EXECUTION-TIME COMMUNICATION DECISIONS FOR COORDINATION OF MULTI-AGENT TEAMS

Maayan Roth

CMU-RI-TR-08-04

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

December 2007

Thesis Committee:
Reid Simmons, Co-Chair
Manuela Veloso, Co-Chair
Carlos Guestrin
Jeff Schneider
Milind Tambe, University of Southern California

ABSTRACT

MULTI-AGENT teams can be used to perform tasks that would be very difficult or impossible for single agents. Although such teams provide additional functionality and robustness over single-agent systems, they also present additional challenges, mainly due to the difficulty of coordinating multiple agents in the presence of uncertainty and partial observability. Agents in a multi-agent team must not only reason about uncertainty in their environment; they must also reason about the collective state and behaviors of the team.

Partially Observable Markov Decision Processes (POMDPs) have been used extensively to model and plan for single agents operating under uncertainty. These models enable decision-theoretic planning in situations where the agent does not have complete knowledge of its current world state. There has been recent interest in Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs), an extension of single-agent POMDPs that can be used to model and coordinate teams of agents. Unfortunately, the problem of finding optimal policies for Dec-POMDPs is known to be highly intractable. However, it is also known that the presence of free communication transforms a multi-agent Dec-POMDP into a more tractable single-agent POMDP. In this thesis, we use this transformation to generate "centralized" policies for multi-agent teams modeled by Dec-POMDPs. Then, we provide algorithms that allow agents to reason about communication at execution-time, in order to facilitate the decentralized execution of these centralized policies. Our approach trades off the need to do some computation at execution-time for the ability to generate policies more tractably at plan-time.

This thesis explores the question of how communication can be used effectively to enable the coordination of cooperative multi-agent teams making sequential decisions under uncertainty and partial observability. We identify two fundamental questions that must be answered when reasoning about communication: "When should agents communicate," and "What should agents communicate?" We present two basic approaches to enabling a team of distributed agents to **A**void **C**oordination **E**rrors. The first is an algorithm that **A**voids **C**oordination **E**rrors by reasoning over **P**ossible **J**oint **B**eliefs (ACE-PJB). We contribute ACE-PJB-COMM, which address the question of **when** agents should communicate. SELECTIVE ACE-PJB-COMM, which answers the question of **what** agents should communicate, is an algorithm that selects the most valuable subset of observations from an agent's observation history.

The second basic coordination approach presented in this thesis is an algorithm that **A**voids **C**oordination **E**rrors during execution of an **I**ndividual **F**actored **P**olicy (ACE-IFP). Factored policies provide a means for determining which state features agents should communicate, answering the questions of **when** and **what** agents should communicate. Additionally, we use factored policies to identify instances of context-specific independence, in which agents can choose actions without needing to consider the actions or observations of their teammates.

ACKNOWLEDGEMENTS

When acknowledging the long list of people who have made this thesis possible, I must first and foremost thank my advisors, Reid Simmons and Manuela Veloso. For the past six years, you have encouraged me, challenged me, supported me, and educated me. You provided me with wonderful, unique opportunities and nudged me back on track when I wandered off in unproductive directions. Thank you.

My thesis committee members, Carlos Guestrin, Jeff Schneider, and Milind Tambe, provided me with valuable guidance and advice. I particularly want to thank Milind and the Teamcore research group for hosting me at the University of Southern California during the summer of 2003. It was during that summer, and with your help, that I came up with the idea that eventually developed into this thesis. Thank you.

The Robotics Institute is an extremely supportive environment in which to be a graduate student, and this is due, in large part, to the tireless efforts of Suzanne Lyons Muth. The members of the CORAL and RASL labs have very generously helped me by proofreading my papers and attending endless practice talks. At various points in time, while working on this thesis, I found myself stuck. During those times, I bounced ideas off Drew Bagnell, Jeremy Kubica, and Frank Broz, and they helped me to get un-stuck. Nick Roy and Vandi Verma kindly provided the LaTeXtemplate used for this thesis document. Thank you.

Having now left Pittsburgh, I'm surprised to realize how much it came to feel like my home, in no small part because of all of the wonderful friends that I made there. It would be impossible to list everyone that I should thank. Alik, Damion, Jeremy, Jonathan, and I started graduate school together, and somehow, we've all made it through. Well, we're still waiting for Jonathan, but I have complete faith that he'll be joining us shortly. Frank and Rachel have helped to keep me healthy. Doug has helped to keep me sane. I can't even begin to list all ways in which the Wasserman family and Jason and Stefanie Small have helped me. Thank you.

I've been lucky enough to have had a lot of great teachers in my life. This thesis wouldn't be complete if I didn't mention those teachers who made the greatest impact on me: Mr. Dalsass and Mr. Perry, my high school chemistry and physics teachers, who taught me to love science, and Richard Pattis, who taught me to write code. You set the standard to which all other teachers should aspire. Thank you.

Finally, saving the best for last, I want to thank my family. Aba, Ema, Nitzan, and Yoel, I love you all very much. Aba, you inspired me to become the second Dr. Roth in the family, and the rest of you made sure that I followed through. Thank you. I couldn't have done this without you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

$\langle \alpha, \mathcal{S}, \mathcal{A}, \mathcal{P}, \Omega, \mathcal{O}, \mathcal{R} \rangle$	a Dec-POMDP
α	number of agents
$\mathcal{S}$	set of states
s^i	one state in $\mathcal{S}$
$\mathcal{A}_i$	set of individual actions for agent i
$\mathcal{A} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_\alpha \rangle$	set of joint actions
a^i	one joint action in $\mathcal{A}$
a^i_j	agent j's individual component of a^i
$\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0 \ldots 1]$	the transition function
Ω_i	set of individual observations for agent i
$\Omega = \langle \Omega_1, \ldots, \Omega_\alpha \rangle$	set of joint observations
ω^i	one joint observation in Ω
ω^i_j	agent j's individual component of ω^i
$\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \to [0 \ldots 1]$	the observation function
$\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	the reward function
$\mathcal{T}$	the time horizon
$\gamma \in [0 \ldots 1]$	the discount factor
$\mathcal{B}$	set of possible beliefs
b^i	one belief in $\mathcal{B}$
Σ_i	set of messages that can be sent by agent i
$\Sigma = \{\Sigma_i\}_{i \in \alpha}$	set of messages
$\mathcal{X}_i$	set of state features observed by agent i
$\mathcal{X} = \{\mathcal{X}_i\}_{i \in \alpha}$	set of state features

CHAPTER 1

Introduction

THIS thesis explores the question of how *communication* can be used effectively to enable the *coordination* of *cooperative* multi-agent teams making sequential decisions under *uncertainty* and *partial observability*. The terms "coordination" and "cooperative" have been defined in many different ways (Parker, 1998; Gustafson and Matson, 2003; Gerkey and Mataric, 2004; Kalra et al., 2005). In this thesis, we use the term with the following meaning:

- A *cooperative* team is one in which the goal of the team as a whole is to maximize a global reward signal. This reward is shared equally among all members of the team, and no team member is assumed to be self-interested or to have individual preferences.
- Broadly, teams *coordinate* when, rather than ignoring its teammates or attempting to model their actions as stochastic components of the environment, each agent considers the effect of the interaction between its local action and the actions of its teammates on the team performance. We are concerned specifically with the problem of *tightly-coupled coordination*, an issue that arises whenever there are instances in the problem space where the actions of one agent affect the optimal action choice for other team members.

In addition to domains such as robot soccer in which a cooperative team is inherent to the problem description (Fujita et al., 2000), multi-agent teams are useful for performing tasks that would be difficult or expensive for a single agent. For example, in domains such as planetary exploration (Goldberg et al., 2003), multi-agent teams provide additional robustness to failure, or may enable a task to be completed faster and more efficiently than would be possible with a single agent.

The behavior of multi-agent teams can be studied at different levels of abstraction, from the high-level problems of coalition-formation and task-allocation (Dias and Stentz,

2000; Vail and Veloso, 2003; Vig and Adams, 2006) down to the granularity of synchronizing the motions of cooperating robots, in domains such as box-pushing and multi-robot assembly (Matarić et al., 1995; Sellner et al., 2006). Some complex domains, like RoboCup Rescue (Nair et al., 2000), may require coordination at multiple levels of abstraction. In this thesis, we model the problem of multi-agent coordination as a problem of sequential decision-making under uncertainty and partial observability.

- *Uncertainty* refers to non-determinism in the outcomes of agents' actions.
- *Partial observability* implies that agents will not always be able to determine, with complete certainty, the current state of their environment.

In our models, agents choose and execute actions according to a pre-computed *policy*, and in the absence of communication, are not informed by any external source about which actions their teammates have chosen. As depicted in Figure 1.1, the collected individual actions of the agents form a joint action that stochastically affects the state of the world. Each agent then receives a local observation, which is a component of a joint observation emitted by the environment. The observations may depend on the joint action of the team at the previous timestep. Again, in the absence of communication, the agents do not know what their teammates have observed. Because the state is partially observable, the agents may not know the state at any given timestep. Instead, each agent must use the information available to it, in this case, its own local action and observation, to form a *belief* that estimates the state. Each agent then uses its belief to choose the next action.

The overall goal of the team is to maximize the cumulative expected reward that depends, at each timestep, on the current state and, possibly, the joint action. Because of our interest in tightly-coupled coordination, we seek to ensure that at every timestep, agents will avoid *coordination errors*. We define a coordination error, or instance of *mis-coordination*, as follows:

> Given a joint policy and some belief about the current state, each agent chooses an individual action. Together, these individual actions form a joint action. This joint action constitutes a coordination error if any agent, upon being informed of the fixed action choices of its teammates, would choose to change its own individual action so as to improve team performance under the current policy and given the agent's current information about the state.

Note that avoiding coordination errors is not equivalent to acting optimally. Rather, coordinated actions are fixed points in the joint policy in which the agents attempt to respond, given limited information, to the possible action choices of their teammates. In our work,

while we do not guarantee that we will enable agents to act optimally, we do guarantee that the agents will act in coordination as a team.

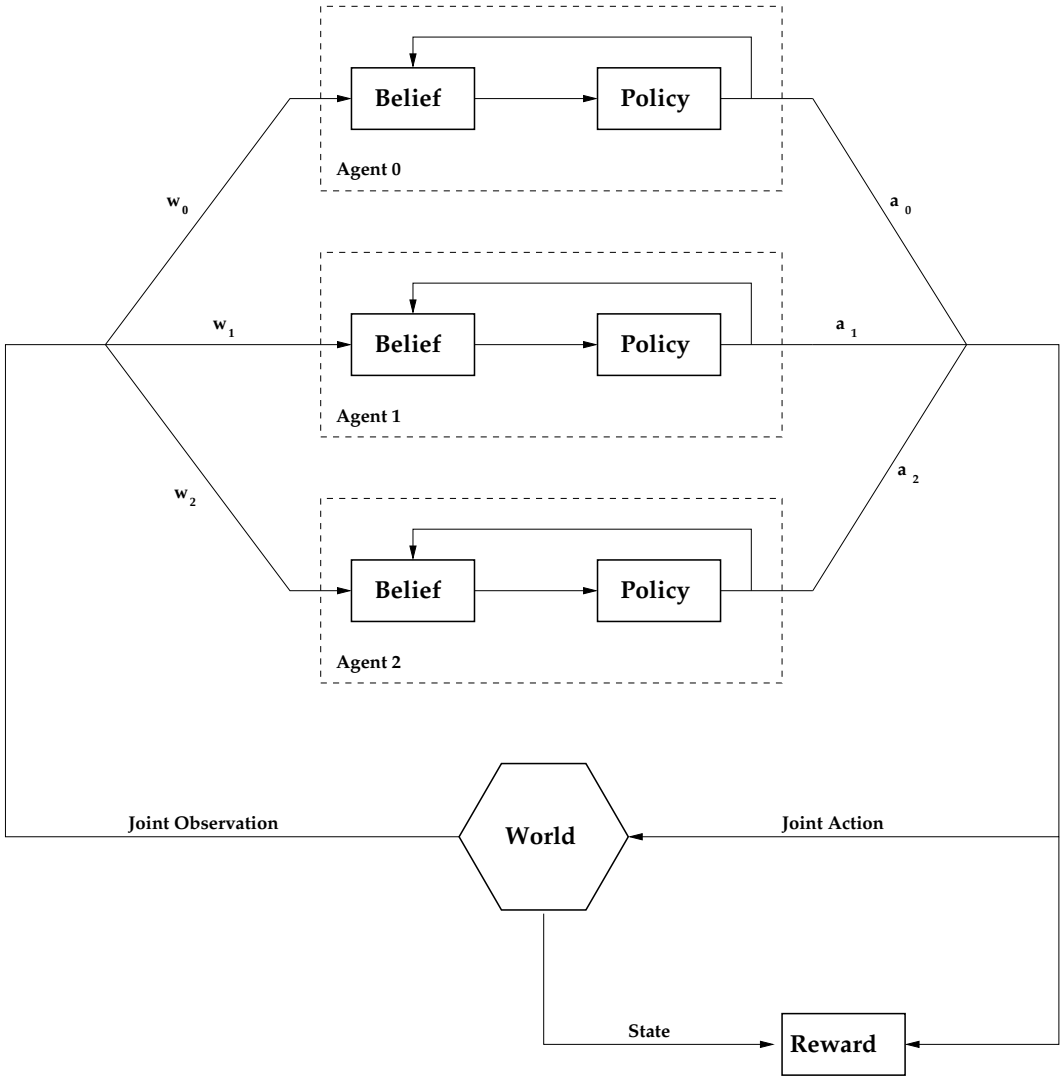


Figure 1.1. Multi-agent sequential decision-making in the absence of communication. The agents choose individual actions (a_0, a_1, a_2) according to their local policies, and receive individual observations (w_0, w_1, w_2). State transitions, observations, and the overall reward accumulated by the team depend on the joint action.

Figure 1.1 shows the information flow within a team of agents making sequential decisions under uncertainty, without communication. Each agent receives observations from its environment and takes actions that affect that environment. However, no information passes directly between the team members. Communication is an important tool that can be used to improve the performance of cooperative agents. When agents share their local observations with their teammates, they are able to form more accurate beliefs about the

state. If agents are informed about the actions chosen by their teammates, they are better able to avoid coordination errors. In this thesis, we address the effective use of communication for maintaining coordination within cooperative teams of distributed agents.

1.1. Motivation

The motivation for this thesis originates in our work on CMPack'02, a four-legged AIBO robot soccer team (Roth et al., 2003; Vail and Veloso, 2003). Sony AIBOs are equipped with a camera, located at the tip of the robot's nose, which provides the primary means by which the robots sense their environment. The robots are fully autonomous, performing all sensing and reasoning on-board. Since 1998, robot soccer in the Four-Legged League has been played by teams of increasingly sophisticated Sony AIBO robots (e.g, (Veloso et al., 1998; Uther et al., 2002)). Figure 1.2 shows the model of AIBO used in the CMPack'02 team. In 2002, a major change was introduced to the robots' capabilities: wireless communication. The potential benefits of this new capability for our research on teams of autonomous robots were enormous.

Figure 1.2. One of the Sony AIBO robots used in the 2002 RoboCup competition. The robot is equipped with the ability for wireless communication.

Before 2002, cooperative behaviors between teammate AIBOs consisted of predefined role assignments, such as *goalie* or *attacker*. Without communication and with limited vision processing, which had to be focused on finding the ball and localization markers, both attacker robots searched for the ball until they found it in their field of view, localized, and kicked the ball towards the goal. Coordination between teammate attacker robots was not explicitly designed, although it could sometimes emerge due to fortuitous spatial distribution of the robots on the field. Because robots were unable to share information with their teammates, each robot had to spend large portions of the game searching for the ball. If two teammates saw the ball simultaneously, they would often get in each other's

way, although they could attempt to use local information to avoid each other (Lenser and Veloso, 2003).

Because the robots could not communicate with their teammates, more complex cooperative behaviors, such as dynamic role assignment and strategic positioning were virtually impossible. Wireless communication in the new 2002 robots provided robot soccer teams with the potential ability to share information about their own positions and the positions of relevant objects not necessarily in a single robot's own field of view, as well as to explicitly coordinate their behavior in terms of dynamic role assignment and switching.

The first approach that we used to try to integrate wireless communication into the CMPack'02 team was simply to direct the robots to communicate their entire local state models. The AIBOs receive and process images at 20 frames per second, and build a world model consisting of their own location (position and heading), the location of the ball, and the positions of their three teammates (Lenser and Veloso, 2000). Because each image that a robot processes may contain only a few (or none) of these relevant state features, the world model also includes a time stamp associated with each element, indicating the last time that it was observed. The most basic way for the AIBOs to share information was for each robot to broadcast its complete world model to its teammates and build a new world model by choosing the elements with the most recent time stamps from each received world model. In this way, it was theorized that the team as a whole could share the most up-to-date information observed by each individual robot.

This approach, however, proved to be ineffective. Communication latency was high. On average, messages took 0.5 seconds to be received by all teammate robots, and in some cases, latency was observed to be as high as 5 seconds. Without synchronized clocks, this latency made it difficult for the robots to determine which observations were, in fact, the most recent. The AIBOs also had limited computational power onboard. Receiving messages broadcast by teammates at 20Hz overloaded the AIBO operating system's message buffers. After a few minutes of attempting to broadcast their world models as they were updated, the robots' motion controllers were affected, causing the robots to slow down and occasionally crash.

We were therefore challenged to research a better communication policy. Given the high latency and the practical need to reduce the broadcasting rate, we devised a strategy that separated each robot's beliefs into an *individual* and a *shared world model*. We further defined a set of rules by which robots updated and used these world models with communicated information (Roth et al., 2003). Robots broadcasted their positions to their teammates

at a rate of 2 Hz, and communicated the position of the ball only if they had actually observed it since the last instance of communication. In addition, robots always trusted their local observations before relying on communicated information. This strategy proved to be quite successful, assisting the CMPack'02 team in becoming the 2002 RoboCup Four-Legged League world champions by enabling novel and effective team coordination (Vail and Veloso, 2003). Our experience working on the problem of communication for the CMPack'02 robot soccer team prompted our interest in the more general problem of controlling the use of communication within robot teams.

In 2002, there were few other teams, besides those involved in robot soccer, of real robots acting in real time. We were very fortunate to have had an opportunity to directly observe *the challenges of communication in teams of robots*. The difficulty of effectively incorporating communication into a team of robots with inevitable computational and sensing limitations served as a motivation for this thesis, although the algorithms detailed in this work are currently too computationally demanding to be applied to real-world problems. Concretely, our experience with RoboCup showed us that, although communication is a potentially valuable resource for robot and agent teams, "broadcast all local state, all the time" is not, in general, a successful communication strategy. The work detailed in this thesis therefore identifies and addresses two fundamental questions that must be answered when developing communication policies for a cooperative team:

- **When** is communication necessary?
- **What** should be communicated?

1.2. Approach

In this thesis, we address the problem of using communication to coordinate the behavior of teams of cooperative agents. We are interested in domains that require tightly-coupled coordination, in which agents need not only to reason individually about uncertainty in their environment, they must also reason about the possible local states and actions of their teammates. To this end, we frame our work within the context of the Decentralized Partially Observable Markov Decision Problem (Dec-POMDP) and Decentralized Markov Decision Problem (Dec-MDP) models, which use decision theory to address the problem of sequential decision making. By representing the effects of combinations of individual agent actions at each timestep on the team reward, Dec-POMDPs and Dec-MDPs accurately represent both the cooperative and tightly-coordinated nature of the domains with which this thesis is concerned.

Unfortunately, for realistically sized models, Dec-MDPs and Dec-POMDPs are too hard to solve optimally. Decision-theoretic models are concerned with the effect, no matter how small, of any single action choice on the overall expected reward. Developing an optimal policy requires one to reason over all possible action choices in all possible states that an agent could encounter. Because agents are distributed and do not know the observations received by their teammates or the actions that their teammates selected, reasoning about the possible states that can be encountered requires agents to reason about all of the possible experiences of their teammates.

Communication provides a valuable tool both for improving the performance of multi-agent teams and, in some cases, for improving the tractability of team coordination. Free and instantaneous communication enables perfect inter-agent coordination by transforming a multi-agent problem into a large, centralized (equivalently, single-agent) problem. Unfortunately, in most domains communication is a limited resource and therefore cannot be treated as if it were free. Limited communication can still be used to improve team performance by allowing agents to share information and thereby generate a more informed estimate of the global state and the local states of each teammate. However, the need to reason about communication decisions as part of the policy-generation process, in order to use limited communication resources effectively or to minimize the cost incurred by communication, means that in general, the presence of communication does not make policy-generation more tractable.

Instead of attempting to make optimal communication decisions at plan-time, this thesis investigates the use of *execution-time communication heuristics*. The overall approach is as follows: At plan-time, a centralized policy is generated for the team as if the agents had free communication available to them. However, because the agents are decentralized at execution-time and communication is not free, the challenge is to enable the agents to execute a centralized policy while avoiding coordination errors. The algorithms detailed in this thesis show how agents can successfully avoid mis-coordination while using communication to improve team performance.

> **Hypothesis -** Reasoning about communication decisions at execution-time provides a more tractable means for coordinating teams of agents under partial observability than including all communication decisions in a comprehensive policy that is computed off-line, and can be done without sacrificing too much in terms of team performance.

The trade-off involved in making communication decisions at execution-time is that agents must reason about the value of communication in the midst of execution, rather than simply following a pre-computed policy. We believe that this trade-off is justified for several reasons: First, doing all computation, for both communication and domain-level actions, at plan-time is intractable for teams acting in complex domains. By generating a centralized policy, we reduce the computation that needs to be done at plan-time to a less computationally complex problem. Secondly, the amount of reasoning involved in making communication decisions at execution-time is less than what would be required to generate a universal policy for communication at plan-time. The problem of generating a communication policy offline requires one to reason about all possible messages that an agent could construct and choose to communicate. When reasoning about communication at execution-time, the agent must consider only the observation history that it has actually observed; there is never a need for the agent to contemplate communicating other, hypothetical, observation histories.

1.3. Issues in Communication

There are several issues that must be addressed when developing communication heuristics. The two most basic issues, on which we focus in this thesis work, are **when** and **what** a team of agents should communicate. Additionally, in teams of more than two agents, there is a question of **with whom** each agent should communicate. Finally, when designing a communication strategy, one must determine whether agents will decide to **tell** what they know to their teammates unprompted, or **ask** their teammates for information when they determine that it is needed.

1.3.1. When is communication necessary?

As agents operate independently with partial observability of the world, their beliefs about the state of the world diverge from those of their teammates. It is clear that if, at a particular timestep, all of the agents on a team share the same synchronized belief about the state of the world, no communication is necessary. The challenge, then, is identifying situations in which the agents' beliefs have diverged sufficiently such that their collective performance suffers, either because the agents begin to make coordination errors or because, individually, the agents do not have sufficient information to choose actions that lead to high expected reward. By re-aligning the agents' beliefs, communication can improve team performance.

How can the need to communicate be quantified? Consider the following: Suppose that at one timestep (e.g. the first timestep, if the agents start with a synchronized belief about the state of the world), the collective state of the team is such that communication is not necessary. Each agent takes an action and receives an individual observation. There are two possible outcomes. Either:

1. communicating anything that has been observed by one or more of the agents over the course of execution will not change the team's behavior in the current timestep, in which case communication is not immediately necessary, or
2. at least one agent has accumulated information that, if communicated would change the subsequent behavior of the team.

An agent can determine whether communication will change the team's behavior, leading to a change in expected long-term reward. This change in expected reward is the myopic *value of the information* to be communicated. If the value of information is greater than the cost of communication, it indicates that communication is cost-effective in the current timestep.

There are two main challenges that must be addressed in order to answer the question of **when** to communicate. First, agents must be able to estimate the effects of communication on team behavior and determine if the joint action and expected team reward will change as a result of a hypothetical communication act. Second, if agents do not communicate at every timestep, there will be situations in which the teammates do not share the same belief about the state of the world. The agents must have a policy that enables them to effectively select individual actions and avoid mis-coordination when their local knowledge differs. Chapter 4 discusses these issues and introduces an algorithm, ACE-PJB-COMM, that enables an agent, at execution-time, to maintain coordination and choose **when** to communicate to its teammates.

1.3.2. What should be communicated?

In addition to deciding **when** communication is necessary, agents must also reason about **what** to communicate. The most restrictive form of communication is *synchronization*, in which, if at least one agent initiates an instance of communication, all of the teammates are forced to communicate all of their observations since the last instance of communication. Synchronization has been commonly used when designing communication paradigms for decision-theoretic cooperative teams, because it has the useful property of ensuring that each instance of communication returns all of the team agents to a shared belief over world states (Xuan et al., 2001; Nair et al., 2004). This thesis introduces a slightly

less restrictive communication paradigm (Roth et al., 2005), discussed in detail in Chapter 4, in which agents need not communicate simultaneously, but an agent who chooses to communicate broadcasts all of its observations that it has not previously communicated. Both synchronization and the paradigm utilized in Chapter 4 essentially answer the question of **what** to communicate by saying "everything," thus reducing the problem of communication solely to a question of **when** to communicate.

One approach to addressing the question of **what** to communicate is to allow agents to choose a subset of their observation histories to communicate to their teammates. There are many criteria by which these observations could be chosen. For example, the most "informative" observations, defined as those observations that cause the highest reduction in the entropy of a teammate's belief, could be communicated (Rosencrantz et al., 2003). By contrast, Chapter 5 discusses an algorithm, SELECTIVE ACE-PJB-COMM, that extends the ACE-PJB-COMM algorithm by selecting observations according to the value of their information, approximated by observing their effect on changing the team's actions (Roth et al., 2006).

Although choosing a subset of "most valuable" observations from a complete observation history does begin to address the question of **what** to communicate, it still restricts communication to complete observations. Consider the robot soccer domain discussed above. In that domain, it is most important for robots to know their own position and the position of the ball. Most of the other state elements that they observe, such as the position of other robots on the field, are frequently irrelevant, or so noisy as to be useless. In many cases, a complete answer to the question of **what** to communicate includes a list of those state features that are currently most valuable to team performance. Chapter 6 shows how a factored model of the state can be used to facilitate the communication of state features (Roth et al., 2007). This allows a more natural understanding of the question of **what** to communicate.

