          if CompareCapabilityAndPriority($decision.context, decisionMade.context$) then
10:             $policy$ ←ComputeITMDP($decision$) /* recompute inaccurate policy */
11:          ProcessDecisionMade($decisionMade$) /* Algorithm 3 */
12:    if $policy[AgentHasAutonomy, currentTime] = execute$ then
13:       $decision.clearHistory$
14:       TeamDecisionTask($decision$)
```

If a human is given the autonomy over the decision (see Line 5 of Algorithm 2), then the agent team waits for a human decision. If the human makes a decision then it is received by an agent (Line 8 of Algorithm 2). Once it is received, the current context (in terms of team capability and decision priority) is compared to the context of the policy to see if the policy is still accurate (Line 9). If it is not, then recompute it (Line 10). Then the decision is evaluated using the accurate policy by executing Algorithm 3 (Line 11 of Algorithm 2). Algorithm 3 determines if the decision is inconsistent and to determine what to do if an inconsistency needs to be resolved, according to the policy.

Algorithm 3 ProcessDecisionMade($decisionMade$)

```
1: $expectedUtilityLoss \leftarrow$ ComputeExpectedUtility($decisionMade$)$-$ComputeExpectedUtility($current$)
2: if $expectedUtilityLoss > threshold$ then
3:    if $policy[HumanDecisionInconsistent, currentTime] = resolve$ then
4:       InstantiateResolvePlan($decisionMade$)
5:       CommunicateToTeammates($resolvePlan$)
6:       ResolveWithHuman($decisionMade$)
7:       while $policy[ResolveHumanDecisionTransitoryState, currentTime] = continue$ do
8:          $msg \leftarrow getMsg()$
9:          if $msg = newDecision$ then
10:            CommunicateToTeammates($newDecision$)
11:            ProcessDecisionMade($newDecision$)
12:         else
13:            if $msg = otherResolve$ then
14:               if OtherResolveTakesPresedence($otherResolve$) then
15:                  StopResolveWithHuman($decisionMade$)
16:         else
17:            if $msg = otherResolveEnded$ then
18:               ProcessDecisionMade($decisionMade$)
19:   if $policy[HumanDecisionInconsistent, currentTime] = execute$ then
20:      if AgentCanOverride($decisionMade$) then
21:         Execute($current$) /* override the human decision */
22:      else
23:         Execute($decisionMade$) /* execute inconsistent decision*/
24: else
25:    Execute($decisionMade$) /* decision is consistent */
26: EndResolvePlan($decisionMade$)
```

Algorithm 3 determines if there is an inconsistency in Line 2, the original ITMDP policy is checked in order to find out if the resolution is beneficial or not and if so, for how long. A major issue that must be addressed is what happens when multiple agents are attempting to resolve at the same time. This algorithm through the communication (see Lines 5,10) and instantiation of a team plan (see Line 4) allows the other teammates to be aware of all possible resolves and choose which of the agents' resolve takes precedence (see Line 14). In this algorithm, precedence is assumed to be given to the resolve that must finish first, however precedence can be determined by any method. Once a resolve of another teammate has ended, then the decision is evaluated again to see if this agent still has an inconsistency that is yet to be resolved (see Lines 17-18). Depending on the domain and state of the world, the agent may have the ability to override the human's inconsistent decision once a resolve action is no longer possible. This is captured in Line 21 and if the agent does not have the capability to override, then it must execute the human's inconsistent decision (see Line 23).

6.3 Experiments

I have conducted two sets of experiments to investigate RIAACT. The first set consists of policy experiments used to explore the advantages of RIAACT over previous adjustable autonomy models. The second set applies these policies to a disaster simulation system, DEFACTO. Both of these experiments use the same motivating scenario as explained earlier in Section 6.1. The experiments section will be divided as follows: (i) model conversion for the CPH solver, (ii) instantiating the scenario and theoretical experiments, and (iii) DEFACTO simulation experiments.

6.3.1 Disaster Response Scenario and Testbed Policy Experiments

I used the RIAACT model as shown in Figure 6.6. In addition, both the testbed policy and DEFACTO simulation experiments use the same scenario that was explained in Section 6.1. To instantiate the scenario for this particular domain, I must assign the action duration functions and the rewards for given action-state pairs.

$Reward(StartingState, Action, FinishingState)$	**Value**
$R(Adc, Execute, Finish)$	6
$R(Adi, Execute, Finish)$	5
$R(Hdc, Execute, Finish)$	10
$R(Hdi, Execute, Finish)$	7.5

Table 6.1: Average rewards for reaching the Finish state after executing an Agent decision that is either consistent (*Adc*) or inconsistent (*Adi*) and a Human decision that is either consistent (*Hdc*) or inconsistent (*Hdi*).

I assume that this is a very dynamic scenario where the situation is very uncertain. Hence the probability of consistency is 0.5 for both the human $P(c, H)$ and the agent $P(c, A)$. Table 6.1 shows the rewards for this scenario. As shown in Section 2.4.1, I will express the reward as $Reward(StartingState, Action, NewState)$. In the disaster response domain, I will measure the reward in terms of buildings saved compared to the maximum (10) that would catch fire if not addressed. In order to estimate the reward I assume that in the best case 10 buildings can be saved, however in the worst case the engines are able to save only 5 buildings with their effort. I assume that the reward for an agent decision is much less than that of a human. Thus, the reward of an agent decision that is consistent $R(Adc, Execute, Finish)$ is 6, whereas an agent's inconsistent reward $R(Adi, Execute, Finish)$ is on average only 5. The reward of a human decision that is consistent $R(Hdc, Execute, Finish)$ is assumed to be the maximum value of 10, whereas a human's inconsistent reward $R(Hdi, Execute, Finish)$ is assumed to be the median of 7.5.

Table 6.2 shows the action duration distributions for this scenario. For machine processes, it is customary to assign an exponential function to the action duration [Younes and Simmons, 2004]. Consequently, for actions not involving the input of a human, I assume the action duration to be a very short duration exponential function with a mean of 0.5 seconds. This includes the $transfer$ of autonomy action, the *decide* action for an agent, and the *execute* decision action (the

Action	Type	Distribution
Transfer Autonomy to Agent	Machine Only	Exponential
Transfer Autonomy to Human	Machine Only	Exponential
Agent Decision	Machine Only	Exponential
Human Decision (Agent Wait)	Human	Normal
Resolve Human Decision	Human	Normal
Resolve Agent Decision	Human	Normal
Execute Any Decision	Machine Only	Exponential
Interrupt Resolve	Machine Only	Exponential

Table 6.2: Action Durations

agents are performing the execution of the fire allocation decision in this scenario). In addition, I assume the interrupt action to take very little time in this domain and it is also modeled with the same exponential function with a mean of 0.5 seconds.

The *decide* action for a human (wait for human decision) and the *resolve* action involve a human and the duration will be modeled as a normal distribution. Although a normal distribution is used here, RIAACT can apply to any arbitrary distribution as it will approximated as a phase type, as seen in Section 6.2.5.4 above. Depending on how available and how experienced the human is, the distribution may change. I will keep the human *decide* action duration at a constant distribution, of a Normal(3,1), which is a normal distribution that has a mean of three and a standard deviation of one. I will use this notation to later describe normal distributions. I will focus on the *resolve* action because it will allow me to demonstrate the distinct benefits of RIAACT: resolving inconsistencies, developing a continuous time policy, and allowing interruptible actions.

I will first experiment in a simple testbed domain, assuming that the *resolve* action has duration distribution of a Normal(9,5). Then I will show experiments with the DEFACTO simulation where I vary the action duration distribution for the *resolve* action.

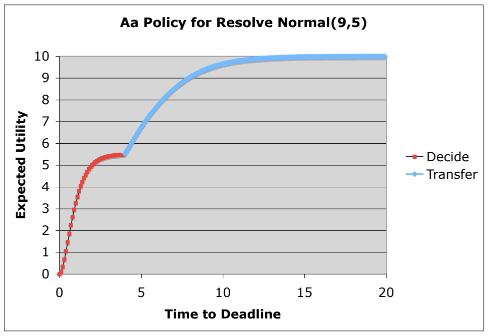
Team	Policy only. 6 engine agents and 1 virtual human, resolve action duration follows a Normal(9,5) distribution.
Scenario	Fires have been ignited at 2 buildings and will start increasing in intensity and possibly spread. The human-multiagent team must coordinate to assign engines and extinguish as many fires as possible. An inconsistency occurs if the human makes an allocation decision to a fire and that agent expects the decision will result in degradation of performance.
Decision	To determine which fire engine, if any, should be allocated to a firefighting role associated with the team plan to fight a fire in a particular building. A policy for adjusting autonomy and, if an agent detects an inconsistency with the human's role allocation decision, a resolve action is available.
Information	Human - Has a top down view of the whole map with intensity of fire in each building. Knows task allocation and geographic location of all agents. Agents - Each agent is aware of the size, height, and fire intensity of buildings within a limited radius, how much water is in its tank, status of only agents that are fighting the same fire, allocation of all teammates.

Figure 6.11: Testbed policy experimental details.

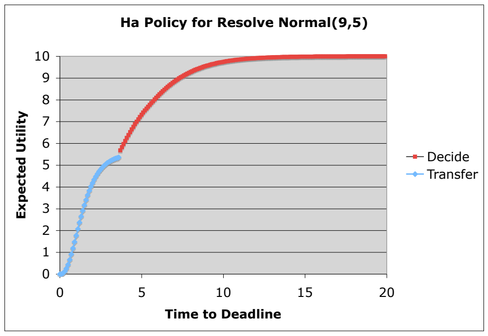
6.3.2 Testbed Policy Experiments

Figure 6.11 explains the details of this set of experiments that were conducted.

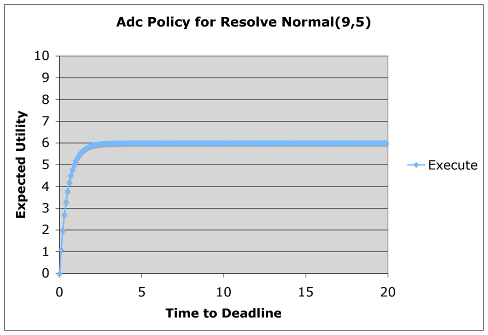
- *Experimental Setup* - Simple testbed domain to construct a policy. 6 agents, *resolve* action duration is Normal(9,5)
- *Reason for Experiment* -To show the benefits in the theoretical model of (i) continuous time, (ii) the resolve action, and (iii) interrupt.
- *Result of Experiment* - Benefits are shown and this confirms the usefulness of the RIAACT model in the policies.



