Planning for Human-Robot Interaction: Representing Time and Human Intention

Allison Bruce
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
abruce@andrew.cmu.edu

June 22, 2005

Committee members:
Reid Simmons (co-chair)
Illah Nourbakhsh (co-chair)
Geoff Gordon
Sven Koenig
Abstract

This thesis proposes a novel approach to planning for a specific class of human-robot interaction domains: those in which robots engage in tasks with humans that are governed by social conventions. When humans perform these social tasks, they try to achieve their own goals in an environment that they share with other people. Because the goals all participants are independent, the interaction is neither purely cooperative nor adversarial in nature. However, the actions of others may have a direct impact on each person’s ability to achieve their goals. Social conventions exist as a guideline for how to interact with others so that all parties involved can achieve their goals efficiently without interfering with one another. Recognizing what goals others are trying to achieve and performing actions at the appropriate time in the interaction are critical abilities for social competence.

Adding a robot into these systems without upsetting the social equilibrium is challenging. Our approach to this problem focuses on creating more accurate models of social tasks. Because the human participants are modeled as a part of the environment, the world state in these problems is dynamic and partially observable. Specifically, we model the intentions of the human as hidden state and explicitly model the time-dependence of action outcomes for both the human and the robot. We introduce the partially observable generalized Markov decision process (POGSMDP), an extension of the GSMDP model that includes hidden state. We explain a general framework for modeling human behaviors and the dynamics of the environment as a POGSMDP, transforming the model into a partially observable Markov decision process (POMDP) approximation, and solving for a policy for the robot. Once a policy is obtained, it is verified against the model to ensure that it does not throw the social system out of equilibrium.

This planning technique provides a general approach to planning for a variety of domains in which social conventions govern behavior. The utility of this approach will be demonstrated by using it to implement a controller for a mobile robot that rides elevators with people and an agent that interacts with human players in an online role-playing game (RPG). Performance will be evaluated by comparing the policies to policies developed using less expressive models, and by evaluating objective performance criteria and people’s subjective responses to interacting with the robot or agent. We believe that POGSMDP-based policies will both outperform competing approaches in terms of task performance and result in interactions that are more acceptable to people.
Contents

1 Introduction ......................................................... 5
   1.1 Motivation ....................................................... 6

2 Background ......................................................... 9
   2.1 Psychology ....................................................... 9
   2.2 Planning and multi-agent systems .......................... 9
   2.3 Markov models ................................................ 11

3 Related Work ..................................................... 13
   3.1 Systems for social interaction .............................. 13
      3.1.1 Human-robot interaction ............................... 13
      3.1.2 Behavior/Plan recognition and dialog management 13
   3.2 Multi-agent Planning ........................................ 14
   3.3 Markov models for decision-theoretic planning .......... 15
      3.3.1 Approximate POMDP solution methods ............... 15
      3.3.2 Semi-Markov and time-dependent models .......... 15

4 Technical Approach .............................................. 17
   4.1 Problem ......................................................... 17
   4.2 General framework .......................................... 19

5 Previous Work .................................................... 22
   5.1 Social interaction with a mobile robot .................. 22
   5.2 People tracking with a goal-based motion model ....... 22
5.3 Representation of time in POMDPs ........................................ 25

5.4 Elevator riding domain ....................................................... 27
  5.4.1 POMDP model ............................................................. 27
  5.4.2 Elevator simulator ....................................................... 30
  5.4.3 Results ................................................................. 33

6 Proposed Research ............................................................ 36
  6.1 Expected contributions ................................................... 36
  6.2 Elevator riding domain ................................................... 36
  6.3 Multiplayer game domain ................................................. 37

7 Schedule ................................................................. 38
1 Introduction

In human-robot interaction, both humans and robots share a common environment in which they are trying to perform tasks to achieve their goals. In many such interactions, each agent (human or robot) is pursuing its own goal, rather than working in collaboration to achieve a common one. At the same time, the interactions are not of a competitive or adversarial nature. However, because of the shared environment, the ability of each agent to achieve its goal may be significantly dependent on the behavior of the other agents. When humans interact with each other under these conditions, the social regulation of behavior helps such interactions go smoothly. Much of this social protocol relies on recognizing what task other people are attempting to perform and performing your part of the protocol in turn. Many everyday tasks fall into this category, including riding elevators, standing in line, and navigating crowded hallways.

The primary goal of this research is to produce a practically useful computational model of this interplay for the purpose of controlling an agent in socially situated tasks. More generally, this work suggests a novel approach to modeling and planning for a particular kind of multi-agent system: one in which self-interested agents pursue their own goals in a shared environment while following some set of guidelines for behavior. In most cooperative and many adversarial domains, the goals of the other agents are known. In domains where the agents are pursuing one of a number of possible goals, the intent of another agent may not immediately be clear. By observing their actions, their intent can be inferred, which will improve an agent’s ability to plan to achieve its own goal.

The planning paradigm on which this work is based is decision-theoretic planning. Decision-theoretic planning is concerned with obtaining policies for acting in situations where there is uncertainty over the outcome of actions modeled as probability distributions over those outcomes. The criteria that decision theory uses to evaluate the quality of policies is the amount of reward obtained. Decision-theoretic approaches have proven themselves useful in a variety of real world planning domains, particularly in robotics, because of the flexibility of defining agents’ goals in terms of reward and the ability to model the uncertainty of action outcomes and observations that so often arises in physical domains.
In this proposal, a general framework for modeling and planning for social interactions is introduced. The intentions that direct people's behaviors are modeled as hidden state. Also, the time dependency of action outcomes are explicitly modeled. The hypothesis of this work is that modeling social interaction by treating human behavior behavior as intentional and goal-oriented and modeling the time-dependent aspects of a dynamic environment achieves better results when compared to decision-theoretic models that treat people as just another random part of the environment. The superiority of this modeling technique will be demonstrated in two ways:

- When global reward is based on both the robot and the humans achieving their goals, the policies will obtain higher reward.
- The policies produced will be subjectively more acceptable to people who interact with the agent.

The approach proposed in this thesis was inspired by observing the characteristics of the problem of planning for a robot to effectively navigate in populated spaces and perform goal-oriented tasks. Our experience with these problems convinced us that the most common ways of modeling problems using decision-theoretic planning formalisms were inadequate to express their fundamental characteristics. The proposed approach explicitly addresses both the time-dependence of action outcomes and the partial observability that arises from not knowing the intentions of other agents in the environment. The utility of this approach will be evaluated by using it to implement a controller for a mobile robot that rides elevators with people and an agent that interacts with human players in a video game.

1.1 Motivation

Social interaction seems mundane because of its regularity, but it is often deceptively simple. It is an interaction between two or more intelligent agents with separate goals. These situations are often incredibly difficult planning problems, because of the infinite regress that can occur when one agent may change its behavior based on changes in the behavior of the other agent. But the constraints on “allowable” behaviors in social interaction make the interactions manageable to model.

Our interest is in domains in which the other agents are all following policies that are constrained, either implicitly or explicitly, by the domain-specific guidelines for behavior that govern their interaction with the environment and other agents. There are a number of different goals corresponding to tasks in the domain that an agent may have at any given time. For each goal, there is a common way of achieving it. This can be thought of as guidelines for action that form a widely agreed upon, heuristically ”good”, policy for achieving the goal. These guidelines define what kind of behavior is desirable at a certain time and place
and what is not. They include instructions on how to interact with and respond to the other agents in the environment, assuming that they are also following guidelines for achieving one of the commonly known goals.

In situations where the agents are people and the goals are achievement of everyday tasks, these sets of guidelines and their interactions are called "etiquette". People make modifications to the general guidelines for behavior in order to optimize their performance for the specific situation. Social interactions are stable systems in which people play different roles based on their goals. But when someone deviates completely from the socially agreed upon patterns of behavior, things go awry. The equilibrium of the system is upset if one person’s actions interfere with the planned behavior of others.

People figure out each others’ intentions in an interaction by taking information gathering actions: observation and communication. Communication is a difficult issue in and of itself. Deciding when and what information to communicate is an open area of research in multi-agent systems [1], [2]. Also, communication increases the cognitive load on the other agents because they must decide on an appropriate response to the request for information. While communication can be modeled as just another possible action, dealing specifically with the issue of communication is outside the scope of this work.