1.3.3. Additional Concerns Regarding Communication

In addition to answering the questions of **when** and **what** to communicate, a comprehensive communication policy should also address the issue of **with whom** agents should communicate, meaning with which of its teammates. Communication in the context of multi-agent systems can be classified as either *broadcast* or *peer-to-peer*. In broadcast communication, when an agent sends a message, that message is received by every member of the team, and the question of **with whom** an agent should communicate is irrelevant. In peer-to-peer communication, however, agents direct messages to particular teammates,

and only those teammates receive the message. Therefore, when an agent chooses to communicate, it must also determine **with whom** it will do so. The algorithms discussed in Chapters 4 and 5 assume the presence of broadcast communication. Chapter 6 shows how factored representations can be used to indicate not only **what** state feature to communicate, but also **which agent** should communicate that feature, making it applicable to peer-to-peer communication.

Finally, there remains the question of how communication is initiated, and in which direction information flows between agents. We consider three communication types (Xuan et al., 2001):

- *tell*, in which one agent determines that its local information may be useful to its teammates and chooses to broadcast its own observations,
- *query*, in which an agent asks one or more of its teammates for a particular piece of information that it believes will be useful,
- and *sync*, in which agents all agents simultaneously communicate their local information to all of their teammates.

These different communication paradigms are useful in different situations. As discussed above, *sync* may be preferable in some domains because it guarantees that agents will reach a unified belief about the world after each instance of communication. On the other hand, *sync* may also be unnecessarily wasteful of communication resources, in that it may force agents to communicate information that will not be useful to their teammates. Chapters 4 and 5 deal with *tell* communication. Because the algorithms detailed in those chapters involve agents modeling the possible states of the team as a whole, it is easier for an agent executing those algorithms to calculate whether its own local observations will be useful to the team than to wonder about the possible observations that its teammates could have observed and ask for them. Chapter 6, however, deals with a situation in which agents do not model the possible beliefs of their teammates. Instead, agents employ *query* communication, and ask their teammates for information whenever they believe that something a teammate could have observed will affect their individual action choice.

1.4. Thesis Contributions

This thesis makes contributions in the area of using communication to improve the coordination of multi-agent teams.

- The approach presented in this work, namely generating a centralized policy for the team at plan-time and making communication decisions at execution-time to facilitate the decentralized execution of this centralized policy, is a novel solution

to the problem of generating heuristic policies for the highly intractable Dec-MDP and Dec-POMDP models. Our focus in this thesis is on developing algorithms that guarantee that agents will **A**void **C**oordination **E**rrors (ACE) during execution.

- This thesis introduces two approaches for enabling the decentralized execution of a centralized policy:
  - Agents **A**void **C**oordination **E**rrors by reasoning about and selecting actions over the *distribution of* **P***ossible* **J***oint* **B***eliefs* (ACE-PJB) of the team as a whole.
  - Agents **A**void **C**oordination **E**rrors by using **I***ndividual* **F***actored* **P***olicies* (ACE-IFP) to identify those portions of the state space in which they can act independently.
- Three algorithms detailed in this thesis address the questions of **when**, **what**, and in the case of our work on factored policies, **to whom** a team of agents should communicate.

The approaches presented in this thesis are verified experimentally in a number of domains, chosen for their illustrative properties. The ACE-PJB-COMM algorithm in Chapter 4 is tested primarily in the Two-Agent Tiger domain (Nair et al., 2003), detailed in Appendix A.1, which has become a standard benchmark problem in the Dec-POMDP literature (Nair et al., 2004; Emery-Montemerlo et al., 2004; Seuken and Zilberstein, 2007). The Two-Agent Tiger problem is particularly useful as an explanatory aide, given its size of only 2 states, 3 individual actions, and 2 individual observations. Our results in the two-agent tiger domain indicate that using the ACE-PJB-COMM algorithm to make communication decisions enables a team to communicate, on average, 48.7% fewer observations and 82.3% fewer messages than a team executing with full communication. With respect to performance, however, ACE-PJB-COMM causes the agents to act more conservatively than they would with free communication, achieving 74.4% of the mean discounted reward. In addition to the tiger domain, we verified the performance of ACE-PJB-COMM in another common benchmark domain, the multi-access broadcast channel (MABC) domain (Hansen et al., 2004; Seuken and Zilberstein, 2007), described in Appendix A.2. We demonstrated that the ACE-PJB-COMM algorithm and the overall approach of reasoning over the distribution of possible joint beliefs is applicable to teams of more than two agents by constructing a three-agent tiger domain.

In Chapter 5, we explore the question of **what** to communicate by presenting an algorithm that selects for communication the most valuable subset of observations from a

complete observation history. While we performed experiments with the SELECTIVE ACE-PJB-COMM algorithm in the two-agent tiger domain, we needed to construct a domain in which different observations have different informative values. In the tiger domain, the two possible individual observations are symmetric with respect to the state and therefore convey the same amount of information when communicated. We therefore introduce a new domain, the Colorado/Wyoming Problem, described in Appendix A.3, in which agents observe 4 possible individual observations of varying qualities that help them to distinguish whether they are in Colorado or Wyoming. The problem is a variant of the common Meeting-Under-Uncertainty benchmark domain (Xuan et al., 2001; Goldman and Zilberstein, 2003; Bernstein et al., 2005), where agents are attempting to meet in a goal location. Our experimental results show that the agents are able to use the SELECTIVE ACE-PJB-COMM algorithm to reason effectively about which of their observations are the most valuable to the team performance, reducing communication by 46.3%, on average, over the baseline ACE-PJB-COMM algorithm, with no significant decrease in performance.

Chapter 6 is concerned with the use of factored representations of state, both to informing agents about **when** and **what** to communicate, and also to enable agents to act independently in some cases without considering the actions of their teammates. The ACE-IFP algorithm described in this chapter is currently applicable only to Dec-MDP domains. We are particularly interested in demonstrating our approach on domains in which the independence of agents from their teammates during portions of execution is intuitively obvious. We therefore performed experiments in a two-agent Meeting-Under-Uncertainty domain, described in Appendix A.4, in which agents must meet their teammates at a goal location before performing a simultaneous SIGNAL action. In this domain, teams executing individual factored policies required communication of, on average, 43.8% fewer state variables than teams executing with full communication, with no loss of expected reward. We also demonstrate the performance of ACE-IFP in a modification of the taxi domain (Dietterich, 1998), detailed in Appendix A.5, a domain that has received significant attention in the multi-agent reinforcement learning literature (Ghavamzadeh and Mahadevan, 2004). In this domain, agents achieved an average communication savings of 46.1% state features over a team with full communication, while accumulating the same mean discounted reward.

1.5. Thesis Outline

- Chapter 2 provides background information on the various decentralized sequential decision-theoretic models that form the framework for this thesis.

- Chapter 3 gives an overview of related work.
- Chapter 4 discusses the ACE-PJB algorithm that enables a team of agents to reason about the *possible joint beliefs* of the team, and introduces the ACE-PJB-COMM algorithm for making decisions about **when** to communicate.
- Chapter 5 presents the SELECTIVE ACE-PJB-COMM algorithm, that reasons about **what** observations a team should communicate.
- Chapter 6 discusses ACE-IFP algorithm, in which a factored representation of state is used to answer the question of **which features** a team should communicate and to enable agents to act independently in some situations, without considering the actions and observations of their teammates.
- Chapter 7 contains concluding remarks and a discussion of possible directions for future work.

CHAPTER 2

Decentralized Decision-Theoretic Models

MARKOV Decision Problems (MDPs) are the basic framework for modeling sequential decision theoretic planning under uncertainty (Howard, 1960). MDPs model *fully observable* domains, domains in which the acting agent has complete knowledge of the current world state at each timestep. When solving an MDP, the goal is to find a *policy* that maximizes the agent's expected reward by specifying which (potentially noisy) action the agent should take in every possible state. Partially Observable Markov Decision Problems (POMDPs) extend the MDP framework to address *partially observable domains*, those domains in which the agent is unable to directly observe the world state, but instead must estimate the state based on noisy or incomplete observations (Sondik, 1971).

This chapter discusses cooperative multi-agent extensions to the Markov Decision Problem framework and gives an overview of some tools used in this thesis. In multi-agent systems, the two classes of observability, full observability and partial observability, take on slightly different meanings. Section 2.1.1 describes the Decentralized Partially Observable Markov Decision Problem (Dec-POMDP), which models teams operating under partial observability, sometimes referred to as *collective partial observability*. Multi-agent teams may also exhibit a class of observability called *collective observability*, discussed in Section 2.1.2. Teams that operate under collective observability, modeled by Decentralized Markov Decision Problems (Dec-MDPs), are those in which, while each agent may not be able to determine the world state independently, the union of the local observations of all of the teammate agents uniquely identifies the world state. Section 2.1.3 discusses the fully observable Multi-agent Markov Decision Problems (MMDPs), in which each agent, without needing to share information with its teammates, is able to uniquely identify the state of the world.

Section 2.2 discusses the computational complexity planning, both centralized and decentralized, for these multi-agent models. The basic Dec-POMDP and Dec-MDP models do not admit explicit communication between agents. In Section 2.2.3, we address the impact of explicit inter-agent communication on the complexity of solving decentralized problems and situate this thesis within the spectrum of observability and communication availability of decentralized decision theoretic models. Section 2.3 discusses POMDP solution methods, particle filters, and factored representations, all of which served as tools for the work in this thesis.

2.1. Decentralized Models

The deliberative, as opposed to behavioral, approaches to the coordination of multi-agent teams tend to be classified as belonging to one of three main philosophies: "Belief-Desire-Intention" (BDI), market-based, and decision theoretic. The models that form the framework for this thesis, Decentralized Partially Observable Markov Decision Problems (Dec-POMDPs) and Decentralized Markov Decision Problems (Dec-MDPs), were chosen because they use decision theory to address the problem of sequential decision making. While BDI frameworks are useful for finding qualitative solutions to team coordination problems, they have difficulty providing a quantitative analysis of team performance (Bratman, 1987; Georgeff et al., 1999). The BDI framework is, therefore, poorly suited to our needs in this thesis, as we are interested in quantitatively analyzing the impact of communication on the performance of multi-agent teams. Market-based approaches are primarily useful for solving the problem of task allocation (Gerkey and Mataric, 2004; Dias et al., 2006), but are difficult to use for coordinating teams of agents at the level of individual actions. Dec-MDPs and Dec-POMDPs represent the effects of combinations of individual agent actions at each timestep on the team reward. In this sense, decision theoretic models accurately represent both the cooperative and tightly-coordinated nature of the domains with which this thesis is concerned.

2.1.1. Dec-POMDP: Collective Partial Observability

There are several equivalent multi-agent POMDP frameworks (e.g., Decentralized POMDP (Dec-POMDP) (Bernstein et al., 2002), Markov Team Decision Problem (MTDP) (Pynadath and Tambe, 2002), Partially Observable Identical Payoff Stochastic Game (POIPSG) (Peshkin et al., 2000)), all of which model cooperative multi-agent teams operating under partial observability. In such teams, the agents take individual actions and receive local observations, but accumulate a joint team reward. Partial observability means that

not only is each individual agent unable to identify the current world state, but that even if the agents were able to share their local observations, these pooled observations would be insufficient to determine the state of the world with certainty. Robot soccer is an example of one such domain. Even if the agents communicate all of their observations to their teammates, the observations still contain noise and do not identify, with certainty, the positions of all of the relevant objects in the robots' environment.

In this work, we describe multi-agent POMDPs using notation which defines a Dec-POMDP as a tuple $\langle \alpha, \mathcal{S}, \mathcal{A}, \mathcal{P}, \Omega, \mathcal{O}, \mathcal{R} \rangle$ (Bernstein et al., 2000). The components of the tuple are:

- α, the number of agents in the team,
- $\mathcal{S}$, the set of n possible world states, $s^1 \ldots s^n$,
- $\mathcal{A}$, the set of possible joint actions of the team, where each joint action a^i is composed of α individual actions, $\langle a^i_1 \ldots a^i_\alpha \rangle$,
- $\mathcal{P}$, the transition function ($\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0 \ldots 1]$), which depends on joint actions and gives the probability associated with starting in a state s^i and ending in a state s^j after the team has executed the joint action a^k,
- Ω, the set of possible joint observations, where each joint observation, ω^i, is comprised of α individual observations, $\langle \omega^i_1 \ldots \omega^i_\alpha \rangle$,
- $\mathcal{O}$, the observation function ($\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \rightarrow [0 \ldots 1]$), which gives the probability of the team observing a joint observation ω^i after taking joint action a^k and ending in state s^j,
- $\mathcal{R}$, the reward function ($\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$), which indicates the joint reward that is received when the team starts in a state s^i and executes the joint action a^k.

We denote the set of possible individual actions of an agent i as $\mathcal{A}_i$, and likewise, the set of possible individual observations as Ω_i. A Dec-POMDP may also have a specified finite time horizon, $\mathcal{T}$, while an infinite-horizon Dec-POMDP may have a discount factor, γ, that ranges between 0 and 1.

In this formulation, the observation function $\mathcal{O}$ depends only on the end state and the joint action of the team, and the reward function $\mathcal{R}$ depends only on the starting state and joint action. This does not limit the generality of the model, as domains in which observation probabilities depend on the start state or rewards depend on the end state, or only on the start state and not the joint action, can be transformed into models that fit this template through a straightforward expansion of the state space.

Although the observation function is written in terms of joint observations, it is important to note that each agent observes only its local component of the joint observation, and

does not know what its teammates have observed.[1] Likewise, without communication, each agent knows only its own individual action and not the actions taken by its teammates. Because the transition and observation functions depend on the joint action of the team and are not, in general, factorable into functions over individual actions, agents must know or reason about the possible actions of their teammates in order to correctly evaluate the likelihood of a particular state transition or observation.

2.1.2. Dec-MDP: Collective Observability

Decentralized Markov Decision Problems (Dec-MDPs) model teams of agents operating under *collective observability* (Bernstein et al., 2002). In a Dec-MDP, the joint observation, composed of the union of individual agent observations at a given timestep, uniquely identifies the current state of the world, even though the individual agents themselves may be unable to determine the state independently. The tuple components for a Dec-MDP are the same as those for Dec-POMDPs.

Because Dec-MDPs are collectively observable, there are restrictions on the structure of the observation function $\mathcal{O}$. Although not all Dec-MDPs take this form, the state in some Dec-MDPs may be factorable into local state components for each agent, such that for each $s^i \in \mathcal{S}$, $s^i = \langle s^i_1 \dots s^i_\alpha, s^i_{shared} \rangle$ where s^i_k is agent k's local component of state s^i and s^i_{shared} is some global state component that is shared by all agents. In such a domain, there is often a one-to-one mapping between agents' local states and observations:

$$\mathcal{O}(\langle s^i_k, s^i_{shared} \rangle, a, \omega^j_k) = \begin{cases} 1.0 & : \quad i = j \\ 0.0 & : \quad i \neq j \end{cases}$$

In this case, the agent deterministically observes its local state and the shared state. For each combination of an agent k's local state s^i_k and shared component s^i_{shared}, the observation function $\mathcal{O}$, independent of the joint action a, returns 1.0 if and only if the individual observation is the corresponding observation ω^i_k. However, this one-to-one correspondence between states and individual observations is not universally true for Dec-MDPs. It is only necessary that the state be deterministically identifiable by the joint observation composed of all individual observations[2]:

$$\mathcal{O}(s^i, a, \omega^j) = \begin{cases} 1.0 & : \quad i = j \\ 0.0 & : \quad i \neq j \end{cases}$$

[1] In this work, we make the common assumption that the individual observations of the agents are conditionally independent given the world state and the joint action. Therefore, the joint observation function can also be written as the cross-product of individual observation functions for each agent.

[2] There need not be a one-to-one mapping between states and joint observations. The mapping may be one-to-many, as long as there is no ambiguity regarding the state identified by each observation.

2.1.3. MMDP: Full Observability

Fully observable domains in which each agent, independent of its teammates, can identify the current world state at every timestep, are modeled by Multi-agent Markov Decision Problems (MMDP) (Boutilier, 1999). Because no observations are necessary to inform the agents about the state of the world, an MMDP model is simply the tuple $\langle \alpha, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where the model components have the same meanings as those defined above for a Dec-POMDP.

2.2. Issues in the Complexity of Planning for Decentralized Models

In this section, we discuss the computation complexity of generating optimal policies for Dec-POMDPs, Dec-MDPs, and MMDPs. Like the work discussed in this thesis, most research on planning for these decentralized models has addressed the problem of centralized planning and decentralized execution. In Section 2.2.2, however, we also discuss a problem that arises during decentralized planning for MMDPs, as it relates to our work in Chapter 6 on transforming centralized policies into decentralized policies. The presence of and limitations on communication in decentralized teams also impacts the complexity of optimal planning, discussed in Section 2.2.3.

2.2.1. Centralized Planning

The problem of generating optimal policies for Dec-MDPs and Dec-POMDPs, even using a centralized planner, is a challenging one. It has been shown that the decision problem for a Dec-POMDP with at least two agents ($\alpha \geq 2$),

> "For a given Dec-POMDP with finite time-horizon $\mathcal{T}$, there exists a policy which yields an expected reward of at least $\mathcal{K}$,"

is NEXP-complete (Bernstein et al., 2002). Furthermore, the problem of generating optimal policies for Dec-MDPs is also NEXP-complete for teams of two or more agents. Single-agent POMDPs, by contrast, are known to be PSPACE-complete (Papadimitriou and Tsitsiklis, 1987). Since NEXP is provably more complex than PSPACE, we know that Dec-POMDPs are fundamentally harder than single-agent POMDPs. Additionally, known upper bounds for complexity hold only for finite-horizon problems where $\mathcal{T} < |\mathcal{S}|$. Problems with longer or infinite time horizons are likely to be more intractable.

The complexity of Dec-POMDPs can be explained intuitively as follows. To plan an optimal policy, an agent modeled by a POMDP must reason about the possible sequences of observations that it could experience. The transition and observation functions for Dec-MDPs and Dec-POMDPs depend on the *joint action* of the team and cannot, generally, be

factored into functions over individual actions. Therefore, an optimal policy for an agent in a multi-agent team must take into account not only the agent's own possible observation histories, but the possible observations of its teammates and the possible actions that they may choose to execute. It is this double branching factor, over both possible observations and possible actions, that leads to the increased complexity of decentralized teams operating under partial observability.

By contrast, the problem of finding an optimal policy for a team modeled by a fully observable MMDP is P-complete, the same complexity as a single-agent MDP (Pynadath and Tambe, 2002; Papadimitriou and Tsitsiklis, 1987). The reasoning is as follows: An MMDP policy is a mapping from states to joint actions. Because agents can identify the complete world state, independently of their teammates, they can therefore also determine what action their teammates will execute. Therefore, agents do not need to reason about the possible actions their teammates could choose; there is only one possibility. In practice, however, MMDPs, even with few agents, grow very quickly. The possible joint actions of the team are formed by the cross-product of the possible actions of individual agents, thereby growing exponentially with α. Additionally, the state space may grow rapidly with the number of agents, as the amount of information needed to capture the complete state of the team increases.

2.2.2. Decentralized Planning

Thus far, we have discussed the challenges of generating policies using a centralized planner. An additional challenge arises if a decentralized approach is used to generate individual policies for the teammate agents: the challenge of avoiding coordination problems introduced by equilibrium selection. This occurs when there are multiple possible optimal joint policies. To ensure optimal behavior, all of the agents must select the same joint policy. Figure 2.1 shows an MMDP in which this problem can arise. The MMDP shown in the figure has two optimal policies. In both of them, agent 1 selects action a from state s^1. Then, in state s^2, to achieve the maximum reward, the agents must execute the same individual action, which can be either a or b. However, if the agents choose different actions, such as the joint action $\langle a, b \rangle$, the team receives a negative reward.

A centralized planner could easily resolve this potential coordination problem by breaking the tie arbitrarily, choosing either joint action and dictating coordinated individual policies to the two agents. However, any decentralized algorithm in which the individual agents compute their own policies separately must concern itself with ensuring that agents will execute complementary policies and avoid coordination errors. In Chapter 6,

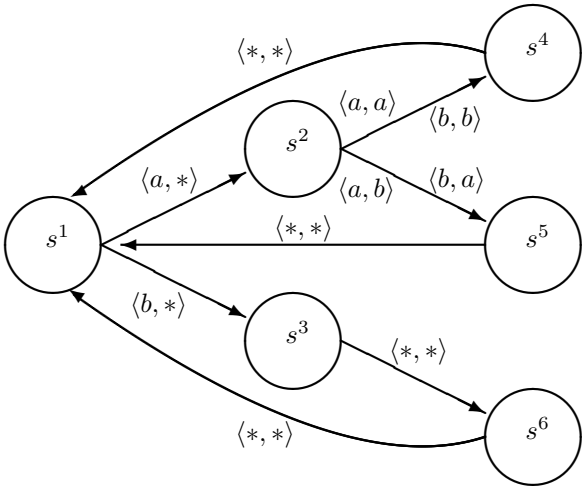


Figure 2.1. A two-agent MMDP with an equilibrium-selection problem (Boutilier, 1999).

we discuss how agents can transform centralized factored policies into individual factored policies. Although we calculate the centralized policies using a centralized planner, the process of transformation has the potential to introduce equilibrium-selection coordination errors. This thesis contributes a means by which it can be ensured that coordination errors are avoided.

2.2.3. Communication

One obvious question is, how does the capability for communication between teammates affect the complexity of generating policies for multi-agent teams? Teams can be split into three categories according to the presence and type of communication available (Pynadath and Tambe, 2002; Goldman and Zilberstein, 2004):

- *no communication*, in which agents either do not have the capability to send messages to each other or the cost of communication is so high as to make it impractical in all cases,
- *free communication*, where messages can be exchanged instantly between teammates with no cost,
- and *general communication*, in which the capability to send messages exists but communication may be limited or incur a cost.

In the absence of communication, the complexity of DEC-POMDPs and DEC-MDPs is NEXP-complete (Bernstein et al., 2000). The COM-MTDP framework is an extension of MTDP (Multi-agent Team Decision Problem) that enables a formal analysis of the effect of communication on computational complexity (Pynadath and Tambe, 2002). MTDP is a

framework that is equivalent to DEC-POMDP, that, for the purpose of analysis, includes an additional parameter, $\mathcal{B}$, defined as:

- $\mathcal{B}$, the set of possible combined belief states for all agents, where the possible individual beliefs of a single agent are defined as the set of the agent's possible observation histories

COM-MTDP adds an additional parameter, Σ, and extends $\mathcal{R}$ to include separate reward functions for communication and the domain-level actions in $\mathcal{A}$. Thus, COM-MTDP is defined as the tuple $\langle \alpha, \mathcal{S}, \mathcal{A}, \Sigma, \mathcal{T}, \Omega, \mathcal{O}, \mathcal{B}, \mathcal{R} \rangle$, where:

- $\Sigma = \{\Sigma_i\}_{i \in \alpha}$, the set of possible messages available to the agents, where Σ_i is the set of messages that can be sent by agent i
- $\mathcal{R} = \mathcal{R}_{\mathcal{A}} + \mathcal{R}_{\Sigma}$, where $\mathcal{R}_{\mathcal{A}}$ is the basic reward function, defined over states and actions as for a Dec-POMDP, and $\mathcal{R}_{\Sigma}$ is the communication reward function ($\mathcal{R}_{\Sigma} : \mathcal{S} \times \Sigma \rightarrow \mathbb{R}$) with which the cost of communication can be expressed

Using the COM-MTDP framework, it can be formally demonstrated that free communication transforms a DEC-POMDP into a POMDP (and likewise, a DEC-MDP into an MDP), with PSPACE-complexity. This proof is shown by assuming that, at every timestep, every agent broadcasts its local observation to its teammates, with no effect on reward. This allows the the team to construction the joint observation and allows policies to be written and executed over joint observation histories. It can also be demonstrated that if communication is free, it is always better, or at least as good, for a team of agents to communicate their observations at every timestep than to apply a communication policy that chooses not to broadcast certain observations. Less intuitively, it is shown that for cases of general communication, where broadcasting a message may incur a negative reward, the problem of finding an optimal policy for a DEC-POMDP, which now includes a communication policy that indicates when messages should be broadcast, remains NEXP-complete (Pynadath and Tambe, 2002). This result has reinforced the motivation of researchers to explore heuristic approaches for generating communication policies, as discussed in Chapter 3. Table 2.1 summarizes the impact of communication on the complexity of generating optimal policies for distributed teams.

	MMDP	Dec-MDP	Dec-POMDP
No Comm.	P-Complete	NEXP-Complete	NEXP-Complete
General Comm.	P-Complete	NEXP-Complete	NEXP-Complete
Free Comm	P-Complete	P-Complete	PSPACE-Complete

Table 2.1. The effect of communication on the complexity of decentralized systems. (Pynadath and Tambe, 2002)

In this thesis, we address multi-agent teams with *general communication*. The ACE-PJB-COMM and SELECTIVE ACE-PJB-COMM algorithms introduced in Chapters 4 and 5 are applicable to teams operating under both *collective partial observability* and *collective observability*, modeled by Dec-POMDPs and Dec-MDPs. In Chapter 6, we limit ourselves to addressing teams with collective partial observability, modeled by Dec-MDPs. Overall, the planning problems that we address are NEXP-complete, motivating us to study execution-time communication heuristics.

2.3. Tools

In this section, we discuss several tools that were used in this thesis. First, we discuss Partially Observable Markov Decision Processes (POMDP) and present several methods for generating POMDP policies. Next, we discuss particle filters, which model arbitrary probability distributions with a fixed amount of memory. Finally, we give an overview of factored Markov Decision Problems.

2.3.1. Partially Observable Markov Decision Problems

We assume that the reader is generally familiar with Partially Observable Markov Decision Problems (POMDPs). Therefore, this section provides only a brief review of the concept of *belief*, and its use in generating and executing POMDP policies.

Policies for Markov Decision Problems, π take the form of a mapping from states to actions ($\pi : \mathcal{S} \rightarrow \mathcal{A}$), specifying an action to be executed in each state. A policy is considered "optimal" if executing it will lead an agent to accumulate the maximum possible expected reward. An agent modeled by a POMDP does not directly observe the world state, and therefore is unable to execute a policy specified in terms of states. Instead, agents in partially observable domains receive observations that allow them to make inferences about the state of the world. A policy for a POMDP could, therefore, be written as a mapping from observation histories to actions.