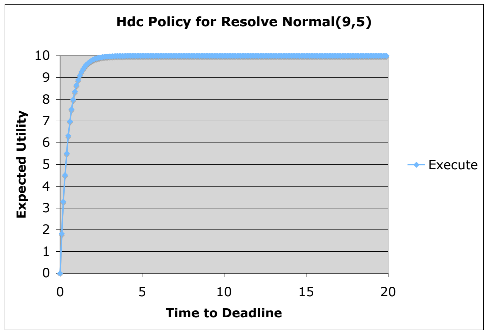
(a) *Aa*



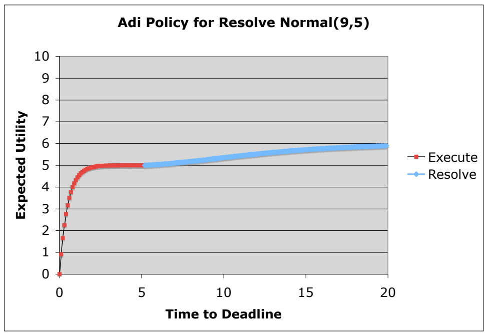
(b) *Ha*



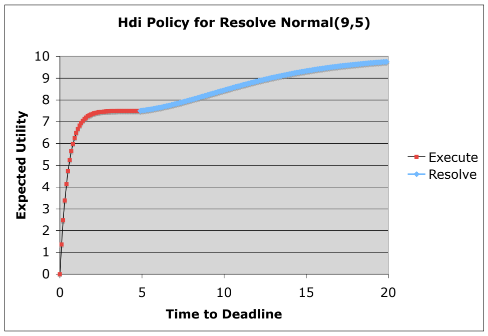
(c) *Adc*



(d) *Hdc*



(e) *Adi*



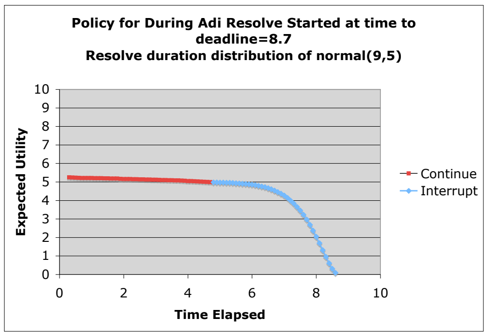
(f) *Hdi*

Figure 6.12: RIAACT Model example policy output of each state (Aa, Ha, Adc, Hdc, Adi, Hdi) given that the resolve action duration fits a Normal(9,5).

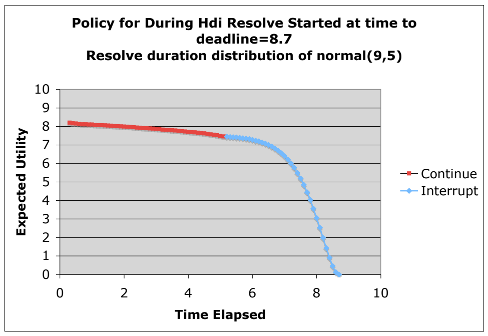
Figure 6.12 shows an example of a policy where the *resolve* action duration distribution is a Normal(9,5). For each general state, the policy shows the optimal action to take and the expected utility of that action as a function over time. On each x-axis is the amount of time left until the deadline and on the y-axis is the expected utility. Thus, if any state is reached, then the time to deadline is referred to and the optimal policy is chosen. For example, if the human has the autonomy (*Ha*) and the time to deadline is greater than 3.6 seconds, then the optimal action is to attempt a human decision. Otherwise, the optimal action is to transfer that decision over to the agent in order to have the agent make a quicker, but lower average quality decision.

Figure 6.12 a and b shows the benefit of using a continuous time approach to modeling the adjustable autonomy problem. Figure 6.12-a shows that the dominant action for the agent has autonomy state, *Aa*, is to transfer the decision to the human up until 3.9[3] seconds before the deadline. On the other hand, Figure 6.12-b shows that the dominant action for the human has autonomy state *Ha* is to *decide* up until 3.6 seconds before the deadline. There are two types of discrete approaches to compare this continuous approach: many intervals and few intervals. First, if a policy of comparable accuracy was desired, then the policy would need to at least have the resolution of 0.1 seconds. Consequently, if the policy were to cover the same 20 second span, then a total of 200 intervals would be used and 200 states would be need for every single state in the current policy. This would result in 1,200 states instead of the 6 in our current simplified model. Second, the discretized approach could alternatively aim to reduce the amount of states and allow for quicker runtimes. For example, if the time were to have been discretized into 8 separate intervals, the state space would be increased by a factor of 8 instead of 200. However, the interval from 2.5 to 5.0 would only have one action associated as the optimal action. The

[3]Note that these values have been rounded to the nearest tenth of a second, however the values are real numbers that each can have an infinite decimal representation.



(a) $\langle Adi, resolve\rangle$



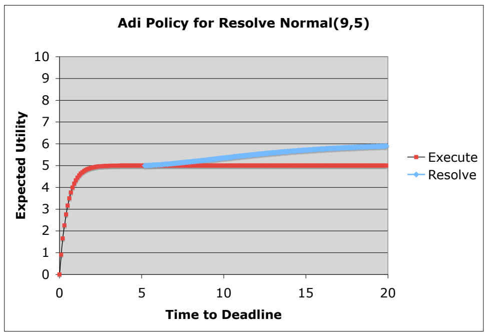
(b) $\langle Hdi, resolve\rangle$

Figure 6.13: RIAACT Model example interrupt policy for resolve action given that the resolve action duration fits a Normal(9,5). These graphs represent interrupt policies for the two transitory states $\langle Adi, resolve\rangle$ and $\langle Hdi, resolve\rangle$.

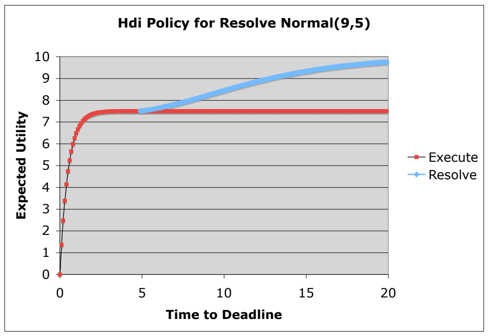
potential gains from knowing the exact point to switch from *Decide* to *Transfer* as the dominant action would be missed. No matter which action is chosen as the optimal, there are periods of time between 2.5 and 5.0 where potential reward would be lost.

Figure 6.12 c and d show only one action because there is only one possible action to take from a consistent decision, *execute*. It is shown here in order to display the entire state space and to give an idea of the expected reward over time of each of the consistent states, *Adc* and *Hdc*. Figure 6.12 e and f show the benefit that is achieved from using the resolve action. Each point where the *resolve* action dominates the execute action shows a region of time where the *resolve* action provides a benefit. One important new aspect of these interruptible policies is that the rewards take into account the ability of certain actions to be interrupted. In this case, human *decision* (waiting for) and *resolve* are both actions that can be interrupted.

Figure 6.13 demonstrates the benefit achieved from allowing an action to be interruptible at any point in continuous time. If the action were not interruptible, then the action inherently



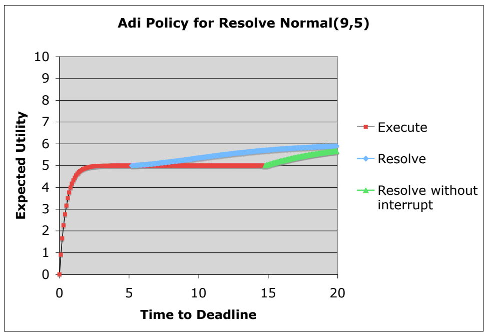
(a) *Adi*



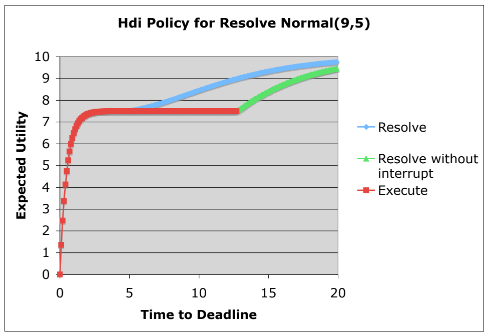
(b) *Hdi*

Figure 6.14: The policies for the states *Adi* and *Hdi* comparing when the *resolve* action is available or not.

chosen over all execution time would be the continue action (because no other action is available). Figure 6.13 shows the policies that are created for a resolve action if it were to be started at 8.7 seconds to deadline and still executing resolve in either the Adi (subfigure a) and Hdi (subfigure b). To express this in terms of the ITMDP model, these are interruption action policies for the transitory states $\langle Adi, resolve\rangle$ and $\langle Hdi, resolve\rangle$ respectively (for more on transitory states, see Section 6.2.1.1). The x-axis here shows time elapsed since the action has started and they y-axis shows the expected utility of the optimal action (continue or interrupt in this case). The time at which the interrupt action becomes the optimal action in the policy shows how long to attempt the *resolve* action. For example if they were to be both started at 8.7 seconds until deadline, then the Adi Resolve action should be attempted for 4.7 seconds whereas, if it were the Hdi Resolve action, then it should be attempted for 5.2 seconds before it should be interrupted. This longer time before interrupting the *resolve* action is due to the greater potential benefit from resolving an inconsistent human decision (7.5 to 10 reward) versus an inconsistent agent decision (5.0 to 6.0 reward).



(a) *Adi*



(b) *Hdi*

Figure 6.15: The policies for the states *Adi* and *Hdi* comparing when the *resolve* action is interruptible and not interruptible.

In order to show the benefit that the *resolve* action provides, a new set of experiments was conducted with the exact same parameters except that the *resolve* action was removed. The results of this experiment can be seen in Figure 6.14. Since there is no *resolve* action possible, the policy for these inconsistent states (*Adi* and *Hdi*) is to always execute. I have superimposed the expected utility that the *resolve* action provided in previous experiments in order to show the difference. In these graphs, the x-axis represents the time to deadline and the y-axis represents the expected utility. As seen in both subfigures a and b, the *resolve* action provides more improvement if there is more time available before the deadline because the *resolve* action takes some time to finish.

I also performed experiments to explore the benefit that the ability to interrupt was providing to the *resolve* action. To see the effects of this, in Figure 6.15 I show the policies of both *resolve* with interrupt and *resolve* without interrupt for each of the inconsistent states *Adi* and *Hdi*. Again in these graphs, the x-axis represents the time to deadline and the y-axis represents the expected utility. As seen in both charts, the standard *resolve* action (with interrupt) provides a higher expected reward and alters the point at which the action ceases to be optimal. For example, as

Team	Application of of Policies to DEFACTO. 6 fire engine agents and 1 virtual human, resolve action duration follows varying distributions of Normal(3,1), Normal(6,4), Normal(9,5), Normal(12,6), and Normal(12,2).
Scenario	The human-multiagent team is coordinating to assign fire engine agents and extinguish as many fires as possible. The human has made an agent allocation decision to a fire and that agent finds this to be inconsistent (expects a degradation in performance).
Decision	An inconsistency about a role allocation to a fire arises. The agent must decide, given the time to deadline, whether to try and resolve the inconsistency and, if so, for how long until the resolve should be interrupted.
Information	Human - Has a top down view of the whole map with intensity of fire in each building. Knows task allocation and geographic location of all agents. Agents - Each agent is aware of the size, height, and fire intensity of buildings within a limited radius, how much water is in its tank, status of only agents that are fighting the same fire, allocation of all teammates, the duration distribution of the resolve action.

Figure 6.16: DEFACTO experimental details.

seen in Figure 6.15-b the policy for *Adi* is to attempt to *resolve* an inconsistency if it is detected with at least 14.8 seconds if the action is not interruptible, however the *resolve* action is beneficial if the inconsistency is detected with at least 5.2 seconds if the action is interruptible.

6.3.3 DEFACTO Experiments

In addition to running this model on CPH, I have also implemented this in a disaster response simulation system (DEFACTO). In this case a distributed human-multiagent team works together to try and allocate fire engines to fires in the best way possible. I utilize the proxy architecture to facilitate the execution of the policies computed by RIAACT.

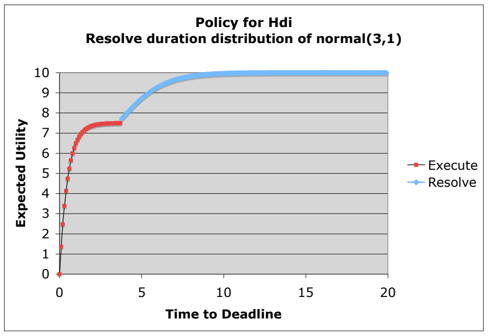
These experiments have been conducted in the DEFACTO simulation in order to test the benefits of the RIAACT policy output. Figure 6.16 explains the details of this set of experiments that were conducted. I have implemented the full policy that RIAACT outputs, however since all of the key elements that I wish to demonstrate are involved with the *resolve* action, I will focus on it. In the scenario that I am using for these experiments, the human has had the autonomy and has made a decision. However, this decision is found to be inconsistent (*Hdi*) by at least one agent and now it runs an ITMDP (following the same transition probabilities as instantiated in the previous section) to determine two things: 1. whether, at this time, a *resolve* action is beneficial and 2. if so, for how long should the *resolve* action be continued before it is beneficial to interrupt it and *execute* the inconsistent human decision *Hdi*.