Therefore, the focus falls on observation as the primary method of gaining information about other agents’ intentions. Often, taking a single observation or even several observations doesn’t yield enough information. The history of an agent’s behavior and the way that it evolves over time is necessary. If the model of behavior that you have for another agent captures the time-dependency of its behavior, then the passage of time itself can be a form of information.

Having a robot autonomously ride elevators in the presence of people is a task of time-dependent interactions with both the environment and people. To be successful, the robot must plan not only to get itself into or out of the elevator while the doors are open, but also infer the intentions of other passengers so as not to interfere with them. For instance, before entering the elevator, the robot should wait for people who are getting off. It cannot wait too long, however, or else the doors will close. Similarly, when on the elevator, the robot should try to predict the intentions of the other people on board in order to know where it should position itself so as to not prevent others from exiting.

Standing in line is also a difficult problem because there can be uncertainty as to who is, or is not, in line. Both spatial and temporal cues can help – people leaving too much space (compared to their personal space) are probably not in line, and neither are people who do not move up when the rest of the line moves. The uncertainty is in how much space is “too much” and how much time to allow before deciding that the person is not in line. Of course, one can always ask the person about his/her intentions, but one would prefer to use that sparingly, since it is intrusive and introduces a large cognitive load. Thus, the ability to reason about the utility of information gathering actions is very important.
There are a number of domains that do not include physical robots interacting with people that share many characteristics with the previous tasks. One example is dialog management: the conversational goals of a person will determine what they communicate, but which goal a person has may not be known to the conversational agent beforehand. The participants in a conversation may have different conversational goals. Also, in conservation, time can provide valuable information. The amount of time it takes a person to respond can give the agent insight into whether its last response was appropriate.

Driving in traffic is also a social domain in which self-interested agents are following a set of guidelines for interaction. The intentions of the other drivers aren’t directly observable. Exogenous events in the environment such as the changing of traffic lights provide external cues that people use to coordinate the interaction. Social rules, such as the order of right-of-way at a 4-way stop sign and the Pittsburgh left, govern how people behave in traffic.

The control of agents in video games is in many ways similar to the control of mobile robots, especially in games in which the agents are embodied in a virtual environment. Even in games in which the overall game-play is adversarial, there are scenarios in which the interactions with agents or other players in the game are not competitive in a zero-sum sense. In particular, many aspects of online role-playing games (RPGs) are social. Characters controlled by people interact with one another in familiar game play situations in order to achieve their own goals.
2 Background

2.1 Psychology

The field of psychology is relevant to this research because of its well-developed methodologies for evaluating subjective measures of human experience and conducting experiments that involve humans [3]. Researchers in human-computer interaction have recognized this and much interesting work on people’s perception of computers has been built on insights drawn from this other discipline, perhaps most notably in the research of Reeves and Nass [4]. Psychology is especially useful for studying the relationships and interactions between people and embodied computer agents or robots [5], [6].

One area of psychology of particular interest for designing algorithms for robots that interact with people is the study of nonverbal behavior [7]. While the concepts nonverbal interaction identified by this research have not been operationalized, they can still provide insight into how people manage physical interactions and what the salient characteristics of these interactions are.

An important psychological concept that is relevant to the design of robots that interact with people is that of cognitive load [8]. The concept is part of a theory of learning concerned with the relationship between human cognitive structure and the formation of long-term memory. The limits on how much information short term memory can contain has an impact on information processing and on the performance of unfamiliar tasks. Unfamiliar tasks place a high cognitive load on a person because they require many pieces of seemingly unrelated information to be held in the person’s short-term memory. Familiar tasks have their structure encoded into the person’s long-term memory, and the relevant pieces of information that must be attended to in the short-term memory are easily identified.

In peoples’ interactions with others, the regularity and familiarity of the interaction serves to reduce the cognitive load on the human participants. Better understanding of what is going on in physical interactions between people helps to design more accurate models of these interactions for use in planning. This results in controllers for robots that will behave in a manner that is both efficient and familiar] to people, making the robots easier for humans to share their environment with and interact with.

2.2 Planning and multi-agent systems

In planning, it is typical to define a problem as a set of states, each of which has a set of actions available for the agent to take in that state. Performing an action may cause a transition from one state to another. Certain states may represent a set of desirable conditions that are defined to be the agent’s goal. In general, planning is the problem of
deciding upon a series of actions that will obtain the agent’s goal. The planning problem can be thought of as the problem of finding a *policy*, a prescription for choosing actions given the state of the world.

Usually, policies are mappings from states to actions, but some problems may require more complex policy representations. For *time-dependent policies*, the time at which an action is taken in a state is important, in addition to what action is taken. For a *history-dependent policy*, the entire history of the agent’s experience up to the current action is used to determine what action to take.

For real world problems, there may be *uncertainty* in the outcome of actions or the transitions from one state to another. Additionally, the agent may not know the true state of the world. The state may be only *partially observable*, with certain observations more likely in certain states.

Decision theoretic planning defines goal achievement in terms of maximizing reward. The criteria for maximization may be the average reward, total reward, or discounted total reward, depending on the formulation of the problem. The models used for decision theoretic planning are sequential decision models, which are used to decide on a sequence of actions to take in a series of states, rather than a one-time decision. There are well-understood sequential decision models that represent uncertainty (MDPs), time-dependence (SMDPs), history dependence (finite horizon solutions to MDPs), and partial observability (POMDPs). One potential weakness of these models is that their framework isn’t developed to explicitly reason about multiple agents with different goals. A group of agents can be combined into a model that represents the entire group as one decision-maker. In order to find a policy for only one agent acting in the presence of other agents, the other agents and their actions must be modeled as a part of the world state. These types of models cannot capture how one agent’s actions may change in response to a change in policy by another agent.

Game theory is a field of mathematics devoted to reasoning about how to act optimally in the presence of other agents, considering that the actions taken by one agent may affect the utility of the actions available to another. For a game to be solved by finding its equilibrium (minimax, Nash, or correlated), one must assume that all players are playing rationally. Constant-sum games, in which the payoff to one player results in an equal loss to another player, are useful for modeling purely adversarial situations. Games that are cooperative or neither purely cooperative or adversarial require more complex solution techniques. Also, much of classical game theory is devoted to solving one-step games, where the players both simultaneously decide on their strategies before the game is played. These approaches cannot always be extended to models that allow sequential decision-making. Some sequential games can be converted into a normal form representation (which makes them appear to be a one-step game). But this change in representation results in a massive increase in the size of the game. Also, the normal form representation is ill-suited to handle situations in which additional information about the game is discovered by the players during game-play. When players have different knowledge about the world (one or more players don’t have access to the true world state), the game is one of imperfect information. Games having this property
cannot be solved using the dynamic programming techniques that are used to solve normal form games.

2.3 Markov models

The most common representation for sequential decision models in decision-theoretic planning are Markov models. This group of models all in some way take advantage of the Markov property as a way of reducing the complexity of the planning problem. The Markov property is defined as:

\[ Pr\{s_{t+1} = s', r_{t+1} = r|s_t, a_t, r_t, ..., r_1, s_0, a_0\} = Pr\{s_{t+1} = s', r_{t+1} = r|s_t, a_t\} \]

In other words, the environment’s response at time \( t + 1 \) depends only on the state and actions at time \( t \), and the rest of the execution history can be ignored.

A Markov decision process is made up of a set of states, a set of actions available in each state, a transition function defining the probability of transition from one state to another under all actions, and a reward function defining the value of each action when taken at each state. \( T \) is the time the system is defined over, either finite or infinite. Actions are taken at fixed decision epochs in the continuous case or timesteps in the discrete case.

\[ \{T, S, A, p_t(\cdot|s, a), r_t(s, a)\} \]

The dynamic programming algorithms for evaluating MDPs may be formulated to find either a finite horizon or infinite horizon solution. A finite horizon policy assumes that the system will terminate after a certain number of decision epochs or time steps. Because time is finite, the rewards and transition probabilities may be history dependent. An infinite horizon policy assumes that these distributions are stationary (i.e., they are the same at every time step or decision epoch). Most planning problems are treated as stationary and solved for an infinite horizon policy. This makes the problem representation more compact and yields a stationary policy as a solution.