A more efficient representation, however, can be developed utilizing concept of a *belief state*, or simply *belief*, that serves as a sufficient statistic for summarizing an observation history (Åström, 1965). A belief is a probability distribution over the possible states of the world. Given a POMDP model and an initial belief distribution $b^0 \in \mathcal{B}$, the belief at the next timestep can be computed recursively from the current belief, taking into account the action a and the observation ω at the previous timestep (Kaelbling et al., 1998):

$$b^{t+1}(s') = \frac{\mathcal{O}(s', a, \omega) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s') b^t(s)}{Pr(\omega | a, b^t)} \tag{2.1}$$

$Pr(\omega|a,b)$ is a normalizing factor that describes the likelihood of seeing a particular observation ω after taking action a, when the current belief is b:

$$Pr(\omega|a,b) = \sum_{s' \in \mathcal{S}} \mathcal{O}(s',a,\omega) \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s')b(s) \tag{2.2}$$

POMDP policies can therefore be written as mappings from beliefs to actions ($\pi : \mathcal{B} \rightarrow \mathcal{A}$).

There has been extensive work done on efficiently finding solutions, both exact and approximate, to POMDPs. The work in this thesis, specifically Chapters 4 and 5, relies on the use of POMDP policies as input sources for generating Dec-POMDP solutions. As such, we have made use of two publicly distributed POMDP solvers:

- *pomdp-solve*, which provides a common syntax for specifying POMDP problem domains and implements several standard algorithms from the POMDP solution literature (Cassandra, 2005)
- and ZMDP, which implements heuristic search algorithms for solving both POMDPs and MDPs (Smith, 2007).

Another POMDP solution method utilized in this work is the Q-MDP heuristic (Littman et al., 1995). Q-MDP can be used to solve challenging POMDPs by first solving the underlying MDP, consisting of the state transitions that occur as a result of agent actions. This underlying MDP is solved by assuming that, at policy-generation time, the world state is fully observable. The MDP solution yields a set of value functions $\mathcal{Q}_a(s)$ that, for each state s, give the expected future reward for taking action a in that state and henceforth acting optimally. At execution time, Q-MDP selects the action that maximizes expected reward, conditioned on the current belief. Because the problem is a POMDP, and not an MDP, at execution time the true state is not known. However, the belief b gives a probability distribution over possible states and can be calculated at run-time from the observation history. The best action can be chosen by examining the average value of each action in every state, weighted by the probability of being in that state at the current time:

$$\text{Q-MDP}(b) = \arg\max_a \sum_{s \in \mathcal{S}} b(s) \times \mathcal{Q}_a(s) \tag{2.3}$$

Q-MDP is a greedy and optimistic heuristic that chooses actions as if the world were going to become fully observable in the next timestep. Therefore, unlike optimal algorithms that consider the value of information that can be gained by the possible actions, Q-MDP will not see potential value in taking information-gathering actions that do not immediately generate reward, causing it to suffer in comparison to other solution methods.

In addition to using Q-MDP to generate a policy for the Colorado/Wyoming domain introduced in Chapter 5 and detailed in Appendix A.3, Q-MDP also serves as inspiration for the Q-POMDP heuristic introduced in Chapter 4. Just as Q-MDP approximates the solution to a POMDP by solving the underlying MDP, Q-POMDP finds approximate solutions for Dec-POMDPs by using the policies for the underlying centralized POMDPs.

2.3.2. Particle Filters

When the state space is large, the amount of memory needed represent belief exactly as a vector of probabilities may be intractable. In continuous state spaces, an exact representation of belief is impossible. Particle filters are a sample-based method for approximating arbitrary probability distributions using a fixed amount of memory, and have been used extensively in POMDP research (Thrun, 2000; Poupart et al., 2001). Particle filters represent probability distributions by storing a fixed number of samples of the possible states being modeled. At each timestep, the state of each sample is propagated forward according to the transition function $\mathcal{T}$, given the action that was taken. Then, a weight is given to the new particle based on the probability of the observation that was received. New particles are resampled according to the weights that were assigned to represent the new belief distribution.

In Chapter 4, we use a particle filter to represent the distribution of the *possible joint beliefs* of a team of agents. As we discuss in that chapter, the number of distinct possible joint beliefs grows approximately exponentially over time. It therefore quickly becomes intractable to model the distribution exactly. A particle filter is useful because it can represent this growing distribution in constant space. However, there is an explicit tradeoff that must be dealt with between the size and the accuracy of a given particle filter. Updates for a particle filter composed of few particles can be performed quickly, but the distribution model encoded by those particles will be inaccurate. A particle filter with many particles can model a belief distribution more accurately, but may be very slow to update. We explore this tradeoff empirically in Chapter 4.

2.3.3. Factored MDPs

Markov Decision Problems use a flat representation of state, in which each possible state is enumerated, and the transition and reward functions indicate the relationship between individual states and actions. Although MDPs are P-complete (Papadimitriou and Tsitsiklis, 1987), there still exist MDP domains in which the state space is so large that existing algorithms for finding an optimal policy based on this flat state representation are

too computationally expensive to be practical. Instead, factored representations have been proposed to both reduce the size of the problem representation and to speed up the computation of policies. The state $\mathcal{S}$ in an MDP can be decomposed into n state features or variables $\mathcal{X} = \langle \mathcal{X}^1 \dots \mathcal{X}^n \rangle$, where each state s^i is an assignment of values to each variable in $\mathcal{X}$ (Boutilier et al., 1999).[3] Therefore, the transition function can be written to represent the probability of each variable taking on a particular value, given the values of its parent variables and the action in the previous timestep. Dynamic decision networks have been used to represent the conditional relationships between parent and child variables across timesteps (Dean and Kanazawa, 1989).

Taking advantage of these compact representations is challenging. In general, there are two classes of approaches that use factored representations to efficiently compute policies for large MDPs. One approach is to define *basis functions* over the state variables (Koller and Parr, 1999). Each basis function depends on some subset of the state variables, and a complete MDP policy is a weighted sum of basis functions. Because each basis function depends on only a small subset of state variables, computing policies over the set of basis functions is more efficient than doing so over the complete state representation. Currently, the fastest MDP solution algorithms utilize this basis function representation (Guestrin et al., 2003; Lagoudakis and Parr, 2003). However, the problem of defining useful basis functions is a difficult one, and is a current area of active research.

We base the work in Chapter 6 on a different approach to factored representations. Instead of basis functions, we are interested in factored policies that can be represented as decision trees, with state variables at the internal nodes and actions at the leaves (Boutilier and Dearden, 1996). One algorithm developed to generate these tree-structured policies is called Structured Policy Iteration (Boutilier et al., 2000). Structured Policy Iteration (SPI) takes as input an MDP representation in which the conditional probability table for each state variable is represented as a decision tree branching over that variable's parent variables from the previous timestep. SPI also requires that the reward function be specified as a tree, and that it depend only on the current state, and not states and actions. Structured Policy Iteration returns a compact tree-structured policy by taking advantage of a property called *context-specific independence* (Boutilier et al., 1996), in which a variable is dependent on some parents only for certain values of other parents, and independent otherwise. As we show in Chapter 6, the concept of context-specific independence can be extended to

[3] This representation is general. Domains in which the state cannot be decomposed can still be represented with factored MDPs, where $|\mathcal{X}| = 1$ and $\forall i, s^i \rightarrow \mathcal{X} = i$.

multi-agent domains, and is a useful property for identifying situations in which one agent can act without needing to consider state variables observed only by its teammates.

CHAPTER 3

Related Work

THE Dec-POMDP and Dec-MDP models that form the underlying representations for the work in this thesis have attracted considerable attention in recent years. In this chapter, we present an overview of prior work aimed at finding policies for teams modeled by these representations. Section 3.1 discusses how Dec-POMDPs and Dec-MDPs can be solved optimally. However, because Dec-POMDPs and Dec-MDPs are NEXP-complete, and therefore optimal algorithms will never allow the solution of large problems, the majority of work in this area has focused on heuristic solutions that generate sub-optimal policies. Section 3.2 describes some of these approximate solution methods. Finally, like the work in this thesis, there has been some attention to the challenges of including communication in reasoning about cooperative multi-agent teams. Section 3.3 discusses other work that has been done in this area, with particular attention to how previous approaches compare to the work presented in this thesis.

3.1. Exact Solutions

There are many decision-theoretic models that, like the Dec-POMDP formulation introduced in (Bernstein et al., 2000), can be used to represent cooperative teams operating with uncertainty in the outcomes of their actions and under partial observability. Some can model both cooperative and non-cooperative teams, such as the Interactive POMDP framework (I-POMDP) (Gmytrasiewicz and Doshi, 2005), which requires agents to maintain recursive beliefs about their teammates and their teammates beliefs about them and so on, and Partially Observable Stochastic Games (POSG), which use a game theoretic representation to model the interactions of agents (Emery-Montemerlo, 2005). The Multi-agent Team Decision Problem (MDTP) includes in its model an explicit representation of the possible belief states of each agent (Pynadath and Tambe, 2002). What all of these models

have in common is their intractability. Nonetheless, there has been some attention to the question of finding optimal solutions to Dec-POMDPs and Dec-MDPs.

Dynamic programming has commonly been used to solve sequential decision-making problems (most notably in work based on the Bellman update (Bellman, 1957)). Hansen *et al.* introduce a dynamic programming algorithm that generates optimal policies for POSGs (Hansen et al., 2004). Because POSGs represent multi-agent decision problems as iterated games, the authors were able to apply game theoretic techniques to iteratively prune those strategies for each agent that are weakly dominated. By reducing the size of the problem space to be searched, this dynamic programming algorithm is able to solve larger problems than can be solved by brute-force search. Unfortunately, the problems that can be solved with dynamic programming are still very small, both in terms of domain size (i.e. state space, number of actions, number of observations) and time horizon. Allen and Zilberstein have recently begun attempting to characterize multi-agent domains by predicting how easily they can be solved by dynamic programming techniques (Allen and Zilberstein, 2007). They define a metric, the *influence gap*, that measures the effect of each agent's actions on the outcome of a domain's transition and reward functions. Preliminary experimental results indicate that domains in which there is a large influence gap, meaning that one agent affects the behavior of the system considerably more than its teammate, can be solved with dynamic programming to longer time horizons than problems with small influence gaps, in which the agents affect the system equally.

Another approach extends the A* algorithm from classical planning to multi-agent domains. Multi-agent A* (MAA*) finds optimal policies for multi-agent POMDPs via heuristic search in the space of possible policies (Szer et al., 2005). Like the classical A* algorithm, MAA* requires a heuristic function that optimistically estimates the future values of states to be expanded, in this case policies at time $t + 1$. With some simple value estimation heuristics, MAA* was shown to solve small Dec-POMDPs with time horizons up to 4. Heuristics that provide a tighter upper bound on the value of a policy could allow for faster solutions, as shown by (Oliehoek and Vlassis, 2007), who compared the performance of MAA* using the Q-MDP (Littman et al., 1995), Q-POMDP (Roth et al., 2005), and Q-BG heuristics (Oliehoek and Vlassis, 2007).

Given the difficulty inherent in solving the general class of Dec-POMDPs and Dec-MDPs optimally, there has been some work done to characterize sub-classes of these problems that may admit more tractable solution. The most significant sub-class to be identified contains Dec-MDP problems possessing the property of *transition independence* (Becker et al., 2003). A transition-independent Dec-MDP is one in which the world state can be

partitioned into a set of local states, one for each agent. An agent's individual actions affect only its own local state transitions and not the state of its teammates. However, because the agents are connected through a shared reward function that depends on the joint action, the system cannot be treated as a set of individual MDPs.

The class of transition independent Dec-MDPs may be further subdivided to separate out those domains in which agents' observations are independent of their teammates' actions. Domains exhibiting both transition and observation independence, in which each agent has full observability of its local state, have been shown to be NP-complete, making them provably less complex than non-independent Dec-MDPs (Becker et al., 2004). Additionally, if the problem is *goal-oriented*, meaning that there is a single goal state that the agents are attempting to reach, and if the cost for each action in the attempt to reach that goal is uniform, then a transition and observation independent Dec-MDP is P-complete (Goldman and Zilberstein, 2004).

One might suppose that because these sub-classes have lower computational complexities than the general case of Dec-POMDPs and Dec-MDPs, problems within these classes can be easily solved by solution techniques designed for Dec-POMDPs and Dec-MDPs. Interestingly, an analysis of agent influence in these domains shows that this may not be the case (Allen and Zilberstein, 2007). Because each agent exerts complete control over its local state, the overall contribution of each agent to the behavior of the system is approximately equal. This small influence gap has been shown experimentally to correspond to an inability by dynamic programming techniques to generate policies for transition independent domains efficiently, indicating that specialized solution techniques will need to be developed to exploit the lowered computational complexity of these domains.

3.2. Approximate Solutions

The majority of recent work in the area of Dec-POMDPs and Dec-MDPs has focused on finding approximate policies. One class of approaches looks at speeding up the calculation of Dec-POMDP policies by restricting the types of policies that can be discovered. Peshkin *et al.* presented a method for solving POIPSGs (Partially Observable Identical Payoff Stochastic Games, or POSGs in which the agents are cooperative and receive a team reward) by using gradient descent methods to learn a finite state controller with fixed memory for each agent (Peshkin et al., 2000). Bernstein *et al.* also restrict the policies that their algorithm can learn to consist of a finite state controller per agent (Bernstein et al., 2005). However, because they allow stochasticity in the controllers' state transitions and action

selections, they are better able to capture the observation sequences of the team. They further improve the performance of these fixed-size policies by introducing a shared source of randomness for all of the agents, what they call a *correlated joint controller*. This shared randomness, which improves team coordination, does not constitute a form of communication, as the joint controller does not observe the behavior of the system.

Another class of approaches looks at reducing the amount of computation that needs to be done to generate policies by placing limits on the granularity at which agents model their beliefs about the states and actions of their teammates. One such algorithm is the Bayesian game approximation (BaGA) introduced in (Emery-Montemerlo et al., 2004). In that work, agents use Bayesian games to model a subset of the possible action and observation histories of their teammates. Agents can then calculate greedy policies by choosing actions that maximize expected reward in combination with how the game predicts their teammates will behave. As agents model larger subsets of their teammates' possible histories, they can expect performance to improve. Likewise, to model a team of agents exactly, I-POMDPs require each agent to keep an infinitely deep recursive belief, called the *interactive state*, about the other agents and their beliefs (Gmytrasiewicz and Doshi, 2005). In practice, the recursion in the interactive state must be terminated at some finite depth. At this terminating depth, agents assume a *no-information model*, supposing that each agent i thinks that agent j models i as if all of i's action choices are equally likely. Shallower recursive beliefs trade off model accuracy for more tractable computation. I-POMDP computation can be further sped up by using a particle filter to represent both the interactive state and beliefs about the world state (Doshi and Gmytrasiewicz, 2005).

There has been some work directed at speeding up dynamic programming for Dec-POMDP policy generation. Szer and Charpillet define a *multi-agent belief state* for each agent consisting of a probability distribution over world states and a belief about the future behavior of the other agents (Szer and Charpillet, 2006). Inspired by sample-based techniques that have been used effectively to solve single-agent POMDPs (e.g. (Pineau et al., 2003)), point-based dynamic programming for Dec-POMDPs samples likely beliefs within the set of possible multi-agent belief states and generates policies accordingly. Memory Bounded Dynamic Programming (MBDP) uses top-down heuristics to search for likely belief states (Seuken and Zilberstein, 2007). These belief states are then used to prune the policy subtrees generated by each timestep of dynamic programming, so that only a fixed number of possible policies is preserved. In this way, MBDP bounds the amount of memory used by dynamic programming. If there are only a few reachable belief states, this

pruning may be very effective, allowing for the computation of policies to much longer time horizons than could be computed with an optimal dynamic programming algorithm.

Finally, there are approaches that focus on finding locally-optimal policies. One such algorithm is JESP, Joint Equilibrium-based Search for Policies (Nair et al., 2003). In this method, agents take turns improving their policies by finding the best response to the fixed policies of their teammates. This process is repeated iteratively until it converges when no single agent can improve the team reward by changing only its own policy. JESP is guaranteed to converge, although not to the globally optimal solution, because the agents are cooperative and change their policies only when this change would cause a net gain in expected team reward. In practice, JESP converges quickly on some small test domains, although in the worst case, the algorithm may iterate through all possible policies. There have been numerous extensions to JESP. One such extension, LID-JESP (Locally Interacting Distributed JESP), uses distributed constraint optimization techniques to speed up policy computation in domains where agent interaction is localized, meaning that each agent's transition and observation functions depend on the actions of only a few of the agent's teammates (Nair et al., 2005). Whereas most Dec-POMDP solution algorithms depend on the problem starting from a single, known distribution over world states, another extension of JESP, CS-JESP (Continuous [Belief] Space JESP) solves Dec-POMDPs with arbitrary initial belief distributions (Varakantham et al., 2005). CS-JESP combines JESP with single-agent POMDP solution techniques that find policies over ranges of beliefs, and speeds up computation by pruning unreachable belief states.

3.3. Communication

The focus of this thesis is on the effective use of communication as a means to facilitate the decentralized execution of centralized policies and improve team performance through information sharing. Although it is known that the presence of communication does not change the complexity of solving Dec-POMDPs and Dec-MDPs unless it is free and unlimited (Pynadath and Tambe, 2002), communication is nonetheless an important resource for multi-agent coordination. There has been some work that looks at how communication can be used during policy generation to facilitate decentralized planning (e.g. (Guestrin et al., 2002; Shen et al., 2003)). In this section, however, we focus on approaches that, like the ideas presented in this thesis, employ a centralized planner and use communication to assist teams with decentralized execution.

There are many ways to classify communication approaches that have been researched in the area of Dec-POMDPs and Dec-MDPs. We identify several here:

- Is the approach applicable to general Dec-POMDP or Dec-MDP domains, or to restricted domains as discussed in Section 3.1?
- With which of the three communication types, *tell*, *query*, and *sync* (Xuan et al., 2001), is the algorithm compatible?
- Does communication replace a domain-level action in the timestep in which it takes place (*OR-communication*), or can both communication and action take place in the same timestep (*AND-communication*) (Emery-Montemerlo, 2005)?

The algorithms detailed in this thesis are applicable to general Dec-POMDPs (Chapters 4 and 5) or Dec-MDPs (all algorithms), and utilize the *AND-* model of communication. The ACE-PJB-COMM and SELECTIVE ACE-PJB-COMM algorithms in Chapters 4 and 5 employ *tell* communication, and the work in Chapter 6 uses the *query* paradigm.

Goldman and Zilberstein point out that, as in unrestricted Dec-POMDPs and Dec-MDPs, the presence of limited or costly communication in restricted domains does not reduce the computational complexity of finding optimal policies. Therefore, finding action and communication policies for Dec-MDPs with independent state transitions and observations remains NP-complete. Nonetheless, there has been significant work addressed at developing heuristic algorithms to utilize communication in transition- and observation-independent Dec-MDP domains. In an early approach, Xuan *et al.* propose to study the impact of *AND-communication* on agent team performance by extending each agent's local state to include a communication history (Xuan et al., 2001). Agent policies, therefore, depend, not only on world state, but on information received from their teammates. Within this context, the authors propose a communication heuristic called "no news is good news." Agents start out by communicating to determine the current world state and calculate an optimal plan based on this initial state. Each agent executes its individual component of the plan, initiating communication when it discovers that the current plan is no longer achievable. When one agent initiates communication, all of the teammates *sync* to discover the global state, and compute a new plan. This approach is primarily applicable in *goal-directed* Dec-MDPs, in which there is a clearly defined goal state or set of goal states.

Another approach, also intended for Dec-MDPs with independent transitions and observations, is similar in concept to the factored approach detailed in Chapter 6 (Xuan and Lesser, 2002). Both approaches look at how communication can be used to process a centralized policy and transform it into individual policies for each agent. As with the work in this thesis, Xuan and Lesser seek to reduce the amount of communication needed during

execution by identifying states in which it can be determined that communication is unnecessary. However, unlike our work, Xuan and Lesser's algorithm instructs agents to communicate their complete local states, rather than identifying relevant state features, and requires agents to *sync* whenever one agent initiates communication. In their approach, each agent executes by maintaining, at each timestep, the complete set of possible world states. This set, called the *local belief set*, consists of that agent's local state and all of the possible local states of that agent's teammates. Every state in the local belief set maps to a joint action in the team's centralized policy. Each agent i examines the set of possible joint actions elicited by its local belief set and determines if its own local component action is the same in every joint action. If so, the agent may execute that action without communication. However, if there is ambiguity in the agent's local action choice, the agent initiates a *sync* communication, and all agents communicate their local states, returning the team to a synchronized belief consisting of the true global state. Communication in this approach is *AND-communication*, and does not replace domain-level actions.

Becker *et. al* formally analyze the limitations of myopic assumptions when making communication decisions in transition-independent Dec-MDPs (Becker et al., 2005). They consider a *sync*, *AND-communication* framework. The myopic assumption explored in their work is one in which agents make communication decisions by supposing that they have the option to communicate in the current timestep, but will never be allowed to communicate again. Under this assumption, agents compute the *value of communication* as the difference between the expected value after a *sync* and the expected value if no communication takes place. In this thesis, particularly in Chapters 4 and 5, we make a different myopic assumption. Agents decide whether to communicate in the current timestep, assuming that there will be free communication in all future timesteps. Despite the difference in the assumptions, Becker *et. al* point out a limitation of myopic approaches that is shared by our work. Agents making myopic communication decisions assume that their teammates never initiate communication. In a *sync* paradigm, this limitation leads agents to systematically under-estimate the expected value of not communicating. In a *tell* paradigm, agents may under-estimate the value of communication because they fail to consider the synergistic effect of multiple agents communicating.

Two models have been proposed to capture the inclusion of communication in general Dec-POMDP domains. One such model, Dec_POMDP_Com (Goldman and Zilberstein, 2003), adds two components to the original Dec-POMDP model:

- Σ - the set of possible messages
- C_Σ - the cost function for each sending messages ($C_\Sigma : \Sigma \rightarrow \mathbb{R}$)

Another model, COM-MTDP, also extends the equivalent MTDP formulation by adding a set of possible messages Σ. However, rather than a fixed cost per message, C_Σ, COM-MTDP extends the reward function to include a component $\mathcal{R}_\Sigma$ that depends on both the message and the current state ($\mathcal{R}_\Sigma : \mathcal{S} \times \Sigma \rightarrow \mathbb{R}$) (Pynadath and Tambe, 2002). In principle, these models can admit either *tell* or *sync* communication, although all experimental results utilize *sync*. Both COM-MTDP and Dec_POMDP_Com explore the problem of generating a *communication policy* for the team. For each agent, its communication policy maps its observation and communication history to a communication decision at each timestep. The models assume *AND-communication*, meaning that agents first decide whether and what to communicate, and then use the outcome of that timestep's communication to make an action decision. Dec_POMDP_Com and MDTP are useful for proving the theoretical properties of Dec-POMDP teams with communication (such as the fact that the decision problem for Dec_POMDP_Com where $C_\Sigma(\sigma^i) > 0$ for some message σ^i or COM-MTDP where $\mathcal{R}(s^i, \sigma^j) > 0$ for some state s^i and message σ^j is NEXP-complete). However, while there may be some domains that can be successfully modeled using Dec_POMDP_Com and COM-MTDP, they cannot be used in practice to model domains with arbitrary communication. Consider the problem of enumerating, *a priori*, all possible messages that a team may wish to communicate. In some cases, it may be useful for agents to communicate any of the possible observation histories that they could experience. The set of possible individual observation histories is $|\Omega_i|^{\mathcal{T}}$, exponential in the number of possible individual observations, and potentially infinite if time horizon $\mathcal{T} = \infty$. Additionally, because MTDP contains an explicit representation, $\mathcal{B}$ of the possible beliefs of each agent, COM-MTDP must define how each possible instance of communication will affect the belief state, which is impractical.

Most work on communication in Dec-POMDPs and Dec-MDPs, including the formal models described above, have assumed an *AND-* model of communication, where each timestep is divided into a communication phase, in which agents may or may not choose to communicate, and an action phase, in which each agent executes a joint action. COMMUNICATIVE JESP uses *OR-communication* to extend the JESP algorithm (Nair et al., 2004). Instead of producing a separate communication policy, to be executed during the communication phase of each timestep, COMMUNICATIVE JESP extends domains by adding an additional domain-level action, SYNC. As indicated by the name, if one agent chooses the SYNC action, it initiates a *sync* in which it, and all of its teammates, broadcast their complete observation histories since the last *sync*. In this way, SYNC returns the team to a single synchronized belief over world states. Because it both leads to a more compact policy and

restricts the action choices of the other agents, communication not only improves team performance but also speeds up policy computation. To further increase policy computation speed, COMMUNICATIVE JESP can be used with a communication heuristic requiring agents to communicate at least once every K timesteps. There is a tradeoff between ease of computation and the value of the computed policy, as larger K-values lead to better performance. [1] As with JESP (Nair et al., 2003), COMMUNICATIVE JESP is not guaranteed to converge to an optimal policy and may, in the worst case, require the same time as a brute force search to converge.

BAGA-COMM (Emery-Montemerlo, 2005) is the approach that is most similar to the ACE-PJB-COMM algorithm detailed in this thesis. Like ACE-PJB-COMM, BAGA-COMM is applicable to general Dec-POMDPs, employs *tell* communication, and is an *AND-communication* approach, with separate communication and action phases in each timestep. The primary difference between the two algorithms is that ACE-PJB-COMM maintains strict joint action coordination among the agents, ensuring that the team avoids all coordination errors, while BAGA-COMM allows occasional instances of mis-coordination while seeking to enable good team performance on average. Like BAGA (Emery-Montemerlo et al., 2004), agents executing BAGA-COMM model a subset of possible team beliefs, called the *types* of the teammate players. Each agent's own *type* encodes its local action and observation history. To choose an action, each agent looks for the team type whose belief is most similar to the belief elicited by its own type and executes the local component of the joint action associated with that team type. Because agents use local information to choose actions, they are not guaranteed to avoid mis-coordination.

In BAGA-COMM, as in ACE-PJB-COMM, agents decide whether to communicate by looking at how communicating their local information would change the team type, choosing to communicate if this change would lead to a gain in expected reward greater than the cost of communication. BAGA-COMM answers the question of **what** to communicate by clustering possible agent types and assigning each possible cluster an identifying number. Then, when an agent decides to communicate, it broadcasts its cluster ID to its teammates. This cluster representation compresses similar agents types, reducing some of the redundant information that would be otherwise communicated by an algorithm like ACE-PJB-COMM that communicates complete observation histories. However, it does not have the flexibility of SELECTIVE ACE-PJB-COMM, which is able to look at the impact of communicating arbitrary subsets of the observation history.

[1] In the extreme cases, $K = \infty$ is unrestricted communication, and $K = 1$ replaces every domain-level action with SYNC.