- *Experimental Setup* - Application of model, DEFACTO Simulation System, 6 agents, Simulated Human, Focuses on inconsistency resolution, *resolve* action duration is varied between Normal(3,1), Normal(6,4), Normal(9,5), Normal(12,6), and Normal(12,2). This is intended to explore the effects of modeling varying *resolve* durations and how they effect the policy and eventually the team performance. The deadline is assumed to be the point in time at which fires spread to adjacent buildings and becomes uncontrollable, which in the simulation is 8.7 seconds on average. Consequently, in each of these experiments, the human decision is found to be inconsistent at a time when there is 8.7 seconds until deadline.

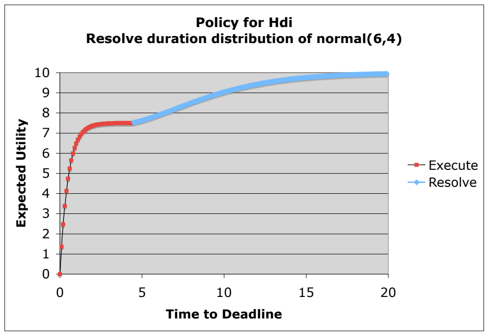
- *Reason for Experiment* -To show the benefits in the application of these policies to the human-multiagent disaster response scenario. Focusing on inconsistency portion of policy allows these experiments to show (i) benefit of the *resolve* action, (ii) advantage of continuous time policy and (iii) the advantage of being able to interrupt.

- *Result of Experiment* - Figure 6.17 and Figure 6.18 shows the policies that were computed and applied to the DEFACTO simulation. These policies determine that if the *resolve* action takes a longer amount of time on average to complete, then the sooner that the action should be interrupted. This was contrary to my hypothesis that since the *resolve* action can be very important to achieving higher reward, the policy would suggest waiting longer to interrupt for longer durations. These policies were then applied to the DEFACTO simulation and the team's performance improved (see Figure 6.19). This shows the usefulness of the RIAACT model in applying to disaster response human multiagent teams.

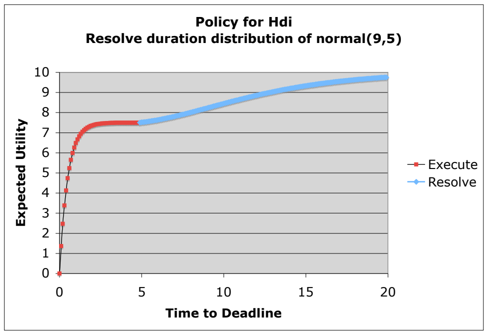
Figure 6.17 shows the policy output for Hdi (Human decision inconsistent) over time with the varying distributions. The time to deadline is shown on the x-axis and the expected utility of the dominant action is shown on the y-axis. For example, looking at subfigure b, the policy shows that if there is enough time to deadline, a *resolve* action should be attempted. However, if the inconsistency is found at any time to deadline that is less than 4.5 seconds, then there is not enough time to *resolve* and the inconsistent decision (*Hdi*) should be executed by the agent team to achieve at least some reward. In the specific scenario that I have focused on, the time to deadline when the inconsistency is determined is always 8.7 seconds. If this is the case, then each of the policies for Hdi shown in Figure 6.17 show the dominant action at 8.7 seconds to deadline to be *resolve*.



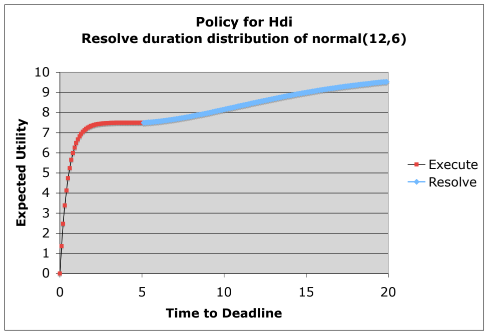
(a) Normal(3,1)



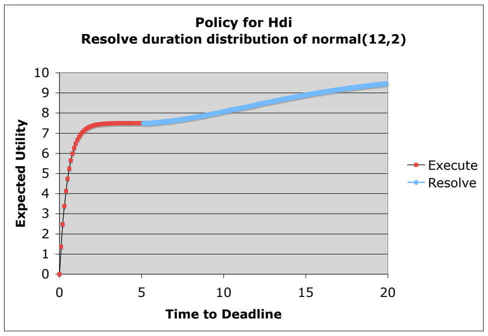
(b) Normal(6,4)



(c) Normal(9,5)



(d) Normal(12,6)



(e) Normal(12,2)

Figure 6.17: Policy predictions for the dominant action (*resolve* or *execute*) when in state Hdi at a given time to deadline.

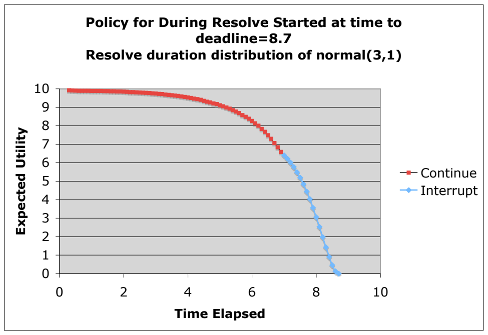
Given this, each of the policies prescribe to start the *resolve* action, however the policy also determines how long an unfinished *resolve* action should continue before it is beneficial to interrupt it. This is done by using the method described earlier in Section 6.2.5. In addition, the belief distribution weights over the intermediate states given the amount of time elapsed since the start of the action is determined using the technique also described earlier in Section 6.2.5.

Once the *resolve* action has begun at 8.7 seconds, the interrupt policies are computed as seen in Figure 6.18. Different from previous graphs, the x-axis now represents time elapsed in seconds. Again on the y-axis is the expected utility of the dominant action (*continue* or *interrupt*). This policy shows that as long as the *resolve* action has not finished, whether it is beneficial to interrupt it or not. For example, looking at the policy for Normal(9,5) in subfigure c, it is beneficial to continue the *resolve* until 5.2 seconds have passed and at that point, the action should be interrupted. The following is a table that shows for each distribution at what point in time it becomes beneficial to interrupt it. These values are then used to complete the team-level adjustable autonomy policy.

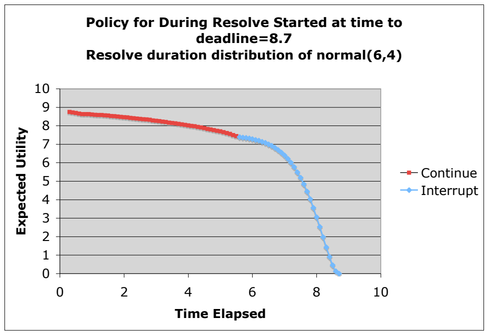
Distribution Normal(Mean,Standard Deviation)	**Interruption Time (Elapsed)**
Normal(3,1)	7.0s
Normal(6,4)	5.6s
Normal(9,5)	5.2s
Normal(12,6)	4.9s
Normal(12,2)	4.8s

Table 6.3: Policies for interruption during the *resolve* action given a start time to deadline of 8.7 seconds. These values are derived directly from the policies shown in Figure 6.18.

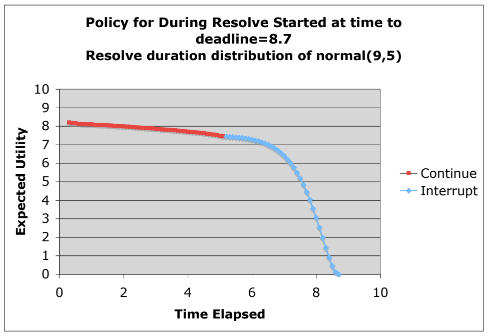
Using the interrupt policies shown in Table 6.3, I conducted experiments where DEFACTO was run with a simulated human with the given action duration distribution. A simulated human was used to allow for repeated experiments and to achieve statistical significance in the results.



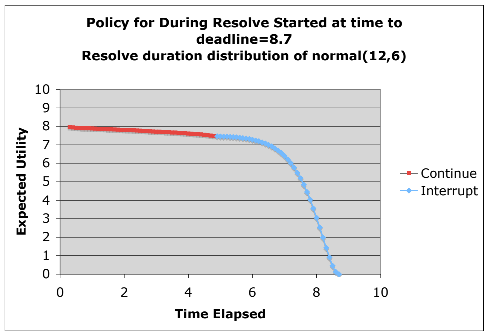
(a) Normal(3,1)



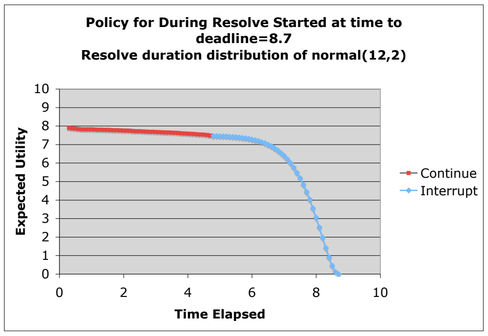
(b) Normal(6,4)



(c) Normal(9,5)



(d) Normal(12,6)



(e) Normal(12,2)

Figure 6.18: Policy predictions for the dominant action (*continue* or *interrupt*) when executing *resolve* and a given amount that has elapsed since the *resolve* action began at time to deadline is 8.7 seconds.

When an inconsistency is detected by the agents in an uncertain world, there is a chance that the agents have raised this inconsistency unnecessarily (false positive). In other words, the agents do not have a better decision to offer and the resolution will result in the same decision as before, however now being consistent. In these experiments I assume this probability $P(IU)$ to be 0.5, however this can be altered along with the average expected reward of the inconsistent state. In fact, this is varied later in Figure 6.21.

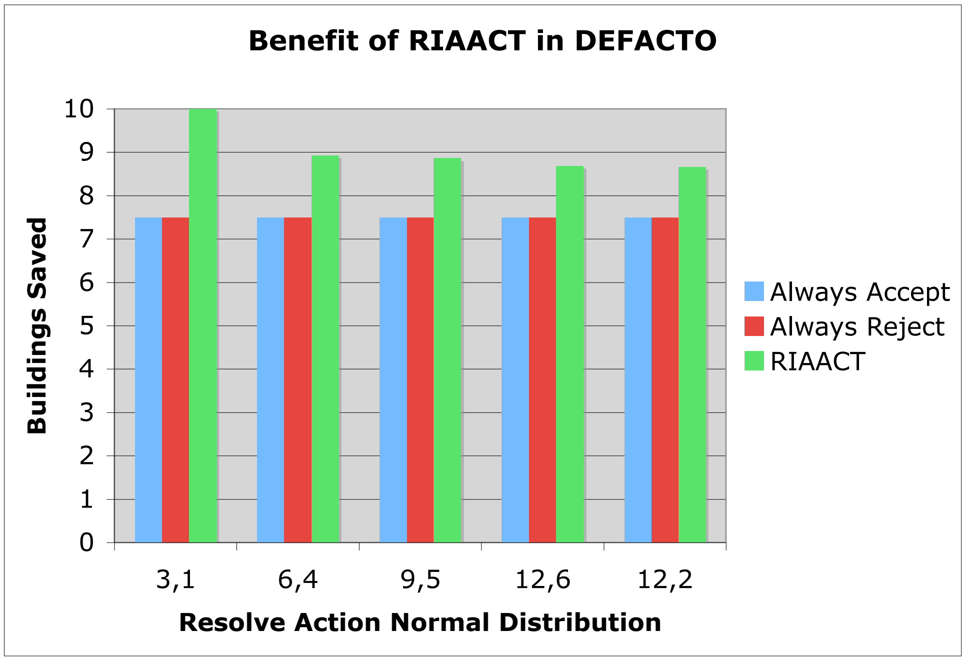


Figure 6.19: This figure shows the increase in team performance in the DEFACTO simulation when using RIAACT to implement a *resolve* action with the ability to interrupt it.