Semi-Markov decision processes (SMDPs) are a generalization of MDPs that allow for the representation of limited non-Markov aspects of the state. SMDPs allow the time spent in a particular state to be modeled as an arbitrary probability distribution. In a MDP, time spent in a state can only be modeled as a self-transition, which has an exponential distribution. Many different formulations of SMDPs exist, including both discrete and continuous representations of time. Policies for SMDPs in the infinite horizon case may be stationary and time-dependent, rather than having the full history dependence of a finite horizon MDP policy.
Semi-Markov models represent an infinite variable (time) as a part of the state space. In discrete representations, time may be treated as finite. Because of the ability to “reset” the time variable associated with a state through self-transition, SMDPs can be used to model a cyclic, time-dependent state duration. In this case, the time variable represents the phase of this cycle in the state. In the discrete case where the time distribution for the state is a delta function at \( t = n \), the SMDP model can be thought of as an MDP with a timer associated with each state that can be reset by self-transition.

The generalized semi-Markov decision process (GSMDP) is a model designed to represent problems with multiple asynchronous events and actions [9]. A GSMDP can be thought of as a collection of concurrent SMDPs running simultaneously, one for each event. Events are any changes in the environment that cause a state transition (including the actions of the agent). The state space is assumed to have a factored representation. Logical formulas over the state variables are used to describe which events are enabled in a state. Each event has an associated time distribution governing how long it remains enabled before it triggers and a state transition function that describes the probability of transitioning to the other states when the event is triggered.

One way of approximating arbitrary distributions (such as the time distributions for a state in an SMDP or GSMP) as a Markov process is with a phase type distribution. A phase type distribution with \( n \) phases is a Markov process with \( n \) transient states and a single absorbing state. The time from entry to absorption in the Markov process approximates the original distribution. Phase type distributions may be either discrete or continuous Markov processes.

The partially observable Markov decision process (POMDP) is similar to the MDP, except that the state is not directly observable by the decision maker. Instead, the decision maker receives observations that arise probabilistically from the underlying state depending on what action was taken. Planning in a POMDP can be thought of as planning over the continuous belief space of the underlying MDP, the probability distribution over which state the agent is actually in when it receives an observation.
3 Related Work

3.1 Systems for social interaction

3.1.1 Human-robot interaction

Recently, there has been a variety of work involving social interaction between robots and humans [10]. What distinguishes our work from this research is the emphasis on planning for social interaction rather than on the physical design of the robot or ethnographic study of human-robot interactions. Additionally, the majority of this research concentrates on conversational interaction between humans and robots rather than on social behaviors that govern physical movement in shared spaces.

Existing approaches to producing social behavior tend to be either hand coded systems or model free learning methods. Policies for interaction are often designed by hand. Whether they are based on domain knowledge about the task to be performed [11] or a more general theory of social behavior [6], the result is a set of hand-crafted rules that trigger the appropriate actions. The main problem with this approach is that designing at set of rules that produce the desired behavior is likely to be very difficult and the method of obtaining the policy will not generalize to other tasks. Learning policies for behavior by either imitation [12] or reinforcement learning [13] is a popular method. The shortcoming of learning by imitation is that a particular way of achieving the goal is learned, not a full policy. The solution learned may be brittle and fail to generalize to future instances. The problems inherent in the model free reinforcement learning approach are the difficulty of obtaining enough training data for learning to occur and the difficulty of producing rewards for the system. In social tasks, rewards are likely to be delayed, and it is difficult to have people assign rewards during a realistic interaction.

3.1.2 Behavior/Plan recognition and dialog management

There is existing research that treats human-robot interaction as a planning problem. One application that bears many similarities to the problems that we are interested in is dialog management for robots, in particular approaches to dialog management that attempt to deal with the ambiguity of human speech in addition to speech recognition error. Work by [14] suggests the use of the POMDP formalism to model dialog tasks for a robotic helpmate. This approach is novel because it explicitly models ambiguity by representing the topic of conversation using hidden state. This makes it possible to find policies that allow the robot to take information gathering speech acts to resolve this ambiguity.

There has been some research in dialog management that uses the goal-directed nature of certain types of conversations to improve performance by using plan recognition [15]. Because
the system has a model of the task that the user is trying to perform, the dialog system can use plan recognition to identify what part of the task a user is asking for information about how to perform. While the representation is different, the idea of explicitly modeling the human’s goals for interacting with the system is similar to our approach. This dialog system has also been extended to include non-verbal communication cues for a robotic hosting task [16].

Another interesting approach to recognition was the modeling of a gesture recognition problem with an active vision system as a POMDP [17]. This allowed the vision system to plan its movements in order to take the most useful observations for the recognition task. Planning actions to improve recognition performance is very useful to the problems we are considering, and the ability to do so is one of the reasons that we have chosen to extend the POMDP formalism.

### 3.2 Multi-agent Planning

Much work in multi-agent planning is concerned with coordinating the planned behaviors of multiple agents. When all the agents are cooperating on a common goal, such as when a team of robots is performing a task, the agents can either be modeled as a single agent taking joint actions or as a game where all the players have common payoffs. These problems are complicated by the fact that the agents do not all have access to the same information about the world, and an accurate centralized model may be impossible to keep up to date with all of the incoming information. Therefore, decentralized approaches are used. One approach, based on game theory, models decentralized robot teams as partially observable stochastic games (POSGs) [1]. Another approach models the problem as a distributed POMDP model and uses communication to share information about observations [2]. In human-robot interaction, we only have control over one of the agents in the environment (the robot). Decentralized planning approaches that rely on communication of state information are a poor fit for the problem because the cooperation of the humans cannot be guaranteed.

In situations that are neither purely cooperative or adversarial, a decision is often made to treat the problem as if it were one or the other. The Electric Elves project models a group of agents trying to schedule meetings on behalf of people who may have conflicting interests as a cooperative team [18]. While this approach provides a framework for coordinating agent actions, the effects of the agents’ policies on each other cannot be reasoned about explicitly.

In purely adversarial domains, game theoretic solutions can guarantee optimal behavior, but the problem may be large enough to make finding such a solution intractable. In this case, hybrid methods may be used to reduce the complexity of the game. In one such system, the problem of planning a path through an environment in which the adversary has placed sensors is modeled as an MDP in which the cost function is chosen by the adversary from a set of possible cost functions. A mixed strategy is chosen by solving a matrix game over the pure strategies of the opponent (possible cost functions) and the agent (the optimal
policies for MDPs with those cost functions). [19]. Finding a game-theoretic equilibrium over policies is a promising way of improving performance in the presence of other agents.

Market-based systems are another popular way of managing interactions between self-interested agents. market-based systems have been used extensively in a variety of multi-robot planning domains [20]. These approaches are concerned with the allocation of tasks and resources rather than on learning policies for task performance, so they are not directly relevant to the planning problems this research is concerned with.

3.3 Markov models for decision-theoretic planning

3.3.1 Approximate POMDP solution methods

There are many different kinds of algorithms for finding approximately optimal POMDP policies, either by approximating the value function [21],[22] or by policy search [23], [24], [25]. There are also greedy heuristics that solve the underlying MDP and make simplifying assumptions about belief during execution [26], [27]. Another class of heuristics makes use of the convexity of the value function over the belief space in order to take information-gathering actions. The Q-MDP heuristic makes the simplifying assumption that the state will become observable after one step [28]. The weighted-entropy heuristic combines the Q-value function of the completely unobservable MDP and the completely observable MDP weighted according to the entropy of the belief state [29].

Current approximate POMDP solution methods reduce the complexity of the value function or the policy representation without compromising the ability to take information gathering actions. Belief compression represents the belief space by a set of sufficient statistics [30]. Point-based value iteration (PBVI) is an anytime algorithm that represents the value function as samples at chosen points of the belief space [31]. The heuristic search value iteration (HSVI), another anytime algorithm, returns a regret bound on the obtained policy with respect to the optimal policy [32]. The VDCBPI algorithms runs bounded policy iteration on a compressed version of the original POMDP [33]. Policy gradient techniques have also been applied to POMDPs [34], [25].

