Sidekick Agents for Sequential Planning Problems

by

Owen Macindoe

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

Author .....
Department of Electrical Engineering and Computer Science
August 10, 2013

Certified by .....
Leslie Pack Kaelbling
Professor
Thesis Supervisor

Certified by .....
Tomás Lozano-Pérez
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Sidekick Agents for Sequential Planning Problems

by

Owen Macindoe

Submitted to the Department of Electrical Engineering and Computer Science
on August 23, 2013, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Effective AI sidekicks must solve the interlinked problems of understanding what their human collaborator's intentions are and planning actions to support them. This thesis explores a range of approximate but tractable approaches to planning for AI sidekicks based on decision-theoretic methods that reason about how the sidekick's actions will effect their beliefs about unobservable states of the world, including their collaborator's intentions. In doing so we extend an existing body of work on decision-theoretic models of assistance to support information gathering and communication actions. We also apply Monte Carlo tree search methods for partially observable domains to the problem and introduce an ensemble-based parallelization strategy. These planning techniques are demonstrated across a range of video game domains.

Thesis Supervisor: Leslie Pack Kaelbling
Title: Professor

Thesis Supervisor: Tomás Lozano-Pérez
Title: Professor

Acknowledgments

To Sophie Mackey, the love of my life, for following her heart and me. To Jan and Neil Macindoe, for their endless encouragement and support from afar. To Whitman Richards, for teaching me how to run marathons and wave flags. To the staff and students of the Singapore-MIT GAMBIT Game Lab, for taking games seriously, without forgetting how to have fun. To Leslie Pack Kaelbling and Tomás Lozano-Pérez, for their wisdom and guidance. To all the members of LIS, for their valuable feedback and insights.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the ongoing goals of AI is to produce intelligent computational agents that can collaborate with a human colleague to work together towards a common goal. In video games in particular, game designs often call for AI programmers to build AI assistants, or sidekicks, that must work alongside human players to achieve the game's objectives. When a human could be pursuing any one of a number of possible objectives in a game, the challenge for an AI sidekick involves understanding what their human colleagues are trying to achieve, planning how the agent ought to help given that understanding, and realizing when they need more information about the world or the human's intentions and determining how to get it. These are challenging problems which game designers are often forced to avoid or mitigate through specific design choices. This thesis develops planning mechanisms for AI assistants that infer and influence the intentions of their human colleagues through observation and communication, adjusting their behavior to help the human based upon their understanding of human intentions.

In video games an AI sidekick may dynamically take over from a player that has lost network connectivity in a cooperative game and carry on in their place, as in World in Conflict [20]. Alternatively, interacting with AI sidekicks may be central to

a single-player video game experience, such as in the case of Yorda in Ico or Elizabeth in Bioshock Infinite [31, 25]. AI sidekicks that can understand human intentions and offer assistance are also critical for personal robotics. For instance, a robot that understands that a human is trying to clean up after a meal may help out by washing the dishes or drying and putting them away, whilst the human takes up the other role.

1.1 Sidekicks in Games

Despite their successes in some games, the repertoire of autonomous unscripted behaviors for sidekick characters has been limited. To explore some of the reasons why, we consider the case of a game where an AI sidekick was notably missing.

In the multiplayer mode of Lara Croft and the Guardian of Light (LCGL), developers Crystal Dynamics focused heavily on cooperative gameplay, with human players taking on the roles of Lara Croft and Totec, the titular Guardian of Light [19]. In single-player mode, however, the gameplay was completely changed to remove the cooperative elements, including removing Totec as an independent character, rather than implementing him as a sidekick character [14]. Crystal Dynamics' choice to completely change the game's design to circumvent the technical challenge of building a controller for Totec is a testament to the difficulty of the sidekick problem, and the capabilities that the controller would have required are a good example of the kinds of problems that a sidekick would need to face in a modern video game.

There are two core elements of gameplay in LCGL: combat and navigational puzzles. Combat involves navigating a three dimensional environment, viewed from an isometric camera, avoiding enemy attacks, and attacking enemies with Lara and Totec's weapons. Choosing the appropriate weapon, dodging attacks, and positioning yourself are key to success. Coordinating with your ally is also important, since your

actions can adversely effect them, for instance by knocking them into a bad position with an explosive, causing them to be surrounded by enemies. The dynamics of combat are non-deterministic, with weapons doing a variable amount of damage, enemy behaviors decided randomly, and game physics behaving unpredictably.

Navigational puzzles in LCGL's multiplayer mode involve combining Lara and Totec's tools and abilities to get to hard-to-reach items or sections of a level. A simple puzzle might involve Totec throwing a spear, which sticks in the wall, allowing Lara to jump onto it and reach a ledge. From the ledge she can throw down a rope for Totec to then climb up. More complex puzzles might involve pushing buttons, activating switches, swinging on ropes, and avoiding boulders, often with the steps chained together in a sequence and requiring timed coordination across different sections of a room. Additionally, there may be multiple possible paths to solving the same puzzles, each requiring Lara and Totec to coordinate in a different way.

During both combat and navigational puzzles, the game state is partially observable on multiple levels, with unobservable elements including the map outside of the player's camera view, the contents of containers such as breakable jars and barrels, the parameters governing the behavior of enemies, and the causal relationship between items in the game, such as buttons, levers, and doors. Aside from the game state, the beliefs, desires, and intentions of the players are also unobservable to one another. This partial observability means that to act reasonably a sidekick would need to keep track of its history of observations, such as where it last saw enemies before they disappeared off the screen, and use them in its deliberations.

A controller for Totec would have to cope with the stochasticity of the game dynamics, the partial observability of the game environment, and, crucially, with the unobservable intentions of the human player, to which it only has clues from the history of its observations of the player's actions. As we will discuss in Chapter 2, these elements make games like LCGL examples of a particular class of problem, that can

formally be modeled as a partially observable Markov decision process (POMDP). Optimally solving POMDPs is a computationally intractable problem, so it is unsurprising that Crystal Dynamics shied away from the challenge.

1.2 Sequential Planning

Throughout this thesis we will be concerned with sidekick AI for sequential planning problems, the simplest of which are fully observable. In a fully observable sequential planning problem an agent is presented with the state of the world, which fully describes every detail about the world which could be important for making a decision about what to do, and is presented with a set of actions to choose from. Picking one of these actions then causes the world state to change, via the world's state transition dynamics, to another world state, in which the agent may pick another action, and so on. State transitions may be stochastic, which the agent needs to take into account when selecting its actions. In addition, after each transition between states, the agent receives a reward signal that tells it how good the chosen action was. Stochastic sequential planning problems are modeled by Markov decision processes (MDPs), which we shall introduce formally in Section 2.3.

A solution to a stochastic sequential planning problem is called a policy. Policies are sets of decision rules that pick an action based on the current state of the world. A good policy is one that maximizes the expected rewards gained over time. Policies can be stationary, meaning that they are constant over time, or non-stationary, meaning that the decision rules can change on each time step. A policy can take many forms, from an explicit mapping between states and actions, computed in a table, to a dynamic procedure that produces an action on the fly for each state with which it is presented. A procedure for finding a policy is called a planner and a procedure that governs the execution of a policy is called a controller.

Solving sequential planning problems involves reasoning about how an agent's actions will affect the state of the world, but in a partially observable sequential planning problem, an agent does not have direct access to the state of the world. Instead, it receives a series of observations, such as readings from a robot's laser rangefinder or a human collaborator's speech, that give it clues to what the true state might be. This complicates matters, since the agent then needs to reason what to do, given its history of observations and the actions that it has taken, without knowing exactly what the true state is. This situation is modeled by partially observable Markov decision processes, which we shall introduce in Section 2.4.

In both the fully and partially observable cases we are interested in building a planner that is robust in the face of uncertainty. In the real world, actions often have uncertain outcomes or have to be made with limited information.

Uncertain outcomes can take the form of pure chance, or represent the limits of modeling fidelity, such as abstractly representing the possibility of a robot's wheel slipping in mud, without fully representing the physics of the situation in the model.

Limited information stems from sensing failures, such as occluded vision, or inaccuracies in a laser rangefinder's readings. For AI sidekicks, another key case of limited information is in reasoning about the beliefs, desires, and intentions of their colleagues. These internal mental states are unobservable by their very nature. A key contribution of this thesis will be to present methods for reasoning about a collaborator's unobserved intentions and planning in the face of uncertainty about them.

An agent can represent its current estimate of the state of the world as a probability distribution over states, called a belief. Given a model of the domain's dynamics and also of how the state of the world produces observations for the agent, the agent can reason about how its actions will update its belief about the state of the world. This kind of reasoning is called belief-space planning. In contrast to the fully observable case, a policy for a partially observable sequential planning problem is a set of

decision rules mapping belief states, rather than world states, to actions.

This thesis focuses strongly on the case of video games, in which there is already a rich tradition of AI sidekicks. Video games have the special benefit, which sets them apart from real world robotics, of being embedded inside of virtual worlds which can be made fully observable to an AI sidekick. Indeed, game developers often explicitly reduce direct access to the state by AI characters in games, through modeling audio and visual sensor dynamics within the game system in an attempt to generate more believable behaviors from their AI and to give human players a sense of fairness. A human collaborator's intentions, on the other hand, are not directly observable and are an important hidden part of the state. We will consider both the case where the human's intention is the only unobserved part of the state and the case where the sidekick must cope with a more general form of partial observability.

1.3 Thesis Structure

This thesis presents two major approaches to tackling the problem of building controllers for AI sidekicks. We first formalize the AI sidekick problem in Chapter 2, introducing decision-theoretic planning, and casting it as a POMDP. As part of this formulation we discuss modeling human beliefs and goals, which is critical for the subsequent two chapters. We also place our work within the context of existing literature on decision-theoretic planning, multi-agent systems, and current practices in video game development.

In Chapter 3 we present an approach to the AI sidekick problem that makes use of offline state-space planning for both constructing human models and planning actions. To deal with partial observability, communication, and information gathering actions, we introduce a range of heuristic methods that build on top of a state-space planning framework. We then evaluate these methods experimentally on two domains, pairing

them with simulated humans.

Chapter 4 presents an alternative approach to the AI sidekick problem that plans online in belief space using Partially Observable Monte Carlo Planning (POMCP), which naturally handles partial observability, communication, and information gathering actions. To improve the performance of POMCP, we introduce ensemble methods for parallelizing the search algorithm. We evaluate several varieties of POMCP planning for the AI sidekick problem over three domains, again pairing them with simulated humans.

We conclude in Chapter 5 by summarizing our key results and contributions and proposing future directions for research.

Chapter 2

The AI Sidekick Problem

There are two related capabilities that an effective controller for an AI sidekick must have. The first is that it must be able to understand the intentions of its human collaborator and the second is that it must be able to choose reasonable actions to help the human, given its understanding of their intentions. An ideal sidekick should also be able to combine these capabilities, so that it if it is uncertain about a human's intentions, and the actions that it ought to take hinge critically upon knowing them, then it should be able to choose actions that can help reveal information about those intentions.

In addition, in many games a sidekick must cope with limited information not only about human intentions, but also about the game world. For instance, a sidekick's vision may be blocked by walls in the game world or limited by the size of the display in the case of a top-down camera. Other entities in the world, such as enemies, may also have hidden state that the sidekick has no access to, but must infer from interacting with them.

The AI sidekick problem is that of building a controller that can choose actions which help to support a human collaborator's intentions in the face of uncertainty about those intentions and also about the true state of the environment. In this

chapter we will formalize this problem as a partially observable Markov decision process, or POMDP. We will first introduce Cops and Robbers, a game that we will use throughout this chapter in explaining the different parts of the POMDP model, and that will also later serve as a test-bed for the sidekick controllers that we develop. We will develop our POMDP formulation in two stages, first by considering the case of trying to coordinate with a human player whose play strategy is known, and then by considering the case where it must be inferred by the sidekick from some fixed set of possibilities.

2.1 Cops and Robbers

In Cops and Robbers, shown in Figure 2-1, the human player and the sidekick are cops, chasing robbers who are fleeing from them through the corridors of a building full of twisting passages and one-way doors. The object of the game is to catch any one of the robbers as quickly as possible, but it takes two cops to overpower a robber, so the player and their sidekick must coordinate to corner and catch a robber. The presence of one-way doors means the cops have to be careful about the routes they take or they could find themselves wasting a lot of time if they go the wrong way. From the perspective of the sidekick, the game state is fully observable, with the exception of the human's intentions which it cannot directly observe.

The player's perspective is from a bird's eye view, with the positions of all of the cops and robbers fully visible. The player and their sidekick take turns moving, with the robbers moving after the sidekick. If there is a cop within two squares of them, a robber will move in the direction that takes them furthest from the nearest cop, breaking ties randomly. A key strategy for the game is to cut off robbers' escape routes so that the two cops can trap them. To help coordinate this, the sidekick is allowed to ask the player which robber they should aim for, but doing so spends a

turn for the sidekick to ask and for the human to respond.

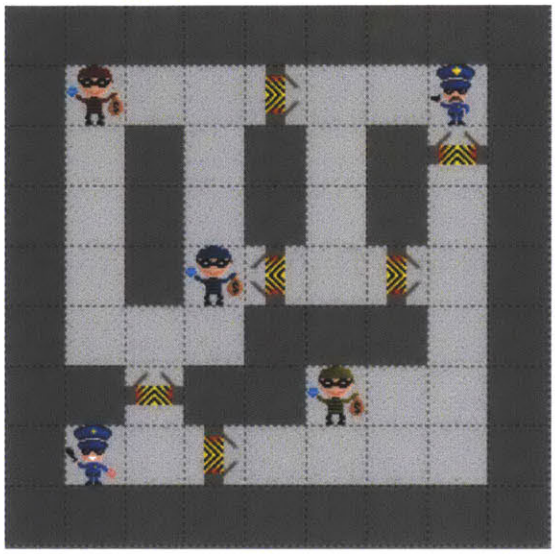


Figure 2-1: The Cops and Robbers play field. The human is in the top right, the sidekick is in the bottom left. Chevrons indicate the direction of one-way doors.

Cops and Robbers is a special domain for two reasons. Firstly, the task of catching each individual robber can be seen as a separate subgoal that can be pursued completely independently of the other robbers. Secondly, assuming that the human knows the rules of the game, believes that the sidekick both knows the human's intentions, and also knows how the sidekick is trying to coordinate with them, then from the human's perspective the game is fully observable. These two properties will be particularly relevant in Chapter 3, where we introduce a planning approach that takes advantage of them.

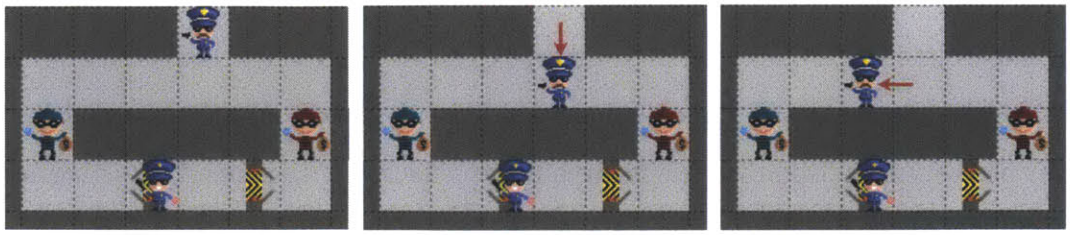
The one-way doors are a key feature of Cops and Robbers, since they accentuate the importance of taking uncertainty about the human's intentions into account. If the sidekick is over-eager in pursuing the wrong robber they may commit to going through a door that prevents them from easily getting to the part of the maze containing the human's real intended target. Although Cops and Robbers in principle

allows for the game to be won by catching any robber, we assume that the human would prefer to be in command, rather than being forced to follow the sidekick's agenda, so it is undesirable for the sidekick to put the two of them into a position where the human is forced to change targets.

2.1.1 Playing Cops and Robbers as the Sidekick

As an example of the kind of reasoning that we would ideally like a sidekick controller to be able to perform, consider the situation in Figure 2-2a, with it being the human's turn to move. Say that the sidekick was initially unsure which of the two robbers, red or cyan, that the human is chasing. If the human were to move south, as shown in Figure 2-2b, its target would remain ambiguous; it could be chasing either robber and would still make the same move. After the human moves, it is the sidekick's turn and it is faced with a choice of moving west to chase the cyan robber, moving east to chase the red robber, or waiting. Note that moving west would cause the sidekick to step through the one-way door on which it is standing, which would mean it would have to take the long way around were the human in fact chasing red. As the sidekick we might reason that waiting for the human to make its move first would be best, since then we would have a clearer idea of their intentions. For instance, if we were to pass and the human to move west as in Figure 2-2c, and we had faith that the human was reasonably competent, then we could be confident that they were chasing the cyan robber. Having seen them make that move, we could then safely step through the one-way door and help them catch cyan.

The key point here is that in doing this kind of reasoning, we are reasoning about our own belief about the human's hidden intentions, and about how our beliefs about those intentions will change as we select actions. We reasoned that in choosing to pass we would have more information with which to make a better-informed decision



(a) Initial state. (b) Human moves south. (c) Sidekick passes, human moves west.

Figure 2-2: Two rounds of moves in a game of Cops and Robbers.

on our next move and could avoid making a blunder. This kind of reasoning about how our actions affect our beliefs is called belief-space planning.

Thinking about how our actions will affect our beliefs is particularly important for reasoning about communication. For instance, if instead of passing, we directly asked the human which robber they were chasing, we can reason that, assuming they were cooperative, their response would almost certainly give us all the information we needed about their intentions to be able to make a good decision about which way to go. So we can conclude that asking a question is beneficial in that situation, because it will allow us to update our belief about the human's intentions in a way that lets us make good decisions in the future.

2.2 Multi-Agent Systems and Decision-Theoretic Planning

Autonomous decision makers are often referred to as agents in the AI community. The study of systems containing interacting agents is the field of multi-agent systems (MAS) and as such, the AI sidekick problem, which focuses upon the interaction between two agents, one being an AI and the other being a human, is a MAS problem. MAS is a broad and interdisciplinary field, involving researchers in psychology,

economics, logic, philosophy and artificial intelligence. Since multi-agent systems subsume single decision makers, to some extent the MAS field subsumes AI itself, but MAS researchers are typically particularly concerned with issues such as how teams of agents can organize themselves to act effectively in concert or adversarially, communicate efficiently, and model each other's beliefs, desires, and intentions. In this thesis we will restrict our discussion of the field to a particular tradition in MAS based upon decision-theoretic planning. For a broader view of MAS, a number of good introductory and survey texts exist [62, 59, 63].

Decision-theoretic planning is a paradigm for studying decision making with its roots in operations research, control theory, and game theory. It is primarily concerned with rational, optimal, sequential decision making, particularly in stochastic and partially observable domains. Operations research typically studies decision-making problems in the context of logistics and scheduling, whereas control theory studies them in the context of continuous dynamical systems. Game theory deals with decision making in the context of multiple, possibly adversarial, decision makers, and is often considered a separate field, but there are strong connections between game theory and decision-theoretic planning, particularly when agents are interacting repeatedly over time [46]. In contrast to the very general setting of game theory, decision-theoretic approaches to multi-agent systems typically assume that the agents are cooperative. Decision-theoretic planning has the virtue of formalizing the notion of decision and plan quality, which distinguishes it from other approaches to decision making in multi-agent systems.

The computational complexity of decision-theoretic planning depends heavily upon the modeling of the planning problem. Throughout this thesis we will make a key assumption, which is that the human will be acting according to some set of behavior profiles that the sidekick knows in advance. The sidekick may not know which particular behavior profile the human is following, but having a human model that

captures these behaviors allows for planning actions that can help reveal which profile the human is following and also assist them. This assumption makes the AI sidekick problem a partially observable Markov decision process (POMDP). Optimal planning for a POMDP is PSPACE-complete if we want to model games that run for a fixed length of time and undecidable if there is no time limit [49, 42]. However, there are a number of approximate planning techniques that we can use.

Requiring that we have an accurate human model is a strong assumption, but relaxing this assumption makes the computational complexity of the problem much more daunting. In the most general case, with no constraints on the kinds of expected human behavior, and in particular with the human and the sidekick potentially having different reward functions, then the problem is a partially observable stochastic game, for which optimal planning is NEXP^{NP}-hard [27]. A less extreme alternative is to assume that the humans will act as if they had previously conferred with the sidekick and come up with an optimal joint plan that respects the uncertainty of the domain, in which case the problem is a decentralized partially observable Markov decision process (Dec-POMDP), for which optimal planning is still NEXP-complete [6].

In our formulation we will not make any assumptions about whether the human is in turn trying to model the sidekick, however there is a related formalism, the finitely nested interactive POMDP, or I-POMDP, that explicitly models this possibility. Agents in a finitely nested I-POMDP are modeled as having a level which determines the degree of belief nesting that they will consider for each other agent in the domain. The finite nesting stipulation enforces the constraint that agents model other agents as having a strictly lower level than themselves, which guarantees that their recursive beliefs about each other will ground out with an agent model that does't model any other agents.

Finding exact an optimal policy for an I-POMDP is still PSPACE-complete for finite horizons and undecidable for the infinite horizon case, as with all POMDPs [17].

State of the art approximation techniques for I-POMDP planning involve representing and updating the agent's nested beliefs using nested particle filters and using value iteration directly on this representation [18].

The human models that we will discuss in this thesis do not attempt to model the sidekick's beliefs, and as such our overall setup is closely related to a level-one I-POMDP, in which one agent models the internal state of the other, but not vice versa. A richer set of human models that allowed for recursive reasoning about one-another's beliefs could make fuller use of the I-POMDP formalism.

In the next section we will begin introducing the POMDP formalism, by considering the completely observable case, where the sidekick knows in advance precisely what the human is planning to do, which makes the problem a completely observable Markov decision process (MDP).

2.3 Markov Decision Processes

A Markov decision process is a formal model used in decision-theoretic planning with their roots in the operations research community. They are used to model problems in which an agent must choose a sequence of actions in interacting with a system whose state updates probabilistically in response to the agent's actions. Games with complete information, and particularly board games whose rules explicitly lay out how the game state updates in response to player actions, are good examples of the kinds of domains that can be modeled as MDPs. There is a rich body of research on Markov decision processes and this thesis will only be able to touch on a small part of it, but a more thorough discussion can be found in other texts [7, 52]. Figure 2-3 shows the structure of a system modeled by an MDP, whose components we will discuss in detail.

At any given time, the system is in some particular state out of the set of possible

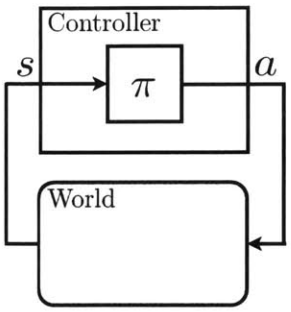


Figure 2-3: An MDP models an agent's interaction with the world.

states, $\mathcal{S}$. The random variable S^t, will represent the state of the system at time t. At each time step, the agent interacting with the system must select an action from $\mathcal{A}$, the set of possible actions. The random variable A^t, will represent the action chosen by the agent at time t. A rule for deciding which action to select at time t given one's experiences up to t is called a policy and the challenge of planning is to find the policy that selects the best such actions.

We will focus on the case where time is divided into discrete points at which the agent is allowed to make decisions, although the MDP model can be applied to the continuous time case, too, at the cost of additional complexity [7, 52]. For the moment we will also only consider discrete states and actions, although when we turn to POMDPs later we will discuss continuous state spaces.