3.4. Summary

The work detailed in this thesis presents a heuristic approach to coordinating multi-agent teams modeled by Dec-POMDPs and Dec-MDPs, using communication to facilitate the decentralized execution of centralized policies. The methods presented here are applicable to unrestricted Dec-POMDP or Dec-MDP domains. We use *AND-communication*, assuming a separate communication and action phase in each timestep, so that communication facilitates better domain-level action selection, rather than replacing domain-level actions. We do not require agents to *sync* whenever one agent chooses to communicate. Instead, we present some algorithms in which agents reason about what to *tell* their teammates and some in which the agents decide to *query* their teammates for information. The unique feature of our research, which is not shared by any of the approaches described in this chapter, is that our algorithms guarantee that teammates will avoid coordination errors during execution. This makes our work particularly applicable to domains that require tight coordination, in which the cost of mis-coordinated joint actions is high.

CHAPTER 4

ACE-PJB-COMM: Choosing *When* to Communicate

BECAUSE the problem of generating optimal policies for Dec-MDPs and Dec-POMDPs is known to be NEXP-complete (Bernstein et al., 2002), heuristics must be found to allow their application to the coordination of teams of agents. The overall approach presented in this thesis is to generate centralized policies for the teams off-line, ignoring the decentralized nature of the problem, and then use communication to facilitate decentralized execution of these policies at run-time. This chapter provides a mechanism by which a decentralized team of agents can execute a centralized policy and **A**void **C**oordination **E**rrors by reasoning about the **P**ossible **J**oint **B**eliefs of the team (ACE-PJB). In this chapter, we also address the question of **when** agents should communicate their observations to their teammates.

Section 4.1 introduces the concept of *joint beliefs*, which can be computed by a centralized team of agents to model the distribution of possible world states, and which would enable the team to execute a joint policy. However, because the teams we are concerned with in this work are decentralized, Section 4.3 discusses how a team can instead execute a centralized policy by modeling the distribution of *possible joint beliefs*. We show how the possible joint beliefs of a team of decentralized agents can be modeled as a tree, where each leaf in the tree is one possible joint observation history of the team. We illustrate decentralized execution based on the distribution of possible joint beliefs with an example in the two-agent tiger domain. The ACE-PJB-COMM algorithm, which addresses the question of **when** communication can assist a team with decentralized execution of a centralized policy, is introduced in Section 4.4. Section 4.5 empirically compares performance of ACE-PJB-COMM to a team with free communication in a number of domains. In Section 4.6, we show how a particle filter can be used to implement a constant-space variant of the ACE-PJB-COMM algorithm, and illustrate via experimentation how the number of

particles used in the particle filter affects the performance of a team executing the ACE-PJB-COMM algorithm. We conclude by discussing some of the issues raised during our work on the ACE-PJB-COMM algorithm that we feel merit further investigation.

4.1. Joint Beliefs

Just as policies for single-agent MDPs are written as mappings from states to actions, policies for single-agent POMDPs are represented as mapping from *beliefs* to actions. An agent executing a POMDP policy can compute the current belief, a probability distribution over possible world states, as described in Section 2.3.1, and choose the action that maximizes expected reward at that belief. Because the transition and observation functions of a Dec-POMDP depend on the joint action, and a complete model of belief would depend on the joint observation, a member of a team modeled by a Dec-POMDP cannot compute the belief distribution independently. Without knowing the actions taken and observations received by its teammates, a single agent is unable to calculate belief, and therefore, cannot execute a policy based on belief. However, as we show in 4.3, agents can compute a *distribution of possible joint beliefs*, which can enable them to execute a centralized policy.

Let us suppose that agents *do* know the actions that their teammates take at every timestep. Each agent could, therefore, determine the *joint action* taken by team, since the joint action is simply the tuple of individual actions taken by each agent. If every agent communicated its local observation to all of its teammates at every timestep, the agents could likewise determine the *joint observation* received collectively by the team. The agents would now have sufficient information to calculate the distribution of possible world states at each timestep of their execution. We call this probability distribution, computed over joint actions and joint observations, the *joint belief* of the team. A single-agent POMDP solver could then compute a policy that maps joint beliefs to joint actions, by treating joint actions and joint observations as atomic. We call this policy, which treats the team as if it were controlled by a centralized controller, a *joint policy*.

4.2. Multi-agent Tiger Domain

We illustrate our approach in this chapter through a running example in the multi-agent tiger domain (Nair et al., 2003), an extension of the classic tiger problem that has been used in POMDP research (Kaelbling et al., 1998). A complete description of the multi-agent tiger domain can be found in Appendix A.1.

The world of the tiger problem consists of two states, SL and SR. These states correspond to two doors, one on the left and one on the right. Behind one door is a tiger, and behind the other is a treasure.

- SL is the state in which the tiger is behind the left door.
- SR is the state in which the tiger is behind the right door.

The world is initialized randomly, with the tiger equally likely to be behind either door (i.e. p(SL) = p(SR) = 0.5). Figure 4.1 shows an illustration of the tiger domain with two agents.

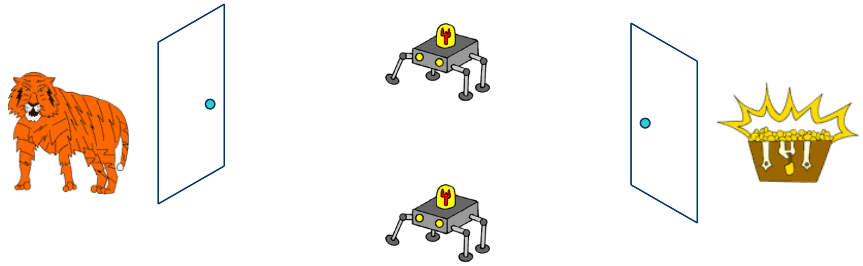


Figure 4.1. The state space of the multi-agent tiger domain consists of two doors. Behind one door is a tiger, and behind the other is a treasure. The agents' goal is to open the door with the treasure behind it.

The goal of the agents in the tiger problem is to open the door with the treasure behind it. To this end, there are three actions that can be performed:

- OPENL opens the left door,
- OPENR opens the right door,
- LISTEN is information-gathering action that does not open a door but provides a noisy observation about the position of the tiger.

In the multi-agent tiger problem, there are α agents acting in the world, each of which can independently perform any of the three actions. For simplicity, we describe here the case in which $\alpha = 2$. The world resets whenever a door is opened, with the tiger placed behind a random door, potentially allowing the problem to have an infinite time horizon. If all agents perform LISTEN, the state is unchanged. A joint action is denoted as a tuple of individual actions. For example, the joint action $\langle$OPENR, OPENL$\rangle$ indicates that agent 1 performed OPENR and agent 2 performed OPENL.

Rewards for the tiger problem are structured such that agents should avoid opening the door with the tiger. The maximum possible reward occurs when both agents open the door with the treasure. For example, if the state is SL and the agents perform $\langle$OPENR, OPENR$\rangle$, they receive the maximum possible reward of +20. An explicit coordination problem is built into the domain by making it preferable for the agents to both open the wrong door together (reward = -50) rather than for each agent to open a different door (reward = -100). There is a small cost of -2 if the agents perform the joint action

$\langle$LISTEN, LISTEN$\rangle$. In the examples below, we assume that there is a marginal cost of ϵ for each instance of communication. We will discuss other models of communication cost in Chapter 5.

To increase their chances of opening the correct door, the agents must increase their certainty about the position of the tiger. At each timestep, each agent receives an independent observation. The possible observations are HEARLEFT (or HL), indicating that the tiger has been heard behind the left door, and HEARRIGHT (HR), indicating that the tiger has been heard behind the right door. If either agent opens a door, the observations received in that timestep have no informative value. If both agents perform LISTEN, the observations are independent (meaning agents may hear different observations in the same timestep) and noisy, correctly identifying the position of the tiger with 0.7 probability.[1]

Figure 4.2 shows the alpha-vectors of the joint policy for this problem, generated using Cassandra's POMDP solver (Cassandra, 2005) with a discount factor of $\gamma = 0.9$. Note that although there are nine possible joint actions, all actions other than $\langle$OPENL, OPENL$\rangle$, $\langle$OPENR, OPENR$\rangle$, and $\langle$LISTEN, LISTEN$\rangle$ are strictly dominated and do not appear in the policy.

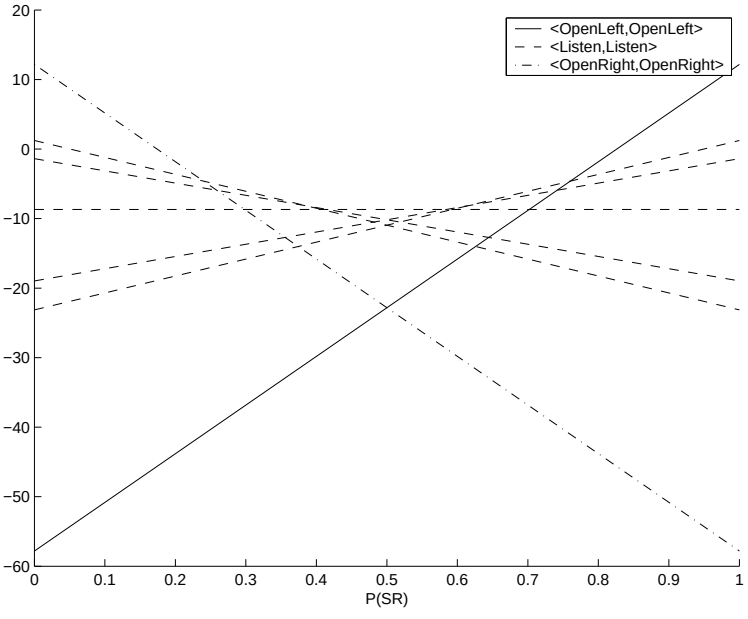


Figure 4.2. Alpha-vectors for an infinite-horizon joint policy for the two-agent tiger domain.

[1]This particular observation model, a slight modification of the model presented by Nair *et al.*, makes it so that the optimal policy for a centralized team would be to hear two consistent observations (e.g. a joint observation of $\langle$HR, HR$\rangle$) before opening a door.

In fact, we can demonstrate that all of the dominated joint actions, ⟨OPENL, OPENR⟩, ⟨OPENL, LISTEN⟩, ⟨OPENR, OPENL⟩, ⟨OPENR, LISTEN⟩, ⟨LISTEN, OPENL⟩, and ⟨LISTEN, OPENR⟩, constitute *coordination errors* as defined in Chapter 1. Concretely, these actions are not fixed points with respect to the policy and that, if an agent, after deciding on an individual action, were to be told that the resulting joint action belonged to this set of dominated actions, the agent would prefer to change its individual action in response. For example, consider a situation in which both agents LISTEN twice, and one agent observes two instances of HEARLEFT. This agent, believing that the tiger is behind the left door with probability 0.845, will wish to perform the individual action OPENR. However, if it were to discover that its teammate will choose to LISTEN, the agent would decide to change its own action to LISTEN. Even given the agent's belief in the high likelihood of the tiger being behind the left door, the infinite-horizon discounted expected reward of ⟨LISTEN, LISTEN⟩, 20.13, is higher than the expected reward of ⟨OPENR, LISTEN⟩, 8.33. More dramatically, if the agent discovers that its teammate will open the left door, it will change its own action to OPENL. The expected reward of opening the wrong door while acting in coordination, -22.70, is still greater than the reward of a coordination error, ⟨OPENR, OPENL⟩, which is -83.62.[2] In the next section, we present an algorithm that guarantees that agents will avoid these coordination errors while trying to maximize expected reward.

4.3. Decentralized Execution of Joint Policies

Execution of a joint policy is straightforward if every team member knows the joint action and joint observation of the team at each timestep. Each agent uses the joint action and joint observation to compute the joint belief according to the Bayes update rule described in Section 2.3.1. This belief maps to a joint action in the joint policy. Each agent then executes its own component of the joint action. However, if two conditions are true, it is sufficient for each team member to know the joint observation:

1. If the team starts with a single, known joint belief b^0.
2. If the joint policy, π, known by all agents, is deterministic, meaning that each belief maps to exactly one joint action.

Given b^0 and the deterministic policy π, each agent can evaluate $\pi(b^0)$ to determine which action $a^0 \in \mathcal{A}$ is the best joint action for the team to execute. If after executing a^0, the

[2]Expected rewards are computed for each action by one-step lookahead. The immediate expected reward for an action a in a belief b is $\sum_{s \in \mathcal{S}} \mathcal{R}(s,a) \times b(s)$. The expected future reward is computed by calculating the possible next beliefs for each possible joint observation ω, b^ω. The total expected future reward is $\sum_{\omega \in \Omega} \mathcal{V}(b^\omega) \times Pr(\omega|a,b)$, where $Pr(\omega|a,b)$ is the likelihood of observing ω after taking action a in belief b, and $\mathcal{V}(b^\omega)$ can be computed directly from the policy. The overall expected reward is the sum of the immediate reward and the discounted future reward.

agents then inform each other of the local observations that they receive, each agent can use a^0 and the known joint observation w^0 to compute the next joint belief b^1, and then the next optimal joint action $a^1 = \pi(b^1)$. The agents will thereby maintain a synchronized joint belief, and avoid potential coordination errors.

This demonstrates that communication at every timestep enables a decentralized team of agents to execute a joint policy. In such a case, where there are no limitations on communication, it seems inaccurate to refer to the team as "decentralized". The challenge is to enable a decentralized team to execute a centralized policy, while **A**voiding **C**oordination **E**rrors (ACE), when there are restrictions on agent communication. We first show how a decentralized team of agents can execute a centralized policy without communicating at all, and then, in Section 4.4, introduce an algorithm by which agents can reason about **when** communication can be used to improve team performance.

As explained above, in our work we make the assumption that agents begin execution with a single, known joint belief. In the two-agent tiger domain, this belief is a uniform distribution over the two possible positions of the tiger (i.e. $b(\text{SR}) = 0.5$). Given this belief and the pre-computed joint policy, each agent can determine that the best joint action is $\langle \text{LISTEN}, \text{LISTEN} \rangle$, and can be certain that its teammates will choose the same joint action. However, after executing its own component of the joint action, each agent observes a local observation. If the team was, in fact, centralized, as assumed when the joint policy was calculated, each agent would be notified of its teammates local observations and could update the joint belief accordingly, allowing the team to continue executing the joint policy.

Because the teams with which we are concerned are decentralized, individual agents cannot calculate a single joint belief. Instead, each agent could attempt to estimate the joint belief based on its own local observation and then act accordingly. This introduces two problems. First, if agents are allowed to act independently based on their local observations, they may suffer from coordination errors. For example, in the tiger domain, if one agent observes HEARRIGHT, it may note that it is likely that its teammate also heard the tiger behind the right door, and therefore choose OPENL. However, its teammate may have actually observed HEARLEFT, and the teammates will therefore open different doors. Second, because each agent does not know what its teammates have observed, if those teammates are allowed to choose their actions based on their observations, agents will not be able to determine the joint action executed by the team at each timestep. As the transition and observation functions ($\mathcal{T}$ and $\mathcal{O}$) are conditioned on the joint action of the team, agents that do not know the joint action will not be able to make correct inferences about

belief. Instead, as agents incorrectly estimate the actions chosen by their teammates, the joint beliefs calculated by each agent will increasingly diverge from each other.

To prevent coordination errors and enable agents to make accurate estimates of joint belief, we make the following assumption in our work. We assume that, at each timestep, every agent knows the individual actions chosen by all of its teammates, and therefore knows the joint action. We enforce the validity of this assumption by requiring agents to act only on the basis of information that is known collectively to all teammates. Agents must, therefore, disregard their local observations unless those observations have been publicized to all teammates via communication. The assumption that agents act based on shared information and thereby synchronize their choice of joint action holds through this and the following chapters. In Chapter 6, we discuss conditions under which this assumption can be relaxed.

4.3.1. Modeling Possible Joint Beliefs

To execute a joint policy while maintaining a synchronized joint action selection and thereby avoiding coordination errors, agents must reason about the shared information collectively known by all teammates. We have already stipulated that agents share a known initial joint belief, b^0, and a deterministic joint policy π. Given this information, they can compute a synchronized joint action, a^0. The question, then, is how the team should proceed after executing this joint action. One possibility is that agents could model the *average joint belief*, $\bar{b}^t$, at each timestep, and execute the joint action associated with that belief. The average joint belief is computed by considering the possibility of the team observing each possible joint observation. For each possible joint observation ω^t, the agents compute the joint belief $b^t_{\omega^t}$ that would result if the team did, in fact, observe ω^t:

$$b^t_{\omega^t}(s') = \frac{\mathcal{O}(s', a^t, \omega^t) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a^t, s') \bar{b}^{t-1}(s)}{Pr(\omega^t | a^t, \bar{b}^{t-1})} \tag{4.1}$$

The weighted average of these beliefs takes into account the likelihood of the team observing a particular joint observation ω^t:

$$\bar{b}^t = \sum_{\omega^t \in \Omega} Pr(\omega^t | a^{t-1}, \bar{b}^{t-1}) \times b^t_{\omega^t} \tag{4.2}$$

The average joint belief is a statistic that the agents can compute identically, without considering local information that is not shared among teammates, and therefore choosing an action based on the average joint belief will enable agents to execute a joint policy without coordination errors. However, it is easy to construct domains in which acting on the basis of the average joint belief would lead to poor team behavior. Consider the policy shown in Figure 4.3.

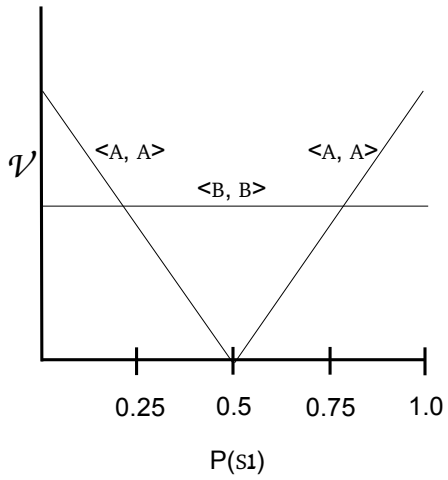


Figure 4.3. Alpha-vectors for a domain in which reasoning about average joint belief leads to poor team behavior.

In this example domain, the optimal joint policy for the agents when the belief is between P(S1) = 0.0 and P(S1) = 0.25 is $\langle \text{A}, \text{A} \rangle$. The same is true for when the belief is between P(S1) = 0.75 and P(S1) = 1.0. Between P(S1) = 0.25 and P(S1) = 0.75, the best joint action is $\langle \text{B}, \text{B} \rangle$. Suppose that the agents start with a synchronized joint belief at P(S1) = 0.5, and after the first timestep, their possible joint observations could lead them to be either at P(S1) = 0.1 or P(S1) = 0.9. At both of those beliefs, the optimal joint action choice would be $\langle \text{A}, \text{A} \rangle$. However, if the agents maintain only the average joint belief, P(S1) = 0.5, they would instead choose $\langle \text{B}, \text{B} \rangle$. In general, the possible joint beliefs may be multi-modal, making the average belief a particularly bad estimate.

We argue that better results are achieved when the agents model the *distribution of possible joint beliefs*. We introduce an algorithm called **A**voiding **C**oordination **E**rrors by reasoning about **P**ossible **J**oint **B**eliefs (ACE-PJB). For clarity, one can think of the distribution of possible joint beliefs as being represented by a tree, with each path through the tree representing a possible sequence of joint observations, or a *joint observation history*. We define $\mathcal{L}^t$, the set of leaves of the tree at depth t, to be the set of possible joint beliefs of the team at time t. Each $\mathcal{L}_i^t$ is a tuple consisting of $\langle b^t, p^t, \overrightarrow{\omega}^t \rangle$, where $\overrightarrow{\omega}^t$ is the joint observation history that would lead to $\mathcal{L}_i^t$, b^t is the joint belief given that observation history, and p^t is the probability of the team observing that history. The initial distribution of possible joint beliefs, $\mathcal{L}^0$, is composed of a single leaf at belief b^0, the starting belief of the team, with probability 1 and an empty observation history. For the tiger domain, this initial distribution $\mathcal{L}^0$ would be the leaf:

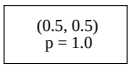


Table 4.1 presents the algorithm for expanding a single leaf in a tree of possible joint beliefs, given the joint action executed by the team. It takes as input $\mathcal{L}_i^t$, a single leaf in the distribution of possible joint beliefs at time t, and a joint action a. GROWTREE begins by initializing the resulting distribution, $\mathcal{L}^{t+1}$, to the empty set.

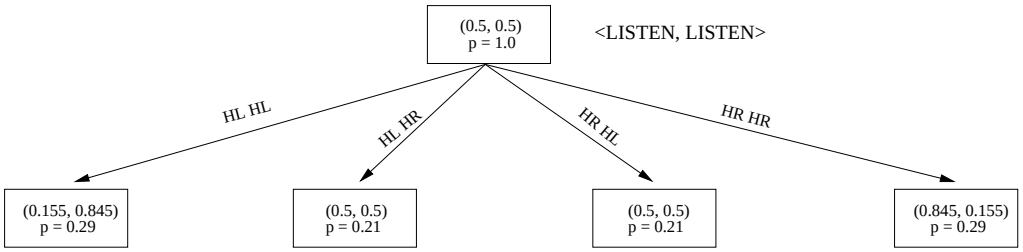
GROWTREE($\mathcal{L}_i^t$, a)

1. $\mathcal{L}^{t+1} \leftarrow \emptyset$
2. $b^t \leftarrow b(\mathcal{L}_i^t)$
3. **for** each $\omega \in \Omega$
4. $\quad b^{t+1} \leftarrow \emptyset$
5. $\quad p^{t+1} \leftarrow p(\mathcal{L}_i^t) \times Pr(\omega|a, b^t)$
6. $\quad$ **if** $p^{t+1} > 0$
7. $\quad\quad$ **for** each $s' \in \mathcal{S}$
8. $\quad\quad\quad b^{t+1}(s') \leftarrow \dfrac{\mathcal{O}(s', a, \omega)\sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s')b^t(s)}{Pr(\omega|a, b^t)}$
9. $\quad\quad \overrightarrow{\omega}^{t+1} \leftarrow \overrightarrow{\omega}(\mathcal{L}_i^t) \circ \langle \omega \rangle$
10. $\quad\quad \mathcal{L}^{t+1} \leftarrow \mathcal{L}^{t+1} \cup [b^{t+1}, p^{t+1}, \overrightarrow{\omega}^{t+1}]$
11. $\quad$ **return** $\mathcal{L}^{t+1}$

Table 4.1. Algorithm to grow the children of one leaf in a tree of possible beliefs

GROWTREE grows a child leaf for every possible joint observation ω. Lines 4-9 show the construction of a single child leaf. $Pr(\omega|a, b^t)$ is the probability of observing ω while in belief state b^t and having taken action a. It is used to calculate p^{t+1}, the likelihood of a complete observation history, beginning with the history stored in $\mathcal{L}_i^t$ and concluding with ω, as composed in Line 9. b^{t+1}, the belief that would result from observing ω, starting in belief b^t and taking action a, is calculated using a standard Bayesian update in lines 7 and 8 (Kaelbling et al., 1998). The child leaf, which is appended to the new distribution, $\mathcal{L}^{t+1}$, is composed of this new belief, b^{t+1}, the probability of reaching that belief (which is equivalent to the probability of receiving this particular observation in the parent leaf times the probability of reaching the parent leaf) and the corresponding observation history.

Suppose, for example, that the team chooses to execute the action $\langle$LISTEN, LISTEN$\rangle$. The agents must consider the likelihood of all four possible joint observations: $\langle$HL, HL$\rangle$, $\langle$HL, HR$\rangle$, $\langle$HR, HL$\rangle$, and $\langle$HR, HR$\rangle$. Therefore, they grow $\mathcal{L}^1$ to contain four leaves:



Note that $\mathcal{L}^1$ is independent of the actual observations of any of the agents. Nonetheless, the likelihood of each of the leaves is not the same. Because the observations are informative and depend on the true state, the two agents are more likely to hear the same observation than different ones. In the two-agent tiger domain, the tree of possible joint beliefs is symmetric. The likelihood of the agents observing $\langle \text{HEARLEFT}, \text{HEARLEFT} \rangle$ is equal to the likelihood of the agents observing $\langle \text{HEARRIGHT}, \text{HEARRIGHT} \rangle$.

4.3.2. Q-POMDP: Independent Selection of Joint Actions

The distribution of possible joint beliefs depends only on the initial joint belief of the team, the joint policy, and the sequence of joint actions that the team chooses to execute. Because the agents do not consider their local observation histories when computing the distribution of possible joint beliefs, each agent will independently compute a distribution identical to those computed by all of its teammates. Agents can therefore choose actions by reasoning over this distribution and, so long as the mechanism for choosing actions is deterministic, be assured that all agents will select the same joint action, thus avoiding coordination errors. The question, then, is how to effectively reason over the distribution of possible joint beliefs to select a joint action.

When there is a single leaf in the distribution of possible joint beliefs, it is clear that the best choice for the team is to execute the optimal joint action for that leaf's belief, as evaluated in the joint policy, $a^t \leftarrow \pi(b(\mathcal{L}_i^t))$. In the tiger domain, the optimal joint action at the belief $b^0(\text{SR}) = 0.5$ is $\langle \text{LISTEN}, \text{LISTEN} \rangle$. When there are many possible joint beliefs, action-selection can be accomplished by the application of any arbitrary function that deterministically maps $\mathcal{L}^t$ to a single joint action. Some simple heuristics include choosing the single most likely belief in $\mathcal{L}^t$ and executing the action associated with it in the joint policy, or a voting scheme in which possible joint beliefs vote for actions, with votes weighted by the likelihood of the belief (Simmons and Koenig, 1995). A particularly risk-averse team may wish to choose the joint action that maximizes worst-case reward over the possible joint beliefs.

In our work, we wish to maximize expected reward over the distribution of possible joint beliefs. To this end, we introduce the Q-POMDP heuristic. Q-POMDP is inspired by the Q-MDP heuristic (described in Section 2.3.1), a greedy algorithm that can be used to select actions for single-agent POMDPs (Littman et al., 1995). Just as Q-MDP finds approximate solutions for POMDPs by solving the underlying fully-observable MDPs, Q-POMDP approximates Dec-POMDPs using the underlying centralized POMDPs. The goal of the Q-POMDP heuristic is to find the joint action that maximizes *expected reward* over the distribution of possible joint beliefs. As input, Q-POMDP takes a centralized policy π, generated for the underlying centralized problem by a single-agent POMDP solver, and a value function $\mathcal{V}^{\pi}$ that, for a belief b, gives the expected reward of acting according to π at the current timestep and henceforth acting optimally.