In order to show the benefit that the *resolve* action gets from being interruptible, experiments were performed with running the *resolve* action until it finishes (see Figure 6.19). The duration is sampled from the same varying normal distributions, shown on the x-axis. Since the progression of the world in non-deterministic and the distributions can be uncertain, in order to assess the overall performance, these are averaged over 50 experimental runs. The y-axis shows performance in terms of amount of buildings saved. The Always Accept policy is the equivalent of previous work in adjustable autonomy where a decision was assumed to be final. The Always Reject policy performs similarly due to the rejection of good human decisions. The RIAACT policy improves over both of these static policies.

Figure 6.19 also shows that as the *resolve* action duration increases, the benefit gained from using RIAACT decreases. This is due to the approaching deadline and the decreased likelihood that the *resolve* will be completed in time. Although, the difference in performance for the Normal(12,2) case may be the smallest, the results show statistical significance $P < 0.05$ ($P = 0.0163$).

As can be seen from Figure 6.20, the benefit in team performance is not only from having the *resolve* action itself, but by allowing it to be interruptible. Again, the x-axis shows the various distributions tested, and the y-axis shows team performance in terms of number of buildings saved. The gains are not as present in Normal(3,1) because the action duration is so short that it need not be interrupted. However, as seen in Normal(9,5), an average of 3.2 more buildings were saved over an uninterruptible action.

Figure 6.21 shows how the benefits that RIAACT brings are affected by the probability that the inconsistency that was detected is useful $P(IU)$. Before this was assumed to be 0.5 since the domain was so uncertain. However, if the probability that the detected inconsistency, if resolved, leads to a better solution increased to 0.7, as in Figure 6.21-c, the performance of RIAACT increases. This is due to the fact that the attempted *resolve* action is more likely to result in higher

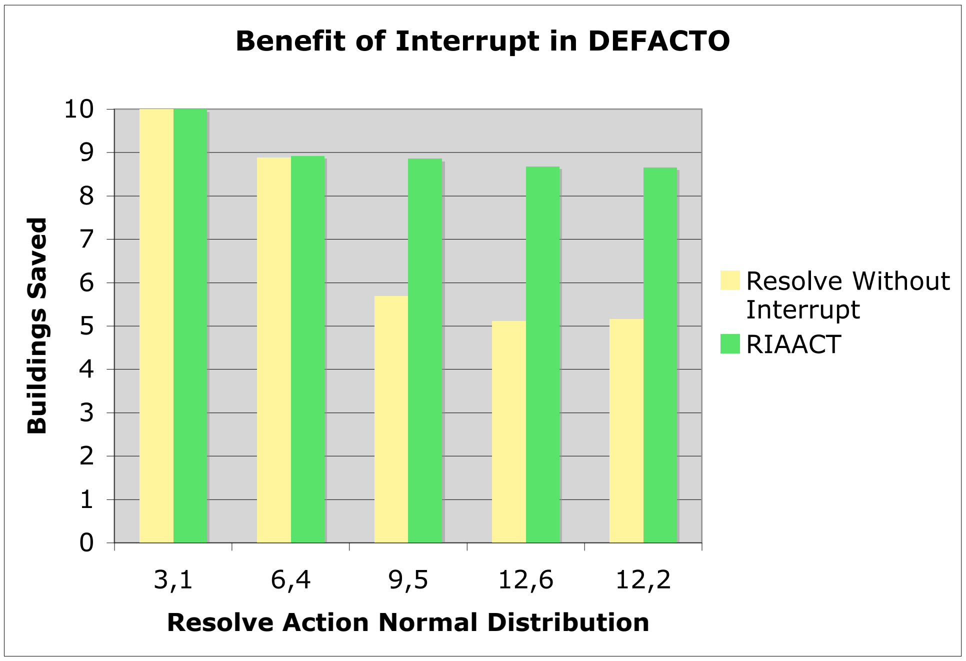
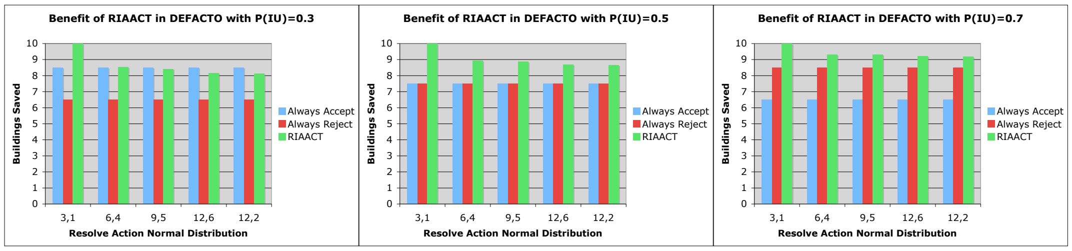


Figure 6.20: This figure shows the increase in team performance in the DEFACTO simulation when using RIAACT to implement a *resolve* action with the ability to interrupt it versus a *resolve* that waits until it is completed.



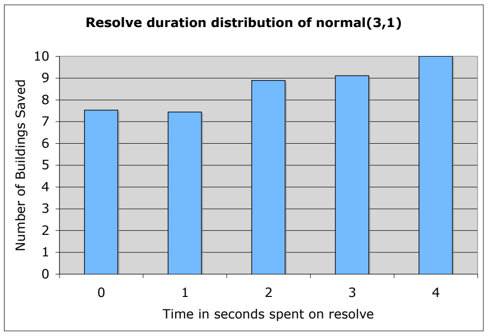
(a) $P(IU) = 0.3$ (b) $P(IU) = 0.5$ (c) $P(IU) = 0.7$

Figure 6.21: Experiments given a simulated human and varying the probability that resolving the inconsistency would be useful $P(IU)$.

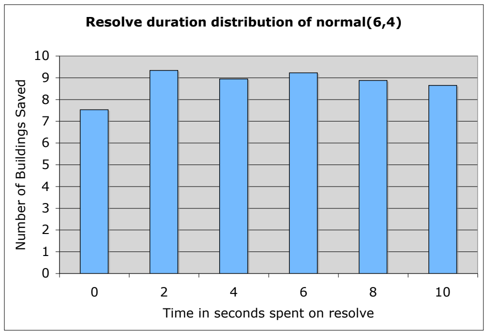
performance. However, if $P(IU) = 0.3$, as in Figure 6.21-a, then the benefits of RIAACT decrease with respect to Always Accept (previous strategy) until in the Normal(9,5) case, RIAACT can be seen to do worse. These experiments highlight how the *resolve* action, and RIAACT in general, provides performance improvement as long as the estimate of $P(IU)$ is accurate and relatively high. Otherwise, if the agent team is poor at detecting inconsistencies that will result in better team performance, then it might be better to not *resolve* these inconsistencies.

In order to explore the benefit of having these policies expressed in continuous time, I conducted experiments that varied the amount of time that the *resolve* would wait before being interrupted. Figure 6.22 shows the results from implementing the *resolve* policy in DEFACTO and running experiments on a simulated human that has a response time sampled from varying distributions. They have been averaged over 50 runs. These distributions are listed in the format of Normal(mean, standard deviation) in minutes. The y-axis shows the average amount of buildings that were saved. On the x-axis is a varying amount of time that was spent on the *resolve* action before interrupting it and executing the inconsistent decision. For example, in the Normal(12,6) graph, waiting for 6 seconds, the expected amount of buildings saved drops to 8.2 versus the 8.7 buildings saved on average by rather than the 4.8 seconds determined by the RIAACT policy seen in Figure 6.19. This shows the benefit of being able to give a policy in continuous rather than a policy with time intervals that would only be able to interrupt at 6 seconds.

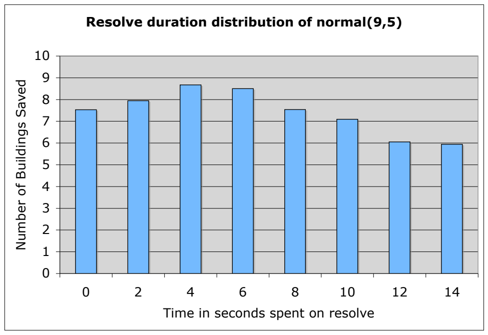
The results of these DEFACTO simulation experiments and the earlier testbed sample policy experiments show the benefits of RIAACT. RIAACT allows for an online hybrid method of reasoning about team-level adjustable autonomy that improves team performance due to: (i) reasoning about the resolving of inconsistencies, (ii) modeling in continuous time, and (iii) planning for the interruptions of actions.



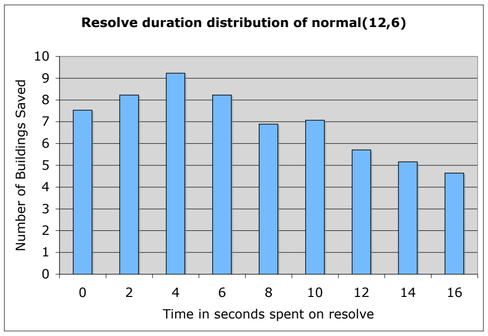
(a) Normal(3,1)



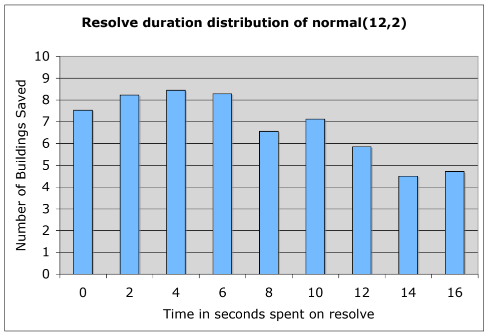
(b) Normal(6,4)



(c) Normal(9,5)



(d) Normal(12,6)



(e) Normal(12,2)

Figure 6.22: Experiments given a simulated human and varying the amount of time before interrupting the *resolve* of an inconsistency.

Chapter 7

Lessons Learned from Initial Deployment Feedback

Through our communication with strategic training division of the LAFD (see Figure 2.4), we have learned a lot of lessons that have influenced the continuing development of our system.