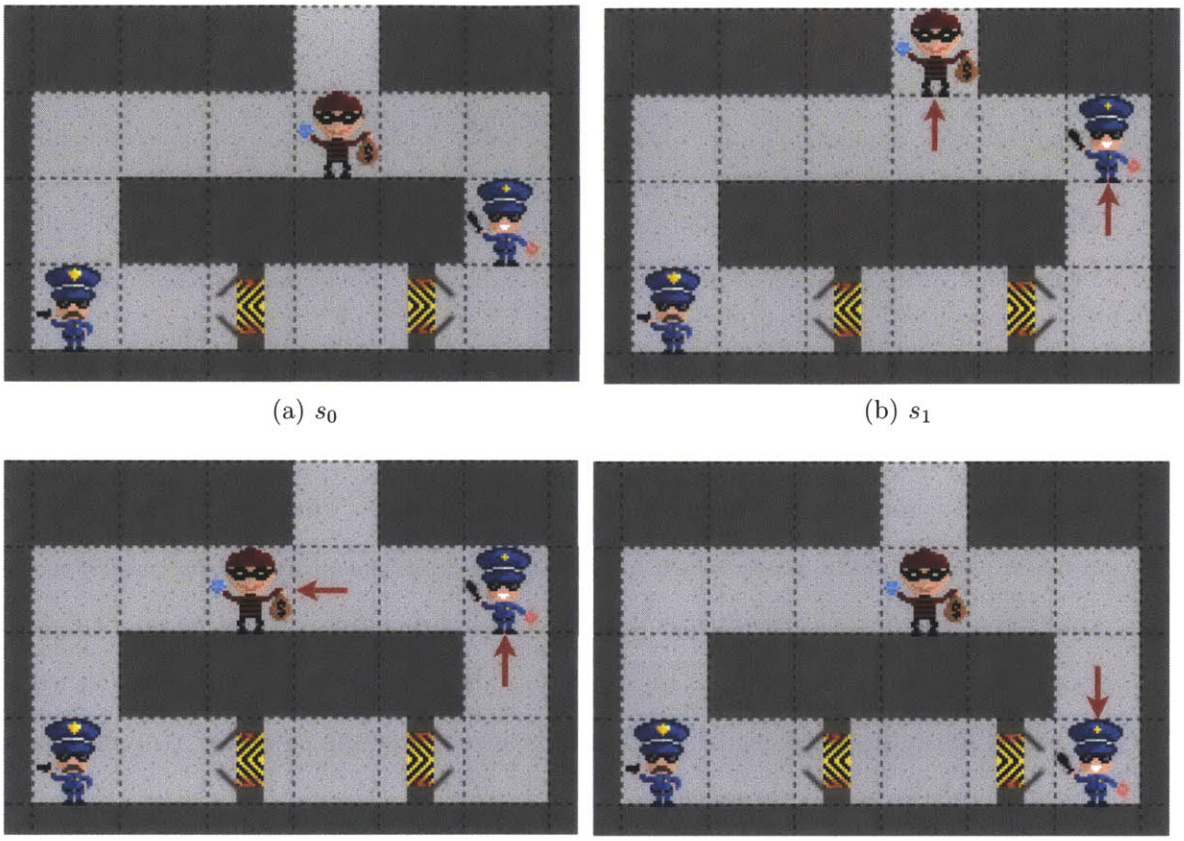
Example: In Cops and Robbers, leaving aside the complication of the human's internal state and intentions, a state is a complete configuration of the game, meaning the positions of each of the robbers, the human, and the sidekick. $\mathcal{S}$ is the set of all such configurations. $\mathcal{A} = \{\mathbf{n}, \mathbf{e}, \mathbf{s}, \mathbf{w}, \mathbf{x}, \mathbf{?}\}$ is the set of movements in each of the cardinal directions, the pass action, and the communication action that asks the human for their target. Note that although Cops and Robbers is a multi-player game, the problem we are solving is choosing sidekick

actions, which is a single-player problem, so we are representing Cops and Robbers as a system experienced specifically from the sidekick's perspective.

An MDP models how a system's states change over time in response to actions via a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$, which maps state and action pairs to a probability distribution over successor states. We will write $\tau(s, a, s') = \Pr(S^{t+1} = s' | S^t = s, A^t = a)$ to indicate the individual transition probabilities. Note that the distribution over successor states depends only on the current state of the system and the most recent action chosen by the agent. This is the eponymous Markov property of the MDP. Almost all perfect-information games have this property, as do many other domains.

Example: Consider the Cops and Robbers game state s_0, shown in Figure 2-4a. We will again leave aside the human and communication actions for the moment. If the sidekick chooses to move north, it will be within 2 spaces of the red robber, causing it to flee. According to the rules of Cops and Robbers, since the spaces immediately to the north and the west are tied for being furthest from the sidekick, it will choose one of them to flee to uniformly at random, so $\tau(s_0, \mathbf{n}, s_1) = 0.5$ and $\tau(s_0, \mathbf{n}, s_2) = 0.5$. If the sidekick moves south, the robber is not in range and won't try to flee, so $\tau(s_0, \mathbf{s}, s_3) = 1.0$. All other actions lead to the sidekick standing still, out of range of the robber, so for each of those actions a, $\tau(s_0, a, s_0) = 1.0$. Furthermore, these are the only possible outcomes, so for every action a, $\tau(s_0, a, s') = 0.0$, where s' is not one of the states previously mentioned.

In games, and many other domains, some states and actions are considered more valuable than others. For instance, in chess the best possible state is one in which your opponent is in checkmate and the best possible action is one which leads to an inescapable checkmate. In other games, players may be trying to maximize their score, and a good action is one that improves their score. A great action, on the



(a) s_0 (b) s_1

(c) s_2 (d) s_3

Figure 2-4: State transitions in a game of Cops and Robbers.

other hand, may be one that scores less now, but sets up strong scoring moves later on. In other cases the reward may not even be an explicit part of the game rules, but rather some more abstract criteria, like coordinating well with other players or satisfying internal desires.

To model the relative immediate value of starting in a given state of performing an action and then making a transition to some other state, we use the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, where the output is a real-valued reward signal. Since we will often be interested chiefly in evaluating the quality of an action in a given state, we introduce the function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which computes the expected immediate reward

of performing a in s,

$$r(s,a) = \sum_{s' \in \mathcal{S}} \tau(s,a,s')\mathcal{R}(s,a,s'). \tag{2.1}$$

Example: In Cops and Robbers the object of the game is to catch a robber by both standing on the same square as it. Let us call all such states capture states. There are several possible reward functions that could model this criteria. One straightforward function is $\mathcal{R}(s,a,s') = 1$ only in the case where s' is a capture state and 0 otherwise. Another possibility is that $\mathcal{R}(s,a,s') = 1$ in the case where s is a capture state, and 0 otherwise. This alternative makes sense if we model the game's ending as being a transition to an absorbing state that occurs after a capture state. Finally, we could change either of these functions to take into account the human's intentions, so that for a capture state to result in a non-zero reward, it further requires that the captured robber was the human's intended target.

Taken as a tuple, the states $\mathcal{S}$, actions $\mathcal{A}$, transition function $\mathcal{T}$, and reward function $\mathcal{R}$ constitute an MDP, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. Solving an MDP means finding a policy that selects actions sequentially such that they maximize some optimality criterion. There a variety of possible criteria, of which we will focus on expected future discounted reward. This criterion combines two key ideas. First, we are interested not just in how good an action is immediately, but also in how taking an action will affect the possibility of good outcomes in the future. Second, and somewhat counteracting this first idea, is that although we care about our future rewards, we weigh the shorter term future more heavily than the longer term future in our considerations.

We model this using a discount factor γ with a value in the range $[0,1]$ that controls how strongly we favor short term rewards over long term ones. The expected future discounted reward is then

$$\mathbb{E}\left[\sum_{t=0}^{T-1}\gamma^t r(S^t, A^t)\right], \tag{2.2}$$

where T is the maximum number of steps or planning horizon. If we introduce the additional constraint $\gamma \neq 1$, we can similarly compute the expectation for the infinite horizon case

$$\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(S^t, A^t)\right], \tag{2.3}$$

where this constraint ensures that the expectation will be bounded.

For some domains, $1-\gamma$ can be interpreted as the probability that on each step the game will end and enter a zero-reward absorbing state. In games domains this could model some probability that the human decides to simply stop playing, so the sidekick should be incentivized to have its actions pay off before the human quits. Alternatively it can be seen as modeling a psychological bias towards short term rewards.

Expected future discounted reward is just one of several possible optimality criteria that can be applied in MDP models. Other possibilities include maximizing immediate rewards or mean lifetime rewards. A fuller account of these kinds of criteria can be found elsewhere [7, 52].

2.3.1 Modeling Humans

So far we have put aside discussion of the human's internal state and actions. The strategy that we will use in this chapter for formulating a tractable multi-agent system model is to treat the human's state as a part of the system's state and both the choice and effects of their actions as part of the transition function. Figure 2-5 shows the structure of the human model in the fully observable MDP case.

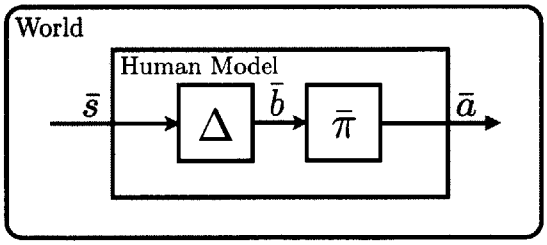


Figure 2-5: Human model structure in an MDP.

The human's state is a part of the system external to the sidekick, but we want to distinguish between the part of the system's state that represents the human and that which represents the rest of the system. Let $\bar{\mathcal{B}}$ be the set of such human states and the random variable $\bar{B}^t$ represent the human's state at time t. Similarly, let $\bar{\mathcal{A}}$ be the set of possible human actions and the random variable $\bar{A}^t$ represent the human's action at time t. Let $\bar{\mathcal{S}}$ then be the set of states excluding the human's state and $\bar{S}^t$ be the corresponding random variable representing the state at time t. We can then write the full state S^t as the tuple $(\bar{S}^t, \bar{B}^t)$. Relatedly, we will be interested in the intermediate state of the world immediately after the sidekick has acted, but before the human has acted, again, excluding the human's internal state. Let $\tilde{\mathcal{S}}$ be the set of such states and $\tilde{S}^t$ be the corresponding random variable representing the state at time t.

We will separate out the effects of sidekick and human actions into multiple steps. The dynamic Bayesian network in Figure 2-6 shows the causal relationship between the different variables involved. The thin dotted lines indicate the functions discussed in this section that are responsible for updating the variables. The thick dashed lines indicate the set of steps modeled by the complete transition function.

The first step, $\mathcal{T}_s : \mathcal{S} \times \mathcal{A} \to \Pi(\tilde{\mathcal{S}})$, models the probabilistic effect of the sidekick's

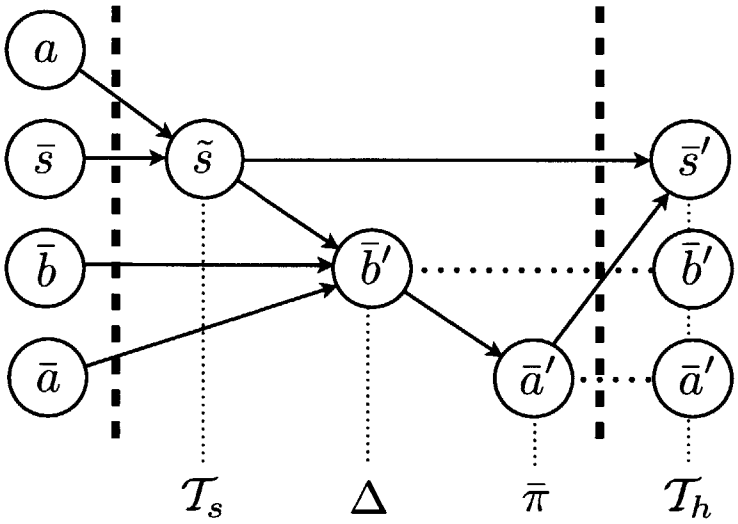


Figure 2-6: The transition function displayed as a dynamic Bayesian network for the MDP case.

actions on state of the world, excluding the human's internal state. We write

$$\tau_s(s, a, \tilde{s}) = \Pr(\tilde{S}^t = \tilde{s} | S^t = (\bar{s}, \bar{b}), A^t = a) \tag{2.4}$$

$$= \Pr(\tilde{S}^t = \tilde{s} | \bar{S}^t = \bar{s}, A^t = a) \tag{2.5}$$

for the individual probabilities. Note $\bar{b}$ has no causal influence on this update, so it can be eliminated from the conditional.

The human's internal state is updated stochastically at each time step by the internal update function $\Delta : \tilde{\mathcal{S}} \times \bar{\mathcal{B}} \times \bar{\mathcal{A}} \to \Pi(\bar{\mathcal{B}})$. We write $\delta(\tilde{s}, \bar{b}, \bar{a}, \bar{b}') = \Pr(\bar{B}^{t+1} = \bar{b}' | \tilde{S}^t = \tilde{s}, \bar{B}^t = \bar{b}, \bar{A}^t = \bar{a})$ to indicate the individual transition probability.

After updating its internal state according to Δ, the human then stochastically selects an action based on its current state and its policy, $\bar{\pi} : \bar{\mathcal{B}} \to \Pi(\bar{\mathcal{A}})$. This policy represents a behavior profile or decision-making process that guides the human's actions, given its current beliefs, desires, and intentions. We write

$m(\bar{b}, \bar{a}) = \Pr(\bar{A}^{t+1} = \bar{a} | \bar{B}^{t+1} = \bar{b})$ to indicate the individual action probabilities. Here the time is $t+1$ because the time step is relative to the sidekick's decisions.

Finally, $\mathcal{T}_h : \tilde{\mathcal{S}} \times \bar{\mathcal{A}} \to \Pi(\bar{\mathcal{S}})$ models the effects of the human's actions on the state. We write $\tau_h(\tilde{s}, \bar{a}, s) = \Pr(\bar{S}^{t+1} = \bar{s} | \tilde{S}^t = \tilde{s}, \bar{A}^{t+1} = \bar{a})$ for the corresponding individual transition probabilities.

A fully observable human model is then the tuple of human states, internal update function, and policy $H = (\bar{\mathcal{B}}, \Delta, \bar{\pi})$. Given this formulation of the human model and transition functions, τ can be written:

$$\tau((\bar{s}, \bar{b}), a, (\bar{s}', \bar{b}')) = \sum_{\tilde{s} \in \tilde{\mathcal{S}}} \tau_s(\bar{s}, a, \tilde{s}) \delta(\tilde{s}, \bar{b}, \bar{a}, \bar{b}') \sum_{\bar{a}' \in \bar{\mathcal{A}}} m(\bar{b}', \bar{a}') \tau_h(\tilde{s}, \bar{a}', \bar{s}'). \tag{2.6}$$

An important consequence of this formulation is that the sidekick's selected action must be recorded as part of the state so that it can be used in the next time-step's state update.

Example: Consider a possible human model for Cops and Robbers. Say that the human had decided on a strategy of chasing the closest robber at each time step, breaking ties between robbers and equally good actions by some arbitrary ordering, and relying on the sidekick to plan around that strategy. We can model a human following this strategy by letting $\bar{\mathcal{B}}$ be the set of intentions to chase each individual robber: $\delta(\tilde{s}, \bar{b}, \bar{a}, \bar{b}') = 1$ if $\bar{b}'$ is the intention to chase the robber closest to the human in $\tilde{s}$ and 0 otherwise. Additionally, $m(\bar{b}, \bar{a}) = 1$ if $\bar{a}$, is the action that moves the human closest to the robber and 0 otherwise.

Formulating the human model in this way makes two key assumptions. The first is that we can actually know Δ and $\bar{\pi}$ for a domain. This is critical, and an assumption that we will carry across to the partially observable case. This assumption means that we know how the human's beliefs, desires, and intentions evolve over time in

response to changes in the game state, how their decision making process works, and how the human will respond to the sidekick's actions. The alternative would be to fall back on a game theoretic analysis that considers all the human's possible responses, which makes the problem far more difficult to both formulate and solve.

This is a strong assumption, since for many domains there is no intuitively obvious model of human action. In Chapter 3 we will present one possible method for automatically generating human models, but as we will see, this method has limits. In many cases, human models will have to either be hand engineered, or possibly learned from data.

There are several possible interpretations of what this assumption means. One possibility is that the human told the sidekick in advance exactly how they were planning to behave. This is similar to the interpretation of Dec-POMDP planning that assumes agents will mutually agree on a strategy that they are going to jointly execute, with the key difference that the human unilaterally declares their plan and forgoes any option to change their mind after telling the sidekick. This means that planning from the sidekick perspective will be a search for an optimal response to a fixed human strategy, rather a search for strategy that is jointly optimal for the human and the sidekick. Another possibility is that the model could represent some ideal human that the sidekick is expecting to be playing with, or even sub-optimal human players whose deficiencies the sidekick expects it may have to make up for. The idea here is that the model is not expected to exactly match the actual human, but the hope is that, given the assumptions that the sidekick makes about the real human's behavior based on the model, the sidekick will be able to plan reasonable actions, even if the model doesn't exactly match reality.

The second assumption is that the sidekick has direct access to the human's current beliefs, desires, and intentions. This makes the planning problem much easier, because the sidekick doesn't have to consider the case that it might be confused or

wrong about what the human wants. We will relax this assumption and consider the partially observable case in Section 2.4.

2.3.2 Solving MDPs

Having formulated the AI sidekick problem as an MDP in this way, with the human model forming part of the MDP dynamics, we could then apply standard MDP solution techniques to solve the MDP and find an optimal policy for the sidekick in the fully observable case. We could then use the policy as a controller for the sidekick. This would be an appropriate thing to do if there could be no doubt as to the human's beliefs, desires, and intentions, so modeling uncertainty about the human's internal state would be unnecessary. We will also be using these solution methods in our approximate methods for approaching POMDPs in Chapter 3.

2.3.2.1 Policies

The most general form of a policy is a mapping of state-action histories to actions. It constitutes a plan for what action an agent ought to take, given any possible sequence of interactions with a system. In the case of a fully observable MDP, it can be shown that optimal behavior can be achieved even if only the current state is considered when selecting an action, and the rest of the history is ignored [52]. A policy that takes advantage of this property is called a Markov policy. The policy that when followed maximizes the optimality criteria for the MDP, which in our case is expected future discounted reward, is called a solution to the MDP. Finding such a policy is called solving the MDP and the optimal policy itself is denoted π^*. In general, an MDP may have multiple solutions.

The structure of a Markov policy for an MDP differs depending on whether the MDP is finite horizon or infinite horizon. A decision rule for time t is a function

$d_t : \mathcal{S} \rightarrow \mathcal{A}$ that maps a state to an action when there are t steps remaining. In the finite horizon case a policy $\pi = (d_T, d_{T-1}, ..., d_1)$ is made up of a sequence of decision rules, one for each time step. A policy with this form is called a non-stationary policy, because the decision rule that applies changes over time.

In the infinite horizon case, a Markov policy can be written in the form $\pi = (d, d, ...)$ with the same decision rule being used at every step, indicated by the lack of a subscript. Such a policy is called a stationary policy. It can be proven that for any infinite horizon MDP, there is a stationary policy that is optimal [52]. We will use π to represent both stationary and non-stationary policies. Whenever there might be confusion, we will draw an explicit distinction between the two.

2.3.2.2 Value Functions and $\mathcal{Q}$-Functions

A key conceptual tool in solving MDPs is the value function of a policy, $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$, which maps a state to the expected reward of starting in that state and selecting actions according to that policy. We will write $V_t^\pi(s)$, with the t subscript for a finite horizon policy, to indicate the expected discounted future reward, or value for short, of starting in state s and following policy π when there are t time steps remaining. This value can be specified recursively with the equations

$$\begin{aligned} V_0^\pi(s) &= 0 \\ V_t^\pi(s) &= r(s, d_t(s)) + \gamma \sum_{s' \in \mathcal{S}} \tau(s, d_t(s), s') V_{t-1}^\pi(s'). \end{aligned} \tag{2.7}$$

which is known as the Bellman equation for the MDP and can be computed efficiently using dynamic programming [5].

We will often be interested in a related function, $\mathcal{Q}^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. $\mathcal{Q}_t^\pi(s, a)$, with the subscript t, is the expected value of taking action a in state s with t steps left to

go and following π from that point on. It is given by the equation

$$\begin{aligned} \mathcal{Q}_0^\pi(s,a) &= 0 \\ \mathcal{Q}_t^\pi(s,a) &= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \tau(s,a,s') V_{t-1}^\pi(s'). \end{aligned} \tag{2.8}$$

In the infinite horizon case, the time dependency is removed, and the equations become simply

$$V^\pi(s) = r(s, d(s)) + \gamma \sum_{s' \in \mathcal{S}} \tau(s, d(s), s') V^\pi(s') \tag{2.9}$$

and

$$\mathcal{Q}^\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \tau(s,a,s') V^\pi(s'). \tag{2.10}$$

An important property of the value function of the optimal policy, V^*, is that it is possible to recover the π^* from the value function using one step of lookahead. For the finite horizon case, if there are t steps remaining, then

$$\pi^*(s) = \operatorname*{argmax}_a \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \tau(s,a,s') V_{t-1}^*(s') \right]. \tag{2.11}$$

Similarly, we can substitute the identity in Equation 2.8 to write

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \mathcal{Q}_t^*(s,a) \tag{2.12}$$

which explicitly shows the relationship between $\mathcal{Q}^*$ and π^*. The infinite horizon case removes the time indexing in both equations.

An important consequence of this property is that if we can compute V^*, then we can use it to build an optimal controller by selecting actions according to π^*. One of

the best known techniques for computing V^*, from Bellman's seminal work on MDPs, is value iteration [5].

2.3.2.3 Value Iteration

The core idea behind value iteration in the finite horizon case is to use dynamic programming to exploit the recursive definition of V_T^* by first computing V_0^* and then iteratively using V_t^* to compute V_{t+1}^*. Algorithm 2.1 solves for V_t^* in this manner. We use an explicit computation of $\mathcal{Q}^*$ to demonstrate how it can be computed in the process of performing value iteration.

Algorithm 2.1 Calculate V^* – Finite horizon

for all $s \in \mathcal{S}$ **do**
  $V_0^*(s) \leftarrow 0$
end for
for all $t \in 1, ..., T-1, T$ **do**
  for all $s \in \mathcal{S}$ **do**
    for all $a \in \mathcal{A}$ **do**
      $\mathcal{Q}_t^*(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \tau(s,a,s') V_{t-1}^*(s')$
    end for
    $V_t^*(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{Q}_t^*(s,a)$
  end for
end for
return V_T^*

A similar approach can also be applied in the infinite horizon case, with the caveat that the removal of time indexing on the value function means that, rather than V_t^* being built up recursively from V_{t-1}^*, we are instead searching for a V^* that converges as the number of iterations of the algorithm increases.

Algorithm 2.2 finds an ϵ-approximation of V^* for an infinite horizon MDPs. In general, there is guaranteed to be some ϵ for which π^* and the policy computed from the ϵ-approximation in this way are the same. The number of steps of value

Algorithm 2.2 Calculate ϵ-approximate V^* – Infinite horizon

$x \leftarrow \infty$
for all $s \in \mathcal{S}$ **do**
  $V^*(s) \leftarrow 0$
end for
while $x > \epsilon$ **do**
  for all $s \in \mathcal{S}$ **do**
    for all $a \in \mathcal{A}$ **do**
      $\mathcal{Q}^*(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \tau(s,a,s') V^*(s')$
    end for
    $V'(s) \leftarrow \max_a \mathcal{Q}^*(s,a)$
  end for
  $x \leftarrow \max_{s \in \mathcal{S}} |V^*(s) - V'(s)|$
  $V^* \leftarrow V'$
end while
return V^*

iteration required to compute this approximation is polynomially bounded by the size of the MDP's representation and the effective horizon, $\frac{1}{1-\gamma}$; however to know that this approximation is optimal requires that the appropriate value of ϵ be known [41]. A discussion of the convergence behavior of value iteration can be found elsewhere [52]. In practice, Algorithm 2.2 is typically run until ϵ is smaller than some arbitrary small threshold or some iteration limit is reached.

Value iteration is just one of several possible approaches to solving MDPs. Another popular approach is policy iteration, which finds a series of policies with increasing quality by alternating steps of policy improvement and value function computation [30].