Because, for a particular domain, there may be actions that are strictly dominated, $\mathcal{V}^{\pi}$ is not guaranteed to give a value for every action in every belief; it gives expected reward only for actions that are part of the policy generated by the POMDP solver. However, as Q-POMDP seeks to maximize expected reward over multiple possible belief states, the best action may not be part of π. To address this, Q-POMDP reasons over the one-step lookahead function $\mathcal{Q}$ that defines expected reward for taking an action a in the current timestep, at belief b^t, and acting according to π in all future timesteps ($b^{t+1}_{a,\omega}$ is the belief that results from taking action a from belief b^t and observing ω):

$$\mathcal{Q}(b^t, a) = \sum_{s \in \mathcal{S}} \mathcal{R}(s, a) b^t(s) + \gamma \sum_{\omega \in \Omega} Pr(\omega | a, b^t) \mathcal{V}^{\pi}(b^{t+1}_{a,\omega}) \tag{4.3}$$

The Q-POMDP heuristic chooses a single joint action that maximizes the expected value of $\mathcal{Q}$ over all the possible joint beliefs in $\mathcal{L}^t$:

$$\text{Q-POMDP}(\mathcal{L}^t) = \arg\max_{a \in \mathcal{A}} \sum_{\mathcal{L}^t_i \in \mathcal{L}^t} p(\mathcal{L}^t_i) \times \mathcal{Q}(b(\mathcal{L}^t_i), a) \tag{4.4}$$

For each possible joint action, Q-POMDP evaluates the expected reward of that action in each possible joint belief in $\mathcal{L}^t$, weighted by the likelihood of that possible belief. The notation Q-POMDP_a is used to refer to the expected value of one particular joint action a:

$$\text{Q-POMDP}_a(\mathcal{L}^t) = \sum_{\mathcal{L}^t_i \in \mathcal{L}^t} p(\mathcal{L}^t_i) \times \mathcal{Q}(b(\mathcal{L}^t_i), a) \tag{4.5}$$

Consider the distribution of possible joint beliefs, $\mathcal{L}^1$, after one timestep of the multi-agent tiger domain:

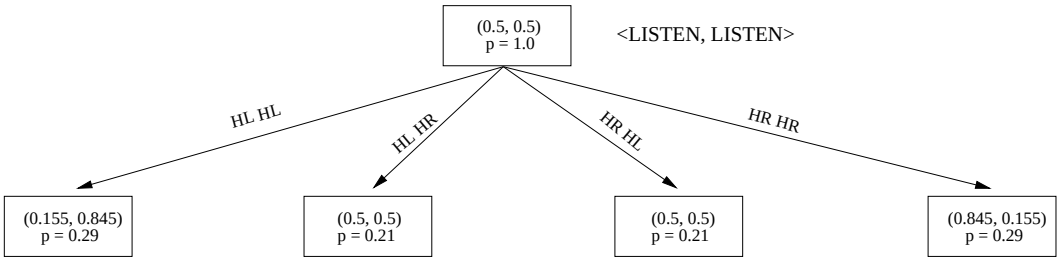


Table 4.2 shows the Q-POMDP$_a$ values calculated for all 9 possible joint actions in the tiger domain, over the $\mathcal{L}^1$ distribution. The action that maximizes expected reward over the possible joint beliefs is $\langle$LISTEN, LISTEN$\rangle$. Intuitively, it should be clear why this is the case. $\mathcal{L}^1$ represents all possible joint observation histories, and does not consider the local observation that has actually been received by the agent computing Q-POMDP. Therefore, there is no evidence, one way or the other, that would lead the team believe that the tiger is behind one particular door. Given this lack of information, the best course of action is to continue to LISTEN.

Action	Q-POMDP Value
$\langle$OPENL, OPENL$\rangle$	10.081
$\langle$OPENL, OPENR$\rangle$	-74.919
$\langle$OPENL, LISTEN$\rangle$	-20.919
$\langle$OPENR, OPENL$\rangle$	-74.919
$\langle$OPENR, OPENR$\rangle$	10.081
$\langle$OPENR, LISTEN$\rangle$	-20.919
$\langle$LISTEN, OPENL$\rangle$	-20.919
$\langle$LISTEN, OPENR$\rangle$	-20.919
$\langle$LISTEN, LISTEN$\rangle$	29.200

Table 4.2. Q-POMDP$_a(\mathcal{L}^1)$ values for all $a \in \mathcal{A}$ in the two-agent tiger domain.

Q-POMDP, like Q-MDP, is a greedy heuristic. Q-MDP asks the question, "What would be the best action in this belief if, after this timestep, the world were to become fully observable?" Q-POMDP asks a similar question: "What would be the best action in this timestep, given the current distribution of possible joint beliefs, if the teammate agents were to communicate their full observation histories in the next timestep and all future timesteps?" Therefore, just as Q-MDP finds no value in information-gathering actions that, while reducing an agent's uncertainty about the state of the world, do not contribute to an immediate increase in reward, Q-POMDP does not consider the true distributed nature of a Dec-POMDP and will not value actions that improve mutual knowledge among teammates.

Because it deterministically selects a single action that maximizes expected reward over $\mathcal{L}^t$, Q-POMDP is a suitable heuristic that enables a distributed team of agents to execute a centralized policy while avoiding coordination errors. The full ACE-PJB algorithm, shown in Table 4.3, combines GROWTREE and Q-POMDP to enable a team of agents to execute a centralized policy, without communication and while avoiding coordination errors. However, like any heuristic that does not take into account the local observations that are actually received by the agents, ACE-PJB is overly conservative and does not allow agents to react to new information that they gain about the state of the world over the course of execution. For example, we showed that, in the tiger domain, the agents will evaluate $\mathcal{L}^1$ and choose to LISTEN. In fact, if the agents continue to use ACE-PJB to make action choices, they will choose the joint action, $\langle$LISTEN, LISTEN$\rangle$ in *every* timestep and never choose to open a door.[3] The challenge, therefore, is to enable agents to utilize the local information that they observe without sacrificing team coordination.

ACE-PJB($\mathcal{L}^t$)

1. $a \leftarrow$ Q-POMDP($\mathcal{L}^t$)
2. execute a_i
3. $\mathcal{L}^{t+1} \leftarrow \emptyset$
4. **for** each $\mathcal{L} \in \mathcal{L}^t$
5. $\quad\quad \mathcal{L}^{t+1} \leftarrow \mathcal{L}^{t+1} \cup$ GROWTREE($\mathcal{L}, a$)
6. **return** $\mathcal{L}^{t+1}$

Table 4.3. Algorithm to execute a centralized policy while **A**voiding **C**oordination **E**rrors by reasoning about the distribution of **P**ossible **J**oint **B**eliefs.

4.4. ACE-PJB-COMM: Using Communication to Introduce Local Information

In this section, we discuss how communication can be used to integrate an agent's local observations into the shared information held collectively by the team so that it can be used by the teammate agents to make coordinated action decisions. Communication provides a means by which the local observations known only by the agent that received them can be shared with that agent's teammates. We discussed above the requirement that agents make action decisions based only on information that is known by all members of the team. Given this requirement, we utilize a *broadcast* communication protocol, in

[3] That the agents will never open a door can be explained as follows: The two-agent tiger domain has symmetric observations, meaning that agents are equally likely to observe HEARLEFT when the tiger is behind the left door as they are to observe HEARRIGHT when the tiger is behind the right door. The agents start out with a uniform belief over the position of the tiger, P(SL) = 0.5. The action that they choose in that belief, $\langle$LISTEN, LISTEN$\rangle$, does not change this belief. Because of the symmetry of the observations, GROWTREE also does not change the overall belief, which is that the tiger is equally likely to be behind the right and left doors.

which, when agents choose to send messages, those messages are delivered to all of their teammates. To effectively use communication to enhance decentralized execution, two questions must be addressed:

- What should prompt agents to communicate, or how should agents determine when communication will be useful?
- How can communicated information be integrated into the distribution of possible joint beliefs so that it can be used for decision-making?

The ACE-PJB-COMM algorithm provides agents with a heuristic for determining when communication will be beneficial to team performance, and gives a mechanism by which communicated information is integrated into the agents' model of possible joint beliefs. ACE-PJB-COMM specifies that an agent should communicate only when it sees that communication has the potential to change the joint action selected by Q-POMDP, and when the change in joint action is estimated to increase the expected reward by more than the cost of communication. Table 4.4 provides the details of the ACE-PJB-COMM algorithm.

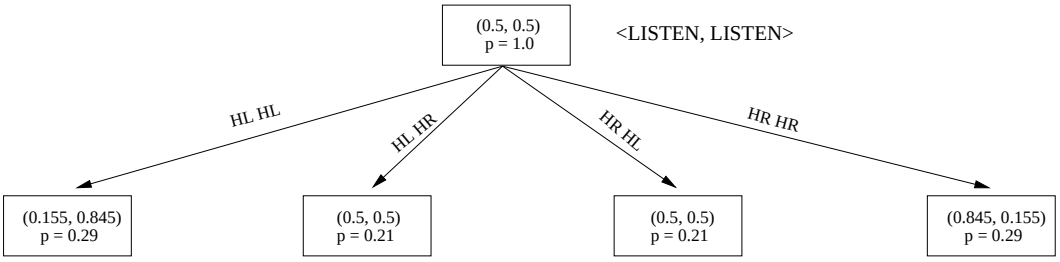
ACE-PJB-COMM($\mathcal{L}^t, \overrightarrow{\omega}_j^t$)

1. $a_{NC} \leftarrow$ Q-POMDP($\mathcal{L}^t$)
2. $\mathcal{L}' \leftarrow$ prune leaves inconsistent with $\overrightarrow{\omega}_j^t$ from $\mathcal{L}^t$
3. $a_C \leftarrow$ Q-POMDP($\mathcal{L}'$)
4. $v_C \leftarrow$ Q-POMDP$_{a_C}(\mathcal{L}')$
5. $v_{NC} \leftarrow$ Q-POMDP$_{a_{NC}}(\mathcal{L}')$
6. **if** $v_C - v_{NC} >$ communication cost
7. communicate $\overrightarrow{\omega}_j^t$ to the other agents
8. **return** ACE-PJB-COMM($\mathcal{L}', \emptyset$)
9. **else**
10. **if** communication $\overrightarrow{\omega}_k^t$ was received from another agent k
11. $\mathcal{L}^t \leftarrow$ prune leaves inconsistent with $\overrightarrow{\omega}_k^t$ from $\mathcal{L}^t$
12. **return** ACE-PJB-COMM($\mathcal{L}^t, \overrightarrow{\omega}_j^t$)
13. **else**
14. take action a_{NC}
15. receive observation ω_j^{t+1}
16. $\overrightarrow{\omega}_j^{t+1} \leftarrow \overrightarrow{\omega}_j^t \circ \langle \omega_j^{t+1} \rangle$
17. $\mathcal{L}^{t+1} \leftarrow \emptyset$
18. **for** each $\mathcal{L}_i^t \in \mathcal{L}^t$
19. $\mathcal{L}^{t+1} \leftarrow \mathcal{L}^{t+1} \cup$ GROWTREE($\mathcal{L}_i^t, a_{NC}$)
20. **return** $[\mathcal{L}^{t+1}, \overrightarrow{\omega}_j^{t+1}]$

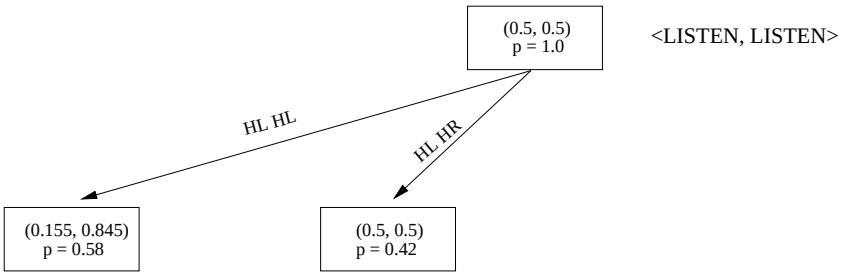
Table 4.4. One timestep of the ACE-PJB-COMM algorithm for an agent j

The inputs to ACE-PJB-COMM are $\mathcal{L}^t$, the distribution of possible joint beliefs at time t, and $\overrightarrow{\omega}_j^t$, all of the observations that agent j has not yet communicated. When deciding whether to communicate, an agent computes a_{NC} (line 1), the joint action selected by

Q-POMDP based on $\mathcal{L}^t$, the current distribution of possible joint beliefs. For example, as detailed above, $a_{NC} = \langle \text{LISTEN}, \text{LISTEN} \rangle$ for the following $\mathcal{L}^1$ distribution in the tiger domain:



In line 2 of the algorithm, the agent then prunes its distribution of possible joint beliefs, removing from $\mathcal{L}^t$ all possible beliefs that are inconsistent with its own local observation history. The resulting distribution, $\mathcal{L}'$, is the distribution of possible joint beliefs that would be held by the team if this agent were to choose to communicate. Suppose, in the tiger domain, that agent 1 observed HEARLEFT in the first timestep. Agent 1 would compute the pruned distribution, $\mathcal{L}'$, by removing all branches from $\mathcal{L}^1$ that are not consistent with that observation and re-normalizing the likelihoods of the leaves:

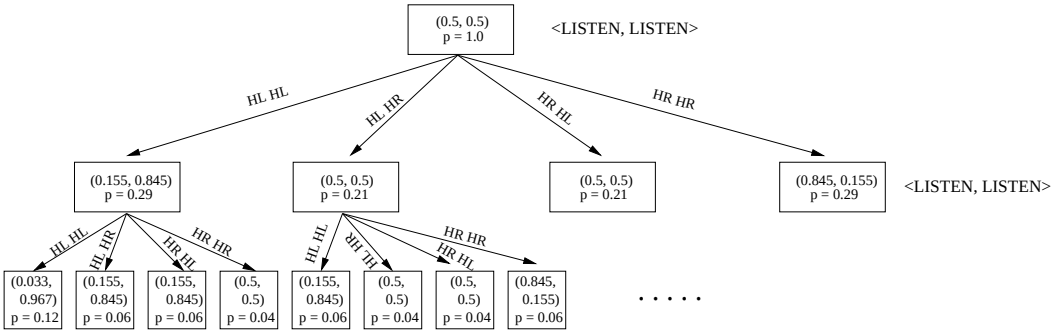


The action selected by Q-POMDP over the pruned distribution $\mathcal{L}'$, a_C in line 3 of the algorithm above, is the action that would be chosen by the team if the agent were to communicate its observation history to its teammates. If the actions differ, this indicates that communication will cause an increase in expected team reward. Lines 4-6 show how the amount of this increase can be computed as the difference between v_C, the Q-POMDP value of a_C, and v_{NC}, the value of a_{NC} computed over $\mathcal{L}'$ (e.g., Q-POMDP$_{a_{NC}}(\mathcal{L}')$).[4] If this increase in expected reward is greater than the cost of communication, the ACE-PJB-COMM heuristic instructs the agent to broadcast its observation history to all of its teammates and

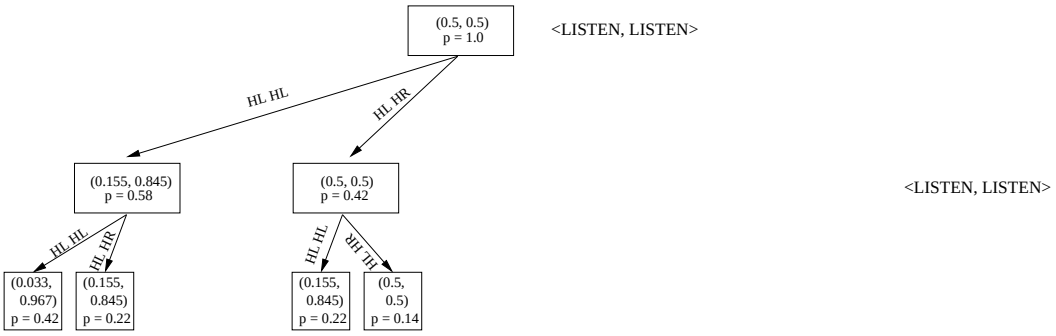
[4]Even though a_{NC} is the joint action that would be selected if the agents do not communicate, its expected value, v_{NC} is computed over the pruned distribution $\mathcal{L}'$. Regardless of whether it ultimately chooses to communicate, the agent doing the computation has received its own observation history and thereby gained additional knowledge about the true state of the world. To make accurate estimates about the relative values of possible joint actions, the agent must take this knowledge into account.

delete its own observation history(lines 7-8), since there will never be value in communicating those observations again.

The Q-POMDP action for the $\mathcal{L}'$ distribution shown above is $\langle$LISTEN, LISTEN$\rangle$, the same as the Q-POMDP action over the unpruned distribution, $\mathcal{L}^1$. Therefore, agent 1 would choose not to communicate in that timestep. Through symmetry, it is easy to see that agent 1 would not have chosen to communicate if it had observed HEARRIGHT instead of HEARLEFT, and that agent 2 will also choose not to communicate in this timestep. Therefore, the agents execute the joint action $\langle$LISTEN, LISTEN$\rangle$ and use GROWTREE to compute L^2, which has 16 leaves (lines 14-20):



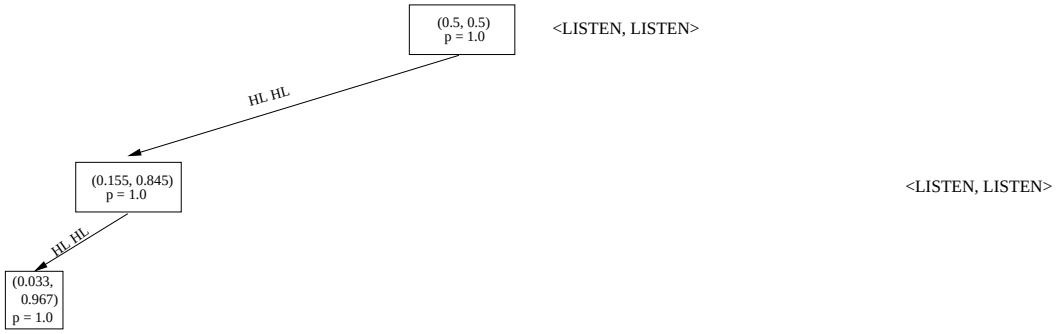
If, at this point, agent 1 were to observe HEARRIGHT, it would once again choose not to communicate. Let us, then, consider the case in which agent 1 observes HEARLEFT. Its observation history would now consist of hearing HEARLEFT twice in a row, and it would generate a pruned distribution $\mathcal{L}'$ as follows:



Now, Q-POMDP over this pruned distribution finds that the best joint action for the team is $\langle$OPENR, OPENR$\rangle$; there is sufficient evidence to believe that the tiger is behind the left door. Q-POMDP$_{\langle \text{OPENR,OPENR} \rangle}(\mathcal{L}') = 34.219$, whereas Q-POMDP$_{\langle \text{LISTEN,LISTEN} \rangle}(\mathcal{L}')$ = 32.758. As the difference in expected reward between these two joint actions is greater

than ϵ, the communication cost in the tiger domain, agent 1 would choose to communicate its observation history to agent 2 (Table 4.4, lines 7-8) .

Once it has communicated, agent 1's observation history is considered part of the shared team information, and can be used by the agents when choosing the next joint action. In the meantime, agent 2 had been simultaneously evaluating the ACE-PJB-COMM heuristic, and several things could happen, depending on the observation history of agent 2. One possibility is that agent 2 could have also observed two instances of HEARLEFT. In this case, agent 2's reasoning would be identical to the reasoning detailed for agent 1 above, and agent 2 would also broadcast its observation history. Agents that receive communication from their teammates prune their distributions of possible joint beliefs to be consistent with the communicated observation history (Table 4.4, lines 10-11). When both agents prune their distributions of possible joint beliefs to include the communicated observations, the resulting distribution would contain a single belief, the belief corresponding to the team observing two instances of $\langle$HEARLEFT, HEARLEFT$\rangle$:



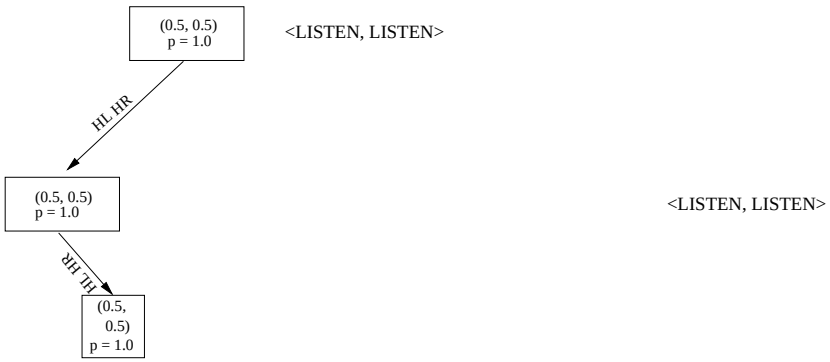
Because receiving new information may prompt an agent to decide to communicate its own observation history, there may be multiple instances of communication in each timestep.[5] Therefore, agents must wait a fixed period of time to allow the system to quiesce before acting. This is indicated in the algorithm above by recursive calls to ACE-PJB-COMM (lines 8, 12). As agents communicate all of their new local observations at each instance of communication, each agent will communicate at most once per timestep. Therefore, ACE-PJB-COMM is guaranteed to converge to a single joint action.

Finally, when no further communication will take place, the agents execute a_{NC}, the Q-POMDP action chosen over the distribution of possible joint beliefs that is consistent

[5]Consider a variant of the tiger domain in which the prior probability of SL is 0.9. In that case, the default action is OPENL and an agent observing HEARLEFT in the first timestep would choose not to communicate. However, if one of its teammates were to communicate an observation of HEARRIGHT, changing the joint action to OPENR or LISTEN, the first agent could discover that it is in the team's interest for it to contradict its teammate and report its HEARLEFT observation.

with all communication acts that have taken place. In the case of the current example, the agents would proceed to open the right door. Each agent receives a new individual observation and grows its observation history and distribution of possible joint beliefs (lines 14-20).

On the other hand, if agent 2 heard one instance of HEARLEFT and one instance of HEARRIGHT, it would not communicate, and the team would open the right door based on the communicated observation history of agent 1. If, however, agent 2 heard two instances of HEARRIGHT, it *would* communicate its observation history, and the team would resolve to continue to LISTEN, based on the pruned observation history below, in which an equal number of HEARLEFT and HEARRIGHT observations have been observed by the team:



4.5. Results

We performed experiments to compare the performance of the ACE-PJB-COMM algorithm to a team executing with full communication and teams making communication decisions stochastically in three domains: the two-agent tiger domain, a tiger domain with three agents, and the multi-access broadcast channel (MABC) domain described by Hansen *et al.* (Hansen et al., 2004). For the stochastic team, we allowed each agent on the team to communicate at a frequency of communication slightly higher than the frequency of communication determined by the ACE-PJB-COMM algorithm. The agents then used Q-POMDP to select their joint action.

4.5.1. Two-agent Tiger Domain

We compared the average discounted reward accumulated by a team of two agents operating in the tiger domain using the ACE-PJB-COMM algorithm to choose actions and make communication decisions to the reward accumulated by a team executing a centralized policy by communicating at every timestep and a team that makes communication

decisions randomly, at approximately the same frequency of communication as ACE-PJB-COMM. We ran 20,000 trials for each algorithm. In each trial, the world state was initialized randomly, and the agents executed for 6 timesteps. Table 4.5 summarizes the results of these trials. μ_{Reward} is the mean discounted reward accumulated over 6 timesteps of execution and $\mu_{Messages}$ is the mean number of messages sent by the team over the course of each trial. When each agent on a team communicates at every timestep, each message consists of exactly one observation. However, the ACE-PJB-COMM algorithm directs agents to send their entire previously uncommunicated observation histories to their teammates when they choose to communicate, so a single message may contain several observations. $\mu_{Observations}$ is the mean number of observations communicated by both teammates per trial. σ is the standard deviation for each value.

	μ_{Reward} (σ)	$\mu_{Messages}$ (σ)	$\mu_{Observations}$ (σ)
Full Communication	7.14 (27.88)	10.0 (0.0)	10.0 (0.0)
ACE-PJB-COMM	5.31 (19.79)	1.77 (0.79)	5.13 (2.38)
Random	-2.18 (14.41)	2.31 (1.51)	5.12 (3.06)

Table 4.5. Summary of results for 20,000 6-timestep trials of the two-agent tiger domain. μ_{Reward} is the mean discounted reward accumulated over 6 timesteps. $\mu_{Messages}$ is the mean number of distinct communication instances per trial, and $\mu_{Observations}$ is the mean number of observations communicated by both agents per trial.

As expected, a team choosing actions using ACE-PJB-COMM accumulates less reward than a centralized team that communicates at every timestep. This difference in performance can be attributed to ACE-PJB-COMM's conservatism. Whereas a team with full communication could potentially open a door after only one observation (e.g., if the joint observation is HLHL), an agent executing according to DECCOMM must hear two consistent observations before it communicates to its teammate. This costs the team an additional $\langle$LISTEN, LISTEN$\rangle$ action, with reward -2. Note that ACE-PJB-COMM execution has a significantly lower variance than executing with full communication. A team with full communication may choose to open a door after only one timestep. Because a team using ACE-PJB-COMM must LISTEN at least twice before opening a door, it is more conservative and therefore less likely to open an incorrect door.

ACE-PJB-COMM's performance is achieved with a significant savings in communication. Whereas a centralized team must communicate a total of 10 observations over the

course of 6 timesteps of execution, ACE-PJB-COMM instructs the agents to communicate slightly more than half as many. Generally, only one agent must communicate in order to direct the team to open a door.[6] Additional instances of communication are needed only when the other agent's observations contradict the information communicated by its teammate (e.g., if one agent observes two instances of HEARLEFT while the other agent observes two instances of HEARRIGHT). Overall, a team using ACE-PJB-COMM communicates less than 20% of the number of messages communicated by a team with full communication. On average, each message contains 2.9 observations.

On average, agents using ACE-PJB-COMM communicated in 17.7% of the timesteps in which they had non-empty observation histories. To demonstrate that communication was used effectively, we compared the performance of a team using ACE-PJB-COMM to a team making random communication decisions. In each timestep, agents chose to communicate with probability 0.2. Although this team communicated slightly more messages than the team that used ACE-PJB-COMM, the messages were significantly less informative, as indicated by the low reward, -2.18, achieved by the random team.