7.1 Adjustable Autonomy in Practice

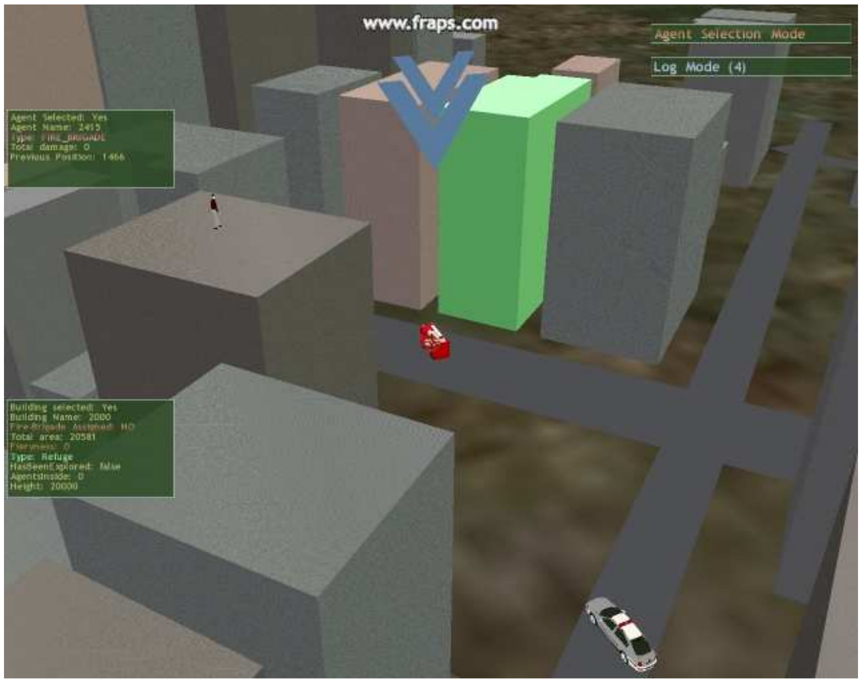
Our most important lesson learned from talking with the LAFD and seeing their exercises is that adjustable autonomy correctly maps over to what happens in the actual disaster response. The adjusting of autonomy is easily seen as the event scales up and down in size and intensity. For a smaller scale response to, for example, a residential single story house fire, the Incident Commander will usually make all allocation decisions and thus practice the A strategy. For a larger scale event, a lot of the burden of most allocations is left to the team and other entities in the hierarchy, while the Incident Commander is left to concentrate on the bigger picture. In this case, the Incident commander is notified of a specifically problematic situation, for example not enough resources to attack a particular fire. This strategy is essentially what we refer to as $A_T H$ in our experiments, in which, the team first tries to assign someone to the fire with the resources they have, and if not able to then pass it off to the Incident Commander for help. As the situation were to die down and the size of the team were to decrease, more autonomy would be shifted to the Incident Commander due to an increased ability to make allocations for the team.

It is very helpful to know that these strategies not only are capable of making our agent teams perform well and interface with the Incident Commander, but that they also reflect similar strategies that current firefighting teams are using. Domain experts [LAFD, 2001] verify that this domain requires the flexible interaction strategies between teams and incident commanders.

7.2 Questioning the Incident Commander

Another lesson that relates to our agent design is that we learned how a team on the ground may possibly not agree with the command (allocation to a fire) given by the Incident Commander. This will usually be due to the fact that the Incident Commander has a broad global view of the disaster, whereas the agents each have a more detailed local view. This mismatch in information can, at times, lead to detrimental team allocations. In an actual disaster response, this is handled by the allocated team both questioning the order and providing the Incident Commander with the missing information.

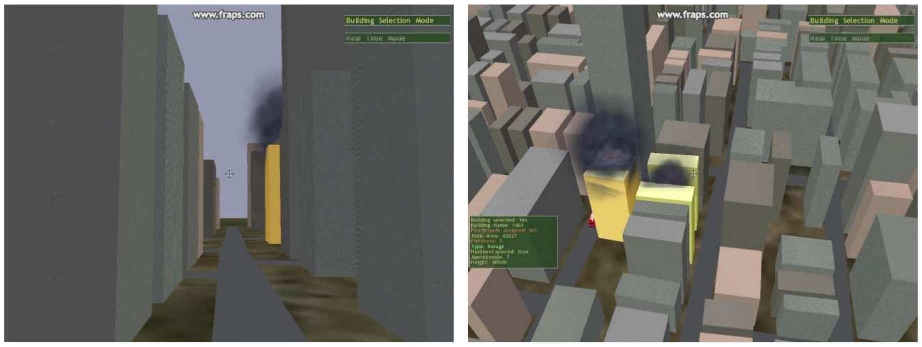
This has led us to consider a team of agents that can disagree with human inputs. This issue has not been addressed in our implementation as of yet, but it is relevant given the results that will be presented later in the training exercise experiments. There are experimental settings in which the team performance would have been improved, had they rejected the Incident Commander's input.



```
Agent Selected: Yes
Agent Name: 2415
Type: FIRE_BRIGADE
Total damage: 0
Previous Position: 1466
```

```
Building selected: Yes
Building Name: 2000
Fire-Brigade Assigned: NO
Total area: 20581
Fieryness: 0
Type: Refuge
HasBeenExplored: false
AgentsInside: 0
Height: 20000
```

Figure 7.1: Selecting for closer look at a Fire Engine.



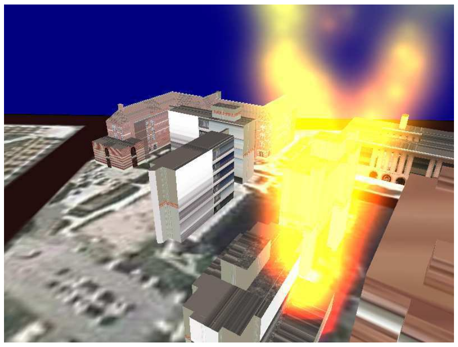
(a) Local Perspective (b) Global Perspective

Figure 7.2: Local vs. Global Perspectives in the Omni-Viewer

7.3 Perspective

Just as in multiagent systems, the Incident commander must overcome the challenge of managing a team, each of whose members possesses only a partial local view. This is highlighted in fighting a fire by incident commanders keeping in mind that there are five views to every fire (4 sides and the top). Only by taking into account what is happening on all five sides of the fire, can the fire company make an effective decision on how many people to send where. Because of this, a local view (see Figure 7.2(a)) can augment the global view (see Figure 7.2(b)) becomes helpful in determining the local perspectives of team members. For example, by taking the perspective of a fire company in the back of the building, the incident commander can be aware that they might not see the smoke from the second floor, which is only visible from the front of the building. The incident commander can then make a decision to communicate that to the fire company or make an allocation accordingly.

The 3D perspective of the Omni-Viewer was initially thought to be an example of a futuristic vision of the actual view given to the incident commander. But after allowing the fire fighters to look at the display, they remarked, that they have such views available to them already, especially in large scale fires (the very fires we are trying to simulate). Often a news helicopter is at the scene and the incident commander can patch into the feed to display it at the command post. Consequently our training simulation can already start to prepare the Incident Commander to incorporate a diverse array of information sources.


(a) Old Fire

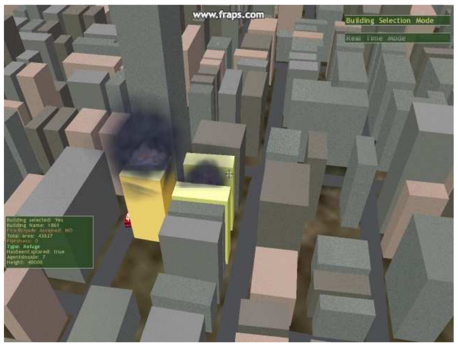
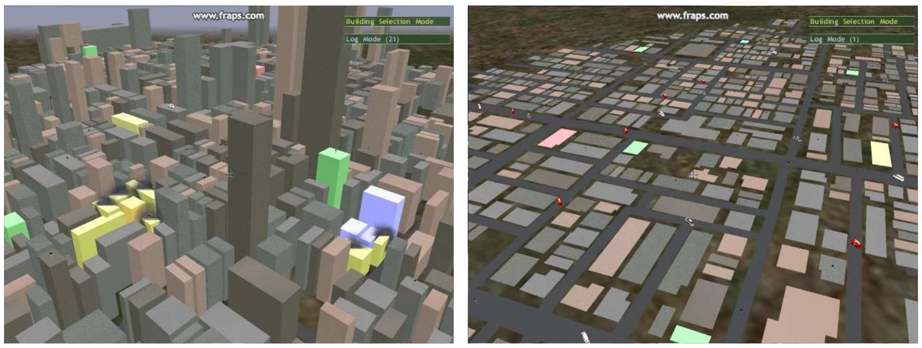

(b) New Smoke

Figure 7.3: Improvement in fire visualization

7.4 Fire Behavior

We also learned how important smoke and fire behavior is to the firefighters in order to affect their decisions. Upon our first showing of initial prototypes to the Incident Commanders, they looked at our simulation, with flames swirling up out of the roof (see Figure 7.3(a)). We artificially increased fire intensity in order to show off the fire behavior and this hampered their ability to evaluate the situation and allocations. They all agreed that every firefighter should be pulled out because that building is lost and might fall at any minute! In our efforts to put a challenging fire in front of them to fight, we had caused them to walk away from the training. Once we start to add training abilities, such as to watch the fire spread in 3D, we have to also start to be more aware of how to accurately show a fire that the Incident Commander would face. We have consequently altered the smoke and fire behavior (see Figure 7.3(b)). The smoke appears less "dramatic" to the a lay person than a towering inferno, but it provides a more effective training environment.



(a) Normal (b) Flat World

Figure 7.4: Improvement in locating resources (fire engines and ambulances)

7.5 Gradual Training

Initially, we were primarily concerned with changes to the system that allowed for a more accurate simulation of what the Incident Commander would actually see. Alternatively, we have also added features, not because of their accuracy, but also to aid in training by isolating certain tasks. Very often in reality and in our simulations, dense urban areas obscure the ability to see where all of the resources (i.e. fire engines) are and prevent a quick view of the situation (see Figure 7.4(a)). To this aim, we have added a new mode using the 3D viewer, but having the buildings each have no height, which we refer to as Flat World (see Figure 7.4(b)). By using this flat view, the trainee is allowed to concentrate on the allocation of resources, without the extra task of developing an accurate world view with obscuring high rise buildings.

7.6 User Intent

A very important lesson that we learned from the LAFD, was that the Incident Commander cannot be given all information for the team and thus the human does not know all about the status of the team members and vice versa. Consequently, this lack of complete awareness of the agent team's intentions can lead to some harmful allocations by the human (Incident Commander). In order for information to be selectively available to the Incident Commander, we have allowed the Incident Commander to query for the status of a particular agent. Figure 7.1 shows an arrow above the Fire Engine at the center of the screen that has been selected. On the left, the statistics are displayed. The incident commander is able to select a particular fire engine and find out the equipment status, personnel status, and the current tasks that are being performed by the fire fighters aboard that engine. This detailed information can be accessed if desired by the Incident Commander, but is not thrown to the screen by all agents, in order to not overwhelm the Incident Commander.

7.7 Training Scale

In addition, we have also learned of new challenges that we are currently attempting to tackle by enhancing the system. One of the biggest challenges in order to start simulating a large urban fire is the sheer scale of the resources that must be managed. According to the fire captains, in order to respond to a single high rise building with a few floors on fire, roughly 200 resources (fire engines, paramedics etc.) would need to be managed at the scene. Coordinating such a large number of agents on a team is a challenge. Also, as the incident scales to hundreds of resources, the Incident Commander ends up giving more autonomy to the team or else faces being overwhelmed. We believe that adjustable autonomy will start to play a bigger and more essential roll in allowing for the Incident Commander to monitor the larger situations.

7.8 Quicker Feedback

In recent discussions with officials from the LAFD, they have pointed out that a very distinct advantage that the simulated training has over their current method is that of quicker feedback. In traditional training, there will be moments of inactivity and a lot of waiting for an incident commander to see the impact of their influence on the disaster response. If the simulation is sped up at this time, they can still see their impact, but much more quickly and can immediately start to analyze what went well and what went poorly. This allows for training exercises to take less time and the training objectives to be more evident to the trainee.

Chapter 8

Conclusion and Vision

8.1 Conclusion

The focus of my thesis is to enable human-multiagent teams to succeed despite having to address real-world constraints including time deadlines, computation limits and the distributed nature of the team. In particular, this thesis focuses on the challenge of applying adjustable autonomy to these human-multiagent teams, in order to increase the performance of the team.

This thesis overall provides four advances to the field that include: (i) defining the new team-level adjustable autonomy problem, (ii) performing human experiments to highlight the challenges of the team-level adjustable autonomy problem, (iii) introducing the RIAACT approach handle these challenges, and (iv) developing the testbed system DEFACTO in order to explore human-multiagent teams.