3.3.2 Semi-Markov and time-dependent models

The major offline MDP algorithms (value iteration, policy iteration, and linear programming) all have well-known SMDP versions [35], [36]. Reinforcement learning algorithms, such as Q-learning and temporal distance learning, have been extended to SMDPs [37]. Convergence results for SMDP Q-learning have been proven [38].

Semi-Markov extensions to Markov models have been used for modeling and planning in a
variety of domains. Hidden semi-Markov models (HSMMs) have been proposed as a more natural way of representing words for speech recognition [39]. HSMMs have also been used to model the plasma etching process for semiconductor manufacturing [40]. Semi-Markov decision processes have been explored as a form of temporal abstraction for reinforcement learning [41]. SMDPs have also been used to model planning problems that allow concurrent actions [42]. An interesting formulation that is closely related to SMDPs is the time-dependent Markov decision process, which models both time-dependence and arbitrary distributions over action durations and was used for planning in a transportation domain [43].

A number of models that are based on SMDPs but represent partial observability have been proposed. A partially-observable semi-Markov decision process (POSMDP) model has been proposed as a way of doing hierarchical abstraction in POMDPs [44]. A similar model, the H-POMDP, hierarchically models concurrent, temporally extended actions [45]. The H-POMDP model has been transformed into a more compact factored representation, a dynamic Bayes network (DBN), used for multi-resolution mapping for robot localization [46]. The focus of this work is on hierarchical abstraction and representing temporally extended actions rather than on modeling time-dependent action outcomes and dynamic environments.

Generalized semi-Markov decision processes are a planning formalism for representing domains with multiple asynchronous events [47]. Rather than solving these models exactly, the semi-Markov states are approximated using phase type distributions. GSMDP models were shown to perform better than a SMDP representation on the “Foreman’s Dilemma” problem and to be capable of representing problems that cannot be modeled as SMDPs because they contain multiple asynchronous events. The planning approach we propose is largely based on this work, with the addition of partial observability to the GSMDP model.
4 Technical Approach

4.1 Problem

In Section 2.3 many different models used for decision-theoretic planning were introduced. They cover a wide range of modeling power and complexity. Choosing an appropriate model is one of the most fundamental steps in solving a planning problem. A model that isn’t expressive enough to represent the most important characteristics of a domain will not achieve adequate performance. But a model that is unnecessarily complex can make a relatively simple problem intractable. When choosing a model, it is important to identify which aspects of a problem are essential and which can be abstracted away.

This thesis hypothesizes that the following are all critical characteristics of social tasks:

- uncertainty in action outcomes
- exogenous events arising from a dynamic environment
- time-dependence of action outcomes
- partial observability of the world state

As a motivating example, consider the problem of a robot attempting to get on an elevator that currently contains human passengers. If the people want to exit the elevator, it is probably best for the robot to yield to them (as etiquette dictates) in order to avoid colliding with someone or blocking them from exiting the elevator. But because the robot is slow compared to a person, it should move toward the elevator as soon as possible in order to avoid having the door close before it can get on. There is a danger of deadlock if the robot doesn’t move to enter the elevator at the right time.
In most real-world domains, the outcomes of an agent’s actions and changes in the environment (including the actions of other agents) are uncertain. Any high-level action (such as moving to enter the elevator) that a mobile robot takes has a possibility of failing or falling short of its intended effect because of unexpected complications caused either by the environment or by problems in actuation. Recognizing this limitation and modeling the range of possible action outcomes results in more robust plans that have a greater chance of success when executed in a physical environment.

Uncertainty in action outcomes is often also used as a way of modeling a dynamic environment, but that approach is appropriate only for modeling state changes that may occur randomly. In interactions with people, however, actions are typically planned, not random. The world state will change according to exogenous events whether the robot acts or not. These events may be caused by the actions of other agents or by a dynamic part of the environment. In the elevator-riding domain, state may change because of the movements of the person riding the elevator or because of changes in the state of the elevator itself.

The outcome of an action may depend on the amount of time spent in a state before executing the action. Purely Markovian models cannot capture this time-dependent information without making time a variable in the state space. For example, the robot may be outside the elevator with the doors open and about to move to get on. However, because the elevator doors operate according to a timer, the robot’s chance of success is affected by how long the doors have been open.

Partial observability is also a common characteristic of most real-world domains. The most familiar kind of partial observability in robotics problems is perceptual aliasing caused by noisy sensors. Physical characteristics of the environment or other agents (such as their relative location, direction of movement, or gaze direction) are observable, but these observations aren’t completely trustworthy and the underlying world characteristics they represent are likely to be changing over time.

In social tasks where the people in the environment are trying to achieve their own goals, the intentions of the people interacting with the robot are another partially observable aspect of the environment. While intention can’t be observed directly, knowing the intentions that agents commonly have in a given domain, and having a model of the actions people take when they have those intentions, makes predicting their behavior much easier. For example, if a person is inside the elevator, it can be assumed that they are either planning to exit at that floor or waiting to get off at another floor. By observing their actions over time, the robot can usually infer their intention.

The dynamics of the environment itself may also be only partially observable. Consider the timer that controls the amount of time that the elevator door is open. The robot may arrive in front of the elevator when the door is already open, so the start time of the timer is unobservable. Even if the robot doesn’t know the amount of time that the timer has left, knowing the time distribution governing the elevator door closing event gives it some information about how likely it is to be able to get on the elevator in time.
4.2 General framework

Here, we will outline a technique to represent problems with both partial observability and multiple time-dependent events as a partially observable generalized semi-Markov decision process (POGSMDP). Both the dynamics of the environment and the actions of people are represented as exogenous, timed events in the POGSMDP. The basic approach to planning using such models is to transform the POGSMDP into a POMDP that can be solved using existing algorithms.

Here is an outline of the proposed approach:

- Model social interaction problem as POGSMDP
- Transform the POGSMDP into POMDP
- Solve the POMDP
- Verify “social correctness” of the policy using the POGSMDP model
- Demonstrate the performance of the policy obtained on the real problem

The POGSMDP model is based on the GSMDP model [47]. Partial observability is modeled by defining a probability distribution over the observations in a state, exactly as it is done in the POMDP model. For the time being, the role of observations can be ignored in order to focus on the design of the state space and reward, which will be described in terms of GSMDPs.

A benefit of the GSMDP model is that it allows one to write a compact representation of the problem while assigning arbitrary time distributions to events and actions. A useful way of approaching the modeling issue is to think of a GSMDP as the composition of a number of SMDPS. This allows one to decompose the problem into easy to manage pieces, one for each agent and exogenous event in the environment. The reward structure can also be decomposed as such, assigning each agent (both human and robot) its own rewards for goal achievement. The global reward for each state may be the sum of all agents’ rewards, or another way of composing the rewards may be chosen, such as weighing simultaneous failures for multiple agents more heavily than failures by a single agent.

Because POGSMDPs will be used to model large, realistic problems, authoring the models by hand may become prohibitively difficult. In order to simplify the process, tools will be developed to compile the models from domain descriptions. PPDDL is an existing domain description language that was used as the input language for the probabilistic track of the 4th International Planning Competition [48]. PPDDL will be extended as needed to use it as an input language for the compilation tools.
Given a GSMDP model of the problem, it can be transformed into a continuous-time MDP by approximating the non-exponential distributions in the model using phase-type distributions [9]. The resulting MDP is the underlying MDP of a POMDP model that approximates the original POGSMDP.

Most widely used POMDP solution techniques require that the underlying MDP have a discrete state space. One way to obtain a discrete MDP is to apply uniformization, an exact transformation from a continuous MDP to its discrete equivalent [49]. Another possibility which was discussed but not explored in Younes’s work [9] is to use discrete phase-type distributions to approximate the GSMDP. One possible benefit of this approach is that discrete phase-type distributions can handle deterministic distributions, while continuous ones cannot. Therefore, discrete phase-type distributions may be better suited to domains where certain events trigger upon expiration of a timer rather than happening probabilistically over a range of time. An elevator door closing or opening is an example of such an event.

When making this transformation, there is a trade-off between the accuracy of the model and the size of the resulting MDP. Each state added adds a dimension to the belief space of our POMDP. One of the goals of this work will be to discover techniques to manage the state blow-up resulting from the phase-type distribution approximation, either through analysis or through heuristics determined by experimentation.