4.5.2. Three-agent Tiger Domain

To show that our work is applicable in domains of more than two agents, we performed experiments in a three-agent tiger domain. This domain has the same reward structure as the two-agent tiger domain. The overall goal is for all three agents to simultaneously open the door opposite the tiger, and the worst coordination error takes place when at least one agent opens each door. We adjusted the observation model so that, like in the two-agent case, a simultaneous observation of, for example, HEARLEFT by all of the agents is sufficient to direct the team to open the right door, whereas two agents observing HEARLEFT while the third agent observed HEARRIGHT would be insufficient. The likelihood of an agent correctly observing the position of the tiger is 0.65. Table 4.6 summarizes the results of 20,000 trials, each 6 timesteps in length, of three agents operating by using the ACE-PJB-COMM algorithm to choose their actions, as a centralized team with full communication, or as a team making communication decisions randomly.

Overall, the total reward in the three-agent tiger domain is lower than reward in the two-agent tiger domain. Both centralized teams and teams choosing actions with ACE-PJB-COMM take longer to become sufficiently certain about the location of the tiger to

[6] In our experiments, we found that, of the timesteps in which at least one agent chose to communicate, its teammate also communicated 27% of the time.

	μ_{Reward} (σ)	$\mu_{Messages}$ (σ)	$\mu_{Observations}$ (σ)
Full Communication	-3.00 (22.66)	15 (0.0)	15 (0.0)
ACE-PJB-COMM	-4.93 (13.76)	1.78 (1.72)	5.01 (4.46)
Random	-7.07 (12.23)	2.83 (1.83)	6.69 (4.05)

Table 4.6. Summary of results for 20,000 6-timestep trials of the three-agent tiger domain. μ_{Reward} is the mean discounted reward accumulated over 6 timesteps. $\mu_{Messages}$ is the mean number of distinct communication instances per trial, and $\mu_{Observations}$ is the mean number of observations communicated by both agents per trial.

open a door. However, as in the two-agent tiger domain, the difference in reward between a centralized team and a decentralized team executing ACE-PJB-COMM is small, corresponding to an additional timestep needed by ACE-PJB-COMM to choose to communicate and open a door. ACE-PJB-COMM again requires substantially less communication than centralized execution, in this case approximately one third the number of observations and 11.87% of the number of messages. In this domain, ACE-PJB-COMM makes significantly more effective communication decisions than a team choosing to communicate at random. The team communicating with probability 0.15 at each timestep achieved an average discounted reward of only -7.07, much less than the reward achieved by the team using ACE-PJB-COMM, in which agents communicate approximately 11.9% of the time.

4.5.3. Multi-access Broadcast Channel Domain

The multi-access broadcast channel domain (MABC) (Ooi and Wornell, 1996) has become a common Dec-POMDP benchmark domain (Hansen et al., 2004; Seuken and Zilberstein, 2007). This problem, which consists of 4 states, 4 joint actions, and 4 joint observations, involves two agents attempting to send messages across the same channel. Agents receive a reward of +1 for every message successfully sent. A collision occurs if both agents attempt to broadcast simultaneously. The agents noisily observe collisions. Complete details of the MABC domain can be found in Appendix A.2.

Although it may be nonsensical to consider the presence of out-of-band communication in this domain, we nevertheless wished to demonstrate the performance of ACE-PJB-COMM on this common benchmark problem. Table 4.7 summarize the results of 20,000 trials, each 10 timesteps long. There is no significant difference between the discounted

reward accumulated by a team that communicates every timestep and a team executing ACE-PJB-COMM, which broadcasts only 52.3% the number of observations and 42.3% the number of messages.

Looking at reward, ACE-PJB-COMM in the MABC domain does not seem to suffer from the same conservatism as in the Tiger domain. Whereas, with full communication, a team executing in the tiger domain may choose to open a door after the first observation, in the MABC domain, agents only observe collisions after attempting to send a message. Because there are no false positive observations in the MABC domain, only false negatives, an agent observing a collision becomes certain about the state of the world and chooses to communicate immediately.

	μ_{Reward} (σ)	$\mu_{Messages}$ (σ)	$\mu_{Observations}$ (σ)
Full Communication	5.23 (0.71)	18.0 (0.0)	18.0 (0.0)
ACE-PJB-COMM	5.23 (0.71)	7.62 (1.30)	9.46 (1.79)
Random	5.26 (0.75)	7.74 (2.47)	15.35 (2.76)

Table 4.7. Summary of results for 20,000 10-timestep trials of the multi-access broadcast channel (MABC) domain. μ_{Reward} is the mean discounted reward accumulated over 10 timesteps. $\mu_{Messages}$ is the mean number of distinct communication instances per trial, and $\mu_{Observations}$ is the mean number of observations communicated by both agents per trial.

4.6. Particle Filter Representation

The distribution of possible joint beliefs, represented in the previous sections as a tree, grows rapidly over time. In fact, this growth is exponential in $|\Omega|$, the number of possible joint observations. When agents communicate frequently, the number of beliefs in the distribution may remain fairly constant, as each instance of communication reduces the distribution of possible joint beliefs to contain only those consistent with the communicated observation histories. However, there may be cases in which agents choose not to communicate for many timesteps.

In the tiger domain, consider the case in which each agent receives alternating observations of HEARLEFT and HEARRIGHT. No agent would ever become sure enough of the position of the tiger to make communication worthwhile. The number of possible joint beliefs in $\mathcal{L}^t$ would rapidly grow beyond what is feasible to store in memory. To address cases in which agents may not communicate for long periods of time, we utilize a fixed-size method for modeling the distribution of possible joint beliefs using a particle filter.

In our particle filter representation, each particle $\mathcal{L}_i^t$ in the distribution $\mathcal{L}^t$ stores a single possible joint observation history, $\overrightarrow{\omega}^t$. Each particle also stores b, the joint belief that would result from taking the actions that the team has chosen and observing the stored joint observation history. The likelihood of a particular belief is indicated by the frequency of occurrence of the particle representing that belief in the particle filter. Each agent stores a particle filter $\mathcal{L}^t$, which represents the joint possible beliefs of the team. At each timestep, the beliefs represented in the particle filter are propagated forward according to the algorithm presented in (Thrun, 2000).

Table 4.8 gives the details of propagating $\mathcal{L}^t$ after taking the joint action a. Its inputs are the current distribution of possible joint beliefs, $\mathcal{L}^t$, and the current joint action, a. Lines 3-12 show the procedure for updating a single particle in the particle filter. First, a single possible joint belief $\mathcal{L}_i^t$ is chosen at random from $\mathcal{L}^t$.[7] b^t is the belief represented by that particle. In lines 5-6, the belief is partially updated, given the joint action taken at time t. Then, in lines 8-12, a new particle is generated for each possible joint observation ω. First, p, the probability of observing ω given the partially updated belief, is calculated. ω is appended to the observation history of the original particle. Lines 10 and 11 show the remainder of the belief Bayes update, in which b' is updated to include the observation of ω. Overall, N $\times|\Omega|$ new particles are generated. From these, N particles are sampled according to their relative likelihoods, p.

The ACE-PJB-COMM algorithm is executed as described in Section 4.4. However, a complication arises when it comes time to prune the particle filters, either to make communication decisions or as a result of observations received through communication. Unlike the tree described earlier that represents the distribution of possible joint beliefs exactly, a particle filter only approximates the distribution. Simply removing those particles not consistent with the communicated observation history and resampling (to keep the total number of particles constant) may result in a significant loss of information. Over time, the number of possible joint observation histories grows exponentially, and many low-probability histories will, because of sampling, not appear in the particle filter. In the worst case, there may not be *any* particles in the particle filter with the exact individual observation history that is motivating pruning.

We provide a means by which agents can recreate observation histories that have been lost due to sampling. Even if the exact observation history driving pruning has been sampled out of the particle filter, it is still possible to identify *similar* observation histories that would lead to similar beliefs. Looking at the tiger domain example above, it is

[7] To ensure coordination, the agents must have synchronized random number generators.

PROPAGATEBELIEFS($\mathcal{L}^t$, a)

1. $\mathcal{L}^{t+1} \leftarrow \emptyset$
2. do N times:
3. draw random $\mathcal{L}_i^t$ from $\mathcal{L}^t$
4. $b^t \leftarrow b(\mathcal{L}_i^t)$
5. **for** each $s' \in \mathcal{S}$:
6. $b'(s') \leftarrow \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s') b^t(s)$
7. **for** each $\omega \in \Omega$:
8. $p \leftarrow \sum_{s \in \mathcal{S}} \mathcal{O}(s, a, \omega) b'(s)$
9. $\overrightarrow{\omega}^{t+1} \leftarrow \overrightarrow{\omega}^t \circ \langle \omega \rangle$
10. **for** each $s' \in \mathcal{S}$:
11. $b'(s') \leftarrow \mathcal{O}(s', a, \omega) \sum_{s \in \mathcal{S}} b^t(s)$
12. $\mathcal{L}^{t+1} \leftarrow \mathcal{L}^{t+1} \cup [\overrightarrow{\omega}^{t+1}, b', p]$
13. normalize $\mathcal{L}^{t+1}$ so that $\sum_{\mathcal{L}_i^{t+1} \in \mathcal{L}^{t+1}} p(\mathcal{L}_i^{t+1}) = 1$
14. $\mathcal{L}^{t+1} \leftarrow$ sample N particles from $\mathcal{L}^{t+1}$ according to p
15. **return** $\mathcal{L}^{t+1}$

Table 4.8. Algorithm to propagate forward the beliefs in a particle filter representation.

easy to see that there is a correlation between the observation histories of the different agents (e.g. if one agent observes $\langle \text{HL}, \text{HL} \rangle$, it is unlikely that the other agent will have observed $\langle \text{HR}, \text{HR} \rangle$). To capture this correlation when pruning, we define a similarity metric between two observation histories (see Table 4.9). When an observation history $\overrightarrow{\omega}_i^t$ is communicated by agent i, the new distribution of possible joint beliefs, $\mathcal{L}_{joint}$, is computed by comparing the observation history in each particle corresponding to agent i ($\overrightarrow{\omega}^t = \overrightarrow{\omega}_i^t(\mathcal{L})_{\forall \mathcal{L} \in \mathcal{L}_{joint}}$) to the communicated observation history, $\overrightarrow{\omega}_i^t$. The comparison asks the question, "Suppose an agent has observed $\overrightarrow{\omega}_i^t$ after starting in belief b^0 and knowing that the team has taken the joint action history $\overrightarrow{a}^t$. What is the likelihood that an identical agent would have observed the observation history $\overrightarrow{\omega}^t$?" The value returned by this comparison is used as a weight for the particle. The particles are then resampled according to the calculated weights, and the agent i observation history for each particle is replaced with $\overrightarrow{\omega}_i^t$. In calculating similarity, we assume that each agent has stored a history of joint actions taken by the team through time $t-1$, $\overrightarrow{a}$.

Overall, team performance increases with the number of particles used to model the distribution of possible joint beliefs, asymptotically approaching the reward achieved by a team that models the distribution exactly, using a tree representation. Figure 4.4 shows the mean discounted reward (with 95% confidence intervals) for 20000 trials of the two-agent

SIMILARITY($\overrightarrow{\omega}_i^t, \overrightarrow{\omega}^t, \overrightarrow{a}^t$)

1. $sim \leftarrow 1.0$
2. $b \leftarrow b^0$
3. **for** $t' = 1 \ldots t$
4. $\quad b' \leftarrow b$
5. $\quad$ **for** each $s' \in \mathcal{S}$
6. $\quad\quad b(s') \leftarrow \frac{\mathcal{O}(s', a^{t'}, \omega_i^{t'}) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a^{t'}, s') b'(s)}{Pr(\omega_i^{t'} | a^{t'}, b')}$
7. $\quad sim \leftarrow sim \times \sum_{s \in \mathcal{S}} \mathcal{O}(s, a^{t'}, \omega^{t'}) b(s)$
8. **return** sim

Table 4.9. The heuristic used to determine the similarity between two observation histories.

tiger problem, using different numbers of particles to model the distribution of possible joint beliefs. Each trial is 6 timesteps long.

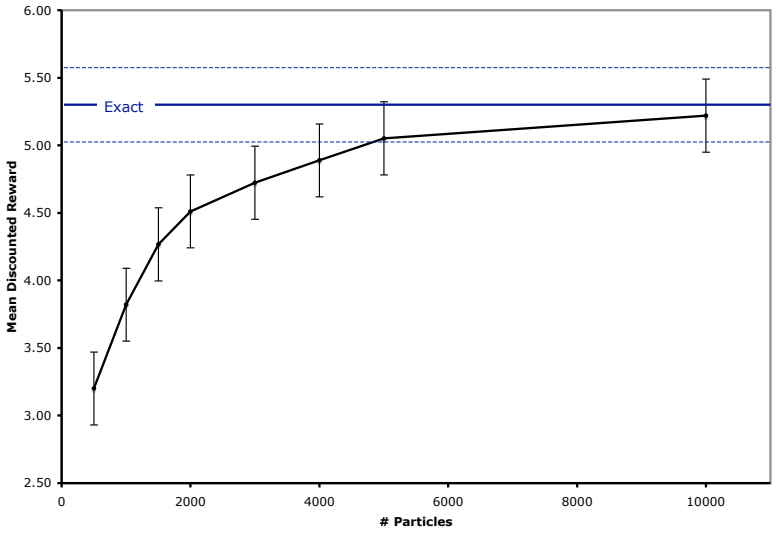


Figure 4.4. Mean discounted reward (with 95% confidence intervals) accumulated over 6 timesteps of the two-agent tiger domain. Teams model the distribution of possible joint beliefs using particle filters ranging from 500 to 10000 particles. The line labeled "Exact" shows the mean reward (with 95% confidence intervals) achieved by a team modeling possible joint beliefs exactly.

A team that models the distribution of possible joint beliefs using a particle filter with an insufficient number of particles performs worse than a team that models possible joint

beliefs exactly because the Q-POMDP values computed by the agents do not accurately reflect the expected outcomes of the actions that the agents consider. Consider the histogram in Figure 4.5, which compares a team modeling the distribution of possible joint beliefs as a tree to a team that uses a particle filter with 500 particles. This histogram shows the frequency of occurrence of different discounted rewards over 20000 trials of the tiger problem.

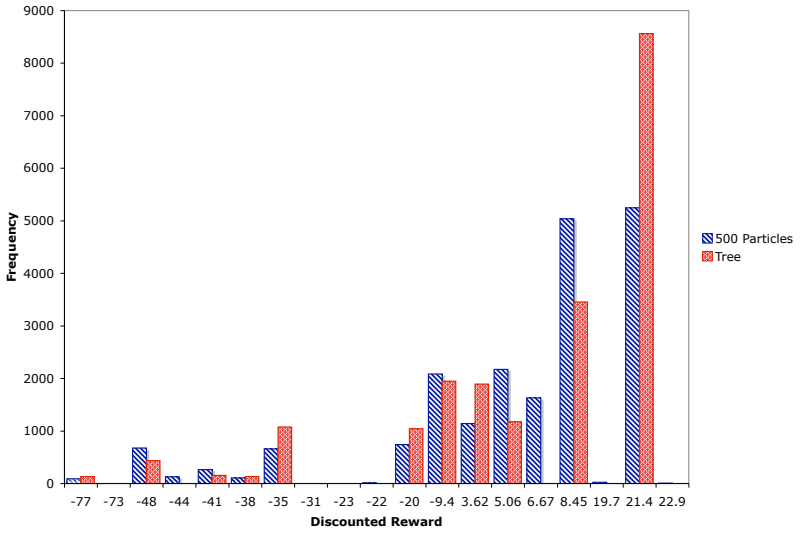


Figure 4.5. Histogram of discounted rewards accumulated by a team of agents executing for 6 timesteps in the two-agent tiger domain. The rewards shown in red stripes were received by a team that models the distribution of possible joint beliefs exactly, as a tree, and the rewards in blue checks were received by a team modeling possible joint beliefs using a 500-particle particle filter.

It is easy to see that the performance of the two teams is significantly different. For instance, the most common discounted reward accumulated by a team over 6 timesteps, both for a team using an exact representation of possible joint beliefs and for a team using a particle filter, is 21.4396. This corresponds to the team listening twice, opening the correct door, listening twice more, and then opening the correct door again, and is the best possible reward that can be achieved by a team executing ACE-PJB-COMM correctly. The particle filter team is unable to open the correct door as frequently, and therefore performs worse than the team modeling the possible joint beliefs exactly.

Additionally, if an agent is modeling the distribution of possible joint beliefs exactly, it will never decide to communicate after hearing only one possible observation. The likelihood that its teammate heard a contradicting observation is too high to change the Q-POMDP action. However, if there are insufficient particles to accurately represent the distribution of possible joint beliefs, an unlucky episode of resampling can slightly bias the particles toward beliefs at the extremes of the distribution. This additional, and unwarranted, weight on beliefs such as the agents both hearing the tiger behind the same door can lead, in rare instances, to the team opening a door after the agents hear only one observation each. Although this can lead to higher rewards (e.g., 22.88302 if the team listens twice, opens a door, and then listens only once before opening the correct door, or 19.6576 if the team listens 3 times, opens a door, listens once and then opens another door), it can also cause the team to mistakenly open the wrong door more often than a team that models the distribution of possible joint beliefs correctly.

Contrast the histograms in Figure 4.5 with those in Figure 4.6, which compare the discounted rewards accumulated by a team using a tree to model belief to a team using a 5000-particle filter. Clearly, the performance of a team using 5000 particles to model the distribution of possible joint beliefs is much more similar to the performance of a team using a tree than a team that uses only 500 particles.

Although using more particles better approximates the true distribution of possible joint beliefs and thereby leads to better team performance, it also requires more computation and slows down execution. Therefore, in each domain, one must determine experimentally the trade-off between reward and execution time and choose the size of the particle filter accordingly. The graph in Figure 4.4 suggests that 5000 particles may be a good size for a particle filter modeling the two-agent tiger domain, as performance approaches the margin of error for an exact model, and overall reward increases slowly with additional particles.

4.7. Discussion

In the previous section, we introduced a particle filter representation that models the distribution of possible joint beliefs of a team of agents in constant space. However, we also showed that the performance of a team using this representation depends heavily on the number of particles used. In our representation, each particle is a joint observation history. Over time, as the number of possible observation histories grows, a particle filter with few particles will capture a less accurate representation of the distribution of possible joint beliefs and lead to increasingly poor team behavior. It is worth noting that different

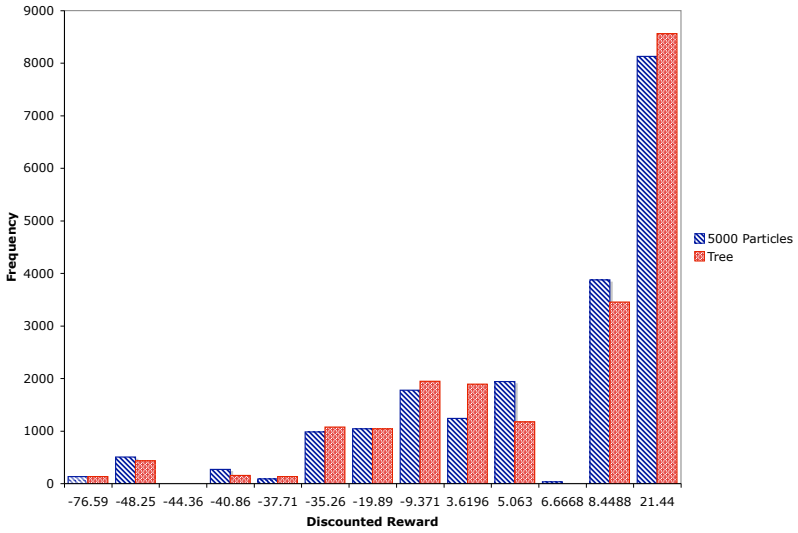


Figure 4.6. Histogram of discounted rewards accumulated by a team of agents executing for 6 timesteps in the two-agent tiger domain. The rewards shown in red stripes were received by a team that models the distribution of possible joint beliefs exactly, as a tree, and the rewards in blue checks were received by a team modeling possible joint beliefs using a 5000-particle particle filter.

joint observation histories may lead to identical joint beliefs. For example, in the two-agent tiger domain, the joint observations $\langle \mathrm{HL}, \mathrm{HR} \rangle$ and $\langle \mathrm{HR}, \mathrm{HL} \rangle$ both result in the joint belief P(SL) = 0.5. In the most extreme case, when the agents choose to open a door, the problem resets, returning the team to the initial belief distribution. At this point, the team can be said to have a *synchronized belief state* (Nair et al., 2004). The observation history leading to this state is irrelevant to the team's future decisions and can be discarded.

In our particle filter model, we chose to make each particle refer to an observation history rather than a belief. If, instead, we chose to model beliefs and ignore the possibility that a single belief could be the result of several different observation histories, it would be much more difficult to prune the particle filter when considering the impact of communication. When deciding whether to keep a particle, we need to know if it is consistent with the communicated observation history. Therefore, in this work, we were unable to take advantage of the potentially more compact representation of possible joint beliefs provided by modeling beliefs rather than observation histories. The problem of more compactly modeling the distribution of possible joint beliefs therefore remains open.

Another avenue for further research is the problem of communication leading to a biased estimate of belief. When using the ACE-PJB-COMM algorithm to make communication decisions, each agent chooses to communicate only when it anticipates that communication of its own local observation history will *change* the joint action of the team. Agents will not choose to communicate observations that confirm their current belief about the world, as these observations will not change the joint action. Instead, agents that observe unexpected, meaning unlikely, events will choose to communicate. Their teammates, who observed contradicting observations, will respond only if, individually, they believe that they have sufficient evidence to change the joint action again. Therefore, one agent's observation and communication of an unlikely event may bias the team's overall belief about the world state. In this chapter, we presented several domains in which agents using ACE-PJB-COMM easily outperformed teams that made communication decisions randomly. However, it is possible to construct domains in which this is not the case. Identifying such domains, and exploring methods to avoid introducing bias into the team's belief via communication remains an open and important problem for future research.

4.8. Summary

Two main ideas were addressed in this chapter. First, we showed how a decentralized team can execute a centralized policy, without communication and avoiding coordination errors, by modeling and reasoning over the possible joint beliefs of the team. We presented an algorithm for calculating the distribution of possible joint beliefs, and introduced a heuristic, Q-POMDP, by which a team of agents can select the joint action that maximizes expected reward over this distribution. We demonstrated that a team that chooses actions based on the distribution of possible joint beliefs is guaranteed to avoid mis-coordination.

Secondly, we showed how communication can be used to improve decentralized execution of a centralized joint policy. The ACE-PJB-COMM algorithm described in this chapter answers the question of **when** a team of agents should communicate. We illustrated the ACE-PJB-COMM algorithm through an extended example in the two-agent tiger domain. We demonstrated the performance of the ACE-PJB-COMM heuristic experimentally in several benchmark domains, comparing it to the performance of teams with free communication. We then showed how a particle filter can be used, in conjunction with ACE-PJB-COMM, to model the distribution of possible joint beliefs of a team using constant space.

CHAPTER 5

SELECTIVE ACE-PJB-COMM: Choosing *Which Observations* to Communicate

THE ACE-PJB-COMM algorithm presented in the previous chapter answers the question of **when** agents should communicate. It does so by directing agents to share their observations with their teammates whenever they find that integrating their local knowledge into the team's distribution of possible joint beliefs will lead to an increase in expected reward greater than the cost of communication. One limitation of ACE-PJB-COMM is that it does not address the question of **what** to communicate. Built into the ACE-PJB-COMM algorithm is the assumption that, when an agent does choose to communicate, it broadcasts all of its observations that have not previously been communicated. This assumption reduces the execution-time computational burden on the agents, as the agents do not need to reason about how their choice of message will affect the team. However, this requirement may often cause the agents to communicate redundant observations.

Consider the two-agent tiger domain discussed in the previous chapter. One property of the domain is that the world resets each time a door is opened, and the tiger is re-located at random behind either of the doors. Observations that were received by the team before they took the OPEN action, therefore, provide no information about the current position of the tiger. Communicating them can add nothing to the team's estimate of the world state. Additionally, $\mathcal{O}$, the tiger problem's observation function, is symmetric with respect to the position of the tiger, meaning, for example, that the probability of correctly observing HEARLEFT when the state is SL is equal to the probability of observing HEARRIGHT when the state is SR. Thus the observations in a history in which an agent first observed HEARLEFT and then HEARRIGHT cancel each other out, resulting in no net change from the initial belief of $b(\text{SR}) = 0.5$. If the agent were to decide to communicate in a subsequent timestep, there would be no advantage in its broadcasting these first two observations.

Making effective use of communication resources requires agents to reason about **what** to communicate, as well as **when**. In Section 5.1, we introduce a new domain, the *Colorado/Wyoming domain*, that is useful for examining the question of **what** to communicate in the presence of limited communication. Section 5.2 of this chapter discusses several paradigms under which limitations on communication can be considered. Section 5.3 then describes an algorithm, SELECTIVE ACE-PJB-COMM, by which decisions about **what** to communicate can be made under the various communication paradigms. This algorithm makes use of a heuristic, BUILDMESSAGE, that iteratively selects the most valuable observations from an observation history. BUILDMESSAGE and SELECTIVE ACE-PJB-COMM are illustrated via an example in the Colorado/Wyoming domain. Section 5.4 presents experimental results demonstrating the performance of SELECTIVE ACE-PJB-COMM in both the Colorado/Wyoming and two-agent tiger domains.

5.1. Colorado/Wyoming Domain

The multi-agent tiger domain, described in the previous chapter, is useful for evaluating communication strategies, in that it encodes a challenging coordination problem in which communication can assist agents in acting jointly to improve expected team reward, it is missing other characteristics that are necessary to illustrate the full range of communication decisions. In particular, the tiger domain has only two possible individual observations. While the tiger domain clearly provides an example of a domain in which observations become redundant over time, as described above, it is less useful for evaluating the question of which observations are the most informative. HEARLEFT and HEARRIGHT are symmetric with respect to the underlying world state; they encode the same amount of information. Therefore, we introduce a new domain, which we call the *Colorado/Wyoming problem*, which has multiple different observations that provide varying qualities of information. Additionally, the Colorado/Wyoming domain consists of many states. As we will show in Section 5.3.2, the agents' decisions about **what** to communicate depend on their state.