Ultimately however, we are not interested in MDP solution techniques in and of themselves. Rather, we are interested in how they can be used to approach the problem of dealing with a human whose beliefs, desires and intentions are unknown to the sidekick, a topic which we will discuss in Chapter 3. More generally still, we

are interested in the case where the whole domain is only partially observable to both the sidekick and the human, which we now turn to.

2.4 Partially Observable Markov Decision Processes

Partial observability is inevitable in the physical world and arises frequently in games. In card games, the cards serve both to randomize and hide the complete state of the game. In first-person video games, walls and other occluding obstacles block a player's vision. Computer-controlled adversaries in video games often have internal state governing their behavior that is not directly observable to a player. These are all examples of partial observability in the game state. The presence of human collaborators and their hidden internal beliefs, desires, and intentions adds another source of partial observability. In this section we will introduce partially observable Markov decision processes (POMDPs), which extend the MDP model to account for hidden state.

Like an MDP, a POMDP models a system whose state updates over time in response to an agent's actions. In contrast to the fully observable case, however, the state is not directly accessible to the agent. Instead, at each time step the agent receives one of the set of possible observations, $\mathcal{Z}$, from the system. The random variable Z^t will represent the observation received by the agent at time t. Observations are generated by the system, possibly stochastically, in response to an agent's actions and the underlying state of the system. This process is modeled by the observation function $\mathcal{O} : \mathcal{A} \times \mathcal{S} \rightarrow \Pi(\mathcal{Z})$, which maps the agent's action at time $t-1$, and the state of the system at time t to a distribution over observations. Note that this process is Markovian, the observations depend only on the current state and most recent action. We write $o(a, s, z) = \Pr(Z^t = z | A^{t-1} = a, S^t = s)$ to indicate the individual observation probabilities.

Example: Cops and Robbers is a domain in which the only partial observability stems from the human's internal state. Here $\mathcal{Z}$ is the set of game states, not including the human's beliefs, desires, and intentions, and $\mathcal{O}$ is a deterministic function that projects the full state into this form, without the sidekick's actions having an influence on the observation. Robotics domains often use stochastic observation functions to model the robot's sensor systems. In these domains $\mathcal{Z}$ is the set of all possible sensor readings and $\mathcal{O}$ models the functioning of the sensors.

In Section 2.3.2.1 we noted that for fully observable MDPs, knowledge of the current state is sufficient for a policy to select an optimal action. Since the current state is unknown in a POMDP, policies must have a different form. In this thesis we will discuss two possibilities. One is a mapping from the set of action-observation histories $\mathcal{H}$ to actions, $\pi : \mathcal{H} \rightarrow \mathcal{A}$. An action-observation history is time-ordered sequence of actions selected by and observations received by an agent. This policy form is useful for the search-based approach that we will use in Chapter 4 and will be discussed there in greater detail.

The other form exploits the fact that the Markovian dynamics of the transition and observation model allow us to compute a probability distribution over the system's state, which is a sufficient statistic for the history. This posterior distribution is called a belief state, or information state, and $\mathcal{B} = \Pi(\mathcal{S})$ is the set of all such belief states. Since a belief state is a sufficient statistic for the history, this allows us to write a policy in the form $\pi : \mathcal{B} \rightarrow \mathcal{A}$. Note that this may be a more compact representation than $\pi : \mathcal{H} \rightarrow \mathcal{A}$, since many different histories may be summarized by the same belief state.

The problem of building a POMDP controller, shown interacting with a partially observable system in Figure 2-7, can be decomposed into two parts; the state estimator, SE that computes the belief-state update every time step, and the policy, π,

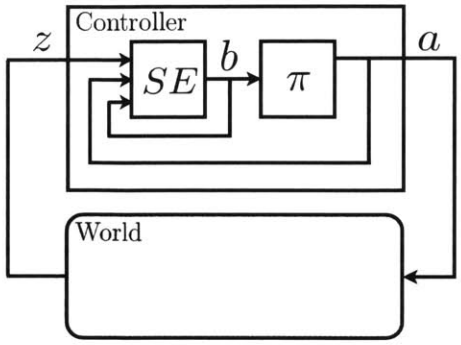


Figure 2-7: A POMDP models an agent's interaction with a partially observable world.

that selects an action, given the current belief state. We can derive SE using Bayes' rule and the probabilities defined in the POMDP model. Let b_z^a be the belief state reached by starting in belief state b, executing action a, and receiving observation z. $b_z^a(s')$, the probability assigned to state s' by this new belief, can be derived

$$\begin{aligned}
b_z^a(s') &= \Pr(s'|b,a,z) \\
&= \frac{\Pr(s',b,a,z)}{\Pr(b,a,z)} \\
&= \frac{\Pr(z|s',b,a)\Pr(s'|b,a)\Pr(b,a)}{\Pr(z|b,a)\Pr(b,a)} \\
&= \frac{\Pr(z|s',a)\Pr(s'|b,a)}{\sum_{s,s''}\Pr(z|b,a,s,s'')\Pr(s,s''|b,a)} \\
&= \frac{\Pr(z|s',a)\sum_s \Pr(s'|b,a,s)\Pr(s|b,a)}{\sum_{s,s''}\Pr(z|a,s'')\Pr(s''|b,a,s)\Pr(s|b,a)} \\
&= \frac{\Pr(z|s',a)\sum_s \Pr(s'|a,s)\Pr(s|b)}{\sum_{s,s''}\Pr(z|a,s'')\Pr(s''|a,s)\Pr(s|b)} \\
&= \frac{o(a,s',z)\sum_s \tau(s,a,s')b(s)}{\sum_{s,s''} o(a,s'',z)\tau(s,a,s'')b(s)}, \qquad (2.13)
\end{aligned}$$

where the final step, Equation 2.13, is obtained by substituting in the observation

and transition function identities.

A range of methods for computing optimal policies for POMDPs make use of the fact that the belief-state dynamics can be used to formulate a POMDP as a continuous state belief-state MDP. The idea is to treat the POMDP as an MDP in belief space, where the states are the belief states and both the actions and discount factor remain the same, but the transition probabilities and immediate reward functions are derived as follows:

Let the conditional probability of receiving some observation z given that the agent took action a and was in belief state b be

$$\begin{aligned}\sigma(b,a,z) &= \Pr(z|b,a) \\ &= \sum_{s',s} b(s)\tau(s,a,s')o(a,s',z). \end{aligned} \tag{2.14}$$

We can then rewrite Equation 2.13 more compactly as

$$b_z^a(s) = \frac{o(a,s,z)\sum_{s'}\tau(s',a,s)b(s')}{\sigma(b,a,z)}. \tag{2.15}$$

The final belief-space transition probability of reaching belief-state b' given that the agent's current belief state is b and that it took action a is then

$$\psi(b,a,b') = \sum_z \sigma(b,a,z)I(b',b_z^a), \tag{2.16}$$

where I is the indicator function

$$I(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}. \tag{2.17}$$

Equation 2.16 gives us the belief-state transition function that we need for a belief-

state MDP. The final element is the immediate reward function on belief states, ϱ, constructed from the reward on regular states, which is

$$\varrho(b, a) = \sum_{s \in \mathcal{S}} b(s) r(s, a). \tag{2.18}$$

A psychological interpretation of this function is that an agent receives reward for believing itself to be in a good state. An alternative interpretation is that this function gives the true expected reward to the agent of being in belief state b and taking action a.

Taken as a tuple, the belief states $\mathcal{B}$, actions from the fully observable MDP $\mathcal{A}$, belief-state transition function ψ, and reward function ϱ constitute an continuous-state MDP, $\Xi = (\mathcal{B}, \mathcal{A}, \psi, \varrho)$, which in principle can be solved to find π^*.

2.4.1 Human Models with Partial Observability

We must adapt our human model for partially observable domains to account for the fact that the human also may not have direct access to the world state and that the sidekick does not have access to the human's internal state. Figure 2-8 shows the structure of the human model in the partially observable case. Note that the model parallels the general POMDP controller. The key difference between this model and the model from Section 2.3.1 is that the human receives observations rather than states. A human model may use these observations in a number of ways, from maintaining a complete action-observation history in their internal state, or summarizing such a history with a belief state that can be used in action selection, to simply reacting to the most recent observation.

The dynamic Bayesian network in Figure 2-9 shows the causal relationship between the different variables involved. The thin dotted lines indicate the functions that are responsible for updating the variables. The thick dashed lines indicate the

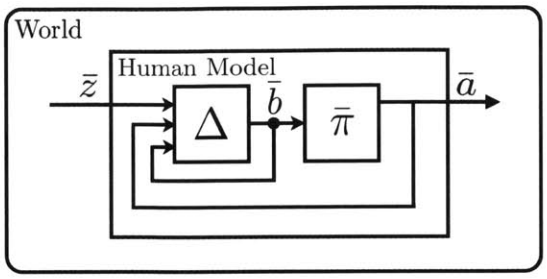


Figure 2-8: The structure of the POMDP human model.

set of steps modeled by the complete transition function.

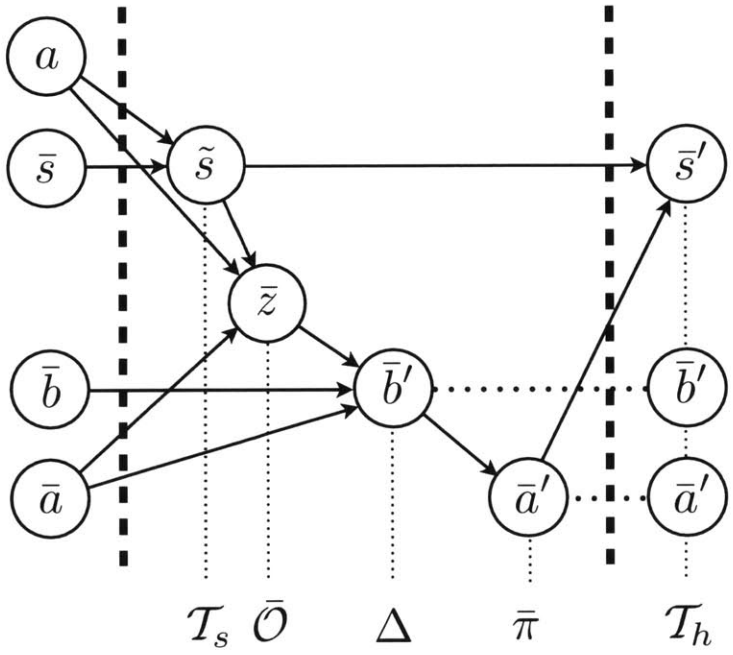


Figure 2-9: The transition function displayed as a dynamic Bayesian network for the POMDP case.

Let $\bar{\mathcal{Z}}$, be the set of possible observations that the human can receive. The random variable $\bar{Z}^t$ will represent the observation received by the human at time t. Similarly to the case of the sidekick, $\bar{\mathcal{O}} : \bar{\mathcal{A}} \times \mathcal{A} \times \tilde{\mathcal{S}} \rightarrow \Pi(\bar{\mathcal{Z}})$ models how the human

probabilistically receives observations from the world, based on the current world state and the last actions of the human and sidekick. We write $\bar{o}(\bar{a}, a, \tilde{s}, \bar{z}) = \Pr(\bar{Z}^t = \bar{z} | \bar{A}^t = \bar{a}, A^t = a, \tilde{S}^t = \tilde{s})$ to indicate the individual observation probabilities.

As mentioned, the human's internal transition function changes in this setting to be $\Delta : \bar{\mathcal{B}} \times \bar{\mathcal{A}} \times \bar{\mathcal{Z}} \to \Pi(\bar{\mathcal{B}})$, which models the human's internal state updating at time $t+1$ after having taken an action and received an observation at time t. We write the individual transition probabilities as $\delta(\bar{b}, \bar{a}, \bar{z}, \bar{b}') = \Pr(\bar{B}^{t+1} = \bar{b}' | \bar{B}^t = \bar{b}, \bar{A}^t = \bar{a}, \bar{Z}^t = \bar{z})$. Note that $\bar{b}$ could be a belief state or action-observation history, or it could be some other model of the human's beliefs, desires, and intentions; in most cases it will not be directly observable by the sidekick.

$\mathcal{T}_s$ and $\mathcal{T}_h$ remain unchanged from the fully observable case. Recall from section 2.3.1 that it is important to include the human's previous action as a part of the state in order to be able to compute Δ correctly. This also holds for $\bar{\mathcal{O}}$, which likewise requires the human's previous action.

A partially observable human model is then the tuple of human states, observations, observation function, internal update function, and policy $H = (\bar{\mathcal{B}}, \bar{\mathcal{Z}}, \bar{\mathcal{O}}, \Delta, \bar{\pi})$.The transition probabilities for the sidekick POMDP can then be written:

$$\tau((\bar{s}, \bar{b}), a, (\bar{s}', \bar{b}')) = \sum_{\tilde{s} \in \tilde{\mathcal{S}}} \tau_s(\bar{s}, a, \tilde{s}) \sum_{\bar{z} \in \bar{\mathcal{Z}}} \bar{o}(\bar{a}, a, \tilde{s}, \bar{z}) \\ \delta(\bar{b}, \bar{a}, \bar{z}, \bar{b}') \sum_{\bar{a}' \in \bar{\mathcal{A}}} m(\bar{b}', \bar{a}') \tau_h(\tilde{s}, \bar{a}', \bar{s}'). \tag{2.19}$$

2.4.2 Solving POMDPs

With the complete formulation of the sidekick POMDP we can now consider solution methods. Solving an MDP can be done in polynomial time, with the caveat that the size of the problem itself scales exponentially with the number of state variables.

The move to partial observability, however, makes solving finite horizon POMDPs PSPACE-hard and uncomputable for infinite horizon POMDPs [41]. The infinite horizon case should not deter us too much, since in general we would be content with a solution for finite, albeit very large, horizon POMDP. The PSPACE-hardness result is, however, a major obstacle to the practical application of POMDP solutions in games.

The transformation of a POMDP into a belief-state MDP allows for variations on value iteration for computing exact solutions. One challenge is that the continuous space could make even representing the value function required for one step of value iteration an intractable problem; however, by leveraging the piecewise-linear and convex properties of a POMDP's value function, it is possible to use finite sets of vectors to represent the function and to approximate it to arbitrary precision or even exactly in the finite horizon case [12]. A variety of exact solution techniques leverage this property and have been compared theoretically and empirically, but are intractable for most games domains [58, 13, 11].

Given that exact solution methods are intractable, we must turn to approximate methods. A variety of approximate planning techniques for POMDPs exist based on different forms of value function approximation, a comprehensive survey of which can be found in [29].

The starting point from which we build our approximation techniques is a decision-theoretic model of assistance introduced by Fern et al. [22, 23]. As we have done, this body of work modeled the sidekick problem as a POMDP. Unlike our model however, they used a special purpose formulation which assumed that the POMDP had a special structure, such that the only hidden state of the world was the human's internal state. They proposed learning the dynamics of their human model from observed data over time, but bootstrapping the learning process by assuming that initially the human acts as if it were perfectly coordinating with the sidekick to

achieve its goal. This assumption allowed them to build a human model by using value iteration. For solving the POMDP itself they used approximate methods based on the value function of the underlying MDP. This technique was applied to games domains in the CAPIR planning framework introduced by Nguyen et al. [45]. A variant of this framework was applied by the Singapore-MIT GAMBIT Game Lab in the game Dearth, a cooperative puzzle game that prominently featured an AI sidekick [38]. We will develop it further in Chapter 3, exploring extensions to allow for communication and information gathering.

One popular set of more general-purpose approaches to POMDP solving use point-based methods which sample points in belief space, compute the value function at those points via value iteration, and use them to form a value function estimate, such as PBVI [50] and SARSOP [37]. By contrast, policy search methods construct compact policy representations, such as finite state controllers, and perform gradient ascent on the value function by trying various policy modifications and accepting the change that best improves the value function [2]. In a recent planning competition, the most successful planner combined SARSOP with the POMCP algorithm, which combines value function estimation from Monte Carlo simulation with particle filtering for representing beliefs [57]. A similar strategy using approximations from state-space planning has also shown success in cooperative planning in teams with unknown team mates [4]. These results make POMCP a promising candidate for attacking the AI sidekick problem, which we will explore in Chapter 4.

2.5 AI in Games

Planning in games has been a fruitful field of research for AI researchers since early in the discipline. The broadest class of successful approaches for adversarial games has combined variations on minimax search with linear function approximation to search

as deep as possible into the tree of state-action sequences and then approximate the value of the states reached from features of these states. This approach has led to super-human performance in a variety of games including chess [10]. An extreme case is checkers, which has been solved via a combination of minimax search and an exhaustive lookup table of end game positions [55]. For the most part, planning in games has focused on creating strong computer opponents, rather than assistants.

Of particular interest for domains with large state spaces, such as the AI sidekick problems with which we are concerned, is recent work on the game of Go. Through combining techniques for minimizing regret on multi-armed bandit problems, ideas from traditional minimax search, value function approximation through Monte Carlo estimates, and heuristic methods for bootstrapping the value function approximations, computer Go players are now competitive with the most accomplished human players on 9x9 boards and are reaching a competitive level on full sized boards [26]. The general framework of combining an exploration policy for selecting actions within a search tree of previously seen states and actions with Monte Carlo estimates of the value function formed from random interactions with a simulation of the domain dynamics, called rollouts, is Monte Carlo Tree Search (MCTS) [15]. The most common choice of exploration policy is the UCB1 algorithm, which combined with uniform random action selection for rollouts gives the UCT algorithm [36].

Numerous variations on these techniques have been demonstrated with good success across games domains, a comprehensive survey of which can be found elsewhere [9]. As of June 2013, the dominant system in general game playing, which tests the ability of AI agents to play previously unseen games, is the MCTS-based system Cadiaplayer [8]. POMCP, mentioned in 2.4.2, extends the MCTS framework to partially observable domains and was used in the winning entry of the 2011 ICAPS probabilistic planning competition [54].

Researchers in academia have largely focused their efforts on developing AI that

competes in adversarial games at a high level. By contrast, game designers and AI programmers in the video games industry have been more concerned with developing AI that gives a compelling gameplay experience whilst using relatively few computational resources. The ideal AI opponent from a commercial games perspective offers the human an enjoyable but beatable challenge and the ideal sidekick AI is an helpful companion whose primary virtues are that the player forms an emotional connection with them and that they don't get in the way.

Game AI uses a heterogenous toolkit of techniques to drive AI behavior in games. Some prevalent techniques include finite state machines, A* search, artificial potential fields, utility-based systems, and domain-specific languages for behavior scripting. Animation selection and blending techniques are also typically closely linked with behavior selection. Game companies are often protective of their proprietary technology, but some notable AI programmers have recently published textbooks covering the major techniques involved in building AI systems for games [56, 43, 44].

In first-person video games in particular there are two prevalent techniques for controlling AI behavior. The first is behavior trees, which combine some of the properties of decision trees and finite state machines, giving designers a domain-specific graphical language for manually specifying how individual behaviors are triggered by an AI's sensors and are sequenced [32]. Behavior trees were developed for the Halo series, first being used in Halo 2, and have since become a standard technique across games, spreading beyond first-person games [61]. Interviews with developers on the recently released games Bioshock Infinite and The Last of Us [25, 16], both of which prominently feature sidekick AI, have suggested that their sidekick character controllers were built using a combination of behavior trees and utility-based reasoning, but in keeping with the secretive nature of the industry, neither of the studios, Irrational Games and Naughty Dog, have officially confirmed this or released a white paper.

The second technique is goal-oriented action planning (GOAP), which first appeared in the F.E.A.R. series and has since been used in a number of other titles, but is less prevalent than behavior trees [48, 51]. GOAP uses a highly optimized STRIPS-like planning system to allow AI characters to autonomously plan sequences of actions to achieve their goals. This style of planning contrasts with decision-theoretic planning, in that there is no notion of a general reward function, but rather some goal state that the AI is trying to reach and a series of available operators that it must sequence together to reach that goal state. This approach has been extended into a hierarchical planning setting in the Killzone series, in which the AI characters plan using hierarchical task networks (HTNs), combining the hierarchical structure of behavior trees with the planning-based approach of GOAP [60, 24].

These two approaches contrast in where development effort is located. In behavior trees the effort is in specifying how to select and carry out appropriate actions, whereas in GOAP and HTN planning it is in specifying a domain model that describes how actions affect the world. Both can require domain insights on the part of designers and engineers to produce good behavior. In many games the rules themselves will specify the domain model, and in particular the transition function, which is a point in favor of the latter approaches, but this is not true of all games. For example, modern computer games often use sophisticated physics simulations that implicitly specify a transition function, but whose effects cannot easily be expressed explicitly. Even if the model is available, the state space, action set, or transition function may be too large or complex for the planner to reason with, so developers must instead design abstractions for the planner, which is a domain engineering challenge in itself. As things stand, behavior trees are wider spread, but neither approach has proven a clear winner among developers.

Our decision-theoretic approach falls in the same camp as GOAP and HTN planning, requiring domain modeling effort to formulate the system of game and human as

a POMDP. As we shall see, the state-space complexity concerns that lead the games industry to use abstract domain models for GOAP and HTN planning will become apparent even for the relatively simple games that we will investigate and will prove to be a challenge. On the other hand, the key strength of our approach is that it provides a way of reasoning about how to take human intentions into account when planning, which is something that GOAP and HTN planning do not address.

In the next chapter we present a method for sidekick planning using our decision-theoretic POMDP model that uses state-space planning to approximate a solution to the POMDP.

Chapter 3

State-space Planning for Sidekicks

3.1 Introduction

Having formulated the AI sidekick problem as a POMDP, we are now faced with the challenge of finding a controller for the sidekick. As mentioned in Chapter 2, the general problem of solving POMDPs optimally is PSPACE hard for finite-horizon problems and uncomputable for infinite horizon problems. For a practical approach we must trade off optimality for tractability by adopting heuristic solutions. However, we will still use the power of the POMDP representation and decision-theoretic planning to motivate these heuristic methods.

The approach in this chapter is to use standard state-space planning techniques to build a heuristic POMDP controller. The key idea is that if the sidekick had an oracle that allowed it to observe the full world state directly, including the human's intentions, it would be able to transform the POMDP into an MDP, solve for the optimal policy of the MDP, and then act according to the policy. As mentioned in Chapter 2, finding an optimal MDP solution can be done in polynomial time, which is a vast improvement on PSPACE-completeness, with the caveat that the size of the problem scales exponentially with the number of state variables.

The trade off in this approach is that the policy for any MDP will not take into account the sidekick's uncertainty about the true state of the world in its planning, including its collaborators' goals, since the MDP formulation implies that the sidekick knows the internal state of its collaborators exactly.

Our approach to building a controller for the sidekick involves four components, labeled in Figure 3.1.

1. A human model H representing the decision process of the human collaborator, with separate behavior profiles for each of the human's possible goals, constructed using state-space planning.

2. A state estimator implemented as a recursive Bayes filter that updates the sidekick's belief state given its actions and observations.

3. The value function, V^*_{MDP}, for the fully observable version of the AI assistant problem.

4. A policy that selects an action, given the sidekick's current belief state, incorporating V^*_{MDP}.

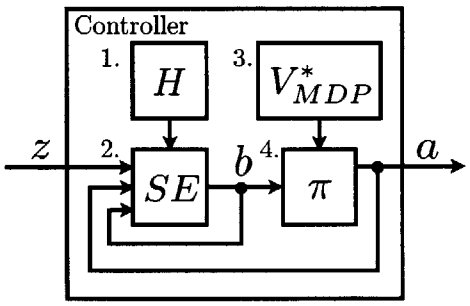


Figure 3-1: The four components of the state-space planning approach.