Once the underlying GSMDP of the POGSMDP has been transformed into an MDP, the correct observation probabilities can be applied to the new MDP states to obtain a POMDP model. This POMDP can then be solved using any of the state-of-the-art approximate POMDP solution algorithms discussed in Section 3.3.1. An additional contribution of this work will be to apply these cutting edge techniques to large, complex POMDPs that model real problems. We expect that this may yield some insights into the relative strengths and weaknesses of these algorithms.

In addition to obtaining a high level of reward, a policy should also result in behavior that is acceptable to the people the robot interacts with. The assumption has been made that an adequate model of the socially acceptable behavior for the people has already been defined as part of the POGSMP model. Therefore, the robot’s policy should not force people to deviate from their own policies in order to obtain higher reward.

How can we determine whether the robot’s policy has this property? One way is to fix the robot’s policy and solve for the human’s responses. If the resulting policy is fundamentally different in structure for the given model of human behavior, the the robot’s policy is not “socially acceptable”. Technically, the verification process depends on the fact that the POGSMDP is the composition of the policies of the humans and the model of the environment. The policy obtained for the robot can be added into this model as if it were one of the human agents. Then, the model governing the behavior of one of the human agents can be removed from the POGSMDP. The new model is solved as before, and the resulting policy for the human agent is compared to the model of its behavior. If the policy is fundamentally different in structure from the model of the human’s behavior, then the robot’s policy is
Changes must be made to the reward structure of the problem to better define criteria for the robot’s success, which should be based on both achieving its own goals and on not interfering with the goal achievement of the humans it interacts with.

The idea of iterating over the policies of agents in a multi-agent system to improve overall performance is also used in the JESP algorithm [50]. Unfortunately, when iterating over agents’ policies, there are no guarantees that the system will converge to a better overall reward. In order to avoid this situation, the policies obtained for the people are not incorporated into the original model. It is the robot’s policy that must accommodate the people’s, not the other way around.

It is possible that there may be multiple optimal policies for a person to achieve their goal. In order to make sure that the changed policy for a person is actually an improvement over their modeled behavior, the reward obtained by the policy should be checked against the reward obtained by their modeled policy. Because a social system is a system in a state of equilibrium, it may be possible to use game-theoretic techniques to determine how the rewards should be adjusted in order to obtain a policy for the robot that does not disturb the human’s policy.

The application domains this approach will be demonstrated on are discussed in Section 6.
5 Previous Work

5.1 Social interaction with a mobile robot

Our earliest experiment with a robot in a social situation involved a greeting task [51]. A robot stood near the wall in a busy hallway and verbally requested that passersby stop to answer a short poll. This task was chosen in order to investigate what behaviors and features a robot can exhibit in order to more fully engage passersby. The behaviors tested were the ability to convey expression with a humanoid face and the ability to indicate attention by turning toward the person that the robot is addressing. We measured the impact of these abilities on the robot’s level of success at attracting people passing by in the hallway. The experimental design was that of a 2x2 full factorial experiment, controlling for the effects of time of day and day of trial. We found statistically significant effects on peoples’ willingness to interact for both of the factors. Our results also indicated that using the movement and the face in conjunction was more effective than using either alone.

Techniques from social psychology were used to design and conduct this experiment. As was discussed in Section 2.1, this field has developed a methodology for experimentation that is designed to control for the complexity of human behavior. In addition to the metrics we measured, we also collected ethnographic data about the interactions that people had with the robot.

We observed many unanticipated failures in interaction. Most important, for the purpose of this thesis was that while the robot was capable of detecting and tracking people, it had no internal model of human behavior. Our algorithm simply chose the first person it detected and would continue to try to engage them, even if they were walking away. In many cases, the robot did this while ignoring other passersby who were clearly interested in interacting. Also, during the interaction, the robot prompted people to step closer to speak into the microphone. Rather than doing this, people stayed the same distance from the robot and talked more loudly. This may have been because standing any closer to the robot would have been a violation of their personal space. People became perplexed and then frustrated when the robot repeatedly prompted them to move closer to continue their interaction.

The unforeseen complications in interaction that we observed during the course of this experiment suggested that having a model of people’s behavior and reasoning explicitly about time might make significant improvements in the robot’s ability to interact.

5.2 People tracking with a goal-based motion model

In work with Geoff Gordon [52], a path planner was incorporated into the motion model of a particle-filter based people tracker in order to make tracking more robust to extended
periods of occlusion. This approach capitalizes on the fact that people usually do not move through spaces in a random manner, but instead follow relatively efficient paths between particular locations in the environment (doorways, for example). These locations can be thought of as a set of goals that people typically navigate toward in a certain space. Given these goals, we assume that any person the robot sees is trying to navigate toward one of them.

First, trajectories of people moving through the space are gathered and divided into clusters that are all following roughly the same path toward a common goal. A hill-climbing algorithm is then used to learn a physical location for each goal that results in plans that fit the observed trajectories assigned to that goal.

Learning the goal location from the trajectory data rather than assigning it by hand is necessary in order to obtain paths from the path planner that mimic the general shape of the people’s trajectories. Figure 2 illustrates the learning process in a case with two clusters of trajectories.

When the tracking filter is initialized, probability mass is distributed equally over all of the goals by assigning each one the same number of samples in the filter. At each timestep, the path planner plans a path from a person’s last observed position to every goal, and the motion model propagates each particles along the path to its corresponding goal. This technique tracks people significantly better than a particle filter that uses Brownian motion as its motion model in cases when trajectories are occluded for long periods of time during
Figure 3: Tracking a person during the occluded part of their path using the Brownian motion model (top) and the plan-based motion model (bottom): a) just after losing sight of the person, b) the Brownian filter’s variance starts to grow, c) the Brownian filter fails to predict where the person is about to emerge from occlusion, d) the Brownian filter’s variance remains higher for several steps after reacquiring the person.

tracking. Figure 3 shows a comparison of the distribution of particles over the course of an instance of tracking.

The particle filter tracks the belief over which goal a person is navigating toward as well as tracking belief over their location, because the samples corresponding to unlikely goals die out during the tracking. A concise parameterization of people’s actions that represent their higher-level intentions (i.e., where their intended goal is) can be learned directly from the data. Because there are very few parameters to learn, this approach is successful with a small amount of data, provided that paths to all of the goals are present in the training set. It is possible to represent the data with such a small number of parameters because we make strong assumptions about people’s behavior: that they are goal oriented and follow relatively efficient paths.

This novel approach to particle filter-based people tracking demonstrates the utility of modeling people’s high-level goals in order to represent their behavior. A better motion model results in better tracking performance, particularly during periods of occlusion when the filter’s hypotheses cannot be frequently updated with observations of the person’s location.
5.3 Representation of time in POMDPs

In order to represent a problem as a POMDP, certain abstractions are made in designing the state space in order to keep problems small enough to be solvable. To illustrate the effect of ignoring information about the dynamics of the world in order to achieve a more compact representation of the problem, we constructed an artificial example based on the ever-popular tiger problem [53], which we call the “teleporting tiger” problem.

In the teleporting tiger problem, a person is standing in front of two doors. Behind one door, there is a treasure, behind the other, there is a tiger that will eat the person if she opens the door. She can listen at the door for the sound of the tiger growling, but she may mis-hear which door the growling is coming from. Every $n$ timesteps, the tiger flips a coin. If it comes up tails, the tiger teleports to the other room. The person knows that the coin will be flipped every $n$ timesteps (she knows the coin-flip event has phase $n$), but she does not know when the coin flip is happening because she cannot see the tiger. Figure 4. shows a diagram of the state transitions of this problem for $n = 3$.

![Figure 4: The dynamics of the state of the teleporting tiger problem with time $n = 3$](image)

An $n$-valued variable representing time can be included in the state space to get a $2 \times n$ state POMDP that accurately captures the relationship between the timestep and the outcome of the actions. This effectively replaces both of the two states representing the tiger’s physical location with $n$ state chains with one state for each timestep between the coin-flips. Or,
the phase of the coin flip event can be ignored by the model. Then the problem can be represented by a two state POMDP where the outcome of listening is more uncertain than it actually is. See Figure 5 to compare the resulting POMDP models. Appendix A includes a full description of the models.