In the Colorado/Wyoming domain, two agents start at random locations in one of two possible 5x5 grid worlds, representing the states of Colorado and Wyoming, and must meet in a predetermined location. If they are in Colorado, the agents' goal is to meet in Denver, at grid position (2,4). If the agents are in Wyoming, they must rendezvous in Cheyenne, located at grid position (5,5) (see Figure 5.1). The agents have 6 individual actions, for a total of 36 possible joint actions: Each agent can move NORTH, SOUTH, EAST, or WEST, with each move succeeding deterministically and incurring a cost of -1. An agent can also

STOP, with cost 0, or send up a SIGNAL. At each timestep, the agents are aware of both their own position and the position of their teammate. Like the multi-agent tiger domain, the Colorado/Wyoming problem contains an explicit coordination problem. If both agents are at the correct goal location when they simultaneously send up a SIGNAL, they receive a joint reward of +20. If they send up simultaneous SIGNALs from an incorrect location, they receive a reward of -50. However, if only one agent SIGNALs, or if the agents signal in different locations, the team incurs a penalty of -100.

Figure 5.1. Figure (a) is one possible configuration of the two agents in Colorado, with the goal, Denver, at (2,4). Figure (b) is one possible configuration of the two agents in Wyoming, with Cheyenne located at (5,5).

In order to progress toward the correct goal location, the agents must observe their environment. Both Colorado and Wyoming contain some flat and some mountainous regions. However, as Colorado is more mountainous than Wyoming, the probability that an agent will observe a MOUNTAIN in Colorado is higher than the probability of observing it in Wyoming (e.g., P(ω = MOUNTAIN | COLORADO) = 0.5 vs P(ω = MOUNTAIN | WYOMING) = 0.3). Likewise, the observation PLAIN is more probable in Wyoming. Additionally, Colorado and Wyoming each contain a distinctive tourist attraction. It is somewhat likely that an agent will see a sign for PIKESPEAK in Colorado or a sign indicating the way to OLDFAITHFUL in Wyoming, but very unlikely that these landmarks would be observed in the opposite state (i.e. P(ω = PIKESPEAK | COLORADO) = 0.15, P(ω = OLDFAITHFUL | COLORADO) = 0.05). Because an agent is much more likely to observe PIKESPEAK in Colorado than in Wyoming, but only somewhat more likely to see a MOUNTAIN, it is clear that a PIKESPEAK observation would be more informative, and therefore more valuable when communicated to a teammate. Unlike the tiger domain, all actions in the Colorado/Wyoming domain lead to informative observations. The observation probabilities are independent of the positions of the agents in the grid world. However, the expected values of the agents' actions change as the agents approach the goal. A complete description of the Colorado/Wyoming domain can be found in Appendix A.3.

5.2. Communication Paradigms

Under certain assumptions about the cost and availability of communication, the questions of **when** and **what** to communicate are equivalent. However, under other, perhaps more realistic, assumptions, agents need to reason explicitly about **what** to communicate. In this section, we present three possible communication paradigms, each of which represents a set of assumptions about communication restrictions. In the remainder of the chapter, we will show how our algorithm, SELECTIVE ACE-PJB-COMM, can be applied to all three paradigms.

5.2.1. Fixed Cost per Communication Instance

The assumption made by ACE-PJB-COMM, and the most common assumption made by algorithms that deal with communication in the context of Dec-MDPs and Dec-POMDPs, is that limitations on communication can be modeled by placing a known, fixed cost on each instance of communication (e.g. (Nair et al., 2004; Xuan et al., 2001; Goldman and Zilberstein, 2003)). Because the cost associated with communication does not change with the size of the message communicated, algorithms operating under this assumption find no benefit in directing agents to communicate fewer observations, thus justifying approaches such as ACE-PJB-COMM and COMM-JESP, in which agents broadcast all previously uncommunicated observations at each instance of communication (Nair et al., 2004). The communication decision is purely a decision about **when** communication may be beneficial. The question of **what** to communicate may be entirely disregarded.

5.2.2. Fixed Cost per Observation

A slightly more complex model for communication limitations is one in which the cost of communication scales with the amount of information to be communicated. We model this by assuming a fixed cost per observation transmitted. Under this assumption, agents primarily reason about **when** to communicate, but must also consider the question of **what** to communicate by attempting to minimize the number of observations communicated. For example, in the tiger domain, once an agent has determined that communicating its entire observation history to its teammate will prompt the team to open a door, it can proceed to compute the minimum subset of that observation history that will still lead the team to open a door, removing, for instance, any observations that were received before the last time a door was opened.

5.2.3. Limited Bandwidth

Often, communication among agents has a strict limit on available bandwidth. For example, in robot soccer, an attempt to communicate the complete and frequent sensory data would easily overload the available communication resources (Roth et al., 2003). In planetary exploration domains, communication bandwidth may be limited by the need of agents to share a small number of satellites or communication relays in order to stay in contact with their teammates (Bhasin et al., 2001). Other domains with bandwidth limitations include distributed surveillance, in which the observations themselves are very large (Rosencrantz et al., 2003). In general, it is possible to quantify the amount of bandwidth available for communication between teammates. For simplicity, we characterize bandwidth limitations by allowing each agent to communicate at most k observations no more frequently than every n timesteps. The challenge, then, is to determine **what** k observations to communicate so as to make the best use of the available bandwidth.

5.2.4. Other Communication Paradigms

It is not our intention to detail *all* possible communication paradigms. We have described a few, relatively simple, communication restrictions and explained how they shape the question of **what** to communicate. One could also consider more complex communication paradigms, such as a domain in which, in addition to a fixed cost per message, there is an additional cost per observation, motivating both fewer instances of communication and smaller messages. Alternately, one could imagine a communication paradigm in which there is both a cost per message and a bandwidth limitation. In this chapter, we describe how SELECTIVE ACE-PJB-COMM can be adapted to any of the three communication paradigms described above. We therefore claim that SELECTIVE ACE-PJB-COMM could be used in any communication paradigm consisting of some combination of these three basic paradigms.

5.3. SELECTIVE ACE-PJB-COMM

The ACE-PJB-COMM algorithm (Chapter 4) answers the question of **when** to communicate, making run-time communication decisions in the context of decentralized execution of a centralized policy by evaluating whether communicating an agent's local observation history would change the joint action selected by its teammates. This section presents SELECTIVE ACE-PJB-COMM, an extension of ACE-PJB-COMM that enables agents to reason about **what** to communicate, and thereby make efficient use of limited communication resources. SELECTIVE ACE-PJB-COMM can be applied under all three communication

paradigms discussed above. It operates by first verifying that communication has the potential to change the joint action and then choosing the k best observations for an agent to communicate, no more frequently than every n timesteps. Choosing **when** to communicate and choosing the minimum subset of observations to communicate can be treated as special cases of choosing k observations at least n timesteps apart:

- If $n = 0$, meaning agents are potentially permitted to communicate every timestep, potentially multiple times per timestep, and $k = \infty$, allowing agents to communicate as many observations as are present in their observation histories, then SELECTIVE DEC-COMM will, like ACE-PJB-COMM, answer the question of **when** to communicate.
- If communication cost is modeled by assuming a fixed cost per observation communicated, and therefore the goal is to minimize the number of observations that it is necessary to communicate in order to change the joint action, SELECTIVE ACE-PJB-COMM should be invoked with $n = 0$, as communication could potentially be beneficial at any timestep. Then, the size-k subset of best observations could be computed, beginning with $k = 1$ and increasing k until the subset is sufficiently large that, if it were communicated, it would change the Q-POMDP action and increase the expected reward by at least ϵ.

Therefore, all further discussions of the details of the SELECTIVE ACE-PJB-COMM algorithm address the problem of choosing k observations to communicate no more frequently than every n timesteps, for arbitrary k and n.

The overall structure of the SELECTIVE ACE-PJB-COMM algorithm, shown in Table 5.1, is nearly identical to the ACE-PJB-COMM algorithm. The SELECTIVE ACE-PJB-COMM algorithm first determines if the agent is permitted to communicate, based on whether more than n timesteps have elapsed since the last time it communicated. If so, the agent uses a helper function, BUILDMESSAGE, to construct a subset of its observation history of size less than or equal to k. Like ACE-PJB-COMM, BUILDMESSAGE first determines if the agent should communicate at all. It does so by comparing the Q-POMDP action that would be selected with no communication to the action that would be selected if the agent were to communicate its entire observation history. If so, BUILDMESSAGE constructs a message in accordance with the domain's communication restrictions. If the message constructed by BUILDMESSAGE contains at least one observation, it is communicated to the agent's teammates.

DEC-COMM-SELECTIVE's arguments are $\mathcal{L}^t$, the distribution of possible joint beliefs at time t, ω_j^t, the observations that agent j has not yet communicated, and T, the number

DEC-COMM-SELECTIVE($\mathcal{L}^t$, $\overrightarrow{\omega}_j^t$, ϵ, k, T, n)

1. **if** $T \geq n$
2. $\quad\overrightarrow{\omega}_C \leftarrow$ BUILDMESSAGE($\mathcal{L}^t$, $\overrightarrow{\omega}_j^t$, ϵ, k)
3. $\quad$**if** $|\overrightarrow{\omega}_C| > 0$
4. $\quad\quad$communicate $\overrightarrow{\omega}_C$ to teammates
5. $\quad\quad\mathcal{L}^t \leftarrow$ prune leafs inconsistent with $\overrightarrow{\omega}_C$ from $\mathcal{L}^t$
6. $\quad\quad\overrightarrow{\omega}_j^t \leftarrow \overrightarrow{\omega}_j^t - \overrightarrow{\omega}_C$
7. $\quad\quad T \leftarrow 0$
8. **if** message $\overrightarrow{\omega}_i^t$ was received from teammate *i*
9. $\quad\mathcal{L}^t \leftarrow$ prune leafs inconsistent with $\overrightarrow{\omega}_i^t$ from $\mathcal{L}^t$
10. $\quad$**return** DEC-COMM-SELECTIVE($\mathcal{L}^t$, $\overrightarrow{\omega}_j^t$, ϵ, k, T, n)
11. $a \leftarrow \arg\max_a$Q-POMDP(a, $\mathcal{L}^t$)
12. take action a
13. receive observation ω_j^{t+1}
14. $\overrightarrow{\omega}_j^{t+1} \leftarrow \overrightarrow{\omega}_j^t \circ \langle\omega_j^{t+1}\rangle$
15. $\mathcal{L}^{t+1} \leftarrow \emptyset$
16. **for** each $\mathcal{L}_i^t \in \mathcal{L}^t$
17. $\quad\quad\mathcal{L}^{t+1} \leftarrow \mathcal{L}^{t+1} \cup$ GROWTREE($\mathcal{L}_i^t$, a)
18. **return** $[\mathcal{L}^{t+1}, \overrightarrow{\omega}_j^{t+1}, T+1]$

Table 5.1. One time step of the DEC-COMM-SELECTIVE algorithm for an agent *j*

of timesteps that have elapsed since the last instance of communication. ϵ, the amount by which expected reward must increase as a result of communication for the agent to decide to communicate, k, the number of observations that can be communicated, and n, the number of timesteps that must pass between instances of communication, are the parameters of execution. The algorithm first determines that the agent j is allowed to communicate in the current timestep, by verifying that T is greater than or equal to n. If so, the agent constructs a message $\overrightarrow{\omega}_C$ by calling BUILDMESSAGE. BUILDMESSAGE may return an empty message if $\overrightarrow{\omega}_j^t$ is empty, meaning that agent j has already communicated its entire observation history, or if communicating all of the observations in $\overrightarrow{\omega}_j^t$ would be insufficient to change the expected reward by at least ϵ.

If the message returned by BUILDMESSAGE, $\overrightarrow{\omega}_C$, is not empty, as in lines 4-7 above, agent j communicates $\overrightarrow{\omega}_C$ to all of its teammates. It prunes $\mathcal{L}^t$ to be consistent with the communicated information, and removes the observations in $\overrightarrow{\omega}_C$ from $\overrightarrow{\omega}_j^t$ so that they will not be considered for future communication. T, the number of timesteps since the previous instance of communication, is reset to 0.

The remainder of DEC-COMM-SELECTIVE, Lines 8-18, function identically to ACE-PJB-COMM. Agent j listens for observations communicated by its teammates (which may,

in turn, prompt agent j to communicate if it has not done so already), updates its distribution of possible joint beliefs and selects an action according to Q-POMDP. After taking this action a, agent j receives an individual observation that it appends to its local observation history, and grows its distribution of possible joint beliefs to take into account all of the possible joint observations that the team could have observed.

5.3.1. BUILDMESSAGE: A Hill-climbing Heuristic

There still remains the problem of, given an observation history of length t, finding the subset of at most k of those observations that would be the most useful if communicated. An obvious possibility is to exhaustively search over all possible subsets of size less than or equal to k for the subset with the highest value of information. However, given that an observation history of length t has $\mathcal{O}(t^k)$ possible subsets of size k, it is clear that exhaustive search quickly becomes impractical.

Neither would it be effective to sort the individual observations in an observation history by informativeness (measured, for instance, by the amount by which they decrease the entropy of the possible joint beliefs or by the value of information of the individual observation) and take the k greatest. Consider, in the Colorado/Wyoming problem, an observation history such as $\langle$MOUNTAIN, OLDFAITHFUL, PIKESPEAK, MOUNTAIN$\rangle$. An agent trying to choose the $k = 2$ best observations might reason that PIKESPEAK and OLDFAITHFUL are, individually, the most informative observations and communicate that pair. However, it is clear that in combination, the two observations would cancel each other out and, rather than being the best size-k subset, would not help in improving the joint action if communicated.

Instead, to facilitate the selection of the subset of observations with a high value of information, we introduce one possible BUILDMESSAGE heuristic, shown below in Table 5.2. The intuition behind the BUILDMESSAGE heuristic is as follows: The agent that is current making a decision about whether and what to communicate first computes a_C, the joint action that the team would perform if that agent were able to broadcast its entire observation history. From this agent's perspective, a_C is the best possible action that the team could take, given all of the available information. If a_C is the same as a_{NC}, the action that would be chosen without any communication, or if the difference in expected rewards between those two actions is less than ϵ (as calculated in lines 1-5 of Table 5.2), then nothing this agent communicates could be expected to improve team performance in the current timestep. If communicating the entire observation history is insufficient to change the joint

action, it is clear that communicating only a subset of this history would also be insufficient. In that case, the agent would choose not to communicate at all. SELECTIVE ACE-PJB-COMM suffers from the same conservatism as ACE-PJB-COMM. Because agents do not choose to communicate until they determine that their local observations are sufficient to change the team action, they may wait longer to act on the basis of their local knowledge than a team with full communication. In a domain with bandwidth limitations, the effects of this conservatism may be even more pronounced. Agents may discover that their local knowledge is sufficient to change the joint action but may lack sufficient bandwidth to communicate those observations. They will have to communicate some observations and wait until their next communication opportunity to complete the transfer of local information. It will, therefore, take them even longer to successfully change the joint action.

BUILDMESSAGE($\mathcal{L}^t$, $\overrightarrow{\omega}_j^t$, ϵ, k)

1. $a_{NC} \leftarrow$ Q-POMDP($\mathcal{L}^t$)
2. $\mathcal{L}' \leftarrow$ prune leaves inconsistent with $\overrightarrow{\omega}_j^t$ from $\mathcal{L}^t$
3. $a_C \leftarrow$ Q-POMDP($\mathcal{L}'$)
4. **if** Q-POMDP$_{a_C}$($\mathcal{L}'$) - Q-POMDP$_{a_{NC}}$($\mathcal{L}'$) $\leq \epsilon$
5. **return** $\emptyset$
6. **else**
7. $\overrightarrow{\omega}_C \leftarrow \emptyset$
8. **while** ($|\overrightarrow{\omega}_C| \leq k$) $\wedge$ ($a_{NC} \neq a_C$)
9. $v_{MAX} \leftarrow -\infty$
10. **for each** $\omega \in \overrightarrow{\omega}_j^t$
11. $\mathcal{L}' \leftarrow$ prune leaves inconsistent with ω from $\mathcal{L}^t$
12. $v \leftarrow$ Q-POMDP$_{a_C}$($\mathcal{L}'$) $-$ Q-POMDP$_{a_{NC}}$($\mathcal{L}'$)
13. **if** $v > v_{MAX}$
14. $v_{MAX} \leftarrow v$
15. $\omega_{MAX} \leftarrow \omega$
16. $\overrightarrow{\omega}_C \leftarrow \overrightarrow{\omega}_C \circ \langle \omega_{MAX} \rangle$
17. $\mathcal{L}^t \leftarrow$ prune leaves inconsistent with ω_{MAX} from $\mathcal{L}^t$
18. $\overrightarrow{\omega}_j^t \leftarrow \overrightarrow{\omega}_j^t - \omega_{MAX}$
19. $a_{NC} \leftarrow$ Q-POMDP($\mathcal{L}^t$)
20. **return** $\overrightarrow{\omega}_C$

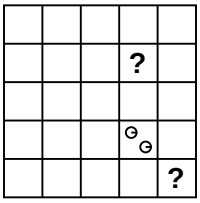
Table 5.2. The BUILDMESSAGE heuristic greedily selects the observations that lead to the greatest difference in expected reward between a_C, the action that would be executed if the agent communicated its entire observation history, and a_{NC}, the action that would be chosen without communication.

If, however, communication could potentially improve the team's performance by changing the joint action selection, then it seems logical to select those observations that do the most to persuade the team to select action a_C. In essence, BUILDMESSAGE is a

hill-climbing heuristic that greedily selects those observations that, when integrated into the distribution of possible joint beliefs, result in the highest expected reward as evaluated over the joint action a_C. Additionally, to maximize the chances of changing the joint action, observations are also selected based on how poorly they effect the expected reward over the joint action a_{NC}. Lines 9-19 of Table 5.2 detail how a single observation is chosen and added to ω_C, the message to be communicated. This process is repeated, with observations added to ω_C until either the maximum number of observations, k, has been selected ($|\overrightarrow{\omega}_C| = k$), or until $\mathcal{L}^t$ has been sufficiently changed such that Q-POMDP($\mathcal{L}^t$) is now a_C. For each observation ω in $\overrightarrow{\omega}_j^t$, BUILDMESSAGE operates by pruning the complete distribution of possible joint beliefs to be consistent with that single observation (lines 10-11), and computing the expected value of a_C minus the expected value of a_{NC} in this pruned distribution (line 12).[1] BUILDMESSAGE compares these observations to find the one that maximizes the change in expected reward between actions a_C and a_{NC} (lines 13-16). This observation can be considered the one that "pushes" the distribution of possible joint beliefs the furthest towards choosing a_C and away from a_{NC}. The algorithm then prunes $\mathcal{L}^t$ to reflect the selection of ω_{MAX}, removes this observation from $\overrightarrow{\omega}_j^t$, computes a new a_{NC} and repeats (lines 17-19).

5.3.2. SELECTIVE ACE-PJB-COMM Example

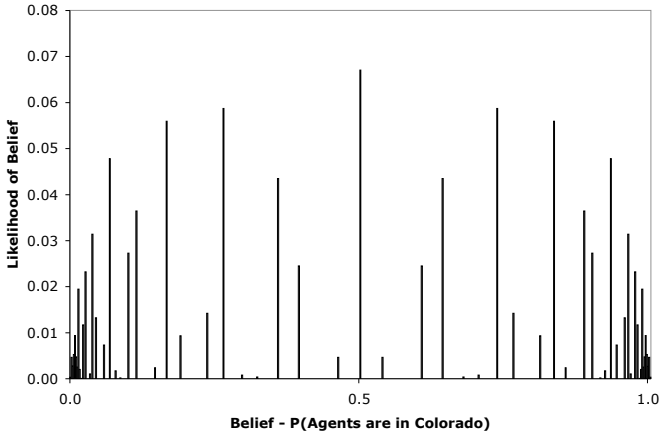
We illustrate how BUILDMESSAGE, in combination with SELECTIVE ACE-PJB-COMM, enables an agent to choose a subset of its observation history to communicate. Consider the observation history in which agent 1 has observed ⟨MOUNTAIN, OLDFAITHFUL, PIKES-PEAK, MOUNTAIN⟩. Suppose that, at the time that the final observation was received, the agents were located in the following configuration, with both agent 1 and agent 2 located at grid position (3,3) (possible goal locations are indicated by question marks):



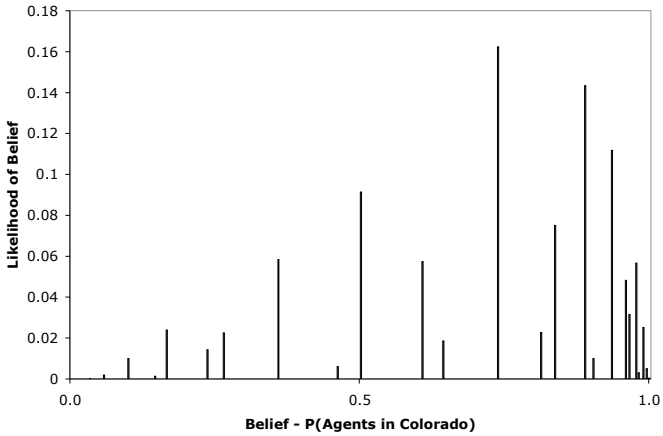
For the sake of clarity, we represent the distribution of possible joint beliefs as a histogram, considering only the portion of the belief that refers to the question of whether

[1] Note that, unlike in ACE-PJB-COMM, the Q-POMDP values of a_C and a_{NC} are computed over a partially, rather than completely, pruned distribution. The value computed in line 12 is therefore not a true expected value, but rather an attempt to estimate the impact on action selection of communicating a particular set of observations.

the robots are in Colorado or Wyoming and assuming that the locations of the two robots on the grid are known. The histogram below represents $\mathcal{L}^4$, the distribution of possible joint beliefs at time $t = 4$, before any communication has taken place. For example, the bars on the extreme left of the histogram correspond to possible joint observation histories in which both agents heard repeated observations of OLDFAITHFUL, making it extremely unlikely that the robots are in Colorado, whereas bars on the extreme right correspond to repeated joint observations of PIKESPEAK:



Given this distribution of possible joint beliefs, there is no reason for the agents to believe more strongly either that they are in Colorado or Wyoming. Therefore, because the agents are in a central location between the two possible goal positions, the best joint action for both agents is to STOP, which costs 0 and does not move them away from either potential goal. However, if agent 1 were to communicate its entire observed history, the following distribution of possible joint beliefs would result:



At this point, sufficient evidence exists for the agents to believe that they are in Colorado. a_C, the best joint action given communication of all of agent 1's observations, is

⟨NORTH, NORTH⟩, which moves both agents towards Denver, the Colorado goal located at (2,4). Suppose, however, that instead of being able to communicate its entire observation history, agent 1 has sufficient bandwidth available to communicate only two observations. To find the best two observations, the agent first finds the single observation that has the highest value of information with respect to a_C. It does this by pruning $\mathcal{L}^4$ four times, each time to be consistent with one of the four observations in $\overrightarrow{\omega}_1^3$, agent 1's observation history before taking an action at timestep 4, yielding the following four distributions of possible joint belief:

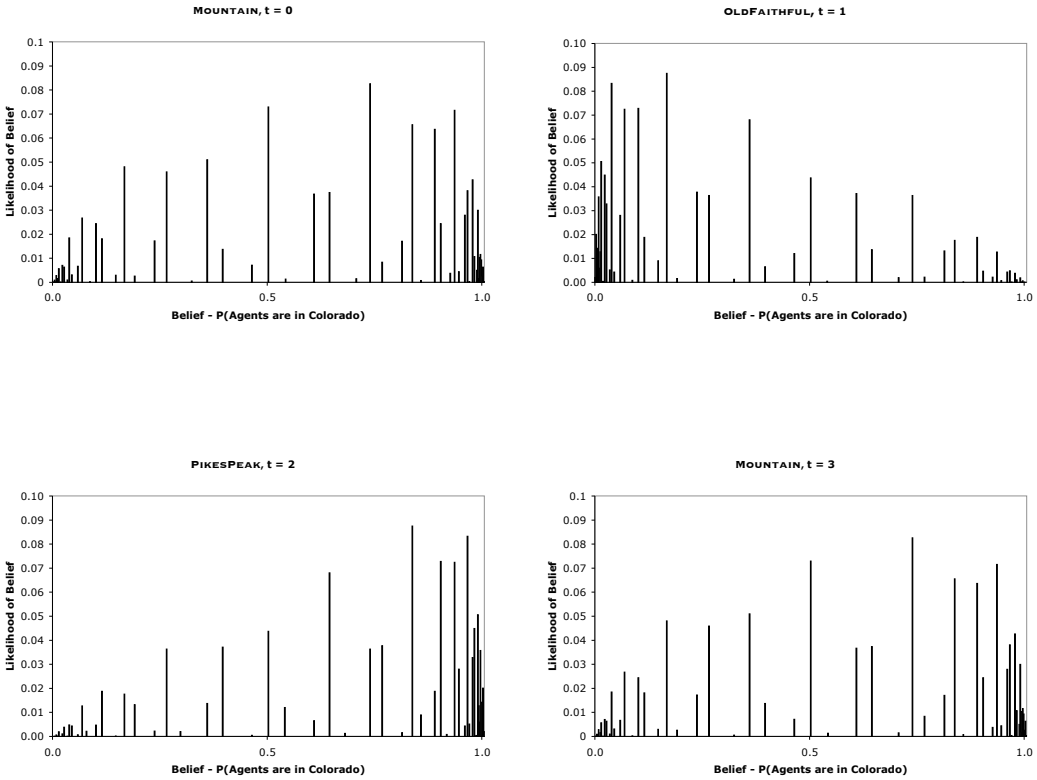
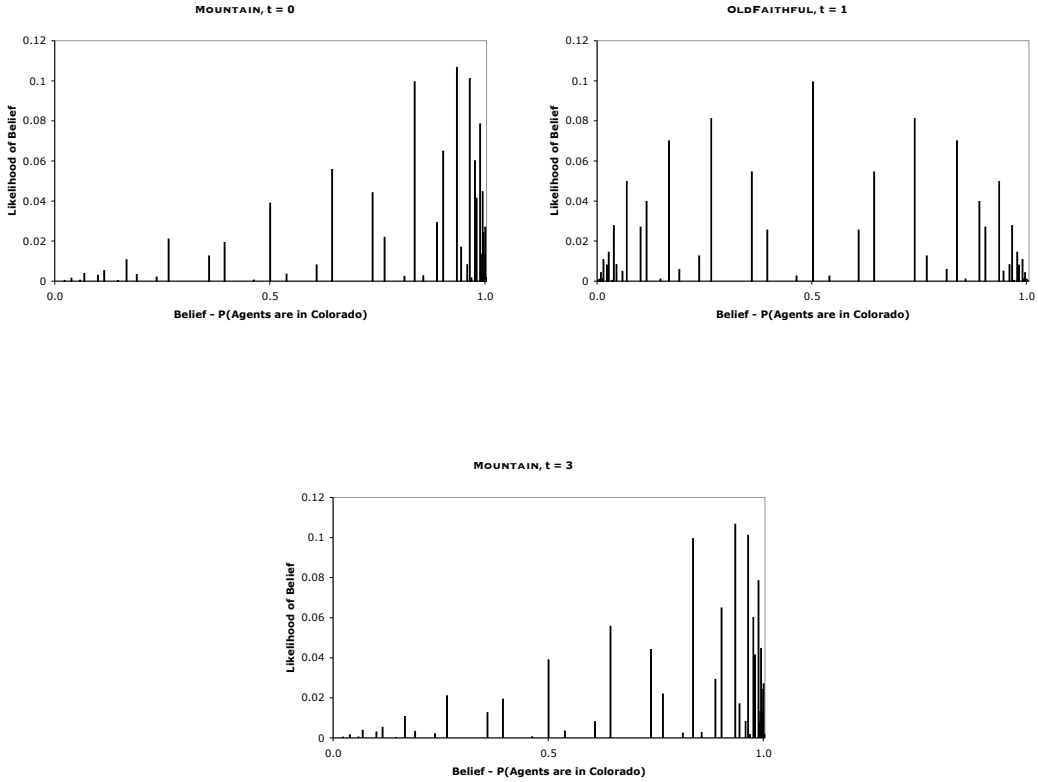


Table 5.3 shows the Q-POMDP values for a_C, ⟨NORTH, NORTH⟩, and a_{NC}, ⟨STOP, STOP⟩, over the four distributions. As expected, communicating PIKESPEAK (observed at timestep 2) would most increase the difference in expected reward between the two actions, and therefore BUILDMESSAGE would choose it first to add to $\overrightarrow{\omega}_C$, the message to be communicated and prune $\mathcal{L}^4$ to be consistent with an observation of PIKESPEAK at timestep 2. Next, the agent prunes the updated $\mathcal{L}^4$ to be consistent with each of the three remaining

$\langle \omega, t \rangle$	**Q-POMDP**$_{\langle \text{NORTH}, \text{NORTH} \rangle}$	**Q-POMDP**$_{\langle \text{STOP}, \text{STOP} \rangle}$	**Q-POMDP**$_{a_C}$ − **Q-POMDP**$_{a_{NC}}$
$\langle \text{MOUNTAIN}, 0 \rangle$	9.07	11.25	-2.18
$\langle \text{OLDFAITHFUL}, 1 \rangle$	6.10	11.25	-5.15
$\langle \text{PIKESPEAK}, 2 \rangle$	10.07	11.25	-1.18
$\langle \text{MOUNTAIN}, 3 \rangle$	9.07	11.25	-2.18

Table 5.3. Q-POMDP$_{a_C}(\mathcal{L}')$ and Q-POMDP$_{a_{NC}}(\mathcal{L}')$ values for all $\mathcal{L}'$ generated by pruning $\mathcal{L}^4$ to take into account each of the four possible individual observations.