The key shortcoming of using state-space planning to evaluate actions is that it assumes that the sidekick has full information about the world, so it will never select

actions solely for gathering information. To make this clear, consider a simple game in which a contestant must pick between one of two opaque boxes, one of which contains a pot of gold, whilst the other contains nothing. We will apply the planning method outlined above. To transform this game into an MDP, we must make the assumption that there is no hidden state, so the position of the gold is completely known and the optimal state-space policy says simply to pick the box with the gold in it. Since the contestant doesn't in fact know the exact state of the world, they must rely on the Bayes filter to tell them which box the gold is in so that they can execute the MDP policy. Since their initial belief about the gold's location should weight each possibility equally in the absence of any other information, the contestant would be left with no better policy overall than picking at random, which is indeed the optimal policy. However, consider how things change if the contestant could pay a penny to be told exactly where the pot of gold is. The optimal policy for the MDP in this modified set up is exactly the same as for the original case, because why would you want to pay a penny if you already knew exactly where the pot is? The problem is that MDP policies will always be too optimistic and discount the value of information, because they assume a fully observable world.

To combat this shortcoming we introduce a range of heuristic approaches to the problem of deciding whether to take information-gathering actions at execution time, which make use of MDP value functions and policies computed by offline planning. These approaches stand in contrast to the approach in Chapter 4, which instead plans directly in belief space, by reasoning directly about how information-gathering actions will affect the sidekick's belief state. Here though, since we simplify the planning problem by setting aside the issue of partial observability, we should expect that planning will be more efficient, albeit with a limit to how well our policy can ever hope to perform.

Finally, for certain kinds of domains, the state-space planning approach offers a

ready answer to the question of how one can automatically build a human model to be used in the POMDP formulation. There are two conditions for this. The first is that we can characterize the set of possible goals or desires that the human may have by a family of reward functions, one for each such specific goal or desire. The second is that the only unobserved state variables in the domain are the internal state of the human.

Given these conditions, we can formulate a set of fully observable MDPs, one for each reward function, in which we assume that both the sidekick and the human have full access to the human's internal state. We can then solve each of these MDPs for joint policies for the human and the sidekick using state-space planning. These joint policies represent the optimal strategies for the human and the sidekick to coordinate under the assumption of full observability. The human's half of these joint policies can then be used to build human models, representing a human acting under the assumption that the sidekick fully understands their intentions and is trying to coordinate optimally with them.

The approach to the AI sidekick problem discussed in this chapter builds upon previous work by Fern et al., who developed a decision-theoretic model of assistance that used the same basic strategy for building human models outlined here, although our formulation is somewhat more general in that we allow for a range of control strategies that attempt to reason about how to mitigate the sidekick's uncertainty about human intentions [22]. They also integrated a learning component for refining human models from experience, which we do not address here, but which would be a possible extension.

Nguyen et al. applied this model of assistance in a games domain, developing a system called CAPIR, which also incorporated a model of task switching that we generalize in our human model's internal update function [45]. The CAPIR framework consists of a specific set of design choices covered in this chapter. It uses a domain-

specfic decomposition of the game into sub-games representing the human's possible subgoals and solves them using value iteration to build a human model, it assumes that humans switch subgoals with a fixed probability on each time step, and it uses the QMDP rule for action selection.

Our contribution to this body of work is to extend this model of assistance by introducing action selection techniques that reason about the sidekick's belief state and can select information gathering actions, which the previous two approaches were incapable of. This allows for us to incorporate communication and question asking into the sidekick's repertoire of actions, as well as all allowing for more subtle reasoning about the effects of regular actions on the sidekick's belief state, extending the sidekick's capabilities.

3.2 Cops and Robbers Revisited

Recall from Section 2.1 that Cops and Robbers is fully observable other than the state of the human, and also that there is a natural way of characterizing the different possible goals of the human in the game, namely chasing each of the robbers. This makes Cops and Robbers an example of the kind of domain that meets the conditions for the state-space planning approach in this chapter to apply.

The combination of these two properties allow for the formulation of the human model as solving and executing its part in a joint MDP policy as outlined previously. Note that if the environment were partially observable from the human's perspective, this kind of modeling would not be possible, since it would mean solving POMDPs instead, which is precisely the source of intractability that is motivating this heuristic approach in the first place.

The asking action in Cops and Robbers is an explicit information seeking action that allows the sidekick to improve its estimate of the human's intentions directly.

It serves as a possibly noisy oracle for revealing the internal state of the human, but doesn't directly advance the pair towards any of the subgoals. As mentioned previously, this means that the asking action will never be selected as part of a MDP policy, however it is often relevant because it can help avoid stepping through a bad one-way door due to uncertainty about the human's intentions. Because the asking action is so strongly diagnostic of human intentions, it can sometimes make Cops and Robbers trivial. When using Cops and Robbers in experiments, we will sometimes compare performance with and without the asking action to tease out the effect of communication. In general, the asking action is very advantageous for planning directly in belief space, whereas for state-space planning with heuristic information gathering its usefulness is limited by the robustness of the heuristic.

3.3 State Estimation

The sidekick does not have direct access to the state, but it can maintain a belief state tracking the possible values of hidden state variables, including the human's intentions, to help it make decisions. This can be achieved by constructing a recursive Bayes filter from the POMDP dynamics. The filter makes use of the conditional probability given in Equation 2.15 from Chapter 2,

$$b_z^a(s) = \frac{o(a, s, z) \sum_{s'} \tau(s', a, s) b(s')}{\sigma(b, a, z)}.$$

Starting from an initial belief state, Equation 2.15 gives the rule updating the sidekick's belief state after taking action a and receiving observation z.

3.4 MDP Conversion

The strategies for action selection discussed in this chapter all hinge on computing an optimal state-space policy, π^*_{MDP}, and expected values of each action under that policy, Q^*_{MDP}. This is done by constructing a fully observable MDP, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, in which each element is the same as in the POMDP formulation, but there are no observations or observation function and the states, including the human's intentions, are fully observable. We then solve for π^*_{MDP} using the value iteration algorithm discussed in Section 2.3.2.3.

3.5 Action Selection

Given the optimal state-space policy, π^*_{MDP}, and the current belief state, b, there are a number of possible decision rules for selecting an action. We investigated four approaches that heuristically incorporate a varying degree of information from the sidekick's belief state and of explicit reasoning about information gathering.

3.5.1 Maximum Likelihood

Maximum likelihood action selection, π_{ML}, uses π^*_{MDP} directly to select the action for the most likely state, according to the belief state. This represents an assumption that uncertainty about the true state is largely irrelevant, given that we know the most likely state, and $V^{\pi_{ML}}$ is a loose lower bound for the POMDP's true value function, V^*. The maximum likelihood policy is given by

$$\pi_{ML}(b) = \pi^*_{MDP}(\operatorname*{argmax}_s b(s)).$$

3.5.2 QMDP

As a decision rule, π_{ML} is largely unresponsive to the effect of uncertainty. By contrast, QMDP action selection, introduced by Littman et al. [40], weights the $\mathcal{Q}^*_{MDP}$ values for the states and actions of the MDP by the state probabilities given by b. The QMDP weighting represents an optimistic assumption that although the state may seem ambiguous now, on the time step after selecting the current action, the world will be completely observable and unambiguous. $V^{\pi_{QMDP}}$ is a tighter lower bound for V^* than $V^{\pi_{ML}}$ [29]. The QMDP policy is given by

$$\pi_{QMDP}(b) = \operatorname*{argmax}_a \sum_s b(s)\mathcal{Q}^*_{MDP}(s,a).$$

3.5.3 Lookahead

Both π_{ML} and π_{QMDP} make the assumption that the state will trivially become completely observable. This means they will never place any value on pure information gathering actions and will not take into account any contribution to the true POMDP value function that comes from an action helping to disambiguate the state. Because they are both based on V^*_{MDP}, this is unavoidable. Instead of using fully fledged belief-space planning, which we will discuss in Chapter 4, we can attempt to mitigate some of the effects of building approximate controllers using state-space planning by introducing control systems that reason heuristically about the hidden world state and can plan to take information gathering actions.

One improvement to the over-optimism of π_{QMDP} is to do a limited lookahead in belief space to assess the value of information gathering actions. We can perform a simple one-step look-ahead in belief space, and then take an expectation over the values of the states we may end up in using V^*_{MDP}, again assuming that from that point on the state will be disambiguated. This gives us the $\mathcal{Q}$-function

$$\mathcal{Q}_{LA}(b,a) = \sum_s b(s)r(s,a) + \gamma \sum_{s'} \sum_z b_z^a(s') V^*_{MDP}(s').$$

The key term which distinguishes this approach from QMDP is the belief-state update b_z^a. From $\mathcal{Q}_{LA}$ we can obtain a policy by choosing the action with the best expected future rewards, given this one-step look ahead,

$$\pi_{LA}(b,a) = \operatorname*{argmax}_a \mathcal{Q}_{LA}(b,a).$$

This approach can be generalized to a k-step lookahead, but the branching factor of the search is $k^{|\mathcal{A}||\mathcal{Z}|}$. We will only consider the one-step case in this chapter. In Chapter 4 we will introduce search methods for belief space that can help cope with the high branching factor.

3.5.3.1 Action Entropy

The entropy of the sidekick's current belief state is a measure of how uncertain it currently is about the state. If the sidekick's current belief state has high entropy, this can be a cue that it ought to try to take information gathering actions to disambiguate it. However, if the ambiguity in the state would have no bearing on the actions selected, then it would be inappropriate to gather information. These considerations motivate the action entropy approach, first introduced by Kaelbling et al. [34]. This approach is a dual-control strategy that heuristically chooses whether to take information gathering actions or actions suggested by the state-space policies based on an entropy analysis.

Since a belief state b is a probability distribution, its entropy is $\mathbb{H}(b) = -\sum_s b(s) \log b(s)$, where $b(s) \log b(s) = 0$ when $b(s) = 0$. The expected entropy of the belief state reached by starting in belief state b and taking some action, a, is then

$$EE(a,b) = \sum_{b'} \psi(b,a,b')\mathbb{H}(b').$$

Because we care specifically about whether there is ambiguity in which action to take, rather than switching control based on $\mathbb{H}(b)$, we should instead switch on the entropy of the distribution over optimal actions, according to π^*_{MDP},

$$w_b(a) = \sum_s b(s) I(\pi^*_{MDP}(s), a),$$

where I is an indicator function.

If the entropy of this distribution is higher than some threshold ϕ, then we will select the action that has the best chance of disambiguating the state. Otherwise we will take the action that maximizes w_b. Thus, the action entropy policy is

$$\pi_{AE}(b) = \begin{cases} \operatorname{argmax}_a w_b(a) & \text{if } \mathbb{H}(w_b) < \phi, \\ \operatorname{argmin}_a EE(a,b) & \text{otherwise.} \end{cases}$$

One concern with the action entropy approach is that information gathering actions may not disambiguate the state enough to reduce the belief-state entropy below the threshold. In this case the agent could become stuck continually trying to gather information. This could be addressed by heuristically restricting the number of information gathering actions that can be taken within some time window, but we have not explored this possibility.

3.6 Human Modeling Using State-Space Planning

Recall from Section 2.4.1 that a human model H is composed of a set of internal states $\bar{\mathcal{B}}$, observations $\bar{\mathcal{Z}}$, an observation function $\mathcal{O}$, an internal update function Δ,

and a stochastic policy $\bar{\pi}$. In general, formulating H for the AI assistant problem is a challenging and potentially ill-defined task, however, if the only partial observability in the domain stems from uncertainty about the human's internal state, then state-space planning methods offer a ready solution to this modeling challenge. Specifically, if we assume that the human expects perfectly optimal collaboration from the sidekick, we can construct $\bar{\pi}$ via value iteration.

3.6.1 Human Types, States, and Observations

In defining our human model we will make use of the notion of types. Types model sets of beliefs, desires, and intentions that drive behaviors. Our model will assume that the human is following one of some fixed set of n possible behavior profiles, each of which is associated with a specific type that represents the human's beliefs, desires, and intentions. For instance in Cops and Robbers there may be one type for each of the robbers, representing the human focusing on catching that specific robber. Let the set of all n such types be $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$. We then define $\bar{\mathcal{B}}$ to be the set of state and type pairs $\bar{\mathcal{B}} = \{\tilde{\mathcal{S}} \times \Theta\}$. A given human state $\bar{b} = (\tilde{s}, \theta)$ represents the most recent state experienced by the human and the human's current type.

Recall that $\tilde{\mathcal{S}}$ is the set of states immediately preceding the human taking an action. Because the only partial observability in the domain is the human's internal state, which the human has access to, we can simply specify that $\bar{\mathcal{Z}} = \tilde{\mathcal{S}}$ and

$$\bar{o}(\bar{a}, a, \tilde{s}, z) = \begin{cases} 1 & \text{if } \tilde{s} = z \\ 0 & \text{otherwise} \end{cases}. \tag{3.1}$$

To construct $\bar{\pi}$ we will introduce a set of policies, $\bar{\Pi}_\theta$, called type-policies. A specific type-policy, $\bar{\pi}_{\theta_x} \in \bar{\Pi}_\theta$, is the policy that implements the behavior profile specified by type θ_x.We will construct type-policies by modeling humans as near-

optimal MDP solvers. In this section we will outline the general approach to doing this, motivated throughout by our example domain, Cops and Robbers.

3.6.2 Constructing MDPs for Human Types

Each type-policy will be derived from the solution to an MDP $\mathcal{M}_\theta$ that embeds the human type θ. Constructing the MDP for a type is a heavily domain-dependent modeling problem, but our example domain can help illuminate the general principles.

We can use an MDP's reward function, $\mathcal{R}$, to model human desires. Suppose for Cops and Robbers that we wanted to model humans as chasing one particular robber with the expectation that the sidekick would take the optimal move to support them in catching that robber. We could do this by using a two-agent MDP formulation of Cops and Robbers with $\mathcal{R}$ constructed such that $\mathcal{R}(s, a, s') = 1$ iff both the sidekick and the human player are standing on the target robber in s'. This two-agent formulation is trivial to construct, by simply assuming that one monolithic controller can select the actions of both the human and the sidekick. Solving this MDP for the optimal joint policy of the human and sidekick gives a decision rule for how the team ought to act, were they optimally coordinating to catch the robber.

We can also use MDPs to model cases in which the human lacks information about the environment or misunderstands the dynamics. Consider for instance if we wanted to model the human chasing a particular robber as before, except that the human doesn't understand that doors are one-way. We could model this as a two-agent MDP in the same way as before, but in this case with $\tau(s, a, s')$ changed so that doors work in both directions. Solving this MDP for the optimal joint policy again gives the behavior of a team optimally coordinating to catch a particular robber, were it the case that they both (falsely) believed that one-way doors were in fact two-way.

Note that the MDP formulation is key, since MDPs, unlike POMDPs, can be

solved relatively efficiently. This means the state must be fully observable and this kind of modeling cannot be used to represent a human reasoning optimally about partially observable domains.

3.6.3 Solving MDPs for Human Types

Given a two-agent MDP, $\mathcal{M}_\theta$, corresponding to some type, θ, we can use value iteration to find its optimal value function $V^*_{\mathcal{M}_\theta}$. From this we can derive the human's half of the joint optimal policy, $\pi^*_{\mathcal{M}_\theta}$, which gives actions that the human should take were it to be acting optimally with a perfectly coordinating sidekick.

In general we should not expect that humans will follow the optimal policy for an MDP, and in fact modeling work by Baker et al. [3], in cognitive science has shown that a better model is that humans select actions with a probability proportional to their Q-values. With this in mind, instead of constructing θ directly from $\pi^*_{\mathcal{M}_\theta}$ we instead construct it via a transformation of $Q^*_{\mathcal{M}_\theta}$.

3.6.4 Constructing Human Policies

There are several reasonable ways to define a probability distribution over actions, given $Q^*_{\mathcal{M}_\theta}$ or $\pi^*_{\mathcal{M}_\theta}$. The simplest is the ϵ-greedy model, which assumes that the human will act according to $\underset{\bar{a}}{\text{argmax}}\, Q^*_{\mathcal{M}_\theta}(\tilde{s}, \bar{a})$ with probability $1 - \epsilon$ and select an action uniformly at random with probability ϵ. However, a potentially more cognitively plausible model, following Baker et al. [3], is to use a Boltzmann distribution,

$$\bar{\pi}(\bar{b} = (\tilde{s}, \theta), \bar{a}) \propto e^{-\beta Q^*_{\mathcal{M}_\theta}(\tilde{s}, \bar{a})},$$

where β is a parameter that controls the amount of noise in the human's actions. For high values of β, the human favors actions with higher expected future rewards, whereas for low values the distribution over actions becomes more uniform.

3.6.5 Defining the Human State Update Function

The internal state update function Δ must still be specified. The simplest modeling choice, though conceptually weak, is to assume that humans are largely indifferent to changes in the world state and most often continue to maintain their type from state to state, but with some small probability, ϵ, change to a random type. This gives the human state update probabilities

$$\delta(\bar{b} = (\tilde{s}, \theta), \bar{a}, \bar{z}, \bar{b}' = (\tilde{s}, \theta')) = \begin{cases} 1 - \epsilon & \text{if } \theta = \theta' \\ \frac{\epsilon}{|\Theta|} & \text{otherwise} \end{cases} . \tag{3.2}$$

A more sophisticated model might make use of a range of variables in $\tilde{s}$ or be dependent on $\bar{z}$. For instance, in Cops and Robbers, proximity to a robber or the actions of the sidekick might play a role.

3.6.6 Communication and Decentralized Beliefs

The modeling approach we have just described makes the strong assumption that humans act as if they shared a common belief about the internal state with their sidekick collaborators. An important consequence of this is that the policies derived through state-space planning in this way will not take into account the true decentralized nature of beliefs in the human's interaction with the sidekick. To see why, imagine that the human player had an action available that revealed the values of its internal state variables to the sidekick. From the state-space planning perspective, sending such a message is a wasted action, since the world is completely observable, and beliefs are centralized, so the team stands to gain nothing from the human taking it. As a result, communication cannot be handled through state-space planning and requires a separate system. This is true in general of reasoning about how human and sidekick actions will affect one another's beliefs. It is something that state-space

planning cannot account for, and to do so optimally would require a more expressive formalism, such as Dec-POMDPs.

For Cops and Robbers, which includes communication actions, we opted for a domain-specific heuristic approach to communication and modeled the human as always responding truthfully to requests for information about its internal state, with some noise. Other modeling options include the action entropy and belief-space lookahead approaches discussed previously.

3.7 Empirical Evaluation

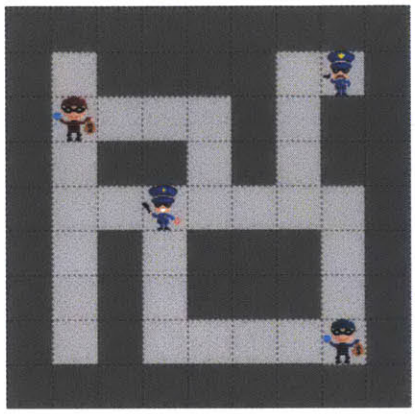
In this section we discuss the domains, controller and model parameters, and experimental setup used for evaluating the action selection methods described in this chapter. We conclude with experimental results and discussion.

3.7.1 Domains

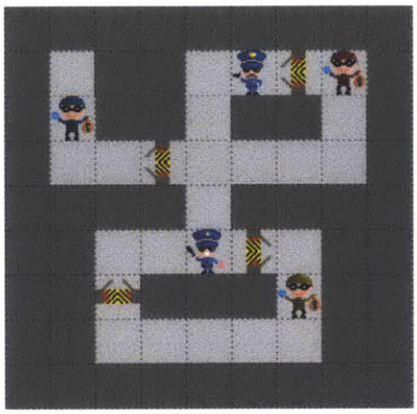
We compared the different approaches to action selection presented in Section 3.5 on two video game domains, Cops and Robbers, described in Section 2.1, and Collaborative Ghostbuster, first introduced with the CAPIR framework [45].

In testing on Cops and Robbers we used five maps, shown in Figure 3-2. Each map is constructed such that the human's first sequence of actions is ambiguous with respect to the robber they're chasing. This means that the sidekick must decide what to do in the face of uncertainty about the human's goal, which is intended to draw out the differences between the action selection methods, in particular favoring actions that can help disambiguate the human's intentions.

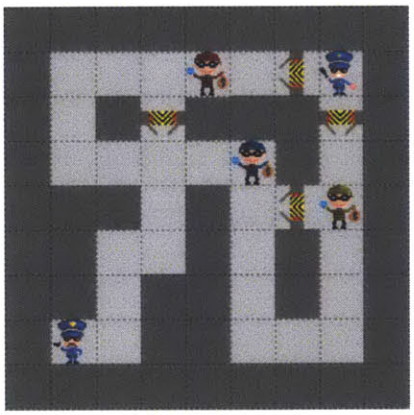
C&R1 is a basic map containing no doors, which lets the sidekick easily recover from having misjudged the human's goal. In C&R2 stepping though the door to the blue robber traps the cop in the northwest part of the map and chasing the yellow



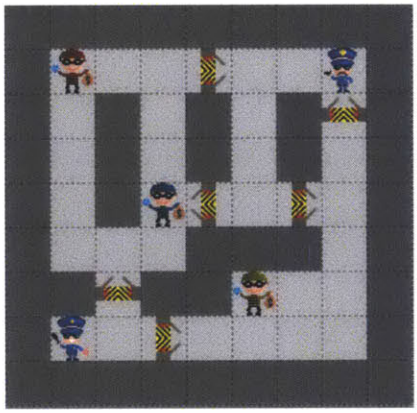
(a) C&R1



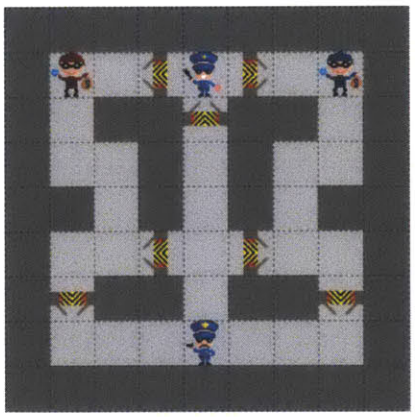
(b) C&R2



(c) C&R3



(d) C&R4



(e) C&R5

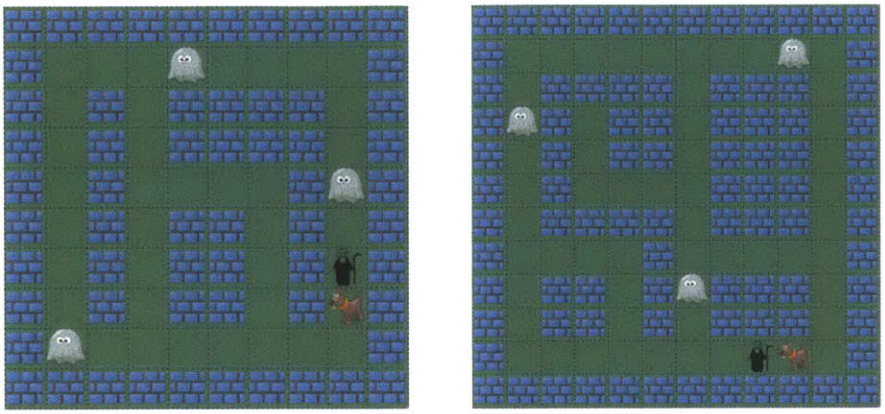
Figure 3-2: The maps used for Cops and Robbers. The human is shown holding a gun, whereas the sidekick holds a doughnut and truncheon. Doors only open in the direction of the chevrons.

robber commits a cop to moving through the whole loop in the south. In C&R3 without any communication the human's intentions will be ambiguous for at least six steps, but because the robbers are initially close to each other, the sidekick can herd them together to hedge their bets about the human's target. The two robbers in the west of C&R4 make it attractive to enter that room, but if the human is chasing the yellow robber, changing course is costly. C&R5 is symmetric, with the choice to step through the west or east doors initially committing the sidekick to chasing the red or blue robber respectively.