Figure 5: State transitions of the listen action for the teleporting tiger problem without representing time in the state (left) and with time as part of the state (right)

Both models were solved and tested using Cassandra’s POMDP solver [54]. The two state model took a fraction of a second to solve exactly using the witness algorithm. However, the six state model could not be solved by the witness algorithm in a reasonable amount of time. Therefore, the PBVI algorithm [31] was used to solve for an approximately optimal policy for each model. The policies were then simulated for 1000 trials of 500 timesteps each. The results are listed in Table 6. The six state POMDP achieved higher average reward than the two state model, at the cost of taking considerably longer to solve. It is possible that the policy obtained for the six state model performed suboptimally, but attempts to solve for a better policy using more belief points resulted in policies which were an order of magnitude larger in the number of alpha vectors and failed to converge. The performance of the policy obtained for the two state model was virtually identical to the performance of the optimal policy found using the witness algorithm.

This toy problem illustrates the inherent trade-off made between modeling time accurately and managing the size of the state space in a POMDP. Putting a discrete time index into the state space as a variable can cause a blowup in the size of the state that can make even relatively simple models intractable to exact POMDP solution algorithms and considerably more computationally expensive even for state-of-the-art approximate solution methods. For
POMDP model | soln time (secs) | belief points | alpha vecs | avg reward | std dev
--- | --- | --- | --- | --- | ---
2 state | 0.74 | 73 | 17 | 1.123 | 37.199
6 state | 5.04 | 141 | 13 | 10.232 | 35.398

Figure 6: Average rewards for teleporting tiger POMDP experiment

a model with $m$ states, adding a time variable to the state space with a phase of $n$ will result in a model with $m \times n$ states. The belief space of the POMDP grows in turn from $n - 1$ dimensions to $m \times n - 1$ dimensions. Clearly, when representing time in a POMDP model, it is important to find methods that increase the accuracy of the model while causing the smallest increase possible in the size of the state.

5.4 Elevator riding domain

Controlling a robot that rides elevators with people is one of the motivating problems for developing the proposed framework for planning for social interaction. We initially thought to model the problem as a POMDP, a commonly used representation for high-level mobile robot planning problems. A simplified version of the elevator riding problem in which the robot learns a policy for entering an elevator that is occupied by one person when it arrives was used to investigate the adequacy of POMDPs as a model. The robot’s goal in this task is to get on the elevator before the doors close, while not preventing the person from exiting the elevator if they are getting off at that floor.

Our original hypothesis was that the performance of a POMDP-based policy would be an improvement over an MDP-based policy that assumes the person’s intention is fully observable from their behavior. However, the results of experiments using these POMDP polices cast doubt on the adequacy of the model for the problem. In particular, the POMDP model did not accurately represent the time-dependence of the door closing event. The distribution over time spent in each state was modeled as a self-transition, as is commonly done with Markov models. Because of the memoryless property of exponential distributions, the model could not capture the time dependence of action outcomes. This caused unanticipated differences in the performance of the policies when run in the elevator simulator as compared to their predicted performance in simulation from the POMDP models.

5.4.1 POMDP model

The POMDP model for the elevator riding task is defined as follows:

- state features
There are 16 possible combinations of these binary valued state features. However, only 12 combinations correspond to states of the world that will actually occur according to our model. In order to create a model in which the person’s intention is not deterministic from trial to trial, we create an additional start state with transitions to both of the beginning world states corresponding to the person’s possible intentions (to exit the elevator or stay on). Figure 7 shows a graphical representation of the states of the POMDP model.

- **actions**

  In each state, the robot has two actions available to it: move or wait. The outcome of these actions is uncertain. When the robot chooses the “wait” action, there is no chance that it’s own location will change. The allowable state transitions will change only the location of the person (if that is possible) or change the state of the door from open to closed according to the dynamics of the model.

  The possible outcomes of the “move” action are more complicated. The probability of the action’s outcome not causing a change in state (i.e., the probability of self-transition) is the same as for the “wait” action. In states where there is no chance of conflict between the goals of the robot and the person (state 8), the remaining probability will all be placed on a transition to a state that represents a successful movement on the part of the robot from outside the elevator to inside it. However, when the robot’s movement could conflict with the person’s goal of leaving the elevator, the remaining probability mass is distributed over transitions to states corresponding to either success or the ways in which the action could fail.

  For a complete picture of the state transitions possible for the POMDP under each action, see Figure 8. For the transition probabilities used for the model, see the Appendix.

- **observations**

  We assume that the robot has the ability to observe the state of the elevator itself perfectly. When the elevator door is closed, the robot receives an elevator closed (ec) observation. When the door is open and there isn’t a person in the elevator (because the passenger has already exited), the robot perceives an elevator empty (ee) observation.

  When the door is open and there is a passenger inside the elevator, the situation is only partially observable to the robot. We treat the intention of the person (whether they intend to exit at this floor or ride the elevator to another floor) as hidden state. The noisy observations that the robot may receive corresponding to these intentions are that the person is exiting (px) or standing still (ps).

  The observations associated with each state transition are shown in Figure 8.
Figure 7: The states of the elevator riding POMDP
• reward

We assign reward to states where the robot achieves its own goal (to get onto the elevator before the door closes) and where the person achieves their goal (either to remain on the elevator, or to exit the elevator before the doors close, depending). However, we want the policy the robot learns to favor self-interested action, so we assign greater reward to states in which the robot’s goals are achieved. Reward for goal achievement is additive and assigned in the end states (the states in which the door is closed).

reward/penalty for robot goal achievement: +60/-60
reward/penalty for person goal achievement: +40/-40

robot succeeds, person succeeds (states S2,S6) : 100
robot succeeds, person fails (state S7) : 20
robot fails, person succeeds (states S1, S5) : -20
robot fails, person fails (state S12) : -100

The POMDP model description in the input format required by Tony Cassandra’s POMDP solver [54] is included in Appendix A.

5.4.2 Elevator simulator

The elevator simulator is a simplified task environment made up of an elevator that travels between multiple floors and the foyer areas outside of the elevator at each floor. The elevator has a working controller that responds to button presses on the different floors and within the elevator in the expected manner. The elevator controller also controls lights that indicate the arrival of the elevator at each floor and its direction of travel.

In our initial experiment, we did not include the detailed steps of calling an elevator in our model. But having an accurate simulator will be necessary to design and debug the larger, more realistic models that will be built in order to learn a controller for elevator riding for a physical robot.

The representation of space used in the elevator simulator is a hierarchical topological map of areas of importance to performing the elevator riding task. For example, the foyer outside the elevator on the second floor contains both the “button press area”, which an agent must be standing in to reach the elevator buttons, and the “doorway area” which an agent must pass through (or can block others from passing through) in order to enter the elevator.

Path planning for both the person and the robot is accomplished by breadth first search through the topological map of the floor that the agent is on. The map updates itself in
Figure 8: Observations, state transitions, and rewards for the elevator riding POMDP model
time with the changes to the elevator state, making links between areas when the door to the elevator is opened and breaking them when it closes. Therefore, search through the map will return a plan to a goal only if it is reachable from the agent’s location at that time. Low level motion control for agents is accomplished by moving the agents along lines from the center of one area in the map to the next at constant speed. At each timestep, an agent is prevented from moving if its movement will result in a collision with a part of the environment or another agent.

For our experiments with the elevator simulator, a reactive, case-based controller for a simulated person was hand-coded. The person controller is able to ride the elevator from one floor to another, making all of the necessary button presses to call the elevator to their current floor and send it to the destination floor. Unlike the controller which is learned for the robot, the person controller is given full access to the internal state of the elevator simulator.

This person controller is not meant to be a high fidelity model of human elevator riding behavior. It can be thought of as a model of the simplest set of behaviors necessary to ride the elevator from one floor to another without considering the possibility of interference from other agents. This simple model was chosen for the sake of expediency. Later, a more realistic model of the social behaviors that govern elevator riding will be designed. A detailed description of the controller is included in Appendix A.
5.4.3 Results

In order to investigate the relationship between the partial observability of the person’s intention and the difficulty of the problem, multiple POMDP models were created by varying the noise level of the observations related to the person’s intentions (ps and px) from 10% to 40% by increments of 10%. Each model was solved for an approximately optimal policy using the PBVI algorithm.