Specifically, I have shown that RIAACT makes four important contributions in order to address the challenges of team-level adjustable autonomy. Firstly, I have shown the importance of modeling the resolution of inconsistencies between human and agent view in adjustable autonomy. This has led to the creation of new adjustable autonomy strategies that recognize inconsistencies and provide a framework to decide if a resolution is beneficial. Secondly, I have shown the improvements in modeling these new adjustable autonomy strategies using TMDPs (Time dependent Markov Decision Problems). This continuous time based solution provides higher quality solutions while remaining computationally efficient. Thirdly, I have introduced a new model for Interruptible TMDPs (ITMDPs) that allows for an action to be interrupted at any point in continuous time. This results in a more accurate modeling of actions and produces additional time-dependent policies that guide interruption during the execution of an action. Fourthly, I have developed and implemented a hybrid approach that decomposes the team level adjustable autonomy problem in a separate ITMDP for each team decision and leverages other techniques in order to provide a feasible online solution.

Toward these contributions, I have developed a research prototype called DEFACTO (Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence) that implements these approaches. Though DEFACTO has been designed to be used for deployed applications, it is initially being used as a modeling and simulation tool to improve on current training methods in the context of an incident commander during disaster response. Experiments have been conducted with DEFACTO that have both actual humans and simulated humans interacting with the teams of agents. I have given an overview of some of the lessons that were learned as a result in working directly with the Los Angeles Fire Department. Together, these approaches have served as beneficial in enabling human-multiagent teams to work together successfully.

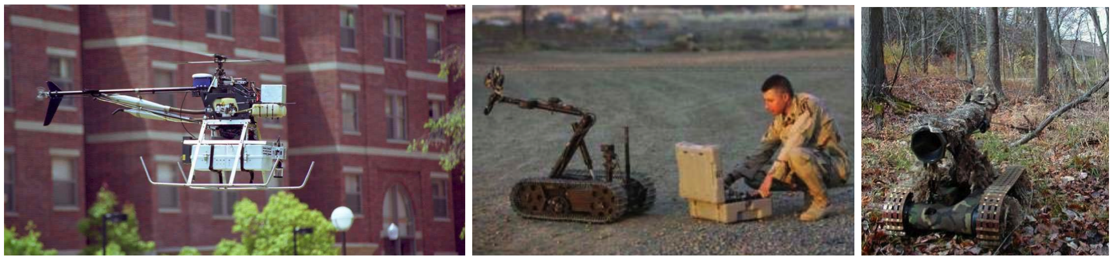
8.2 Vision

There are many possible directions for future research in the area of human-multiagent teams. Human-multiagent teams are being deployed in ever expanding real-world domains, which make the challenges that they bring even more important to address. In Chapter 9, I have placed a short discussion of some of these issues that these human-multiagent teams will present. In particular, I explore how Isaac Asimov's Laws of Robotics can influence the design of these human-multiagent teams of the future.

In addition, there are many possible future directions for research in the specific area of team-level adjustable autonomy. One example of this is to explore the area of multihuman-multiagent teams. This thesis focuses on teams including one human with multiple agents and some of these insights will be applicable, however new challenges will arise when trying to deal with authority and hierarchy with multiple humans. The RIAACT approach may even help prepare for the future where multiple humans will be with the team. I propose that there will be opportunities to leverage these techniques when dealing with humans with conflicting decisions. The humans could employ these resolution strategies among themselves and decide how long to attempt to do so.

Also, a related challenge that will arise is how to approach more heterogeneity in the team, in both the capabilities of the team member and the information that they have access to. One could imagine many intermediate steps between the local and global views described in this thesis. This would create varying information gaps to be addressed.

Another important area of future research is to look at these team-level adjustable autonomy issues in the context of other domains. RIAACT specifically can apply to some of the



(a) AVATAR helicopter (b) TALON with human controller (c) TALON in camouflage

Figure 8.1: The MAAF project will include TALON unmanned ground vehicles and AVATAR autonomous helicopters coordinating with a human controller.

other projects that I have been closely involved with. For example, the Software for Distributed Robotics (SDR) project aimed to coordinate a large group of robots (around 100) during a sustained activity of surveillance and continual recharging. The project would have benefited from allowing the large team and the human user to work closely together and resolve inconsistencies. For more details on this domain, please refer to [Schurr et al., 2003].

Another closely related current project is the Multiagent Adjustable Autonomy Framework (MAAF). This project has been developed to support multi-robot, multi-human teams during tactical maneuvers (see Figure 8.1). This is a very exciting project because it will soon extend past simulation and start to address the challenges of applying adjustable autonomy to the real world. Furthermore, it will include a heterogeneous team of unmanned ground vehicles, TALONs (see Figure 8.1 b and c), and air vehicles, AVATARs (see Figure 8.1-a), along with the human controller.

I hope that by looking into these future research areas, these new human-multiagent teams will become important and useful fixtures in our society.

Chapter 9

Discussion

In this chapter, I discuss some of the issues that I propose will arise in the future as more human-multiagent teams are deployed in the real world. I find these new challenges exciting and this area is overflowing with potential for future research. I believe that a lot of these forward-thinking research topics can be inspired by or even influenced by some of the science fiction literature that has begun to explore some of these issues. A prime example of this is the work of Isaac Asimov. In particular, his books concerning human-robot relations and the Laws of Robotics are very relevant [Asimov, 1990]. I provide some very initial experiments that serve to show some of the benefits of casting the general human-multiagent coordination problem in this manner.

In deploying multiagent teams that can function as teammates with humans, it is crucial to provide some kind of guarantees about their behavior. These multiagent teams have been designed to perform well on their own and wish to improve performance by interacting with a human. But there is a risk that the human will unintentionally degrade the team performance. This can be due to a lack of complete detailed knowledge of the whole team's situation. The challenge is then how to provide the guarantees that will make certain that human involvement will not cause performance to degrade significantly.

I propose a method for counteracting potentially harmful human input whereby the agents communicate locally and determine if local performance will suffer greatly. If that is so, then the team must let the human know. In this chapter, I will first introduce the Asimovian Laws as a potential framework for the guarantees. Then I will apply this local method for detecting degradation and maintaining guarantees to Asimov's Laws and show some initial results of an implementation.

9.1 Asimovian Guarantees for Human Agent Teams

In this chapter, I will focus on Asimov's three laws of robotics from his science-fiction stories that provide us a starting point for such behavior guarantees. We do not claim that these laws are the only or the best collection of similar rules. However, the laws outline some of the most fundamental guarantees for agent behaviors, given their emphasis on ensuring that *no harm* comes to humans, on obeying human users, and ensuring protection of an agent. Indeed, these laws have inspired a great deal of work in agents and multiagent systems already [Weld and Etzioni, 1994; Gordon, 2000; Pynadath and Tambe, 2001]. However, in operationalizing these laws in the context of multiagent teams, three novel issues arise. First, the key notions in these laws (e.g. "no harm" to humans) are specified in very abstract terms and must be specified in concrete terms in implemented systems. Second, while the laws were originally written for interaction of an individual robot and an individual human, clearly, our systems must operate in a team context. Third, since, in many realistic domains, agents or humans may not have perfect information about the world, they must act based on these laws despite information uncertainty and must overcome their mutual information mismatch.

Indeed, as mentioned earlier, researchers have in the past advocated the use of such laws to provide guarantees in agent systems [Weld and Etzioni, 1994; Gordon, 2000; Pynadath and Tambe, 2001]. However, previous work only focused on a single law (the first law of safety) and in the process addressed two of the issues mentioned above: defining the notion of harm to humans and applying the laws to teams rather than individual agents. The key novelty of our work is going beyond previous work to consider the second of Asimov's laws, and more importantly in recognizing the fundamental role that uncertainty plays in any faithful implementation of such a law. In particular, Asimov's second law addresses situations where an agent or agent team may or may not obey human orders — it specifies that in situations where (inadvertent) harm may come to other humans, agents may disobey an order. However, in the presence of uncertainty faced either by the agents or the human user about each others' state or state of the world, either the set of agents or the human may not be completely certain of their inferences regarding potential harm to humans. This chapter illustrates that in the presence of such uncertainty, agents must strive to gather additional information or provide additional information. Given that the information reduces the uncertainty, agents may only then disobey human orders to avoid harm.

To the best of our knowledge, this is the first time a concrete implementation has been provided that addresses the three key issues outlined above in operationalizing Asimov's laws. As mentioned earlier, our implementation is focused on that of disaster rescue simulations. The real-time nature of this domain precludes the use of computationally expensive decision-theoretic techniques, and instead agents rely on heuristic techniques to recognize situations that may (with some probability) cause harm to humans.

9.2 On Asimov's Laws[1]

In 1942, having never had any personal contact with a robot, Isaac Asimov sat down to write his short story "Runaround" [Asimov, 1990] and in doing so enumerated for the first time his three laws of robotics:

- *First Law: A robot may not injure a human being, or, through inaction, allow a human being to come to harm.*
- *Second Law: A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.*
- *Third Law: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.*

Asimov believed these three laws were both necessary and sufficient for human-robot interaction, an idea he set out to illustrate in his series of robot stories. While he believed that enforcing his three laws would prevent a robot from becoming the nightmarish monster of Frankenstein that was the fodder of many science fiction stories, Asimov admitted that the operationalization of his three laws would not be simple or unambiguous. In this chapter, I focus on operationalization of the first two laws, which requires several key issues be addressed in concretely applying them to our domains of interest: (i) Providing definition of "harm" so central to the first law; (ii) Applying these laws in the context of teams of agents rather than individuals; and (iii) Addressing these laws in the presence of uncertainty in both the agent's and the human user's information about each other and about the world state. Previous work has only focused on the first law, and thus on techniques to avoid harm via agents' actions [Gordon, 2000; Pynadath and Tambe, 2001;

[1]This section was worked on in joint with Emma Bowring and Pradeep Varakantham.

Weld and Etzioni, 1994]. This previous work dealt with both a single agent and a team of agents, but the emphasis remained on the autonomous actions of these agents. In contrast, the second law emphasizes interactions with humans, and thus its relevance in the context of heterogeneous systems that involve both humans and multiagent teams, that are of interest in this thesis.

Indeed, among the issues that must be addressed in concretely applying these laws, the first two — defining harm and applying the laws to teams instead of individuals — are addressed in previous work (albeit differently from our work). However, the uncertainty of information that the agents and the human user may suffer from is the novel issue that must be clearly addressed when addressing the second law. In the following, I provide a more detailed discussion of these three issues, with an emphasis on the issue of uncertainty. Nonetheless, in contrast with previous work, this thesis is the first (to the best of my knowledge) that addresses these three issues together in operationalizing the two laws.

9.2.1 Definition of Harm

What constitutes harm to a human being? Must a robot obey orders given it by a child, by a madman, by a malevolent human being? Must a robot give up its own expensive and useful existence to prevent a trivial harm to an unimportant human being? What is trivial and what is unimportant? pg 455 [Asimov, 1990]

The notion of harm is fundamental to Asimov's laws. Yet, Asimov himself did not imply that harm had to be necessarily physical harm to humans. Indeed, in the story "LIAR" harm is purely mental harm (e.g. someone not getting a promotion they wanted) [Asimov, 1990]. So whereas the notion of harm as physical harm to humans is obviously relevant in one of our domains mentioned earlier (disaster rescue), it is also relevant in an office assistant domain, where harm may imply harm to some business (e.g. products not delivered on time) where the office assistant team is deployed. Indeed, in previous work in software personal assistants that is motivated by Asimov's laws [Weld and Etzioni, 1994; Pynadath and Tambe, 2001], the notion of harm includes such effects as deletion of files or meeting cancellation.