Collaborative Ghostbuster is a game with asymmetric roles for the sidekick and human. As with Cops and Robbers, the human, represented as a shepherd, and the sidekick, represented as a dog, are pursuing enemies through a maze. The enemies are ghosts who flee from the players within 4 cells of them. The goal for the human is to eliminate all the ghosts. They can do so via an action which emits a short ranged zap that destroys any ghost within 4 cells in each orthogonal direction. The sidekick, on the other hand, cannot destroy ghosts, and must instead help by herding them towards the human to be zapped. Our implementation of Ghostbusters differed slightly from the one reported in the CAPIR paper in that ghosts stayed still, rather than performing a random walk, when neither the human nor the sidekick were near.

Figure 3-3 shows the maps used for evaluation on Collaborative Ghostbuster, both of which were introduced along with CAPIR [45]. In GB1 the ghost near the human can be zapped immediately, leaving only two ghosts remaining, which are relatively easy for the sidekick to round up for the human to zap. In GB2 the sidekick must reach the northern part of the map to drive the ghosts back south and west towards the human and the human's initial moves may be ambiguous as to the ghost they are chasing.

We chose to model both Cops and Robbers and Collaborative Ghostbuster as infinite horizon problems with an absorbing zero-reward state that the game enters



(a) GB1 (b) GB2

Figure 3-3: The maps used for Collaborative Ghostbusters. The human is shown as a shepherd and the sidekick as a dog.

when it terminates. Infinite-horizon MDPs have the advantage of only requiring us to solve for a single stationary policy. Were we to model these domains as finite-horizon MDPs, we would instead have to solve for multiple time-indexed, non-stationary policies, one for each time step.

For both Cops and Robbers and Collaborative Ghostbuster we solved for policies of abstracted versions of the problem by ignoring the non-target robbers and ghosts respectively. This abstraction works in the case of Cops and Robbers because the robbers have no influence on each others' behavior and have no impact on the outcome of the human or sidekick behaviors. In Collaborative Ghostbuster however this is not a lossless abstraction, since the game doesn't end after the first ghost is shot. In fact, strategically herding several ghosts to all be shot at once may be the optimal strategy. On the other hand, this abstraction dramatically reduces the space cost of value iteration and was previously used in the CAPIR framework.

3.7.2 Controller and Human Model Parameters

Each of the four action-selection methods used a human model constructed using value iteration on the multi-agent formulation of the domain as described in Section 3.6, with $\epsilon = 0.001$ for value iteration convergence. The noise parameter, β, was 0.1, which for these domains typically represented a 60% chance of selecting the highest $\mathcal{Q}$-valued action when $\mathcal{Q}$-values were very close, but strongly favored the on-policy action there was one clear winner according to $\mathcal{Q}$-values. For belief-state updates, the Bayes filter made the assumption that the human would maintain its type with probability 0.9 and select a new type uniformly at random otherwise.

To compute $\mathcal{Q}^*_{MDP}$ for the sidekick, value iteration was performed on the single-agent MDP incorporating the human model with the convergence threshold $\epsilon = 0.001$. In contrast to belief-state updates, this computation assumed that humans would not change targets. This was in line with the strategy of estimating the human's current goal and selecting actions to support it, without hedging against the possibility of the human changing goals in the future and in doing so follows the approach of the CAPIR framework.

For the action-entropy controller, the entropy threshold ϕ was 0.6, which was selected empirically for the best performance on all maps out of values ranging between 0 and 1.

3.7.3 Simulated Humans

We evaluated each action-selection method with a simulated human partner. Our simulated human represents an ego-centric greedy strategy from a below-average human player that doesn't consider how the sidekick is likely to respond to their actions, assuming that the sidekick will play around its poor choices. This contrasts with the expectation encoded in the sidekick's human model that the human is actively think-

ing about coordination with the sidekick. We took this as representative of play strategies that are actively attempting to win the game, but are not particularly effective, so that results produced using the simulated human are hopefully indicative of the performance of the controller when paired with a sub-optimal human player.

For Cops and Robbers, the simulated human selected a robber uniformly at random and myopically chased after that target, taking the shortest path to it, computed using A*. Once adjacent to its target it would wait until the sidekick was also adjacent before stepping into the robber's square to avoid letting the robber escape. With 0.1 chance it would instead select an action uniformly at random. Simulated humans never changed targets, even if that target became unreachable. If the simulated human was asked for its target by the sidekick it would respond with its true target with probability 0.9 and otherwise choose a response uniformly at random.

The simulated human for Collaborative Ghostbuster similarly selected a ghost uniformly at random and chased it myopically after it using A*. If it ever came within range of any ghost it would activate its zapping pulse. Like in Cops and Robbers, with 0.1 chance it would instead select an action uniformly at random. At each time step, with a probability of 0.1, it reselected a ghost to chase uniformly at random from the ghosts not yet zapped.

3.7.4 Trials

For both Cops and Robbers and Collaborative Ghostbuster we ran 1000 trials for each controller on each map. We used a turn limit of 100 for Cops and Robbers and 200 for Collaborative Ghostbuster. We measured the number of steps required to complete the game, either by catching a robber in Cops and Robbers or by eliminating all the ghosts in Collaborative Ghostbuster. Our tests were run on a system with a quad-core 2.3GHz Intel i7 processor and 8GB of RAM. In each case, the wall clock time

required for action selection was less than 20 milliseconds.

3.7.5 Results and Discussion

The results of our experimental trials differed distinctly across the two domains. We present and discuss each of them in turn.

3.7.5.1 Cops and Robbers

For each of the maps and action-selection methods we present the mean number of steps taken to catch a robber, both in aggregate across all targets and broken down by the particular target that the simulated human was chasing. This break down is particularly revealing of the kinds of errors that the controllers can make. Since the simulated humans never changed targets, they would never compromise by re-picking a robber that was easier to catch, given the sidekick's actions. In the worst case, the game could reach an unwinnable state if the sidekick trapped itself, since the myopic play of the simulated human would not help it recover.

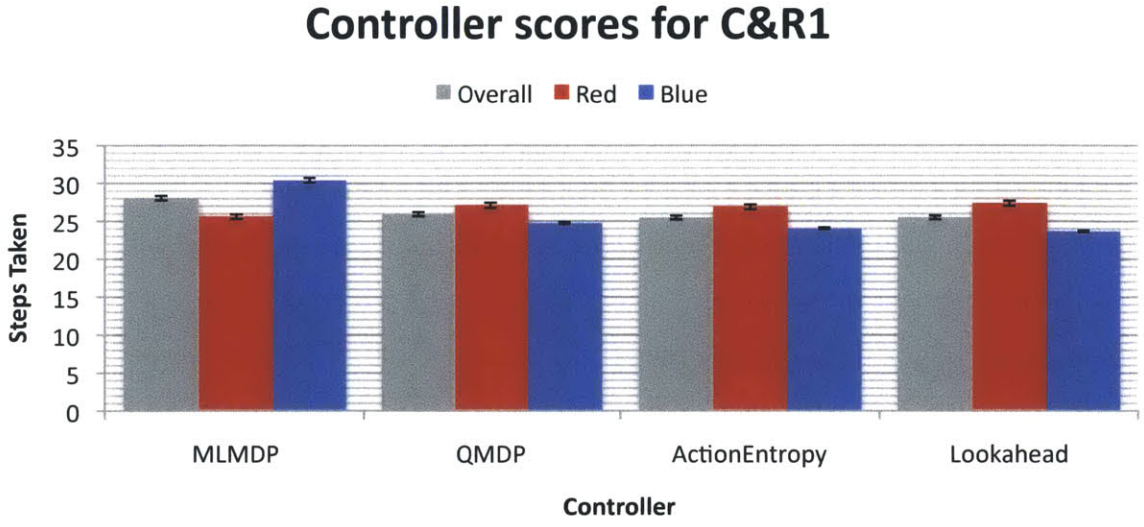


Figure 3-4: Mean steps taken to catch a robber in C&R1.

Figure 3-4 shows the results for C&R1. In this chart and throughout this section grey bars show the mean number of steps taken by each controller, averaged over 1000 trials. The colored bars show the mean number of steps taken, broken down by robber, so for instance the blue bar shows results specifically for the cases where the simulated human was chasing the blue robber. Error bars show the standard error of the sample mean.

The results for C&R1 demonstrate that when actions are easily reversible, the performance of each of the controllers is roughly comparable, with MLMDP performing slightly worse due to committing to actions moving east and south before having seen a disambiguating action from the human. This comes from the fact that MLMDP will always take an on-policy action for collaboratively chasing either of the two robbers. Action entropy prompted the player for their target an average of just over once per trial, and was still tied for the best performer despite having spent two steps to ask and hear the reply. One-step lookahead, by contrast, asked 0.3 times per trial.

As shown in Figure 3-5, on C&R2 the controllers all perform poorly for the yellow robber. This happens for the same reason in each case, which is that their model does not accurately predict the human's myopic behavior. Once it becomes clear that the human is moving south from the center of the map, the sidekick heads east through the door in the bottom corridor, expecting the human to head west, according to the joint policy for catching that robber. The simulated human, however, follows the sidekick through the door, resulting in the two having to chase the yellow robber around the map and try to pin it in a dead end. Even if the sidekick correctly identifies the human's target, this thwarts them.

The high mean and variance for MLMDP and action entropy chasing red stem from the sidekick misinterpreting noise in the player's actions as a commitment to chasing one of the other two robbers, causing the sidekick to step through either the eastern or western doors. In particular, a noisy pass action or bumping into a wall is

seen as a cue that the human is chasing yellow because it indicates that the human is waiting for the sidekick to drive yellow to the north. The resulting belief-state update typically causes action entropy to prompt for the human's goal, even when the true goal will soon be revealed through the simulated human's actions, which causes the chase to take longer than for MLMDP. By contrast, QMDP tends to early, which is typically the action that compromises between the different possible robbers, and avoids these problems.

The belief-space lookahead controller suffers a systematic failure when chasing the blue robber. It reaches the western door, at which point its one step of belief-space lookahead reveals that moving west is terrible if the goal is to chase either of the other targets, but that asking whether the human truly is chasing the blue robber will help let it know whether it really is worth stepping west. However, the belief-state update upon receiving an answer from the human will never result in it being certain enough to finally decide to step through, and one step of belief-space search is not deep enough to detect this, leading to the sidekick being stuck in a question

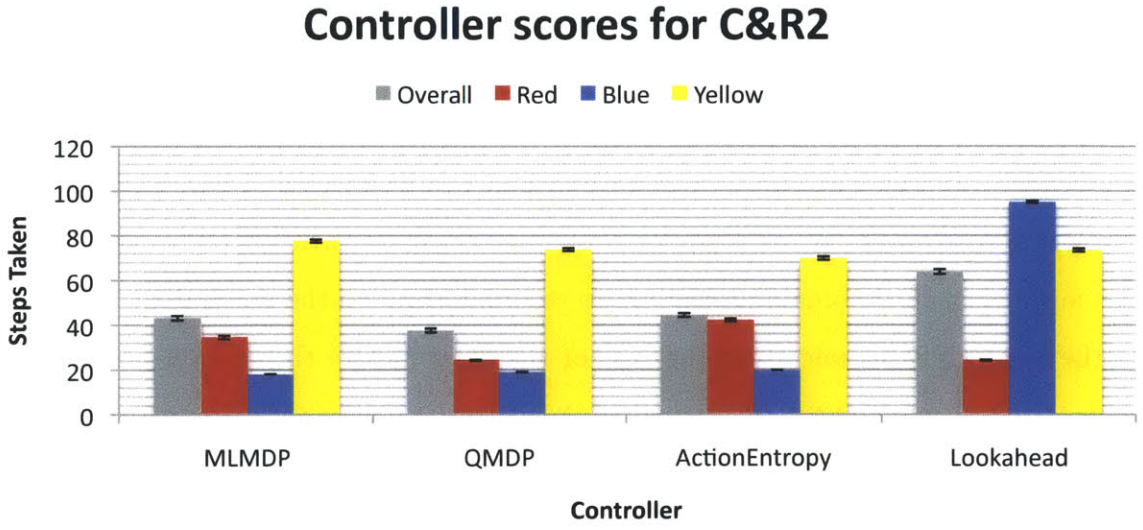


Figure 3-5: Mean steps taken to catch a robber in C&R2.

asking loop.

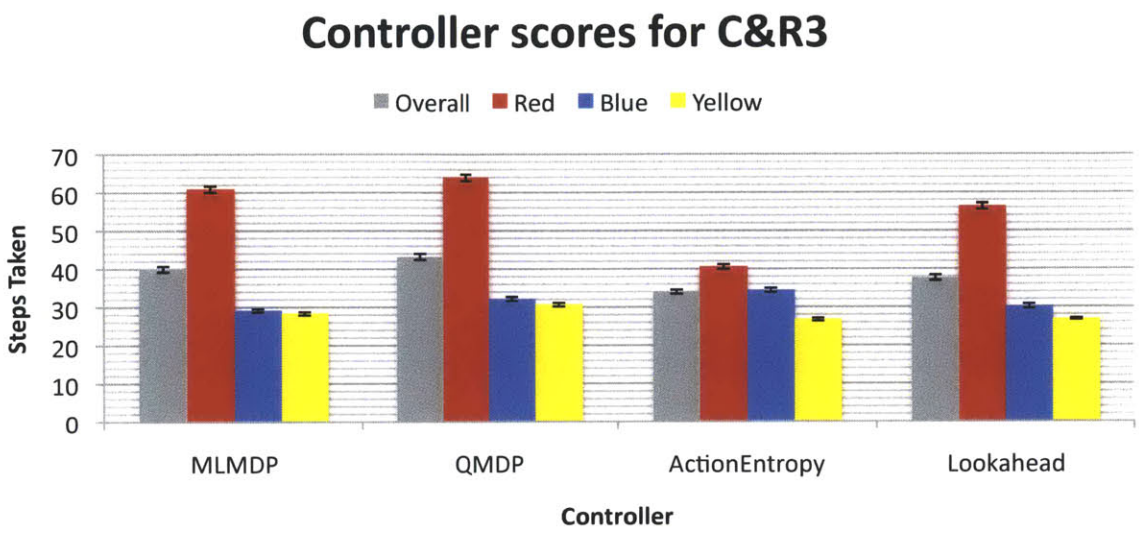


Figure 3-6: Mean steps taken to catch a robber in C&R3.

The results for C&R3 in Figure 3-6 show that MLMDP, QMDP, and one-step lookahead all perform similarly however action entropy's performance in chasing red gives it an edge. Action entropy always opens by asking the human for their target as its first action on this map, which allows it to set up red for an earlier capture when the human indicates that red is its target.

Figure 3-7 shows the results for C&R4. In each case the myopic simulated human frequently sabotages attempts at coordination, resulting in high variance scores as the sidekick attempts to make up for the human's missteps. MLMDP's early actions are contingent on tie breaking, leading to stepping through the north door two thirds of the time and the east door the rest of the time, before the human has revealed their intent through their actions, which leads to the higher variance in its scores.

Both action entropy and one-step lookahead consistently ask the human for their target before leaving their starting area, actively chasing the human's target. QMDP on the other hand waits until the human reveals its target by moving west to the room

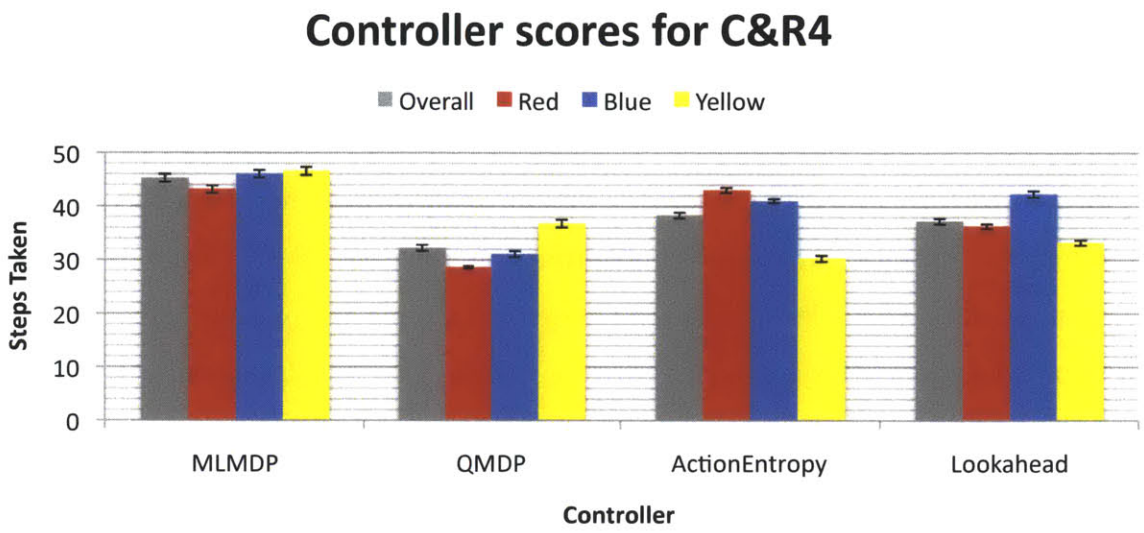


Figure 3-7: Mean steps taken to catch a robber in C&R4.

with red and blue or heading east to yellow. The particular interaction between the myopic actions of the human and the fact that the QMDP controller enters the room containing the target robber later than the other two controllers gives the QMDP controller an edge overall, particularly in the case of chasing red.

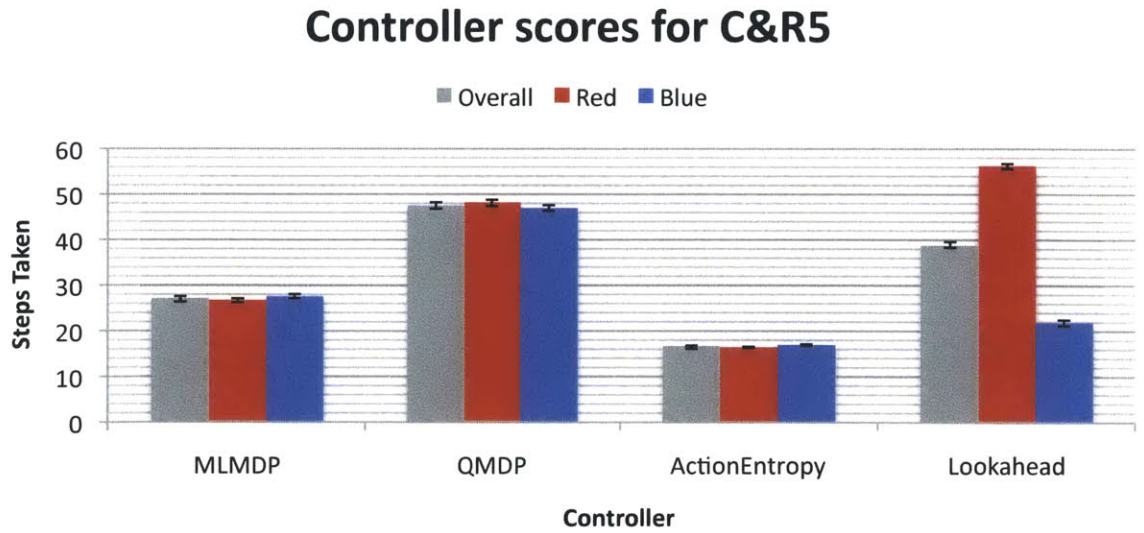


Figure 3-8: Mean steps taken to catch a robber in C&R5.

Figure 3-8 shows the results for C&R5. MLMDP's tie breaking causes it to arbitrarily pick one robber, and step through the door to chase it. If it happens to pick the same one as the human model, this works well, but if not it leads to a long chase. Action entropy consistently asks for the human's target on its first turn and chases it, leading to an optimal path almost every time. By contrast, QMDP waits to see what the human does, which leads to the simulated human moving up the central corridor, as opposed to going through a side corridor, which it does if the sidekick acts sooner to drive its target closer to it, leading to a circuitous chase.

Single-move belief-space lookahead has two failure modes. One is similar to the QMDP case, where not acting immediately causes the human to move up the central corridor. The other occurs when noise causes the simulated human to move west from its starting location. This triggers a particularly bad chain of reasoning, where the sidekick would like to step through the western door and drive the red ghost to the human. However, one step of lookahead reveals that the human is likely to move back into the central corridor, which from the sidekick's perspective is a lower value state than the current one. To prevent this move, the sidekick asks the human for its target, forcing it to respond and leading to a repeat of the same state. Similar to the case with C&R2, the single step of lookahead, and the ad hoc integration of question asking into the policy, means that the search will not reveal that change to the belief state will not be enough to prevent the same circle of reasoning on the next step, which leads to an infinite loop of question asking.

3.7.5.2 Collaborative Ghostbuster

The results for both GB1 and GB2 are presented in Figures 3-9 and 3-10. In contrast to Cops and Robbers there is no breakdown by target, since the goal is to zap all ghosts. All four controllers perform similarly. This stems from the fact that actions in Collaborative Ghostbuster are largely reversible and do not commit the sidekick

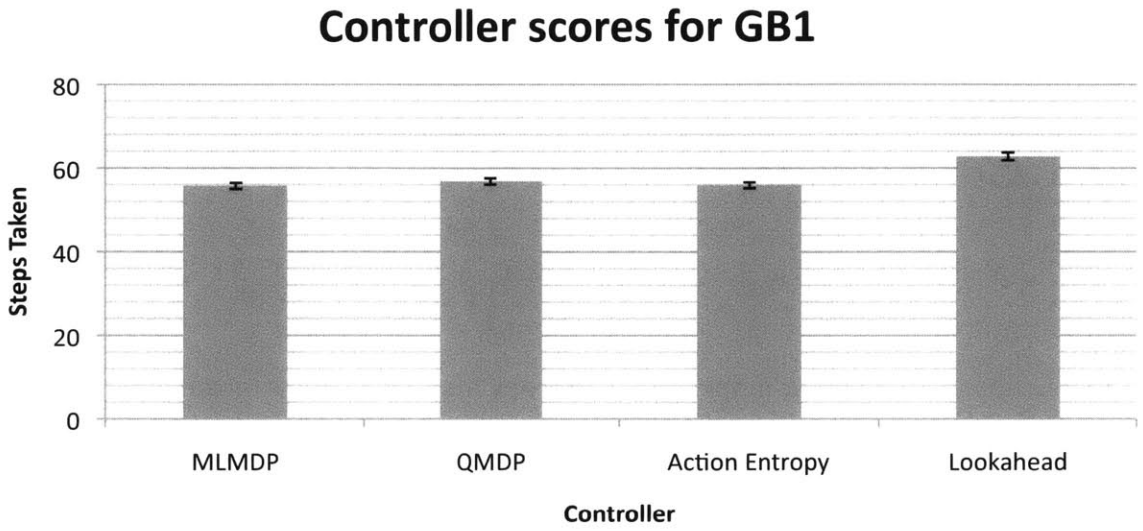


Figure 3-9: Mean steps taken by each controller in GB1.

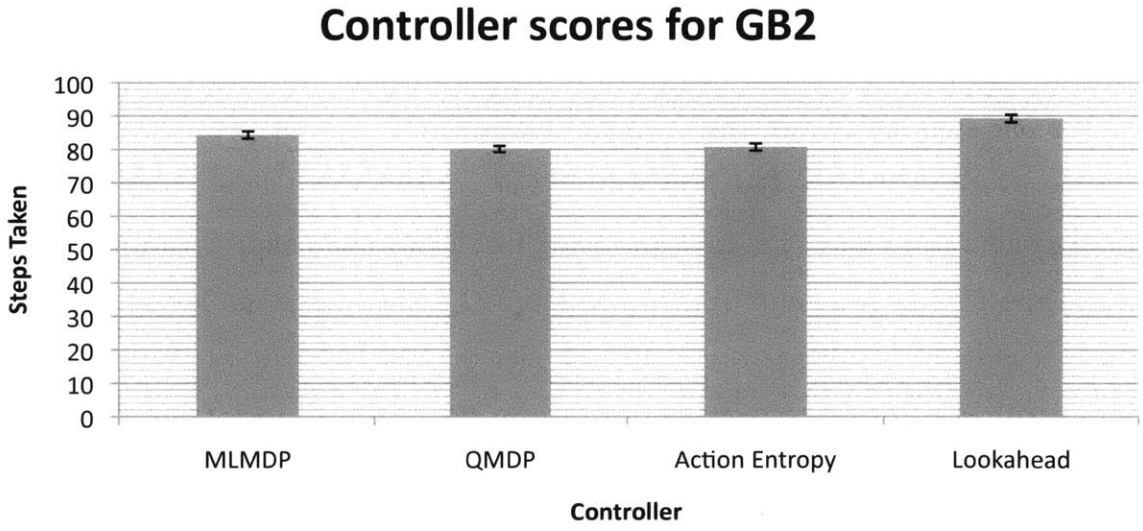


Figure 3-10: Mean steps taken by each controller in GB2.

to a particular course of action, unlike the case of Cops and Robbers, where doors make recovering from errors challenging. These characteristics are shared with C&R1, which showed similar results. Additionally, the absence of communication actions in Collaborative Ghostbuster removes the advantage that the action entropy and one-

step lookahead controllers have through using those actions.