The average reward obtained for 50 trials in the elevator simulator at different noise levels in the observations are shown in Table 10. At the start of each trial, the person’s intention was chosen randomly with 50% probability on each case. The same seed was used for each experiment in order to make a clear comparison of the policies’ performance. All of the policies performed well, regardless of the noise level of the model that they had been trained on. The performance of the policies trained on the 10%, 20%, and 30% noise level POMDP models were the same (the policy trained on the 40% model performed only slightly worse). In order to show the cases in which the policies failed, the results are divided according to the person’s intention. The elevator door stayed open long enough that even if the robot needed to make observations for longer periods of time before acting, it could still enter the elevator before the doors closed. The failures at higher noise levels for the exit case were the result of the robot attempting to enter the elevator before the person had finished exiting and blocking them in the doorway. It is understandable that the policies trained on low noise levels performed this way, because they could be expected to be overconfident in the observations. But it is surprising that the same behavior was observed in the policies trained at high noise levels. These policies seem to choose moving immediately over waiting and taking more observations in order to determine what the person intends to do.

<table>
<thead>
<tr>
<th>noise level</th>
<th>average reward</th>
<th>exit</th>
<th>stay</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>20%</td>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>30%</td>
<td></td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>40%</td>
<td></td>
<td>59</td>
<td>94</td>
</tr>
</tbody>
</table>

Figure 10: Average rewards for POMDP policies in elevator simulator

Initial results show that the low noise POMDP policies perform well in simulation. However, when the model has a high noise level, the policy obtained ignores observations and causes the robot to get on the elevator immediately, blocking the person from exiting if they intend to leave the elevator at that floor. This occurs because representing the time before the elevator door closes using self-transition in a single POMDP state (which has an exponential distribution) models that event as having a certain probability of happening at each timestep in that state. This is a poor model of the world. This problem with the representation could be solved by using phase-type distributions to accurately represent the amount of time that passes before the elevator door closes, demonstrating the utility of more complex representations of time in a real-world problem.
The policies obtained were also run in simulation against all of the POMDP models. Each policy was simulated for 1000 trials on each model. Table 11 shows the average reward obtained per trial. Note that the policies performed worse when run against the POMDP models than they did in the elevator simulator at the same noise levels. We believe that this difference in performance arises from the way that time is represented in the POMDP model.

<table>
<thead>
<tr>
<th>policy</th>
<th>model</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>70.6</td>
</tr>
<tr>
<td>20%</td>
<td>70.6</td>
</tr>
<tr>
<td>30%</td>
<td>70.6</td>
</tr>
<tr>
<td>40%</td>
<td>70.6</td>
</tr>
</tbody>
</table>

Figure 11: Average rewards for elevator riding POMDP simulation

Each policy performed about the same on all of the models, regardless of the noise level of their observations. We interpreted these results to mean that the actions chosen by the robot did not have a great effect on the reward that it could obtain. In order to gain insight into this interpretation, we decided to make the model fully observable so that we could obtain optimal policies using exact solution techniques. The POMDP was divided into two different MDPs, one for the part of the state space where the person intended to stay on the elevator and the other for the part where the person intended to exit at the robot’s floor.

Optimal policies for both MDPs were found and then run in simulation against the model for 1000 trials each. The average reward obtained is shown in Table 12. The combined average reward for both cases is close to the average reward obtained by the POMDP policies on the 10% noise level POMDP model, suggesting that the performance of those approximate policies is close to optimal.

<table>
<thead>
<tr>
<th>MDP</th>
<th>avg reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td>61.8</td>
</tr>
<tr>
<td>stay</td>
<td>83.4</td>
</tr>
<tr>
<td>combined</td>
<td>72.6</td>
</tr>
</tbody>
</table>

Figure 12: Average rewards for elevator riding MDP simulation

The policies obtained for both MDPs were very simple. If the person was staying, the optimal solution found for the MDP was to always move at every timestep. If the person was exiting, the optimal solution found for the MDP was to wait until the elevator was empty and then move.

These policies are intuitively reasonable policies for elevator riding. Because the MDP models have no uncertainty in the observations, our expectation was that the policies should be able to obtain a reward of 100 (perfect performance). However, this was not the case.
Assuming that the robot has time to move into the elevator before the doors close, these policies should always succeed. But because the MDP (and POMDP) model doesn’t explicitly represent the fixed amount of time that passes before the doors close, the door closing can only be modeled as an event that has a non-zero probability of occurring in any state at which the doors are open. This way of modeling the door closing event is obviously inaccurate, and exaggerates the probability of failure for the robot. We believe that it is this characteristic of the model that causes the POMDP policies trained on noisy observations to fail at high noise levels. Because the model indicates that the door could close at any timestep, the policies learn to move onto the elevator when they could actually take more observations. This leads to failures when the person is trying to exit and the robot blocks them in the elevator doorway.

Our next step is to develop a model of the problem with a more accurate representation of time without causing the state space of the POMDP to grow so large that it becomes intractable to solve.
6 Proposed Research

6.1 Expected contributions

This thesis work will demonstrate a novel way of modeling social systems involving people and robots and designing controllers for robots in human-robot interaction domains. The proposed approach models both the partial observability of the state and the time-dependence of action outcomes in these problems, two issues that have not previously been addressed simultaneously in planning. A new model, the POGSMDP, will be developed, and its performance will be tested against existing commonly used decision-theoretic models in two different implementation domains. Authoring tools for POGSMDP models will be designed that will allow these models to be created from descriptions written in a general domain description language. As a part of the planning framework, a method for verifying that robot policies for human-robot interaction are “socially acceptable” to the human participants will be developed. Because the POGSMDP models will be transformed into POMDPs before being solved, this work will provide insight into the performance of current approximate POMDP solution algorithms on large, real-world problems. Additionally, this work will contribute to the field of human-robot interaction by experimentally evaluating the performance of the policies developed on a physical robot interacting with people.

6.2 Elevator riding domain

First, build a phase-type model for the Elevator simulator and demonstrate that it is an improvement. The next step of the research in the elevator riding domain is to model the problem as a POGSMDP and test the resulting policy in simulation. To verify that the additional complexity of our approach is necessary, we will compare our solutions to those found using less expressive MDP and POMDP models. An MDP model cannot capture the partial observability of intention or the asynchronous, time-dependent nature of the problem. A POMDP model (as we have shown in our previous work with the elevator simulator) still fails to accurately model the asynchronicity of events. Using a common reward structure for all of the policies, we will evaluate their performance in the simulator according to the reward obtained.

Next, we will build a POGSMDP-based controller for a mobile robot that is able to ride an elevator with human passengers. This will involve scaling up the elevator riding model we used for the simulator to a level of detail that can capture the complexity of a real world problem.

Once again, we will compare the performance of our policy to that of an MDP and POMDP policy. The comparison will be done both objectively and subjectively. Some objective measures include: time to complete a task, percentage of times the task is successfully
completed (e.g., the percentage of people the robot successfully interacts with), and the reward obtained by the policy. Subjective measures include asking the people about their perceptions of the interaction and cognitive load they experienced while interacting with the robot. Both measures are important in assessing the utility of our approach.

6.3 Multiplayer game domain

The second implementation domain for the thesis will be a multiplayer game, in which one of players will be an agent running a policy developed using the proposed technique. The platform for development to be used will be a “mod” (a user customized game) based on the UnReal Tournamnet (UT) game engine [55]. The UT engine is designed to give programmers a high level of flexibility in designing their own games or making extensions to existing games. It is used by several research groups as a platform for research for AI in video games [56] and for simulation in robotics[57].

While many online games are adversarial in nature, role-playing games (RPGs) are story-based games that may contain adversarial, cooperative, and social elements. One implementation option is to identify a game-play scenario that exhibits the type of social interaction we are interested in within an existing RPG game, such as Neverwinter Nights [58]. In the unlikely situation that an existing game scenario that exhibits both uncertainty in people’s intentions and time-dependence cannot be found, the UT engine will be used to author an original game for the experiments to be conducted with.