In this chapter, the notion of harm is operationalized as a "significant" negative loss in utility. So if actions cause a significant reduction in an individual agent's or team's utility, then that is considered as constituting harm. In our disaster rescue simulation domain, such negative utility accrues from loss of (simulated) human life or property. An example of this can be seen in Figure 5.3 when human inputs are followed by 6 fire engine agents, resulting in more buildings being burned than if the inputs were ignored.

9.2.2 Applying Laws to Teams

I have dealt entirely with the matter of the interaction between [a] single robot and various human beings. ... Suppose two robots are involved, and that one of them, through inadvertence, lack of knowledge, or special circumstances, is engaged in a course of action (quite innocently) that will clearly injure a human being – and suppose the second robot, with greater knowledge or insight, is aware of this. pg. 479-480 [Asimov, 1990]

Diana Gordon-Spear's work on *Asimovian agents*[Gordon, 2000] addresses teams of agents that guarantee certain safety properties (inspired by the first law above) despite adaptation or learning on part of the agent team. The key complications arise because the actions of multiple agents interact, and thus in preserving such safety property, it is not just the actions of the individual, but their interactions that must be accounted for, in terms of safety. In our work (particularly as seen in the disaster response domain of Sections 2.2 and 9.4), similar complexities arise when applying the laws to teams of agents. No single individual may be able to detect harm by itself; rather the harm may only be detectable when the team of agents is considered as a whole.

9.2.3 Uncertainty

Even a robot may unwittingly harm a human being, and even a robot may not be fast enough to get to the scene of action in time or skilled enough to take the necessary action. pg 460 [Asimov, 1990]

The second law in essence requires that agents obey human orders unless such orders cause harm to (other) humans. Thus, this law opens up the possibility that the agent may disobey an order from a human user, due to the potential for harm. In many previous mixed agent-human systems, human inputs are considered final, and the agent cannot override such inputs — potentially with dangerous consequences as shown earlier. Asimov's second law anticipates situations where agents must indeed override such inputs and thus provides a key insight to improve the performance of agents and agent teams.

Yet, the key issue is that both the agents and the human users have uncertainty; simply disobeying an order from a human given such uncertainty may be highly problematic. For example, the agents may be uncertain about the information state of the humans, the intellectual or physical capability of the human users, and the state of the world, etc. In such situations, agents may be uncertain about whether the current user order may truly cause (inadvertent) harm to others. It is also feasible that the human user may have given an order under a fog of uncertainty about the true world state that the agent is aware of; and in such situations, the agents' inferences about harmful effects may be accurate.

One of the key insights in this thesis then relates to addressing situations under the second law where an agent may disobey human orders due to its potential for causing harm to other humans: given the uncertainty described above, an agent should not arbitrarily disobey orders from humans, but must first address its own or the human users' uncertainty. It is only upon resolution of such uncertainty that the agent may then disobey an order if it causes harm to others.

In addressing the simulated disaster rescue domain, the issue centers on potential uncertainty that a human user must face. Here, an agent team acting in the simulated disaster-rescue environment potentially has more certainty about the world state than the human user does. Unfortunately, the disaster is spreading rapidly, and the agent team may have little time to deliberate on the uncertain situation, and come up with a detailed policy (as with a POMDP). Instead, the agent team quickly brings up to the notice of the human user key possible sources of potential harm due to the human's order. At this juncture, the human user may be able to reduce his/her uncertainty about the world state, and thus possibly rectify his/her order.

9.3 Operationalizing Asimov's Laws

In the initial adjustable autonomy strategy experiments, appropriately obeying the first and second laws would have improved the situation. Specifically in the second law, the caveat where human directives should be followed, *unless it causes harm to humans*, is not being paid attention to. Instead, as mentioned in Chapter 2, agents blindly obey human commands, which is problematic. However, as mentioned earlier, in complex domains, there is significant uncertainty. Given such uncertainty, it is quite feasible for the humans to provide imperfect inputs or orders; yet agents may not be certain that these orders are in error, due to the uncertainty that they face. Our position is that in order to start constructing teams of agents and humans that perform well, they must not always take human input as final, yet must only do so after resolving uncertainty.

Given that humans may (unintentionally) provide problematic input, I propose that there are 5 general categories of agents' reactions to problematic human input. In particular, agents may:

- A. Follow the human input exactly
- B. Ignore the human input completely
- C. Make the human aware of the alleged problem in the input
- D. Make the human aware of the alleged problem in the input and offer non-problematic option(s)
- E. Limit human input to only pre-defined non-problematic options to be chosen from by the human

Due to the uncertainty (mentioned in Section 9.2.3), Options A and B become infeasible. Option A results in suboptimal performance and does not take advantage of the team's resources and potential as seen in Chapter 2. Option B may end up in better performance, but not only results in angry or confused humans, the agents may also be mistaken due to its own uncertainty and poor performance. Option E can be very difficult for dynamic domains, because there are so many options to explore and it is challenging to identify the appropriate choices to offer the human. It is then desirable to engage in some of the dialogue described in Options C or D. Our aim is to have the joint performance of the agents and humans result in an improvement over either of them separately, that is to have the agents correct problems of humans and vice versa.

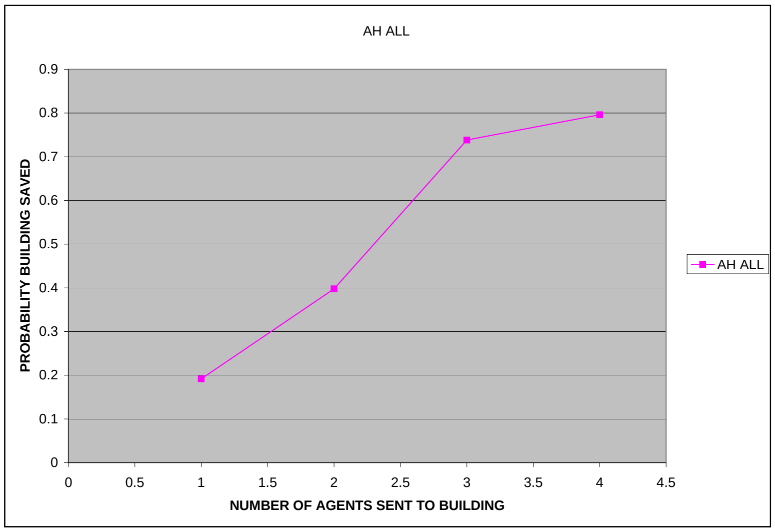


Figure 9.1: AH for all subjects.

9.4 Applying Laws for Guarantees in Disaster Response

Our goal was to have the agent team be able to detect the problematic input seen in the previous experiments and then be able to engage in some type of dialogue with the human user. In order to do this, I continued an in depth analysis of what exactly was causing the degrading performance when 6 agents were at the disposal of the human user. Figure 5.5 shows the number of agents on the x-axis and the average amount of fire engines allocated to each fire on the y-axis. AH and $A_T H$ for 6 agents result in significantly fewer average fire engines per task (fire) and therefore lower average. For example, as seen in Figure 5.5, for the $A_T H$ strategy, subject 3 averaged 2.7 agents assigned to each fire when 4 agents were available, whereas roughly 2.0 agents were assigned to each fire when 6 agents were available. It seems counterintuitive that when given more agents, the average number that was assigned to each fire actually went down. Furthermore, this lower average was not due to the fact that the human user was overwhelmed and making fewer decisions (allocations). Figures 9.3(a), 9.3(b), and 9.3(c) all show how the number of buildings attacked

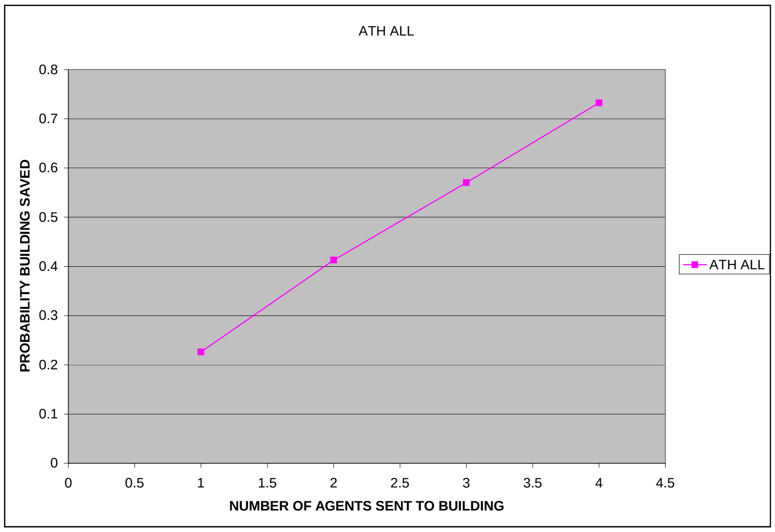
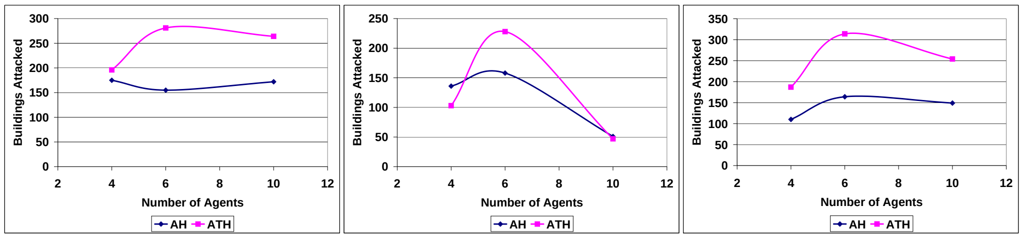


Figure 9.2: ATH for all subjects.



(a) Subject 1 (b) Subject 2 (c) Subject 3

Figure 9.3: Number of buildings attacked.

does not go down in the case of 6 agents, where poor performance is seen. Figures 9.1 and 9.2 show the number of agents assigned to a building on the x-axis and the probability that the given building would be saved on the y-axis. The correlation between these values demonstrate the correlation between the number of agents assigned and the quality of the decision.

We can conclude from this analysis that the degradation in performance occurred at 6 agents because fire engine teams were split up, leading to fewer fire-engines being allocated per building on average. Indeed, leaving fewer than 3 fire engines per fire leads to a significant reduction in fire extinguishing capability.

9.5 Domain Independent

Up until now, the analysis and deduction of what was causing the harm (performance degradation) was, for the most part, domain dependant. However, I have started to develop some domain independent methods for detecting a decrease in performance. We cannot view the problem of detecting performance degradation from the perspective of traditional one-on-one tutoring systems, where an agent has access to the same information as the human trainee and can thus provide immediate inputs when its inference of what is right contradicts those of the trainee. In our work, however, agents have access to far more limited, but detailed information that is local, and the human has access to global but abstracted information. This is especially so, in the long term where we envision a deployed system in the real-world. Given this dissimilarity of information, agents can perform a local check to detect if a local evaluation suggests that the performance of their team has degraded. If local evaluation suggests a degradation beyond a certain bound, then that may trigger their attempts to resolve the situation with the human user.

In many complex systems, agents use a locally optimal algorithm to perform resource allocation (i.e. allocation of roles to agents). For example, in DEFACTO, as mentioned in Chapter 4, we use LA-DCOP, a locally optimal DCOP (Distributed Constraint Optimization) algorithm to allocate fire-engines to fires – a globally optimal algorithm is often unusable in such complex dynamic domains. LA-DCOP settles on a local equilibrium when allocating fire-engines to fires. When human inputs are received, agents may locally check if the quality of the solution

Reject Orders to Split?	Buildings Damaged	Fires Extinguished
No	27	3
No	29	3
No	33	1
Yes	14	5
Yes	18	5
Yes	20	5

Table 9.1: Benefits to team when rejecting orders allows split of team. In top half, team accepted all human orders, and in bottom half, problematic orders were rejected.