3.8 Conclusions

As mentioned in the previous section, in cases characterized by the ability to easily recover from being mistaken about the human's target, such as C&R1 and both Collaborative Ghostbuster maps, action entropy and belief-space lookahead gave little benefit if any over the methods that did not actively reason about the effect of the sidekick's actions on their beliefs. For C&R 2-5 the results were mixed. Both action entropy and belief-space lookahead were able to incorporate information gathering, and communication in particular, into their action repertoire, but this only made a notable improvement for action entropy over QMDP and maximum likelihood on C&R 3 and 5.

For one-step lookahead, the communication action in some cases was an active hinderance, since its one step of lookahead was not enough to detect that, in asking for the sidekick's target, it could find itself in a belief state that would cause it to repeat itself. This is an effect of the value function that one-step lookahead uses to assess actions not taking into account the fact that the sidekick will be performing lookahead again to select actions in the future. We used one step of lookahead here, but in general, belief-space lookahead would not be able to detect a potential action loop outside of its search horizon.

The domains in this chapter represent the practical limit of the size of the state space that our value-iteration implementation was able to handle, with the largest map, GB2, containing 56^5 states, reduced to 56^3 since in our abstraction we only reasoned about the position of the human, sidekick, and target ghost. In general, the memory required for solving an MDP with value iteration scales with the size of the state space, which grows exponentially with the number of state variables, unless a

compact representation of the value function can be found. Commercial video games have very large state spaces, so ideally we would like to avoid representing the entire value function if it is not required.

This scalability concern, along with the shortcomings of limited lookahead in belief space, motivate our approach in the next chapter. Rather than attempting to solve the fully observable MDP and use its value function for action selection, we will use POMCP to search directly in the action-observation tree. By using estimates from Monte Carlo simulations we will avoid having to represent the entire value function. Additionally POMCP's sample-based search allows it to manage the branching factor of the search space and search all the way to the planning horizon, thereby avoiding the kinds of looping problems we found with belief-space lookahead.

Chapter 4

Belief-space Planning for Sidekicks

4.1 Introduction

The controllers based on state-space planning for building sidekick controllers from Chapter 3 suffered from key weaknesses that we address in this chapter.

Each state-space planning approach, as well as our human model, made use of value iteration on an MDP formulation of the AI sidekick POMDP or the game dynamics. Value iteration, however, has the drawback that in the worst case it must explicitly represent $V^*(s)$ for each s in $\mathcal{S}$, and $|\mathcal{S}|$ is exponential in the number of state variables. Some domains allow for V^* to be factored so as avoid this, but it is true in the general case. For commercial video games domains, even those with modest scopes, the number of state variables will often be too large for an approach using value iteration to be practical.

In addition to the problems of using value iteration, our decision-theoretic human model required that, other than its own internal state, the rest of the domain be fully observable. Ideally we would like to be able to build sidekick controllers for domains with a larger amount of partial observability.

Although the belief-space lookahead and action entropy controllers did incorporate

information gathering and communication actions into the sidekick's repertoire, they did so in an ad hoc way. For lookahead, the result was that the sidekick failed to reason correctly about what its future strategy would be, because its lookahead assumed that it would be selecting actions according to its state-space policy, when in fact it would be selecting actions on the basis of its lookahead. In some cases this caused behavioral loops. Action entropy, on the other hand, performed better on our test domains, but both parts of its dual control strategy were ad hoc. It selected actions suggested by policies that assumed that all actions subsequent to the one being chosen would be on-policy, when this was not in fact the case. The same applies to the QMDP and maximum likelihood controllers, with the additional downside that they could not incorporate information gathering and communication.

To address these problems we turn to the Partially Observable Monte Carlo Planning (POMCP) algorithm as a combined action selection and state estimation method for building a sidekick controller [57].

POMCP plans by searching directly in the space of action-observation histories, which we will refer to, slightly inaccurately, as belief-space planning. Strictly speaking, belief-space planning implies that we are planning in space of belief states, however it will be more practical for us to use histories. If necessary a history can be converted to a belief state using Bayes' rule. Working directly in belief space, or with histories, alleviates the need for ad-hoc integration of communication and information gathering actions, because the search will inherently take into account the effect of the sidekick's actions on their belief state.

There are trade-offs to searching in the tree of action-observation histories versus searching in the tree of beliefs. The primary advantage of using action-observations histories is that it, as we shall discuss later in detail, it allows us to use a black-box generative model of the dynamics of the domain to do the search using sampling, rather than requiring that we have an explicit model of the domain. It also alleviates

the need to do a possibly costly belief update at each step of the search. The key disadvantage however is that when two different histories map to the same belief, we will not take advantage of this fact to save on time and space as we would if we were to use the belief tree.

One worry about the move to belief space may be that the scalability issues experienced in value iteration may be exacerbated. However, POMCP also has the property that it focuses its planning efforts in the region of belief space reachable from the agent's current belief and that is most likely to be reachable via the algorithm's estimate of π^*. This avoids needing to represent a large part of its estimates for V^* or π^*, which helps address the scalability concerns.

Planning in belief space means we must contend with the PSPACE-Completeness of POMDPs and will need to be content with only approximating the optimal POMDP policy. POMCP is an approximate, anytime algorithm, meaning that, if parameterized correctly, it continually improves its estimate of π^* as it runs, converging to the true optimal policy in the limit, but it can be stopped at any time to give an approximation to π^* [57]. The asymptotic convergence rate is domain independent, but may be slower than is feasible for games domains, which often require fast decisions from a sidekick.

Building human models for partially observable domains is a challenging problem, since modeling them as decision-theoretic planners as we did in the previous chapter is intractable given partial observability. We would still like to demonstrate POMCP as a solution method for the AI sidekick problem in these domains however, so we fall back on heuristic human models.

Our contribution in this chapter is to demonstrate how POMCP can be applied to the AI sidekick problem and to introduce an ensemble-based parallelization strategy for improving the performance of POMCP. We demonstrate these techniques on a variety of games domains, using heuristic human models.

4.2 POMCP

POMCP is comprised of two steps. A search step called Partially Observable UCT (PO-UCT), and a belief update step that uses particle filtering to approximate an exact belief-state update. The key idea behind PO-UCT is to sample states from the current belief state, and use each of these samples as a starting point to generate a history, by selecting actions and generating observations from a simulator that implements the dynamics of the POMDP until a terminal state or effective horizon is reached. The action selection that constitutes the search is guided partly by statistics gathered from previous simulations, and partly by random action selection.

After a set timeout or number of histories have been generated, POMCP stops sampling and takes the action with the highest expected value at the root of the tree, according to its current estimate from the simulations it has produced. After taking the action and receiving an observation from its true environment, it updates its belief state via particle filtering and begins its search step again.

In the next sections we discuss some of the core elements of POMCP, noting any special requirements for the AI sidekick problem, and then present the algorithm itself.

4.2.1 Simulator

The use of a simulator, $\mathcal{G}$, that implements the transition, observation, and reward dynamics of a POMDP distinguishes POMCP from approaches that use explicit probabilistic models such as value iteration. One step of simulation, $(s', z, \rho) \sim \mathcal{G}(s, a)$, takes the state s and action a as input and outputs a tuple of successor state s', observation z, and reward ρ sampled from the dynamics of the POMDP. By repeatedly using successor states and new actions as inputs to the simulator an action-observation history can be produced by concatenating the sequence of input actions and output

observations.

Two basic approaches to creating a simulator for a video game domain are to use the game engine for simulation and to construct a simulator from an explicit probabilistic model of the game's dynamics.

The engine approach requires the game engine to be built such that it can be set to an arbitrary state and can produce successor states upon demand, given actions, and provide information about what the player and sidekick can observe in the resulting state. These requirements are not trivial, since in many engines, resetting the game state may be costly, and that cost must be paid for each trajectory.

The explicit model approach avoids dealing with the engine at the cost of a greater modeling effort. In this approach the game must be expressed in an explicit POMDP formulation. For many current video games this is non-trivial and, as with the case of the engine approach, may involve discretizing continuous elements of the game. Given such a formulation however, it is straightforward to implement a simulator. Given a state and action, the transition, observation, and reward functions can be sampled to give the outputs that POMCP requires.

There are challenges involved in both of these approaches that are general for any POMDP formulation of the problem. For real-time games with continuous action spaces, the notion of a time step or even a discrete set of actions may be unclear. In these cases either the game must be designed to work around this or POMCP must interact with the engine through an interface that provides this layer of abstraction or in the explicit model approach the model must make this abstraction. Additionally, many games have no concept of reward in the decision-theoretic sense, in which case the reward signal would need to be generated by some other process or specified in the explicit model.

For the AI sidekick problem the simulator must also allow for a human model to select actions. With an explicit model this requires formulating the POMDP to

include the human model as outlined in Chapter 2. Using a game engine would typically require building a wrapper around the engine to convert a two-player game into a single-player game. The process for doing this is game engine specific, but in short the simulator must implement $\mathcal{T}_h$ and $\mathcal{T}_s$ as generative processes. Sampling $\mathcal{T}_h$ given a state and sidekick action gives the intermediate state immediately after the sidekick's action but before the human's. The human's state update and action selection are then sampled from the human model H and used as input for sampling from $\mathcal{T}_h$, which gives the resulting state and observation for the sidekick. The game engine may also be used for $\bar{\mathcal{O}}$ in H. The wrapper also needs to ensure the correct properties of the state, in particular that the human's last action is embedded in the state in order for the $\mathcal{T}_h$ to extract it correctly in the next step. It is also responsible for sampling $\mathcal{R}$, which again may need to be specified externally to the game engine.

4.2.2 Belief Function

POMCP searches in the space $\mathcal{H}$ of action-observation histories and we will often want to know what belief state corresponds to one of these histories. The belief function $\mathbb{B}(s,h) = \Pr(s|h)$, performs this conversion. Given an initial belief $\mathcal{I}$, using Equation 2.13 from Section 2.4 allows us to write

$$\mathbb{B}(s', haz) = \frac{\sum_{s\in\mathcal{S}} \mathbb{B}(s,h)\tau(s,a,s')o(s',a,z)}{\sum_{s\in\mathcal{S}}\sum_{s''\in\mathcal{S}} \mathbb{B}(s,h)\tau(s,a,s'')o(s'',a,z))}. \tag{4.1}$$

where haz is a sequence of actions and observations that ends in the action a and observation z.

Although in principle $\mathbb{B}$ could be computed using Equation 4.1, POMCP makes the assumption that the POMDP dynamics are unavailable, and instead uses a particle filter to compute $\mathbb{B}(s,h)$ during its belief-state update step. For convenience we will often write the belief state b using this notation as $\mathbb{B}(\cdot,h)$.

4.2.3 Search Tree and Returns

POMCP uses a tree data structure $\mathbb{T}$ that contains a set of nodes storing statistics about histories. The node $\mathbb{T}(h)$ stores a tuple $(\mathbb{N}(h), \mathbb{V}(h), \mathbb{B}(\cdot, h))$ of the number of times that the history h has been visited, a running estimate of the value of h, and a set of particles representing the belief state for h.

A return, $\mathbb{R}^t$, is the total discounted reward accumulated in a simulation from time t onwards, with $\mathbb{V}(h)$, being an estimate of the expected return of the optimal policy, starting from the belief state $\mathbb{B}(\cdot, h)$.

4.2.4 PO-UCT Search

Algorithm 4.1 shows the PO-UCT step of POMCP, which performs UCT search on the tree of histories rooted at the sidekick's current history [36]. It generates these histories via Monte-Carlo simulation using the simulator and two different action selection policies. Which policy is used depends on the stage of the search. The stages are Simulate and Rollout.

Each round of Simulate begins by sampling a state from $\mathbb{B}(\cdot, h)$. During Simulate, actions are selected according to the UCB1 rule, which chooses actions in one of two ways. If there are one or more actions, a, for which $\mathbb{N}(h, a) = 0$, then UCB1 picks one of them uniformly at random, and takes that action. If each action has been selected at least once in h, then the UCB1 selects $\underset{a}{\operatorname{argmax}}\, \mathbb{V}(ha) + c\sqrt{\log \mathbb{N}(h)/\mathbb{N}(ha)}$, where c is a constant that controls the ratio of exploration to exploitation in the search. If c is chosen appropriately, then PO-UCT's estimate of V^* will converge to the true value in the limit. After selecting each action, Simulate uses the simulator to generate an observation, successor state, and reward, appending the action and observation to the search's history. The Rollout phase begins when the search reaches a history that has no node in $\mathbb{T}$.

During Rollout, actions are selected according to a stochastic rollout policy, typically uniformly at random, although convergence can be faster if a more informed policy is used. In order to preserve PO-UCT's convergence guarantees, the rollout policy must have a non-zero probability of choosing each available action for a given history. Rollout ends when a terminal state is reached or the planning horizon d_{max} is reached. For infinite horizon problems the effective horizon $d_{max} = 1/(1-\gamma)$ can be used.

Once Rollout completes, statistics on the rewards accumulated and history visitation counts throughout the simulation are recorded in $\mathbb{T}$. Additionally $\mathbb{T}$ is extended to include the node for the history at which the Rollout phase first began. The Rollout phase then ends and a new Simulation phase begins with a new state being sampled. This continues until a maximum number of simulated histories k is reached. Alternatively simulations can be continued until a timeout is reached.

4.2.5 Belief Update

POMCP uses particle-filter-like rejection sampling to approximate $\mathbb{B}(\cdot, h)$. The filter is initialized by sampling a set of particles, $\mathcal{K}$, from the initial belief. After performing an action a in the real game, and receiving the real observation z, POMCP performs its belief update step using rejection sampling to generate $|\mathcal{K}|$ new particles. In practice, it is possible to reuse some of the state samples generated during the tree-search step to reduce the number of particles that must be generated during the belief update step. Particle reinvigoration techniques, such as introducing artificial noise into the current set of particles, can also be introduced to avoid particle deprivation.

Algorithm 4.1 Partially Observable UCT (PO-UCT)

procedure SEARCH(h)
  for $i = 1 \rightarrow k$ **do**
    if $h = \emptyset$ **then**
      $s \sim \mathcal{I}$
    else
      $s \sim \mathbb{B}(\cdot, h)$
    end if
    SIMULATE($s, h, 0$)
  end for
  return $\operatorname{argmax}_a \mathbb{V}(ha)$
end procedure
procedure SIMULATE(s, h, d)
  if $d > d_{max}$ **then**
    return 0
  end if
  if $h \notin \mathbb{T}$ **then**
    for all $a \in \mathcal{A}$ **do**
      $\mathbb{T}(ha) \leftarrow (\mathbb{N}_{init}(ha), \mathbb{V}_{init}(ha), \emptyset)$
    end for
    return ROLLOUT(s, h, d)
  end if
  $a = \operatorname{argmax}_{\hat{a}} \mathbb{V}(h\hat{a}) + c\sqrt{\log \mathbb{N}(h)/\mathbb{N}(h\hat{a})}$
  $(s', z, \rho) \sim \mathcal{G}(s, a)$
  $\mathbb{R} \leftarrow \rho + \gamma$.SIMULATE($s', haz, d + 1$)
  $\mathbb{B}(\cdot, h) \leftarrow \mathbb{B}(\cdot, h) \cup \{s\}$
  $\mathbb{N}(h) \leftarrow \mathbb{N}(h) + 1$
  $\mathbb{N}(ha) \leftarrow \mathbb{N}(ha) + 1$
  $\mathbb{V}(ha) \leftarrow \mathbb{V}(ha) + (\mathbb{R} - \mathbb{V}(ha))/\mathbb{N}(ha)$
  return $\mathbb{R}$
end procedure
procedure ROLLOUT(s, h, d)
  if $d > d_{max}$ **then**
    return 0
  end if
  $a \sim \pi_{\text{rollout}}(h, \cdot)$
  $(s', z, \rho) \sim \mathcal{G}(s, a)$
  return $\rho + \gamma$.ROLLOUT($s', haz, d + 1$)
end procedure

4.3 Human Models

Since we wish to generalize our planning to problems beyond the restricted set that we considered in Chapter 3 in which only the internal state of the human was hidden, the modeling approach presented in the Section 3.6 is not applicable. The generalization of the joint MDP solving method to partially observable domains is solving a Dec-POMDP, which is intractable. Instead we will be forced to rely on domain-specific heuristic human models that we will introduce in Section 4.5.3.

4.4 Parallelization

Search parallelization in UCT has been shown to improve performance in two ways across a range of MDP domains, which motivates our parallelization approach for the POMDP case [21].

The first kind of improvement is parallel time advantage, in which combining the results of multiple searches with k simulations performed across n cores produces higher quality actions than a single search done with k on one core. The existence of parallel time advantage implies that adding extra cores can improve action quality, keeping the time cost fixed.

The second kind of improvement is single-core time advantage, in which combining the results of multiple searches with k simulations performed across n cores produces actions with at least as high quality as a single search done with $n \cdot k$ simulations on one core. The implication of single-core time advantage is that it is possible to reduce the time cost of producing actions by running multiple short searches in parallel, rather than a single long search.

A naive approach to parallelizing PO-UCT might be to split the work of performing the Simulate and Rollout phases across cores. However, since PO-UCT requires

both read and write access to $\mathbb{T}$ in order to select actions and update return statistics, synchronizing these operations on a common data structure is a challenge. We sidestepped this problem and developed ensemble PO-UCT, based on ensemble UCT [21], in which threads operate solely on local data, only requiring one point of synchronization at the end of the search.

Ensemble UCT is a parallelization approach that is parameterized by an ensemble size, n, a number of simulated histories, k, and the UCT exploration constant, c. It uses a root parallelization strategy that splits the work into n separate UCT searches, distributing them across n cores, generating k state-action histories in each search, and then combining the statistics at the roots of each of the search trees once they are all complete. With a small modification, this strategy can be applied to PO-UCT.

Ensemble PO-UCT uses the same parallelization strategy as Ensemble UCT, with two exceptions. First, $\mathbb{B}$ is passed as an additional parameter to each core, so that initial states can be sampled, and second, the search is over action-observation histories rather than state-action histories.

Let $\mathbb{V}^i(ha)$ and $\mathbb{N}^i(ha)$ be respectively the value function estimate and visit counts at depth 1 of $\mathbb{T}$ on i-th core after PO-UCT is complete. The root parallelization value function estimate is then

$$\mathbb{V}_{RP}(ha) = \frac{\sum_{i=1}^{n} \mathbb{V}^i(ha) \cdot \mathbb{N}^i(ha)}{\sum_i \mathbb{N}^i(ha)}. \tag{4.2}$$

Additionally, the particles generated during PO-UCT at depth 2 in each tree can be combined for the belief update step can be combined. Let $\mathbb{B}^i(\cdot, haz)$ be the set of particles stored in node $\mathbb{T}(haz)$ on i-th core after PO-UCT is complete. The combined particle set is then simply

$$\mathbb{B}(\cdot, haz) = \bigcup_{i=1}^{n} \mathbb{B}^i(haz). \tag{4.3}$$

The rationale behind these ensemble methods is that performing n searches with k simulated histories in parallel and combining the root statistics is likely to outperform a single search with k simulated histories, since averaging across the independent approximations of $V^*(ha)$ at the root reduces the variance from the Monte Carlo simulations. Clearly, however, this reduction in variance comes at the cost in memory of performing n independent searches in parallel.

Empirically, both parallel time advantage and single-core time advantage have been demonstrated across a range of fully observable domains, including Backgammon and Yahtzee. These results were contingent on $n \cdot k$ being sufficiently large [21]. We will present evidence for both kinds of advantage for the Cops and Robbers domain in Section 4.5.6.5.

4.5 Empirical Evaluation

In this section we discuss the domains, planner and model parameters, and experimental setup used for evaluating the methods described in this chapter. We conclude with experimental results and discussion.

4.5.1 Domains

We used three domains in our evaluation: Cops and Robbers, Collaborative Ghostbuster, and Hidden Gold.

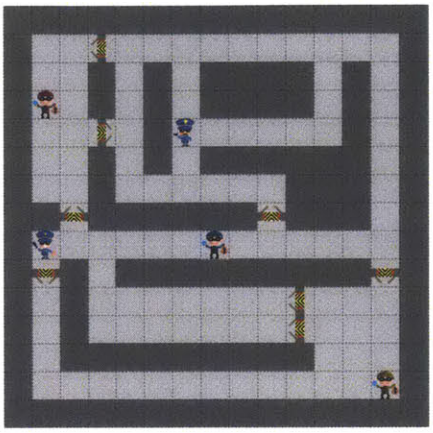
Cops and Robbers is described in Chapter 3. We used the same maps to give a point of comparison. To test the scalability of the POMCP approach we also introduced two larger maps, L1 and L2, shown in Figure 4-1, that were intractable for our value iteration implementation. L1 has 110^5 states, with the shortest possible action sequence to catch the furthest robber, yellow, being 44 steps. L2 has 339^5 states, with the shortest possible action sequence to catch the furthest robber, blue,

being 72 steps. Although our state-space planner only tried to solve for the optimal plan for catching a single robber, which requires considering 110^3 and 339^3 state projections of L1 and L2's state-spaces respectively, this was still beyond the memory limits of our value iteration implementation and hardware.

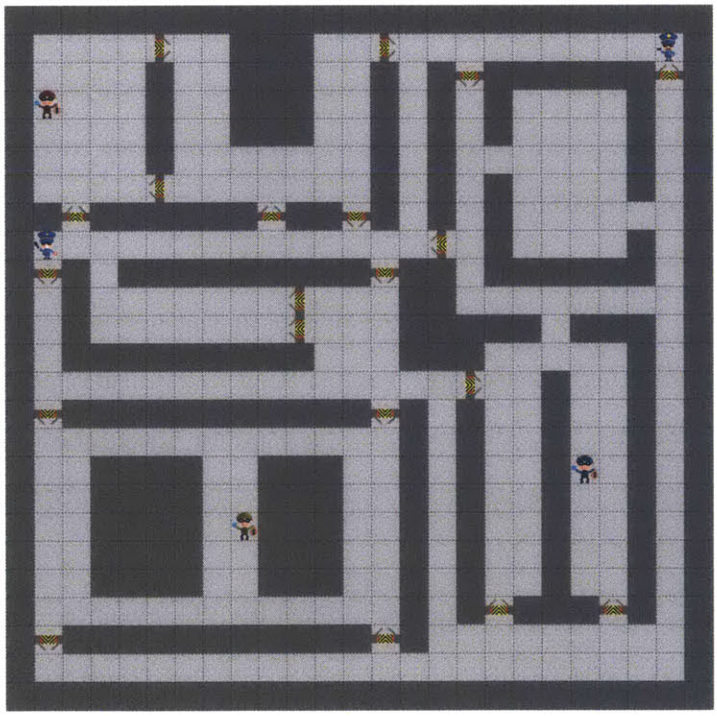
Collaborative Ghostbuster and the maps used were described in Section 3.7.1.