Evaluation of the system will be similar to the evaluation of the elevator riding domain. The POGSMDP-based policy will be compared to policies generated using MDP and POMDP models. The quality of the policies will be evaluated based on the reward obtained. The policies will also be evaluated against the level of reward obtained by a human player controlling the same agent. There will also be a subjective evaluation of the quality of game play that results from interacting with each of the policies (including the human player’s policy).

There are a number of reasons for choosing a computer game as the second evaluation domain rather than another task performed by a physical robot. One goal is to show the general applicability of this approach to modeling social interactions between humans and artificial agents. The characters are physically embodied within the game environment, so the planning problems are similar to those in robotics. However, many difficulties of working with a mechanical system are absent in dealing with software agents. Also, the popularity of video games makes this an interesting demonstration domain to a wide variety of people. Finally, the online game setup will make evaluating the system with human users relatively simple.
7 Schedule

- **Fall 2005**
  - Develop tools for authoring POGSMDP models and automatically transforming them into approximate POMDP models using phase-type distributions.
  - Explore issues of representation for phase-type distributions to control POMDP state blowup.
  - Design POGSMDP model for elevator riding domain. Test in the elevator simulator against MDP and POMDP models.

- **Spring 2006**
  - Formalize verification process for “socially acceptable” policies.
  - Design and implement controller for non-player character in online RPG.
  - Run tests evaluating performance of video game controller.

- **Fall 2006**
  - Implement elevator riding task on a physical robot.
  - Run tests evaluating robot’s performance at elevator riding task.
  - Write and defend thesis.
Appendix A

Tiger problem POMDP description files

2 state version:

discount: 0.95
values: reward
states: tiger-left tiger-right
actions: listen open-left open-right
observations: tiger-left tiger-right

start:
0.5 0.5

T: listen
0.83 0.17
0.17 0.83

T: open-left
deadlock

T: open-right
deadlock

O: listen
0.85 0.15
0.15 0.85

O: open-left
deadlock

O: open-right
deadlock

R: listen : * : * : * -1
R: open-left : tiger-left : * : * -100
R: open-left : tiger-right : * : * 10
R: open-right : tiger-left : * : * 10
R: open-right : tiger-right : * : * -100
6 state version:

discount: 0.95
values: reward
states: tiger-left0 tiger-right0 tiger-left1 tiger-right1
tiger-left2 tiger-right2
actions: listen open-left open-right
observations: tiger-left tiger-right

start:
uniform

T: listen
0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0
0.5 0.5 0.0 0.0 0.0 0.0
0.5 0.5 0.0 0.0 0.0 0.0

T: open-left
0.0 0.0 0.5 0.5 0.0 0.0
0.0 0.0 0.5 0.5 0.0 0.0
0.0 0.0 0.0 0.0 0.5 0.5
0.0 0.0 0.0 0.0 0.5 0.5
0.5 0.5 0.0 0.0 0.0 0.0
0.5 0.5 0.0 0.0 0.0 0.0

T: open-right
0.0 0.0 0.5 0.5 0.0 0.0
0.0 0.0 0.5 0.5 0.0 0.0
0.0 0.0 0.0 0.0 0.5 0.5
0.0 0.0 0.0 0.0 0.5 0.5
0.5 0.5 0.0 0.0 0.0 0.0
0.5 0.5 0.0 0.0 0.0 0.0

O: listen
0.85 0.15
0.15 0.85
0.85 0.15
0.15 0.85
0.85 0.15
0.15 0.85
O: open-left
uniform

O: open-right
uniform

R: listen : * : * : * -1

R: open-left : tiger-left0 : * : * -100

R: open-left : tiger-right0 : * : * 10

R: open-left : tiger-left1 : * : * -100

R: open-left : tiger-right1 : * : * 10

R: open-left : tiger-left2 : * : * -100

R: open-left : tiger-right2 : * : * 10

R: open-right : tiger-left0 : * : * 10

R: open-right : tiger-right0 : * : * -100

R: open-right : tiger-left1 : * : * 10

R: open-right : tiger-right1 : * : * -100

R: open-right : tiger-left2 : * : * 10

R: open-right : tiger-right2 : * : * -100
Elevator riding POMDP description file

This is a copy of the description file for the elevator POMDP model. It is in the format required by Cassandra’s POMDP solver. For this model, there is a 10 percent chance of receiving an incorrect observation about the person’s intention.

discount: 0.95
values: reward

states: start DcRoPoPGx DcRiPoPGx DoRoPoPGx DoRiPoPGx
        DcRoPiPGs DcRiPiPGs DcRiPiPGx DoRoPiPGs
        DoRiPiRGs DoRoPiPGx DoRiPiPGx DcRoPiPGx

actions: wait move

observations: ec px ps ee

R:*:DoRoPiPGs:DcRoPiPGs:* -20
R:*:DoRoPoPGx:DcRoPoPGx:* -20
R:*:DoRiPiRGs:DcRiPiPGs:* 100
R:*:DoRiPiPGx:DcRiPiPGx:* 20
R:*:DoRiPoPGx:DcRiPoPGx:* 100
R:*:DoRoPiPGx:DcRoPiPGx:* -100

start:
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

T:wait
0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 0.0 0.1 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.12 0.0 0.88 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.12 0.0 0.88 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.2 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.12 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.12 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.105 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0

42
T: move
0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 0.0 0.1 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0

O: wait
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0
0.0 0.0 0.0 1.0
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
0.0 0.1 0.9 0.0
0.0 0.1 0.9 0.0
0.0 0.9 0.1 0.0
0.0 0.9 0.1 0.0
1.0 0.0 0.0 0.0

O: move
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0
0.0 0.0 0.0 1.0
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
0.0 0.1 0.9 0.0
0.0 0.1 0.9 0.0
0.0 0.9 0.1 0.0
0.0 0.9 0.1 0.0
1.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
Reactive person controller for elevator simulator

The binary features that make up the state space for the controller are:

- AD  at destination floor
- IN  in elevator
- EC  elevator called
- P_B  person in button area
- CE  can exit/enter
- ED  elevator direction (relative to direction to destination)
- P_OFF_T  person getting off at this floor
- P_ON  person getting on elevator at this floor

The high level actions that a person may execute are:

- enter/exit elevator
- move to button area and push button
- ask another person to push button
- move to waiting area and wait

Not at destination floor, outside elevator:

1) !AD,!IN,!EC,!P_B,!CE,*ED, *P_OFF_T
elevator not called, no one in button area, door closed
move to and press button

2) !AD,!IN,!EC,P_B,!CE,*ED, *P_OFF_T
elevator not called, person in button area, door closed
ask person to press button for you

3) !AD,!IN, EC, *P_B,!CE, *ED, *P_OFF_T
elevator called, door closed
move to waiting area and wait
   elevator not going in your direction
   wait

5) !AD, !IN, *EC, *P_B, CE, ED, P_OFF_T
   elevator going in your direction, door open, person getting off at this floor
   wait

6) !AD, !IN, *EC, *P_B, CE, ED, !P_OFF_T
   elevator going in your direction, door open, no one getting off at this floor
   enter elevator

Not at destination floor, inside elevator:

7) !AD, IN, !EC, !P_B, *CE, ED, *P_OFF_T
   elevator not called, no one in button area, elevator going in your direction (or no direction)
   move to and press button

8) !AD, IN, !EC, !P_B, *CE, !ED, *P_OFF_T
   elevator not called, no one in button area, elevator not going in your direction
   move to waiting area and wait

9) !AD, IN, !EC, P_B, *CE, *ED, !P_OFF_T
   elevator not called, person in button area, no one getting off at your destination floor
   ask person to press button

10) !AD, IN, !EC, P_B, *CE, *ED, P_OFF_T
    elevator not called, person in button area, someone getting off at your destination floor
    move to waiting area and wait

    elevator called
    move to waiting area and wait

At destination floor, inside elevator:

12) AD, IN, !EC, *PB,!CE, *ED, *P_OFF_T
door closed, elevator not called
   go to button area and press button

   door closed, elevator called
   wait

   door open
   exit elevator
References


[20] Nidhi Kalra Anthony Stentz M. Bernardine Dias, Robert Zlot, „”.


[54] Anthony R. Cassandra, “Pomdp solver software,”.