has locally improved. Local improvement is feasible, given that they were at a local equilibrium. However, if that new local DCOP has a substantially less quality (reward) than the current DCOPs reward, then the agent must let the human know so that this can be resolved. By resolve, I mean that either (i) the human is made aware of some local changes and determines to retract the input or (ii) the human was aware of that local decrease in reward, but sees the overall team performing better in the long run.

9.6 Remedy

Given this domain independent method for predicting human input to be problematic, I implemented the ability for the agent team to detect if a reallocation is pulling a teammate from a working group of 3 or more. Once this is detected, there is a high probability that the team performance will be degraded by following the human input. But since there is some uncertainty in the final outcome, the agents do not blindly follow (Option A from above) or ignore (Option B from above). Instead they present the possible problem to the human (Option C from above).

In order to evaluate this implementation I set up some initial experiments to determine the possible benefits of the team being able to reject the splitting of a coordinated subgroup. We used a team comprised of a human user and 6 agents. We used the same map and the same $A_T H$ strategy as were used in previous experiments. Each of these results were from a short (50 time step) run of the DEFACTO system. The only variable was whether the agents were allowed to raise objections when given allocation orders to split up. In these experiments, the human user listened to all agent objections and did not override them. The results of this initial experiment can be seen in Table 9.1. In Table 9.1, I present results for three problem instances. Performance is measured by calculating the amount of buildings damaged (less is better) and the number of fires extinguished (more is better). As seen from the number of buildings damaged, by allowing the agents to reject some of the human input (see bottom half of Table 9.1), they were able to more easily contain the fire and not allow it to spread to more buildings.

Bibliography

Advanced Systems Technology. Epics - emergency preparedness incident commander simulation. In *http://epics.astcorp.com*, 2005.

James F. Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom. Plow: A collaborative task learning agent. In *AAAI*, pages 1514–1519, 2007.

Isaac Asimov. *Robot Visions (collection of robot stories)*. Byron Preiss Visual Publications Inc, 1990.

Jeremy W. Baxter and Graham S. Horn. Controlling teams of uninhabited air vehicles. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of MDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence(UAI-00)*, pages 32–37, 2000.

Craig Boutilier, Raymond Reiter, Mikhail Soutchanski, and Sebastian Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 355–362, 2000.

J. Boyan and M. Littman. Exact solutions to time-dependent MDPs. In *NIPS*, pages 1026–1032, 2000.

CALO: Cognitive Agent that Learns and Organizes. CALO http://www.ai.sri.com/project/CALO, http://calo.sri.com, 2003.

William J. Clancey, Maarten Sierhuis, Charis Kaskiris, and Ron van Hoof. Advantages of brahms for specifying and implementing a multiagent human-robotic exploration system. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, pages 7–11, 2003.

Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2–3):213–261, 1990.

J. Collins, C. Bilot, M. Gini, and B. Mobasher. Mixed-initiative decision-support in agent-based automated contracting. In *Proceedings of the International Conference on Autonomous Agents (Agents)*, 1998.

Jacob W. Crandall, Curtis W. Nielsen, and Michael A. Goodrich. Towards predicting robot team performance. In *SMC*, 2003.

G. Dorais, R. Bonasso, D. Kortenkamp, P. Pell, and D. Schreckenghost. Adjustable autonomy for human-centered autonomous systems on mars. In *Mars*, 1998.

Z. Feng, R. Dearden, N. Meuleau, and R. Washington. Dynamic programming for structured continuous MDPs. In *UAI*, 2004.

G. Ferguson and J. Allen. Trips : An intelligent integrated problem-solving assistant. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI)*, pages 567–573, 1998.

G. Ferguson, J. Allen, and B. Miller. Trains-95 : Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems*, pages 70–77, 1996.

Terrence Fong, Charles Thorpe, and Charles Baur. Multi-robot remote driving with collaborative control. *IEEE Transactions on Industrial Electronics*, 2002.

M. Goodrich, D. Olsen, J. Crandall, and T. Palmer. Experiments in adjustable autonomy. In *IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, 2001.

Michael A. Goodrich, Timothy W. McLain, Jeffrey D. Anderson, Jisang Sun, and Jacob W. Crandall. Managing autonomy in robot teams: observations from four experiments. In *Proceedings of the Second ACM SIGCHI/SIGART Conference on Human-Robot Interaction, HRI*, pages 25–32, 2007.

Diana F. Gordon. Asimovian adaptive agents. *Journal of Artificial Intelligence Research (JAIR)*, 13:95–153, 2000.

C. Guestrin, M. Hauskrecht, and B. Kveton. Solving factored mdps with continuous and discrete variables, 2004.

R. Hill, J. Gratch, W. L. Johnson, C. Kyriakakis, C. LaBore, R. Lindheim, S. Marsella, D. Miraglia, B. Moore, J. Morie, J. Rickel, M. Thiebaux, L. Tuch, R. Whitney, J. Douglas, and W. Swartout. Toward the holodeck: integrating graphics, sound, character and story. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 409–416. ACM Press, 2001.

Randall W. Hill, Johnny Chen, Jonathan Gratch, Paul Rosenbloom, and Milind Tambe. Intelligent agents for the synthetic battlefield: A company of rotary wing aircraft. In *Proceedings of Innovative Applications of Artificial Intelligence (IAAI)*, 1997.

Eric Horvitz and Johnson Apacible. Learning and reasoning about interruption. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 20–27, New York, NY, USA, 2003. ACM Press.

US JFCOM Joint Warfighting Center. Jcats - joint conflict and tactical simulation. In *http://www.jfcom.mil/about/fact_jcats.htm*, 2005.

Samin Karim and Clint Heinze. Experiences with the design and implementation of an agent-based autonomous uav controller. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsushi Shinjoh, and Susumu Shimada. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE SMC*, volume VI, pages 739–743, Tokyo, October 1999.

LAFD. High-rise incident command system. In *Los Angeles Fire Department*, 2001.

L. Li and M. Littman. Lazy approximation for solving continuous finite-horizon MDPs. In *AAAI*, pages 1175–1180, 2005.

Janusz Marecki, Zvi Topol, Sven Koenig, and Milind Tambe. Coordinators autonomy module. In *Coordinators Project Technical Report*, 2006.

Janusz Marecki, Sven Koenig, and Milind Tambe. A fast analytical algorithm for solving markov decision processes with real-valued resources. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, January 2007.

Mausam, Emmanuel Benazera, Ronen I. Brafman, Nicolas Meuleau, and Eric A. Hansen. Planning with continuous resources in stochastic domains. In *IJCAI*, pages 1244–1251, 2005.

Karen L. Myers and David N. Morley. Policy-based agent directability. In *Agent Autonomy (ed., Henry Hexmoor, Cristiano Castelfranchi and Rino Falcone)*. Kluwer Academic Publishers, 2003.

Ranjit Nair and Milind Tambe. Hybrid bdi-pomdp framework for multiagent teaming. *Journal of Artificial Intelligence Research (JAIR)*, 23:367–420, 2005.

Sean Owens, Paul Scerri, Robin Glinton, Bin Yu, and Katia Sycara. Synergistic integration of agent technologies for military simulation. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1443–1444, New York, NY, USA, 2006. ACM Press.

A. Paivio. Pictures and words in visual search. *Memory & Cognition*, 2(3):515–521, 1974.

M. Petrik. An analysis of laplacian methods for value function approximation in mdps. In *IJCAI*, pages 2574–2579, 2007.

M. Puterman. *Markov decision processes*. John Wiley and Sons, New York, 1994.

D. V. Pynadath and M. Tambe. Automated teamwork among heterogeneous software agents and humans. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 7:71–100, 2003.

D V. Pynadath and Milind Tambe. Revisiting asimov's first law: A response to the call to arms. In *Intelligent Agents VIII Proceedings of the International workshop on Agents, theories, architectures and languages (ATAL'01)*, 2001.

David V. Pynadath, Milind Tambe, Nicolas Chauvat, and Lawrence Cavedon. Toward team-oriented programming. In *Proceedings of Agent Theories, Architectures, and Languages Workshop*, pages 233–247, 1999.

J. Reitsema, Wendell Chun, Terrence W Fong, and Randy Stiles. Team-centered virtual interactive presence for adjustable autonomy. In *American Institute of Aeronautics and Astronautics (AIAA) Space 2005*, September 2005. AIAA-2005-6606.

Charles Rich and Candace L. Sidner. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3-4):315–350, 1998.

A. Richardson, D. Montello, and M. Hegarty. Spatial knowledge acquisition from maps and from navigation in real and virtual environments. *Memory and Cognition*, 27(4):741–750, 1999.

R. Ruddle, S. Payne, and D. Jones. Navigating buildings in desktop virtual environments: Experimental investigations using extended navigational experience. *J. Experimental Psychology - Applied*, 3(2):143–159, 1997.

P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *AAMAS*, 2003.

Paul Scerri, Alessandro Farinelli, Stephen Okamoto, and Milind Tambe. Allocating tasks in extreme teams. In *Proceedings of the international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

Jesse Schell. Haz-Mat Hotzone. In *http://www.etc.cmu.edu/projects/hazmat/*, Spring 2007.

Nathan Schurr, Paul Scerri, and Milind Tambe. Impact of human advice on agent teams: A preliminary report. In *Workshop on Humans and Multi-Agent Systems at AAMAS*, 2003.

Nathan Schurr, Janusz Marecki, Paul Scerri, J. P. Lewis, and Milind Tambe. The DEFACTO System: Training Tool for Incident Commanders. In *The Seventeenth Innovative Applications of Artificial Intelligence Conference (IAAI)*, 2005.

Martijn Schut, Michael Wooldridge, and Simon Parsons. Reasoning about intentions in uncertain domains. In *ECSQARU '01: Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 84–95, London, UK, 2001. Springer-Verlag.

Brennan Peter Sellner, Frederik Heger, Laura Hiatt, Reid Simmons, and Sanjiv Singh. Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE - Special Issue on Multi-Robot Systems*, 94(7):1425 – 1444, July 2006.

M. Sierhuis, J. Bradshaw, A. Acquisti, R. Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice. In *In Proceedings of the seventh International Symposium on AI, Robotics and Automation in Space. Nara, Japan, 2003.*, 2003.

William Swartout and Michael van Lent. Making a game of system design. *Communications of ACM*, 46(7):32–39, 2003. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/792704.792727.

M. Tambe, E. Bowring, H. Jung, G. Kaminka, R. Maheswaran, J. Marecki, P. J. Modi, R. Nair, S. Okamoto, J. P. Pearce, P. Paruchuri, D. Pynadath, P. Scerri, N. Schurr, and P. Varakantham. Conflicts in teamwork: hybrids to the rescue. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 3–10. ACM Press, 2005.

Willem A. van Doesburg, Annerieke Heuvelink, and Egon L. van den Broek. Tacop: A cognitive agent for a naval training simulation environment. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

P. Varakantham, R. Maheswaran, and M. Tambe. Exploiting belief bounds: Practical pomdps for personal assistant agents. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

Thomas Wagner, John Phelps, Valerie Guralnik, and Ryan VanRiper. COORDINATORS: Coordination Managers for First Responders. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2004.

D. Weld and O. Etzioni. The first law of robotics: A call to arms. In *AAAI*, Seattle, Washington, 1994. AAAI Press.

H. Younes and R. Simmons. Solving generalized semi-MDPs using continuous phase-type distributions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence AAAI*, pages 742–747, 2004.

Michael Zyda. From visual simulation to virtual reality to games. *IEEE Computer, September Issue*, 2005.

Michael Zyda, Alex Mayberry, Jesse McCree, and Margaret Davis. From viz-sim to vr to games: How we built a hit game-based simulation. In *Organizational Simulation: From Modeling and Simulation to Games and Entertainment*, 2005.