Hidden Gold is a collaborative search game in which the human and the sidekick are pirates searching for buried treasure. The pirates have an unreliable map with multiple Xs marked on it, indicating the possible location of the treasure. Any pirate walking next to, or on top of the treasure receives a glitter observation in its direction. The pirates are always visible to each other. Digging up the treasure and winning the game requires both players to stand on top of it, which gives a reward of 1. All other state-action combinations give 0 reward. Pirates are able to move in any of the cardinal directions, pass, and shout "Yarr!" to one another to indicate that they've found something or simply fool around. As with the other domains, the intentions of the human are a hidden part of the state from the sidekick's perspective. However, unlike the others, there is partial observability in the general environment too. To do well, the sidekick must reason about what their own observations reveal about the location of the treasure, as well as what observations the human has likely received, and where the human is planning to search, so that the sidekick can plan its own efficient search path.

We tested our POMCP controller on four Hidden Gold maps, shown in Figure 4-2. HG1 is a simple symmetrical map, with Xs in two corners. HG2 has two regions in the north, one containing two Xs close to each other and the other containing only one. In HG3 it is possible to check the X in the east on the way to the X in the southeast. HG4 is similar in structure to HG2, in that there are two Xs in a region of the map that are relatively close to each other and one farther away, but it takes more of a commitment to check both of the close Xs than in HG2.

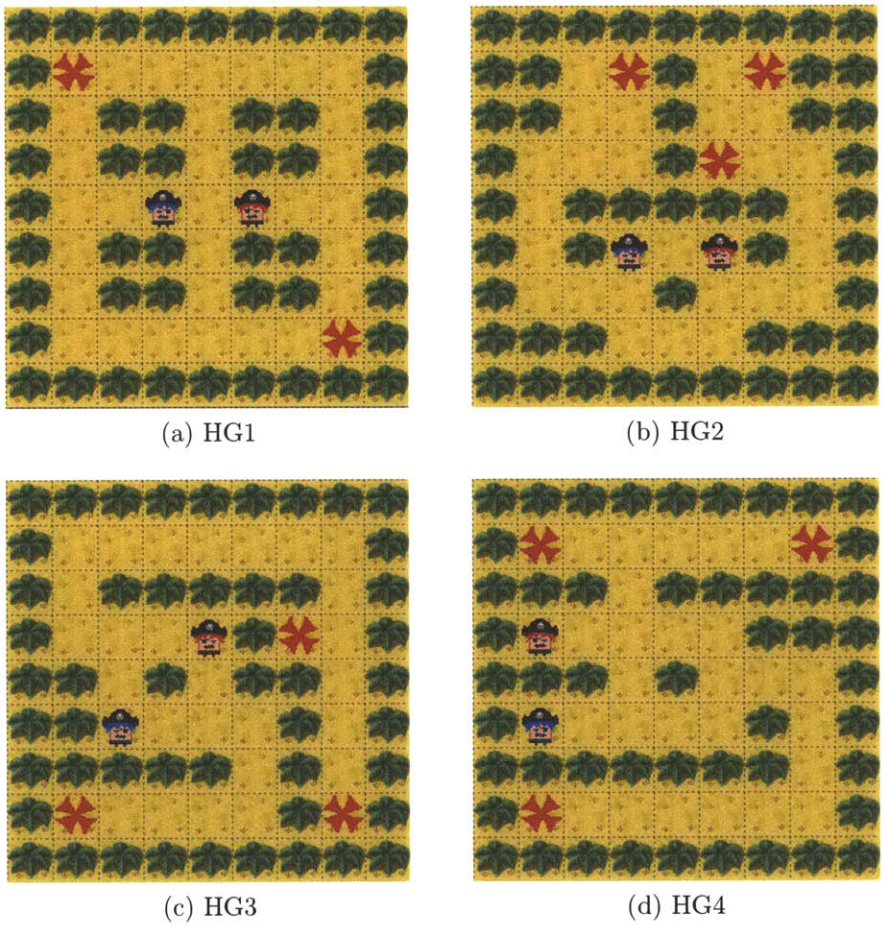


(a) L1



(b) L2

Figure 4-1: The maps used for Cops and Robbers scalability tests. The human is shown holding a gun, whereas the sidekick holds a doughnut and truncheon. Doors only open in the direction of the chevrons.



(a) HG1 (b) HG2

(c) HG3 (d) HG4

Figure 4-2: The maps used for Hidden Gold. The human has blue hair and the sidekick has red hair.

4.5.2 Planner Parameters

For Cops and Robbers, POMCP's UCB1 parameter c was 1, chosen empirically, and γ was 0.99. We used the particles generated from forward search for the belief update, as described in section 4.2.5.For C&R1 through 5 we terminated simulated histories after 100 steps. For our rollout policy we biased action selection towards minimizing the Manhattan distance to the sampled human model's target, such that these actions were twice as likely to be selected. We used 50,000 simulated histories per action for C&R1 through 5. We discuss the settings used for L1 and L2 in section 4.5.6.4.

For Collaborative Ghostbuster we used the same POMCP settings as for Cops and Robbers, but with $c = 3$, again chosen empirically, and terminating simulated histories after 200 steps.

For Hidden Gold we again used the same settings, but with $c = 1$ and uniform random action selection as the default policy during the Rollout phase.

Since no state-space planning based controller could be built for Hidden Gold, we also built a heuristic planner as a point of comparison. The planner behaved in the same way as the Hidden Gold human model, which we will describe in the following section, but unlike that model it kept track of which Xs had been investigated by both players. When selecting a new target after having found no gold at the X it was investigating, it would only choose from among the Xs that hadn't been visited be either player. It did not have the random target switching behavior of the model.

We did not use particle reinvigoration except in the case of complete particle deprivation, where we attempted to recover by sampling another 1000 particles via rejection sampling in the same manner as for the belief update. This happened in less than 5 cases over all trials.

Throughout the results section we show the results of the QMDP planner using the settings presented in Chapter 3 as a point of comparison.

4.5.3 Human Models

For Cops and Robbers, we tested POMCP with two different human models. The first was a decision-theoretic model constructed via state-space planning as in Chapter 3. It used the same configuration and parameters as described in Section 3.7.2 with one exception. In the previous previous chapter the internal state-update function Δ assumed no type switching for value iteration tractability reasons, and then type switching was later accounted for in the belief update. For the model used in POMCP however, Δ switched types each step with probability 0.1 as described in Section 3.6.5. This means that POMCP's value function estimates assumed that the human may switch type in the course of pursuing a robber.

Since L1 and L2 were intractractable for our value iteration implementation we were forced to implement a heuristic human model. As we have argued in the case of the simulated human, a reasonable expectation of human behavior in Cops and Robbers is for them to have some target robber in mind and to simply move by the shortest path towards their target, relying on the sidekick to react to their actions appropriately. We implemented such a model by replacing our decision-theoretic models action selection criteria for constructing $\bar{\pi}$ with the policy used by the simulated human in Chapter 3. This policy moved the human towards their target by A*, although avoiding stepping on a Robber prematurely, with probability 0.9 and selected a random action with probability 0.1. The heuristic model was otherwise the same as the decision theoretic model, including Δ.

For Collaborative Ghostbuster we also built a heuristic model using the same strategy. As with the Cops and Robbers model we replaced $\bar{\pi}$ with the action selection strategy used by the simulated human in Chapter 3.

For Hidden Gold we again built a heuristic model that represents a suboptimal human player who wanders around searching for the gold without remembering where

it or the sidekick have already looked. We modeled the intention to search at each X as a type-variable in the human's internal state, as with the other two domains. $\bar{\pi}$ and Δ were constructed according to the following rules: The human's default action was to move towards the X corresponding to its target using A*. If the human received a glimmer observation it switched its type to that of the X at which the glimmer was spotted. Upon reaching its target X, if it received a glimmer observation it stood on the X and switched to saying "Yarr" with 0.5 probability each time step. If not, it switched to a new type selected uniformly at random. If it heard the sidekick say "Yarr" while standing within one square of an X it switched to that X's type. If not standing on gold, it also switched types with probability 0.1 to a new one chosen uniformly at random. With probability 0.1 the human's action would instead be selected at random.

4.5.4 Simulated humans

We used simulated humans as partners in evaluating each controller. For Cops and Robbers and Collaborative Ghostbuster we used the same simulated humans as in Chapter 3. For Hidden Gold we used the human model described above as the simulated human, but without the random target switching behavior. In all domains, the initial targets of the simulated humans were chosen uniformly at random.

4.5.5 Trials

For all maps in all domains we ran 100 trials for each controller. We used a turn limit of 100 for Cops and Robbers on C&R1 through 5 and 200 for every other domain and map. We measured the mean number of steps required to complete the game, either by catching a robber in Cops and Robbers, eliminating all the ghosts in Collaborative Ghostbuster, or retrieving the treasure in Hidden Gold. We also measured the mean

wall clock time taken to select an action. Our tests were run on a system with a quad-core 2.3GHz Intel i7 processor and 8GB of RAM.

4.5.6 Results and Discussion

4.5.6.1 Cops and Robbers

We compared three variants of the POMCP controller to investigate the contributions of different aspects of the controller to its performance in Cops and Robbers, particularly in comparison to the QMDP controller from Chapter 3. These were POMCP, POMCP-NC, and POMCP-Q.

POMCP used the heuristic model and had full access to communication actions. It was the most advantaged controller. Its human model's policy matched the simulated human's, although its internal state update function did not, and communication gave it a near oracular action for discovering the human's true goal with minimal effort. This gave it a double advantage over QMDP, whose model did not match so closely and which did not have access to communication.

POMCP-NC used the heuristic model with communication actions disabled. It was forced to infer human intentions solely from their actions, which put it on a more even footing with QMDP, but its model still closely matched the simulated human.

POMCP-Q used the decision-theoretic human model with communication actions disabled, making its capabilities the most directly comparable to QMDP.

Figure 4.5.6.1 shows the mean steps taken by each controller in C&R1 through 5. Error bars show standard the standard error of the sample mean. POMCP was the best performer over all, with POMCP-NC only marginally worse in most cases. As a result of its belief-space planning, on C&R2 though 5 POMCP consistently used its communication action within its first four actions to disambiguate the human's goal and was able to respond well from that point onwards.

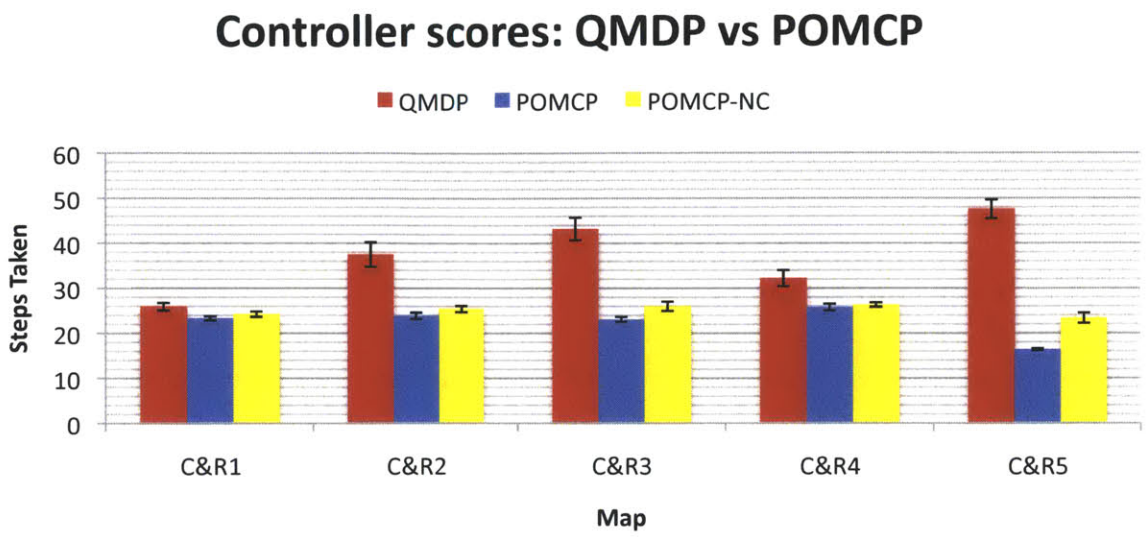


Figure 4-3: Mean steps taken by each controller to catch a robber in C&R1 through 5.

Like QMDP, POMCP-NC was forced to wait for the human's actions to disambiguate their goal or overcommit early, resulting in worse performance. Overcommitment to chasing the wrong target was most apparent in C&R5, where the controller typically followed a strategy of gambling on stepping through a door early in the hope of picking the right target and preventing the human from walking all the way up the center, which it would do if the sidekick instead waited. This led to a worse mean score and higher variance.

On most maps the QMDP and POMCP-Q controllers showed similar behavior, with the exception of C&R2, in which POMCP-Q performed very poorly. On this map POMCP-Q consistently misinterpreted the human's initial actions as chasing the blue robber when it was in fact chasing yellow or green, leading them to enter the western door and become stuck. A combination of factors produced this behavior. First, the decision-theoretic human model judged any actions other than passing as diagnostic for blue. This is because, in the optimal joint policies for red and yellow, the human is

expected to wait for the sidekick to drive the robbers closer before the human needs to move. Second, the model that POMCP-Q uses assumes a cooperative human that will reason correctly about the sidekick's predicament and attempt to drive the robber that they are actually chasing through the western door to it. However, the simulated human is in fact a myopic A* agent that doesn't consider how important it is to coordinate, which contributes to the POMCP-Q controller underestimating how bad committing to step through the door is. Finally, the sidekick's model suggested that there was some chance that the human may change targets and chase blue, leading them to misjudge how much of a commitment stepping through the west door was. QMDP, by contrast, plans under the assumption that the human's target will never change, which leads to a more dire assessment of committing to chasing blue too early.

It is clear from these results that having an accurate human model has a large impact on POMCP's performance. In particular, the optimistic assumption that the human will be cooperative and coordinate well, when in fact they behave myopically, can lead to making bad decisions that are hard to recover from. This highlights the need for careful modeling of human behavior for good performance in domains with actions that can have actions whose effects are not easily reversed and require close coordination with the human. Attempting to learn the true human model over time could also help to mitigate these problems, but setting up this learning task is a challenge in itself.

For a domain like Cops and Robbers, one possibility would be to generate two sets of types and type-policies to include in the human model, one myopic and one cooperative. The tradeoff is that this would increase the dimensionality of the state-space, potentially making particle filtering belief updates less accurate.

Table 4.1 shows the sample mean and standard deviation of the wall clock time taken for each of the POMCP controllers to select an action, averaged across all maps

with 100 runs each. POMCP-Q performed substantially worse than the other two, with decision times that are unacceptable for play with a real human. We suspect that this was the result of a bad interaction between the Manhattan distance minimization rollout bias and the actions of the decision-theoretic model causing POMCP to find it hard to reach a terminal state before the step limit, particularly in the early stages of the game. A large part of the variance in mean decision time was due to POMCP ending simulated histories early when they reached a terminal state. This led to decision times being lower when the sidekick and human were close to catching a robber. We will return to the interaction between the number of simulated histories, decision time, and action quality in the Section 4.5.6.4.

Controller	Mean decision time
POMCP	1.409s (0.46s)
POMCP-NC	1.875s (0.75s)
POMCP-Q	8.924s (1.11s)

Table 4.1: Mean decision times in seconds for POMCP to produce an action on C&R1 through 5 averaged over 100 trials for each map. Sample standard deviations are in parentheses.

4.5.6.2 Collaborative Ghostbuster

In contrast to Cops and Robbers, POMCP and QMDP performed at similar level in Collaborative Ghostbuster, despite the differences in their human models. As Figure 4-4 shows, the mean number of steps was comparable, particularly given the standard error shown in the error bars. Qualitatively, the strategies pursued by both planners were similar.

A combination of factors led to this result. The lack of elements in the domain that punish overcommitment to pursuing the wrong goal, such as the one-way door traps in Cops and Robbers, made undoing a mistake such as moving to herd the

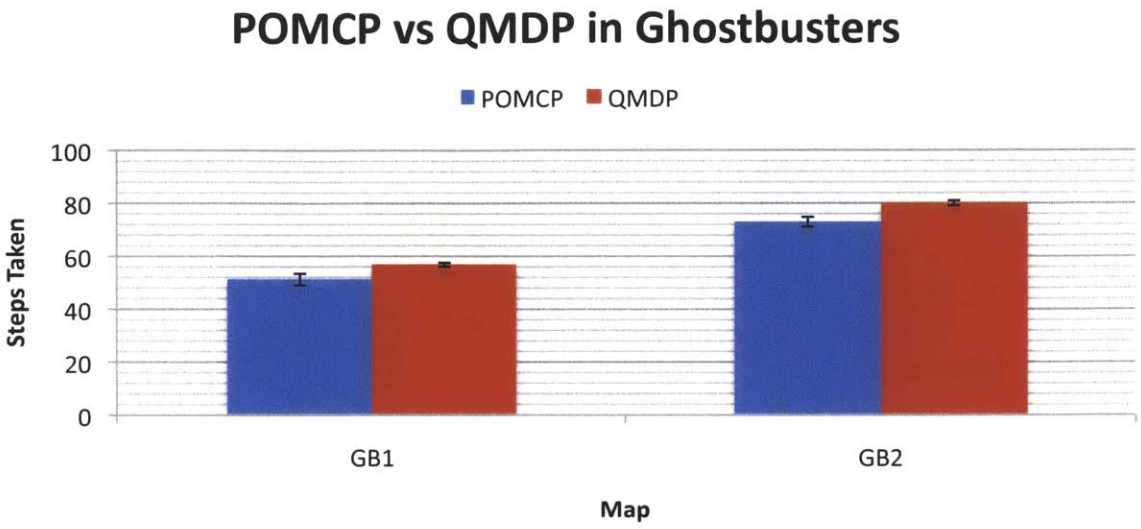


Figure 4-4: Mean steps taken by each controller to zap all ghosts in GB1 and 2.

wrong ghost relatively cheap. This could also be seen in C&R1, which had no doors. In contrast to Cops and Robbers, for most cases in Collaborative Ghostbuster the actions of the simulated human and the actions of QMDP's decision-theoretic model matched for each target ghost, which helped QMDP with both action selection and state estimation.

Table 4.2 shows the mean decision time for POMCP to select an action. There is a dramatic difference between the two maps, which likely stems from the fact that in GB1 the human could zap the ghost to its north on its first step, making the subsequent searching much easier since there were only two possible active human types from that point onwards and the length of the full plan required to catch the remaining ghosts was nearly one third shorter. We did not test the impact of reducing the number of simulated histories on action selection time and action quality, but the results reported for Cops and Robbers in Section 4.5.6.4 are likely applicable for this domain too.

Controller	Mean decision time	
GB1	5.99	(0.29)
GB2	13.28	(0.15)

Table 4.2: Mean decision times in seconds for POMCP to produce an action on GB1 and GB2 averaged over 100 trials for each map. Sample standard deviations are in parentheses.

4.5.6.3 Hidden Gold

Results for the Hidden Gold maps are shown in Figure 4-5. In HG1 through 3 the POMCP player and the heuristic player were comparable in their performance, but in HG4 the POMCP player significantly outperformed the heuristic player. On this map the sidekick consistently chose to go north and check both Xs, which is the most efficient search path. By contrast, the heuristic controller picked an X at random, and even if it picked one of the top Xs, had only a 0.5 chance of picking the other top X if it investigated the first and found that the gold was not there, leading to more backtracking in expectation.

By contrast, the heuristic planner matched up well with the POMCP controller on the other maps, largely due to the layout of the Xs. On HG1 the behaviors of the two controllers were largely indistinguishable. In HG2 and 3 the layout of the Xs meant that if either the player or the sidekick by chance happened to choose to search at either of the northeast Xs in HG2 or at the southeast X in HG3, they also incidentally searched at another X on the way there. In both HG2 and HG3 POMCP's strategy was to search in locations closest to it first, which was only marginally more efficient than the heuristic sidekick.

Table 4.3 gives the mean decision times for each of the maps. Again, early terminations in searches lead to a high variance in search times, resulting in faster searches once the gold has been found or narrowed down to one location. The lack of a heuristic for rollouts likely contributed to the longer decision times than in the other domains.

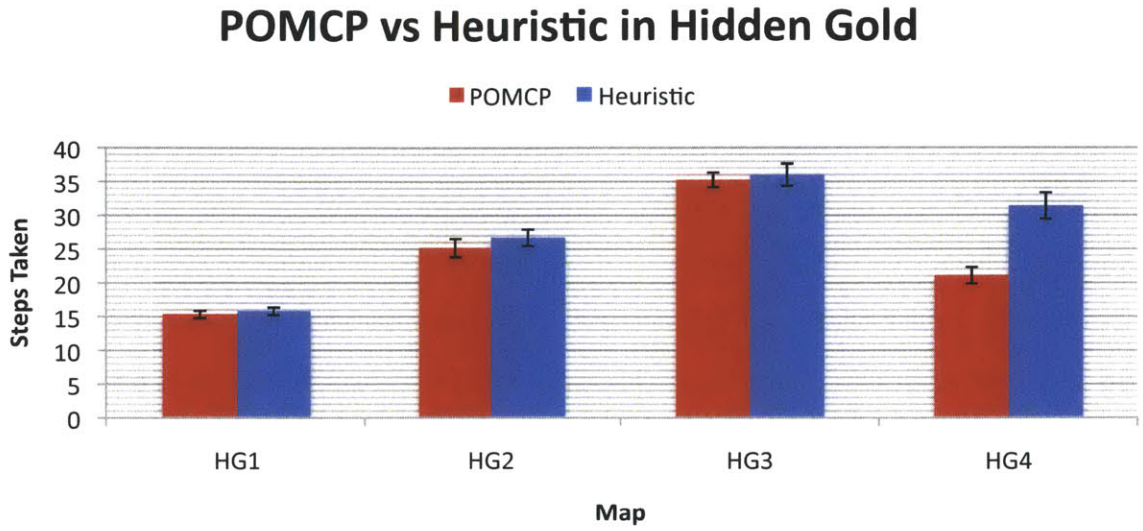


Figure 4-5: Mean steps taken by each controller to retrieve the gold in HG1 through 4.

Map	Mean decision time
HG1	7.57 (5.42)
HG2	9.97 (6.56)
HG3	13.24 (7.61)
HG4	11.61 (7.77)

Table 4.3: Mean decision time in seconds for POMCP to produce an action on HG1 through 4 averaged over 100 trials for each map. Sample standard deviations are in parentheses.

4.5.6.4 Larger Domains

We saw across the different domains that mean decision time could vary widely depending on the plan length required, even with a fixed number of simulated histories. For modern video game titles, more than a few milliseconds of decision time is considered unacceptable for real time responsiveness, and for these simple domains the mean decision time has been orders of magnitude higher than this. Ideally, we would like to reduce the mean decision time by using fewer simulations, while keeping action

quality high. We would also like to know how POMCP performs in domains larger than those tractable for the controllers in Chapter 3.

For testing the effect of varying the number of simulations, we used the two large Cops and Robbers maps, L1 and L2, which were intractable for our value iteration implementation due to the state space size. The search horizons for these maps were significantly longer that for C&R1-5, with the best-case shortest path to catch the furthest robber in each map being 44 and 72 steps respectively, making them very challenging. For each map we measured the mean number of steps taken and mean decision time for POMCP with the planner settings described in Section 4.5.2 and a varying number of simulated histories from 10 through 100,000, incrementing in multiples of 10. Communication actions were allowed. We used the standard myopic simulated human and the heuristic human model with the same settings as for the trials on C&R1-5. We averaged the results for each controller setting over 100 trials.

The mean number of steps taken to catch a robber are shown in Figure 4.5.6.4 and the mean decision times are given in Table 4.4. The mean decision time scaled roughly linearly with the number of simulated histories, which is as expected.

For L1, the mean number of steps taken decreased approximately logarithmically with the number of simulated histories before plateauing at around 70 steps. At 10,000 simulated histories the mean number of steps taken to catch the furthest robber was 104.6, substantially higher than the 44 steps minimum. By contrast, the number of steps taken to catch the closest robber was 22.4, relatively close to the 18 step minimum. This aligns with our intuition that POMCP should find it harder to construct plans that require searching deeply in the tree of action-observation histories.

In L2, the furthest robber was never caught in any trial. The overall performance of POMCP improved much more slowly and plateaued much faster, reflecting the longer search horizons required to find action sequences resulting in a catching a

robber. Even with 100,000 simulated histories, the closest robber was only caught in 72.7 steps on average, when the best-case shortest path was 46 steps. The next closest, at 56 steps in the best case, was only caught in one out of the 100 trials, suggesting that the effective search horizon fell somewhere between these two cases.

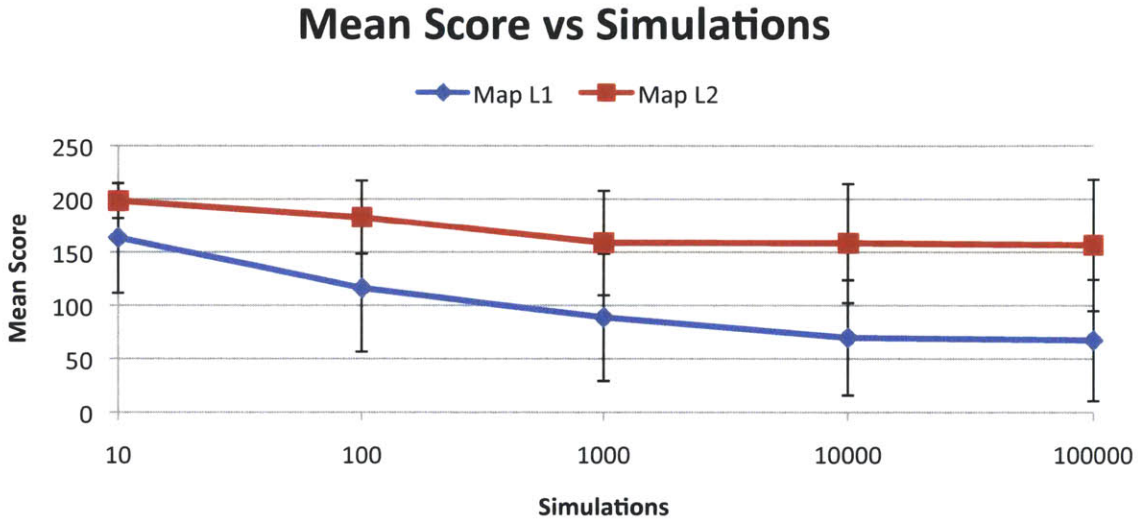


Figure 4-6: Score scaling results for maps L1 and L2. Mean score improves logarithmically with the number of simulations before plateauing.

Simulated histories	L1		L2	
10	0.002	(0)	0.01	(0)
100	0.03	(0.01)	0.05	(0.01)
1000	0.2	(0.03)	0.37	(0.07)
10000	1.79	(0.21)	3.23	(0.58)
100000	16.38	(2.55)	31.05	(5.38)

Table 4.4: Mean decision time in seconds for POMCP to produce an action on two large maps averaged over 100 trials for each map. Wall clock time scales linearly with the number of simulated histories. Sample standard deviation is given in parentheses.

4.5.6.5 Ensemble PO-UCT

One hope for reducing mean decision time and improving action quality is to use ensemble POMCP. We should bear in mind that in the worst case we will be searching in a tree that branches exponentially in the search horizon, and there is only so much that linearly scaling the search through adding additional cores can ultimately help. Recall from Section 4.4 here are two possibilities for improving performance. The first is to gain parallel time advantage, which occurs if an ensemble of searches split across multiple cores with k simulated histories per search outperforms a single-core search with k simulated histories. If parallel time advantage exists then it means that a controller can be improved by running using the same amount of time to perform work across multiple cores and achieve higher quality actions. The second is to gain single-core time advantage, which occurs if an ensemble of size n with k simulated histories per tree equals or outperforms a single-core search with $n \cdot k$ simulated histories. Single-core time advantage is particularly beneficial, since it implies that the mean decision time can be reduced by performing multiple faster searches in parallel, rather than one expensive search, without a loss of quality.

In evaluating POMCP with ensemble PO-UCT on Cops and Robbers we used the same controller, model, and simulated human settings as for the scalability tests. We focused on L1, which was the largest map in which each of robbers could be caught by a controller running a reasonable number of simulated histories. We ran trials for a range of trajectory counts and ensemble sizes, with 100 trials per combination, recording the mean number of steps taken to capture a robber. Tables 4.5 through 4.7 break down the results by the human's target.

The number of simulated histories increases by powers of two down the columns and the ensemble size increases similarly across the rows. A parallel time advantage corresponds to a decreasing values across a row.

Along the diagonals, the total number of simulated histories for a trial are equal. For instance in Table 4.5 the two shaded cells both represent trials with 2^{12} total simulated histories, in one case on a single core, and in the other split across two cores. A single-core time advantage corresponds to decreasing values from top to bottom along a diagonal. The shaded cells in Table 4.5 are an example of this.

Table 4.5 gives the results for catching the red robber, which has the shortest minimum number of steps required. The results show that finding a close to optimal strategy for this is relatively easy, with mean scores around 23 and low variance being the norm once the total number of simulated histories reaches 2^{13}. There are two exceptions to this in the first and third rows, which may be due to noise in the human model allowing the robber to escape in a trial. Only the highlighted cells demonstrate a significant single-core time advantage. This is most likely because with only a relatively small number of simulated histories, the optimal path consistently falls within the search tree, so there is no advantage to be gained from performing the search on multiple cores.

Simulated histories per tree	Ensemble Size		
	1	2	4
2^{11}	34.79 (2.99)	28.71 (0.62)	34.5 (3.41)
2^{12}	32.31 (2.69)	23.94 (0.39)	24.53 (0.46)
2^{13}	25.86 (0.51)	24.12 (0.37)	23.26 (0.43)
2^{14}	25.93 (0.79)	22.72 (0.32)	22.23 (0.45)
2^{15}	23.72 (0.57)	22.94 (0.35)	23.5 (0.41)
2^{16}	24.29 (0.57)	28.54 (3.34)	22.35 (0.37)

Table 4.5: Scores for range of ensemble settings on L1 with Red as the target. Standard error for the 100 trials is given in parentheses.

The results for catching the blue robber, which required the second shortest plan, are given in Table 4.6. Although the results are noisy, there is evidence to suggest the existence of both parallel time advantage and single-core time advantage. Increasing

the number of cores to 2 improved performance in 4 out of 6 cases and increasing it to 4 improved it in 5 out of 6 cases. For configurations with 2^{14} or more total simulated histories, a parallel time advantage can be seen in 3 out of the 4 diagonals in the table. Our interpretation is that there is a minimum number of simulated histories per tree required to regularly find a good solution in expectation, and that single-core time advantage is most apparent once this threshold is reached.

Simulated histories per tree	Ensemble Size		
	1	2	4
2^{11}	103.24 (5.98)	102.42 (6.06)	87.03 (5.46)
2^{12}	81.27 (4.91)	100.09 (5.97)	72.5 (4.54)
2^{13}	83.4 (5.6)	78.06 (4.98)	66.45 (3.83)
2^{14}	86.48 (5.45)	67.56 (4.29)	91.00 (6.56)
2^{15}	83.11 (5.81)	87.56 (5.94)	62.46 (3.45)
2^{16}	93.51 (6.16)	66.00 (4.36)	64.16 (3.95)

Table 4.6: Scores for range of ensemble settings on L1 with Blue as the target. Standard error for the 100 trials is given in parentheses.

Table 4.7 gives the results for catching the yellow robber, which requires the longest plan of the three. With both 2 and 4 cores there was a parallel time advantage in only 3 out of 6 cases and only one diagonal demonstrates a single-core time advantage. Since even with 100,000 simulated histories, almost twice as many as shown here, the mean score achieved by POMCP for catching yellow was 102.59, this suggests that catching yellow sooner consistently would require substantially more simulated histories for POMCP to consistently choose higher quality actions. Our interpretations is that these results are largely statistical noise, and that more total simulated histories would be required for either parallel or single-core time advantage to be apparent.

Simulated histories per tree	Ensemble Size		
	1	2	4
2^{11}	99.68 (4.45)	100.00 (4.58)	127.71 (6.23)
2^{12}	112.54 (5.72)	106.28 (5.82)	97.94 (5.44)
2^{13}	118.21 (5.96)	93.5 (5.48)	91.55 (5.22)
2^{14}	86.5 (4.66)	108.77 (6.37)	94.53 (5.57)
2^{15}	89.97 (5.31)	105.28 (5.92)	73.26 (3.64)
2^{16}	100.59 (6.22)	96.57 (5.81)	94.06 (5.33)

Table 4.7: Scores for range of ensemble settings on L1 with Yellow as the target. Standard error for the 100 trials is given in parentheses..

4.6 Conclusions

This chapter demonstrated an approach to the AI sidekick problem that extended the capabilities of the sidekick beyond the constraints of the state-space planning based approaches of the previous chapter. POMCP naturally accounts for the effects of information gathering and communication actions on its belief state and also can be applied in domains with partial observability that extends beyond hidden human intentions. However, in making the move to more challenging partially observable domains we were forced to abandon our decision-theoretic approach to building human models. As the results in this chapter showed, the fidelity of a human model can have a strong impact on the quality of actions, particularly when there are irreversible detrimental actions in the domain. This places a heavy modeling burden on a developer who is interested in using POMCP to address the AI sidekick problem in a real game.

In terms of scaling, for Cops and Robbers, and similar stochastic shortest path problems, there appear to be three cases that this chapter demonstrates, which can be characterized by the length of the plan required to reach a goal state.

The first case occurs when goal state is well outside of the search horizon covered by the search tree, meaning it is very infrequently reached by rollouts. In this case

increasing the number of simulated histories in POMCP may give a worse than logarithmic improvement in performance in the number of simulated histories until the search tree becomes deep enough for the goal state to be consistently reached from its leaves. This seemed to be the case for the two furthest robbers in L2, where in each case the goal was so distant that the number of simulated histories required for the search tree to grow close to them reliably in expectation was higher than was reasonable for a responsive sidekick controller. This also seemed to be the case with the yellow robber in L1, where neither increasing the number of simulated histories, nor the number of cores used in ensemble PO-UCT, reliably produced a gain in action quality for the values we tested.

The second case is when goal states are at a search depth that is not consistently reached by the search tree, but is potentially reachable by rollouts from the leaves. In this case increasing the number of simulated histories can help, trading time for action quality by reducing the variance in POMCP's V^* estimates. Another option is to keep the number of simulated histories constant and increase the number of cores, gaining parallel time advantage through ensemble PO-UCT. There may also be a range of total simulated histories, as demonstrated in the case of the blue robber in L1, for which using an ensemble of size n with k simulated histories can give as good or better performance as a single search with $n \cdot k$ simulated histories, yielding single-core time advantage.

The final case is when the goal reliably falls within the range of POMCP's search, possibly within the search tree itself, as with the red robber in L1. In this case adding further simulated histories or cores gives marginal or no benefit.

Chapter 5

Conclusion

5.1 Contributions

This thesis brings together two threads of research in game AI, user modeling and automatic planning, and addresses them within a decision-theoretic planning framework. The decision-theoretic approach to building sidekick controllers for games was recently pioneered by Nguyen et al. in their CAPIR framework [45]. CAPIR itself built upon the decision theoretic model of assistance developed by Fern et al., whose core innovation was to build human models from the joint value function derived from solving a MDP representing a human and their AI sidekick perfectly collaborating, and then use approximate POMDP solution methods to plan actions [22]. Our work started from that idea, and developed it further.

Our first contribution was to give an explicit formulation of the AI sidekick problem as a POMDP, specifically with an embedded human model that has internal states that are potentially hidden, a process that cause those states to be updated based on observations, and a process that causes actions to be selected on the basis of those states. This general formulation covers the specific case that Fern et. al and CAPIR pioneered, but also generalizes the problem to allow for human models that

act in domains that are only partially observable by them as well as the sidekick, whereas the previous formulation assumed that the domain was fully observable by the human. Our formulation of the problem as a POMDP is a compromise between generality and tractability. We assume that we know the dynamics of the human model, which allows us to avoid the intractability of richer decision-theoretic models such as POSGs, but at the cost of having to define those models.

Formulating the problem in this way allowed us to consider extensions to the capabilities of an AI sidekick that went beyond CAPIR's. Our second contribution was to explore a variety of approximate solution techniques that first recreated the capabilities of CAPIR and then extended them to include actions that specifically aim to gather information about the hidden state of the world. This allowed us to incorporate communication and question asking into our sidekick agent's behavioral repertoire via a range of heuristic action selection methods that were based on state-space planning. We demonstrated this in the Cops and Robbers domain and showed that these techniques also perform on a par with the QMDP approach that CAPIR used in the Collaborative Ghostbuster domain.

From there we addressed two limitations of the the methods based on state-space planning. The first was a scalability limitation imposed by the need to exactly solve large MDPs. The second was a representational limitation that required that domains be fully observable, aside from the human's internal state. We introduced the use of POMCP, a POMDP solution technique introduced by Silver and Veness that scales well in large POMDPs, to address these problems jointly. This, allowed us to consider how to interact with a more general class domains and of human models, but also precluded automatically generating human models through joint MDP solving.

Our third contribution was to build controllers using POMCP as a planning method and demonstrate empirically that they could perform at least as well as the methods introduced earlier for small domains, and could perform reasonably well

on larger domains that were unapproachable by value iteration.

Finally, our fourth contribution was to extend POMCP itself by introducing ensemble methods for parallelization, which we demonstrated allowed us to choose higher quality actions in less time than standard POMCP in some cases. We also offered some suggestions as to what characterized the cases where this was possible on the basis or our empirical results.

5.2 Future work

There are three key directions that we have identified for extending the work in this thesis; user studies, human modeling, and scaling.

Although we have demonstrated the performance of our sidekick controller when paired with simulated humans and noted the failure cases in that setting, we have not conducted any extensive user studies. User studies would help to understand the relative merits of the different controllers from an HCI perspective. User studies could also help us to understand human expectations of sidekicks and motivate new directions for developing controllers. Additionally, data collection from the studies would give us a data set from which we could explore ways to learn human models.

As we saw from the results in Cops and Robbers on planners that used a decision-theoretic human model but were paired with a heuristic simulated human, having an incorrect human model can have a major negative impact on sidekick performance, particularly when actions are irreversible or could have serious negative consequences. Ideally we would like a sidekick to adapt to a specific human partner's play style. There are several possible approaches.

As alluded to, we may try learning models from historical data of humans playing. This approach has been explored by the BotPrize community for games domains. In 2012 the bot UT^2 used recorded data from humans playing Unreal Tournament

2004 to generate controllers for AI opponents that mimicked human behavior, and convinced judges that it was human in over 50% of cases [35]. Given the success of this approach as a generative model for human-like behavior, it may hold promise as general human modeling strategy.

Instead of learning from recorded data, we could try to learn the human model on the fly, by treating the game as a Bayes-adaptive POMDP, which would lets us set up a reinforcement learning problem to simultaneously learn the model and plan actions [53]. However, current state-of-the-art planners are not yet at a point where they would be able to scale even to the size of the smaller Cops and Robbers problems. Recent advances in approximate methods for fully observable Bayes-adaptive MDPs, however, may point the way to improvements in planners for partially observable domains [28].

An alternative approach, that faces similar tractability challenges, is to generalize the decision-theoretic human model that we constructed with state-space planning by introducing partial observability. In this set up, we would assume that the human expects that they are perfectly coordinating with the sidekick, but in this case with the added challenge of partial observability, which makes the problem a decentralized POMDP. We could then construct a human model using the Dec-POMDP value function in a similar manner to our state-space planning approach. Again, however, this would require us to solve a large Dec-POMDP. One possible approach could be to approximate the value function instead. A recent survey of the progress being made in this direction in the Dec-POMDP community was compiled by Oliehoek [47] and there has also been recent work on combining this approach with Bayes-adaptive methods [1].

As seen in the results for the larger Cops and Robbers maps, scaling to larger domains remains a challenge, even with heuristic human models. Commercial video games typically have domains with orders of magnitude larger state spaces and longer

planning horizons than the largest Cops and Robbers map for which our planner showed adequate performance. Directly applying a POMCP planner in a commercial video game may be possible, but would likely require the problem to be represented at a higher level of abstraction than low level states and actions.

Part of what makes large Cops and Robbers maps challenging for the planner is the navigation problem of getting through the maze, even when a fine level of coordination is not required. One strategy for scaling up the Cops and Robbers planner would be to offload some of the navigational concerns to a specialized controller and instead plan at a more abstract level, where actions move the sidekick to different locations in the map, executed by the navigation system, rather than in the cardinal directions. The planner could then potentially switch to a lower level of abstraction when the sidekick was close to the robber for finer grained reasoning about coordination.

We suspect that in general this kind of hierarchical decomposition of the planning problem into different levels of abstraction has the most potential for making POMCP applicable in real commercial games. Using abstract macro-actions for planning has been explored by other researchers and has been shown to be helpful in scaling Monte Carlo based POMDP solution methods to large domains [39]. This does however introduce the modeling problem of selecting appropriate macro actions and state abstractions. In the fully observable case, researchers have developed methods for automatically forming hierarchical MDPs and solving them efficiently [33], but the partially observable case remains largely unexplored.

Bibliography

[1] C. Amato and F. A. Oliehoek. Bayesian reinforcement learning for multiagent systems with state uncertainty. In *Multi-Agent Sequential Decision Making in Uncertain Domains*, 2013.

[2] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo. Monte Carlo value iteration for continuous state POMDPs. In *International Workshop on the Algorithmic Foundations of Robotics*, 2011.

[3] C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. In *Neural Information Processing Systems*, 2009.

[4] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Autonomous Agents and Multi-Agent Systems*, 2011.

[5] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[6] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4), 2002.

[7] D. P. Bertsekas. *Dynamic Programming and Optimal Controls Vols. 1 and 2.* Athena Scientific, 1995.

[8] Y. Bjornsson and H. Finnsson. Cadiaplayer: A simulation-based approach to general game playing. *IEEE Computational Intelligence and AI in Games*, 2009.

[9] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. In *IEEE Computational Intelligence and AI in Games*, 2012.

[10] M. Campbell, A. J. Hoane, and F. Hsu. Deep Blue. *Artificial Intelligence*, 134, 2002.

[11] A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, Brown University, 1998.

[12] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: Discrete Bayesian models for mobile robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[13] H. T. Cheng. *Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, University of British Columbia, 1988.

[14] Computer And Video Games. Lara Croft: Guardian of Light, Crystal Dynamics on Lara's new path. http://www.computerandvideogames.com/247348/interviews/lara-croft-guardian-of-light/, 2010. Accessed 3/1/2013.

[15] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*, 2006.

[16] Naughty Dog. *The Last of Us.* Sony Computer Entertainment, 2013. Playstation 3.

[17] P. Doshi. *Optimal sequential planning in partially observable multiagent settings.* PhD thesis, University of Illinois, 2005.

[18] P. Doshi and P. J. Gmytrasiewicz. Monte Carlo sampling methods for approximating interactive POMDPs. In *Journal of Artificial Intelligence Research*, 2009.

[19] Crystal Dynamics. *Lara Croft and the Guardian of Light.* Eidos Interactive, 2010. XBox 360.

[20] Massive Entertainment. *World in Conflict.* Ubisoft, 2007.

[21] A. Fern and P. Lewis. Ensemble Monte-Carlo planning: An empirical study. In *International Conference on Automated Planning and Scheduling*, 2011.

[22] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli. A decision theoretic model of assistance. *International Joint Conference on Artificial Intelligence*, 2007.

[23] A. Fern and P. Tadepalli. A computational decision theory for interactive assistants. *Neural Information Processing Systems*, 2010.

[24] Guerrilla Games. *Killzone 2.* Sony Computer Entertainment Europe, 2009.

[25] Irrational Games. *Bioshock Infinite.* 2K Games, 2013. XBox 360.

[26] S. Gelly and D. Silver. Achieving master level play in 9x9 Go. *Artificial Intelligence*, 2008.

[27] J. Goldsmith and M. Mundhenk. Competition adds complexity. *Neural Information Processing Systems*, 20, 2008.

[28] A. Guez, D. Silver, and P. Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Neural Information Processing Systems*, 2012.

[29] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. In *Journal of Artificial Intelligence Research*, 2000.

[30] R. Howard. *Dynamic Programming and Markov Processes.* MIT Press, 1960.

[31] Team Ico. *Ico.* Sony Computer Entertainment Japan, 2001. Playstation 2.

[32] D. Isla. Handling complexity in the Halo 2 AI. *Game Developer's Conference*, 2005.

[33] T. Lozano-Pérez J. L. Barry, L. P. Kaelbling. Hierarchical solution of large Markov decision processes. In *International Conference on Automated Planning and Scheduling*, 2010.

[34] L. Kaelbling, A. Cassandra, and James A. Kurin. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[35] I. V. Karpov, J. Schrum, and R. Miikkulainen. Believable bot navigation via playback of human traces. In Philip F. Hingston, editor, *Believable Bots*, pages 151–170. Springer, 2012.

[36] L. Kocsis and C. Szepsvari. Bandit based Monte-Carlo planning. In *Neural Information Processing Systems*, 2006.

[37] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Robotics: Science and Systems*, 2008.

[38] Singapore-MIT GAMBIT Game Lab. *Dearth.* MIT, 2009. PC.

[39] Z. W. Lim, D. Hsu, and L. Sun. Monte Carlo value iteration with macro actions. In *Association for the Advancement of Artificial Intelligence*, 2002.

[40] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, 1995.

[41] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving Markov decision problems. In *Uncertainty in Artificial Intelligence*, 1995.

[42] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable decision problems. In *Artificial Intelligence*, 1999.

[43] D. Mark. *Behavioral Mathematics for Game AI.* Cengage Learning, 2009.

[44] I. Millington and J. Funge. *Artificial Intelligence For Games.* CRC Press, 2009.

[45] T. H. D. Ngyuen, D. Hsu, W. S. Lee, T. Y. Leong, L. P. Kaelbling, T. Lozano-Perez, and A. H. Grant. CAPIR: Collaborative action planning with intention recognition. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.

[46] F. A. Oliehoek. *Value-based planning for teams of agents in partially observable environments.* PhD thesis, University of Amsterdam, 2010.

[47] F. A. Oliehoek and A. Visser. A decision-theoretic approach to collaboration: Principal description methods and efficient heuristic approximations. In R. Babuska, editor, *Interactive Collaborative Information Systems.* Springer, Berlin, 2010.

[48] J. Orkin. Symbolic representation of game world state: Towards real-time planning in games. In *AAAI Workshop on Challenges in Game AI*, 2004.

[49] C. H. Papadimitiou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3), 1987.

[50] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27, 2006.

[51] Monolith Productions. *F.E.A.R.* Vivendi Universal, 2005. PC.

[52] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley and Sons, 1994.

[53] S. Ross, B. Chaib-draa, and J. Pineau. Bayes-adaptive POMDPs. In *Neural Information Processing Systems*, 2008.

[54] S. Sanner. The ICAPS2011 IPPC website. http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/, 2011. Accessed 3/1/2013.

[55] J. Schaeffer, N. Burch, Y. Bjornsson, A. Kishimoto, M. Muller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *Science*, 317(14), 2007.

[56] B. Schwab. *AI Game Engine Programming.* Cengage Learning, 2008.

[57] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Neural Information Processing Systems*, 2010.

[58] E. J. Sondik. *The Optimal Control of Partially Observable Markov Decision Processes.* PhD thesis, Stanford University, 1971.

[59] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8, 2000.

[60] R. Straatman, T. Verweij, and A. Champandard. Killzone 2 multiplayer bots. In *Paris Game AI Conference*, 2009.

[61] Bungie Studios. *Halo 2.* Microsoft Games Studios, 2004.

[62] N. Vlassis. *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2007.

[63] M. Wooldridge. *An Introduction to Multi Agent Systems.* Wiley, 2002.