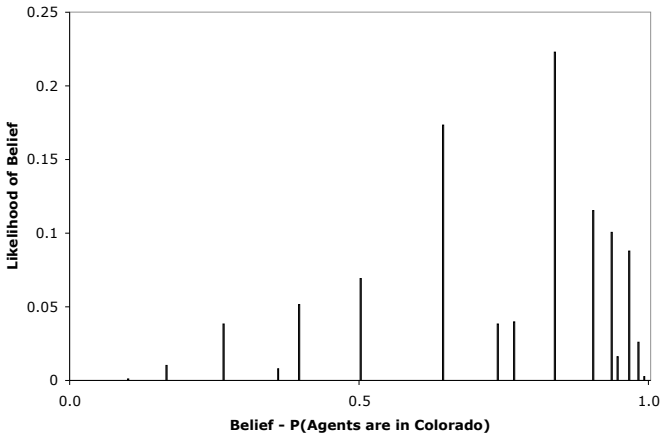
observations:



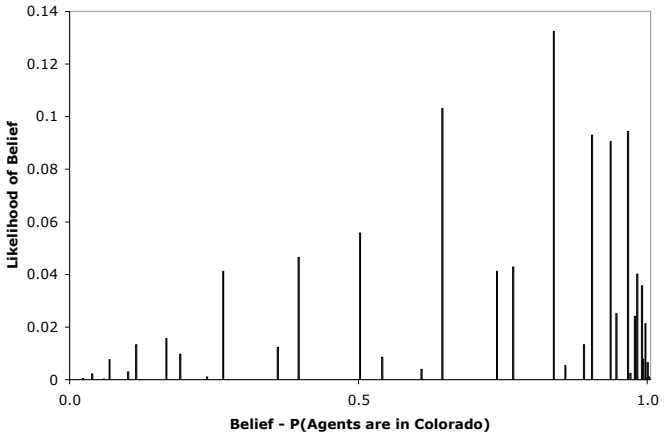
Computing the difference between Q-POMDP$_{a_C}$ and Q-POMDP$_{a_{NC}}$ over the three distributions, BUILDMESSAGE finds that the MOUNTAIN observations best complement the PIKESPEAK observation that has already been chosen and adds one of them to $\overrightarrow{\omega}_C$, completing the set of two observations that agent 1 is allowed to communicate.

A slightly different execution of BUILDMESSAGE can be demonstrated by considering the case in which agent 1 observes $\langle \text{MOUNTAIN}, \text{PLAIN}, \text{PIKESPEAK} \rangle$. In this case, agent 1 will find that communication has the potential to change the joint action after only 3

observations, as opposed to the 4 timesteps that passed in the previous case before communication was found to be potentially beneficial. The pruned distribution that would result if agent 1 were to communicate its whole observation history is:



As before, a_C is $\langle \text{NORTH}, \text{NORTH} \rangle$ and, as in the previous example, the first observation chosen by the BUILDMESSAGE heuristic is PIKESPEAK. However, at this point, when BUILDMESSAGE computes a new Q-POMDP action over an $\mathcal{L}^3$ distribution (shown below) that has been pruned to be consistent with the chosen observation, it finds that the best joint action is $\langle \text{NORTH}, \text{NORTH} \rangle$, the same action as would be chosen if agent 1 were to communicate its entire observation history.



Even though agent 1 is allowed to communicate as many as two observations, it terminates after choosing only one observation, as that observation is sufficient to change the joint action to the Q-POMDP action that would be chosen if it communicated its entire observation history. The BUILDMESSAGE heuristic automatically minimizes the number of observations communicated, thus making it applicable to the communication paradigm

in which a fixed cost is incurred per observation communicated, avoiding the need to try different values of k. It can simply be invoked with $k = t$.

5.3.3. Limitations of a Greedy Approach

It must be noted that BUILDMESSAGE is a greedy heuristic, and as such, is sub-optimal. In particular, because BUILDMESSAGE attempts to hill-climb towards a_C, the joint action that would be selected if the communicating agent were able to broadcast its entire observation history, it may miss opportunities to use communication to move the team towards the selection of an intermediate action. Consider a domain with three possible joint actions, x, y, and z, in which agent 1 is deciding whether and what to communicate. Let x be a_{NC}, the action that, given the current distribution of possible joint beliefs, would be selected if no communication takes place. Let z be a_C, the action that would result if agent 1 communicated its entire observation history. Now suppose that agent 1 is only allowed to communicate k observations, and that there is no subset of size k that will change the joint action to a_C. Suppose also that there is some size-k subset of observations $\overrightarrow{\omega}^*$ that, if communicated, would change the joint action to y, where the expected reward of y is greater than that of x but less than the expected reward of action z. An exhaustive search over subsets of observation histories would successfully find $\overrightarrow{\omega}^*$, whereas BUILDMESSAGE, by attempting to choose observations that push the team towards choosing a_C (i.e. joint action z), may choose a subset $\overrightarrow{\omega}_C$ that leaves the joint action at x. In this way, BUILDMESSAGE may sacrifice optimality to avoid doing an exhaustive search over the space of possible subsets of the observation history.

5.4. Results

We tested the effectiveness of the SELECTIVE ACE-PJB-COMM algorithm in the Colorado/Wyoming and Two-agent Tiger domains.

5.4.1. Colorado/Wyoming Domain

We performed several sets of experiments in the Colorado/Wyoming domain. For each experiment, we ran 3000 trials, with each trial consisting of allowing the agents to execute for at least 15 timesteps, or until at least one agent performed a SIGNAL action. First, we used the BUILDMESSAGE heuristic and SELECTIVE ACE-PJB-COMM to minimize the number of observations communicated. To do this, we invoked SELECTIVE ACE-PJB-COMM with $k = \infty$ and $n = 0$, meaning that agents are permitted to communicate at

every timestep, possibly multiple times per timestep, and when they choose to communicate, they may broadcast as many observations as are needed to change the joint action to a_C. However, since BUILDMESSAGE terminates as soon as enough observations have been appended to the message to change the joint action, SELECTIVE ACE-PJB-COMM avoids communicating unnecessary observations. The results are summarized in Table 5.4.

	μ_{Reward} (σ)	$\mu_{Messages}$ (σ)	$\mu_{Observations}$ (σ)
ACE-PJB-COMM	-3.80 (4.05)	3.75 (2.36)	12.0 (6.29)
SELECTIVE ACE-PJB-COMM	-4.01 (4.09)	4.50 (2.65)	6.44 (3.67)

Table 5.4. Summary of results for 3,000 15-timestep trials of the Colorado/Wyoming domain, comparing a team that uses ACE-PJB-COMM to make communication decisions to a team that uses SELECTIVE ACE-PJB-COMM to minimize the number of observations communicated. μ_{Reward} is the mean discounted reward accumulated over 15 timesteps. $\mu_{Messages}$ is the mean number of distinct communication instances per trial, and $\mu_{Observations}$ is the mean number of observations communicated by both agents per trial.

The team using SELECTIVE ACE-PJB-COMM to make communication decisions achieves almost the same overall discounted reward as the team that uses ACE-PJB-COMM (the difference in means is not statistically significant), while communicating 45.11% fewer observations, indicating that ACE-PJB-COMM is clearly communicating redundant information, while SELECTIVE ACE-PJB-COMM is able to be more efficient. It is interesting to note, however, that using SELECTIVE ACE-PJB-COMM leads a team to communicate slightly more *messages*, and that this difference is statistically significant ($p < 0.0001$). The cause of this difference seems to be because, in the case where SELECTIVE ACE-PJB-COMM is used to minimize the number of observations communicated, each agent communicates only enough observations to change the joint action, leaving room for its teammates to decide to communicate in response and change the action back. In some cases, this may lead to further instances of communication, in which the original communicating agent then broadcasts those observations it had previously held back, changing the joint action again. This result indicates that one should use caution when considering a communication policy that minimizes the message size, in domains where the cost of communication corresponds to the number of messages communicated, rather than the size of the messages.

Next, we wished to demonstrate that the BUILDMESSAGE heuristic effectively selects which observations to communicate when there is a strict bandwidth limitation of no more than k observations, at least n timesteps apart. We did this by comparing a team that

uses BUILDMESSAGE to select which observations to communicate to a team that selects observations randomly. The team selecting observations randomly was still restricted to communicate only in cases where communicating the entire observation history would change the joint action. The agent communicating then picked observations at random until it either built a message of size k or until the message it had assembled was sufficient to change the joint action. Note, unlike in ACE-PJB-COMM, in which agents may communicate more than once per timestep, in this case, even if an agent chooses to communicate fewer than k observations in a given timestep, it must still wait at least n timesteps until it can communicate again. The rewards for the BUILDMESSAGE team are shown in Table 5.5 and the rewards for a team that selects observations randomly are in Table 5.6. Clearly, BUILDMESSAGE and SELECTIVE ACE-PJB-COMM select which observations to communicate more effectively, as indicated by the higher mean reward achieved by a team using BUILDMESSAGE.

N / K	1	2	3	4	5
1	-4.53 (3.52)				
2	-4.82 (3.10)	-4.42 (3.81)			
3	-5.15 (2.70)	-4.54 (3.43)	-4.44 (3.44)		
4	-5.34 (2.55)	-4.79 (3.08)	-4.66 (3.30)	-4.67 (3.64)	
5	-5.59 (2.50)	-5.07 (2.91)	-4.92 (3.06)	-4.92 (3.16)	-4.90 (3.04)

Table 5.5. Summary of mean discounted rewards for 3,000 15-timestep trials of the Colorado/Wyoming domain, comparing different values of n (rows) and k (columns) for a team using the BUILDMESSAGE heuristic to choose **what** to communicate.

It is important to note that BUILDMESSAGE with $k = 1$ and $n = 1$ is not equivalent to ACE-PJB-COMM. Although the team is permitted to communicate in every timestep, it may communicate only once per timestep. Recall that SELECTIVE ACE-PJB-COMM chooses to communicate only if it sees that communicating all observations would be sufficient to change the joint action. Consider a situation in which an agent discovers at timestep 2 that communicating both of its observations (for instance, if the agent observed MOUNTAIN twice) would change the joint action. If there were no bandwidth limit, the agent could communicate both observations and impact the joint action immediately. However, in this case, because $k = 1$, the agent may communicate only one observation at

N / K	1	2	3	4	5
1	-6.05 (6.88)				
2	-7.60 (6.45)	-5.72 (7.13)			
3	-8.60 (5.91)	-6.30 (6.68)	-4.96 (6.98)		
4	-9.16 (5.79)	-6.87 (6.32)	-5.75 (6.70)	-5.17 (7.16)	
5	-9.25 (5.20)	-7.79 (6.11)	-6.60 (6.16)	-5.89 (6.52)	-5.35 (6.58)

Table 5.6. Summary of mean discounted rewards for 3,000 15-timestep trials of the Colorado/Wyoming domain, comparing different values of n (rows) and k (columns) for a team choosing **what** to communicate randomly.

this timestep and must wait until the next timestep to communicate an additional observation.

In addition to accumulating a greater discounted reward, the team using BUILDMESSAGE to select observations communicated, on average 2.46 *fewer* observations than the team that built messages randomly. The reason for this is straightforward; because both teams chose observations until they either reached their bandwidth limit or until they were able to change the joint action, the team that selected observations intelligently was able to communicate smaller messages that still succeeded in changing the joint action. Additionally, the team using BUILDMESSAGE required an average of 0.76 fewer instances of communication because the agents were able to set the team on the correct path of actions by communicating the most relevant observations.

5.4.2. Two-agent Tiger Domain

We performed the same experiments in the two-agent tiger domain. For each experiment, we ran 20,000 trials, with each trial consisting of allowing the agents to execute for 6 timesteps.[2] The results shown in Table 5.7 demonstrate that, in the tiger domain as well, a team choosing observations using BUILDMESSAGE achieves the same reward as a team that only reasons about **when** to communicate, while sending 28.7% fewer observations.

Even though the two individual observations in the two-agent tiger domain have the same informative value with respect to the true state, when bandwidth is limited, it is still

[2]We ran many more trials in the tiger domain than in the Colorado/Wyoming domain because the variance in the reward of the tiger domain is much higher. On average, the agents in the tiger domain choose to open a door much more frequently than the agents in the Colorado/Wyoming domain choose to signal. In large part, this is because of the number of timesteps that it takes the agents in the Colorado/Wyoming domain to reach a goal location. Since the agents in the Colorado/Wyoming domain have more timesteps in which to receive observations, they make fewer mistakes than agents in the tiger domain.

	μ_{Reward} (σ)	$\mu_{Messages}$ (σ)	$\mu_{Observations}$ (σ)
ACE-PJB-COMM	5.30 (19.79)	1.77 (0.79)	5.13 (2.38)
SELECTIVE ACE-PJB-COMM	5.31 (19.74)	1.81 (0.92)	3.66 (1.67)

Table 5.7. Summary of results for 20,000 6-timestep trials of the two-agent tiger domain, comparing a team that uses ACE-PJB-COMM to make communication decisions to a team that uses SELECTIVE ACE-PJB-COMM to minimize the number of observations communicated. μ_{Reward} is the mean discounted reward accumulated over 6 timesteps. $\mu_{Messages}$ is the mean number of distinct communication instances per trial, and $\mu_{Observations}$ is the mean number of observations communicated by both agents per trial.

important to select consistent observations that lead the team to open the correct door. Table 5.8 summarizes the mean discounted rewards (with standard deviations) accumulated by a team communicating k observations at most n timesteps apart, chosen according to the BUILDMESSAGE heuristic, while Table 5.9 shows mean discounted rewards for a team that chooses what to communicate randomly. As in the Colorado/Wyoming domain, BUILDMESSAGE is consistently more successful at deciding **what** to communicate. While there is no significant difference between the number of messages communicated by the team communicating randomly and the team using BUILDMESSAGE, the team using BUILDMESSAGE communicated, on average, 0.37 fewer observations than the team making random communication decisions.

N / K	1	2	3	4	5
1	4.47 (20.52)				
2	4.42 (20.45)	4.84 (20.20)			
3	0.061 (9.48)	0.72 (11.33)	0.71 (11.30)		
4	-0.72 (7.76)	0.18 (9.02)	-0.26 (8.69)	-0.20 (8.65)	
5	-1.76 (7.15)	-0.52 (8.35)	-0.48 (8.29)	-0.48 (8.37)	-0.51 (8.42)

Table 5.8. Summary of mean discounted rewards for 20,000 6-timestep trials of the two-agent tiger domain, comparing different values of n (rows) and k (columns) for a team using the BUILDMESSAGE heuristic to choose **what** to communicate.

N / K	1	2	3	4	5
1	1.87 (18.37)				
2	2.01 (17.76)	3.84 (19.25)			
3	-1.68 (9.34)	-0.39 (11.39)	0.38 (11.24)		
4	-4.34 (7.39)	-1.67 (8.39)	-0.61 (8.25)	-0.20 (8.59)	
5	-5.29 (6.38)	-2.48 (7.65)	-0.91 (7.71)	-0.45 (8.34)	-0.43 (8.36)

Table 5.9. Summary of mean discounted rewards for 20,000 6-timestep trials of the two-agent tiger domain, comparing different values of n (rows) and k (columns) for a team choosing **what** to communicate randomly.

5.5. Summary

In this chapter, we began to address the question of **what** agents should communicate. We identified situations in which algorithms like ACE-PJB-COMM, which instruct agents to broadcast their entire observation history whenever they make the choice to communicate, are wasteful of limited communication resources by communicating non-informative observations. We then identified three paradigms for modeling the cost of communication:

1. Fixed cost per instance of communication
2. Fixed cost per observation communicated
3. Limited bandwidth

The SELECTIVE ACE-PJB-COMM algorithm, introduced in this chapter, extends ACE-PJB-COMM to answer the question of what to communicate within any of these three paradigms. To enable the selection of the most valuable subsets of observations in an observation history, we introduced BUILDMESSAGE, a greedy heuristic that computes the action an agent would prefer if it could communicate all of its observations, and then chooses those observations most compatible with that action. We illustrated SELECTIVE ACE-PJB-COMM and BUILDMESSAGE with an example in the Colorado/Wyoming domain, which has multiple observations of varying information qualities, and demonstrated the performance of our algorithm experimentally in the Colorado/Wyoming and Two-agent Tiger domains.

CHAPTER 6

Communicating State Features

In the previous chapter, we began exploring the question of **what** a team of agents should communicate. We looked at how an agent can select the most valuable observations from its observation history. However, we claim that there are many domains in which the question of what to communicate is best answered by examining the individual *state features*. Consider, for example, the AIBO soccer team discussed in the introduction to this thesis. The robots in that domain capture camera images of their environment, color-segment them, and extract an observation consisting of 9 features: the robot's own location, the position of the ball, and the positions of the 7 other robots on the field. In any given observation, many of these elements may be uninformative, as the objects that they refer to are currently outside the robot's field of view. Also, while there are certainly scenarios in which it is helpful for a robot to know the position of its teammates or opponents, in most cases, the most important piece of information for the robot to know is the location of the ball. The approach presented in this chapter uses a *factored representation* of state to enable agents to communicate useful state features to their teammates.

There is an additional issue to be considered. The ACE-PJB-COMM and SELECTIVE ACE-PJB-COMM algorithms *guarantee* that agents will avoid coordination errors while using communication to execute a centralized policy. However, they do so by making a limiting restriction on team behavior - they require the agents to synchronize their choice of joint action. The algorithms ensure that the team's action choice is synchronized by allowing agents to choose their actions based only on information that is known to all teammates. We do not permit agents to choose actions based on their local observations unless they first communicate those observations to all of their teammates. In this chapter, we relax this restriction by using factored policies to identify portions of the state space in which agents can act *independently* of their teammates, while still guaranteeing that the team will avoid coordination errors.

We consider *collectively observable* domains, modeled by Decentralized Markov decision problems (Dec-MDPs). Like in Dec-POMDP domains, the agents modeled by a Dec-MDP cannot determine the world state by themselves. However, unlike in a Dec-POMDP, the union of the team's observations *does* uniquely identify the current world state at every timestep. Therefore, the underlying centralized problem for a Dec-MDP that could be constructed if the team had free communication is an MDP. Because Dec-MDPs with limited communication are, like Dec-POMDPs, NEXP-complete, our overall approach in this chapter is the same as in previous chapters. We first solve the underlying centralized MDP and then show how a decentralized team can execute that policy by reasoning about communication at execution-time.

In previous chapters, we addressed communication in teams modeled by Dec-POMDPs. In those domains, agents receive noisy observations and use those observations to form *belief* estimates, probability distributions over the possible states. Merging the beliefs of different agents is an open problem (see Section 7.2.1). We therefore restrict Dec-POMDP agents to communicating their observations, as the observations of one teammate can be integrated straightforwardly into the belief of another. In Dec-MDP domains, agents observe portions of the state directly and without noise. Therefore, in this chapter, we use the phrase *state feature* to refer interchangeably to a component of the state and to an agent's observation of that component.

In Section 6.1, we introduce a Meeting-Under-Uncertainty domain that we use as a running example throughout this chapter. Section 6.2 discuss how factored representations of state can be used to discover instances of *context-specific independence* between teammate agents. Section 6.3 introduces the idea of decision tree-structured MDP policies. Section 6.4 shows how a team of agents can execute individual tree-structured policies, and how these policies can be used to instruct the agents both **when** and **what** they should communicate. The algorithm described in Section 6.5 transforms tree-structured joint policies into individual policies that are guaranteed to prevent coordination errors, while attempting to maximize the number of timesteps in which agents can act independently. Section 6.6 presents empirical results that demonstrate, in two domains, the communication savings that can be achieved by using individual factored policies to control a team of agents.

6.1. Meeting-Under-Uncertainty Domain

In this chapter, we introduce a version of the common Meeting-Under-Uncertainty domain (Xuan et al., 2001; Goldman and Zilberstein, 2003; Bernstein et al., 2005) in which

agents may act independently during most of their execution, but face a potential coordination error upon reaching a goal state. In our domain, n agents must meet at a predetermined location in a grid world, and when all are at the goal location, simultaneously send up a SIGNAL. The other individual actions available to the agents are: NORTH, SOUTH, EAST, WEST, and STOP, with movement actions succeeding with 0.9 probability. The agents receive a reward of +20 for signaling together, and receive penalties for either mis-coordinating their SIGNAL actions (-100) or signaling from the wrong location (-50). The team incurs a cost of -1 for each timestep that it takes them to reach the goal location and SIGNAL. Each agent observes its own location (e.g., $\mathcal{X}_0$ and $\mathcal{Y}_0$ for agent 0, and $\mathcal{X}_1$ and $\mathcal{Y}_1$ for agent 1 in a two-agent domain) but does not know the positions of its teammates. The problem ends when at least one agent SIGNALS. Clearly, this is a domain with many instances of potential independence between agents. While the agents are moving towards the goal location, they can act independently. The agents need to communicate only once they have reached the goal. Figure 6.1 shows one possible 3-by-3 grid world, which we use as the example domain throughout this chapter. Appendix A.4 presents the complete details of this domain.

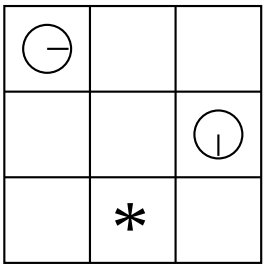


Figure 6.1. Agents in a 3-by-3 grid world. The goal is at (2,3).

6.2. Factored Representations and Context-Specific Independence

Factoring is an approach that has been applied successfully to the problem of speeding up policy computation for large Markov decision problems (Boutilier et al., 2000; Guestrin and Gordon, 2002). Unlike standard MDP representations that enumerate all possible states, factored MDPs take advantage of *conditional independences* among state variables to allow more compact problem representations, and in some cases, more compact policies. We are interested in Dec-MDP domains in which the state space, $\mathcal{S}$, is comprised of n state features, or variables:

- $\mathcal{X} = \langle x_1 \ldots x_n \rangle$, the set of state features

Each state s_i is an assignment of values to every feature in $\mathcal{X}$. The features may be binary, multi-valued, or both. In the general case, the set of joint observations Ω of a Dec-MDP is defined such that each joint observation corresponds to a single world state, and there is at least one such joint observation per state. We are interested in domains in which Ω can be defined in terms of state features. In these domains, each agent observes, without noise, some subset of the features, such that:

- $\mathcal{X} = \cup_{i \in \alpha} \mathcal{X}_i$, where $\mathcal{X}_i$ is the set of state features observed by agent i

We sometimes refer to $\mathcal{X}_i$ as the set of *local features* belonging to agent i. These feature sets may be overlapping. The domain's collective observability is guaranteed by the requirement that each variable must be observed by at least one agent. Agents may also observe some noisy observations, drawn from the set Ω_{NOISY}, such that the complete set of joint observations is:

- $\Omega = \mathcal{X} \times \Omega_{\text{NOISY}}$

In practice, however, we are interested only in the deterministically observable portion of Ω, the state features in $\mathcal{X}$, since they are sufficient to uniquely identify the current state.

As in previous chapters, each agent i has a set $\mathcal{A}_i$ of possible individual actions. Because our model is factorable, the relationship between state features and joint actions can be represented as a Dynamic Decision Network (DDN) (Dean and Kanazawa, 1989). For each joint action, there is a Bayes network that specifies the relationship between state variables at time t and time $t + 1$. A state feature at time $t + 1$ is dependent on the values of its parent features in time t, indicated in the network diagram by arcs from parent to child nodes. The feature is conditionally independent of all other, non-parent, features. Associated with each variable is a conditional probability table (CPT) describing the probability of that variable achieving a particular value assignment given the different possible values of its parents. A variable x^1 is said to exhibit *context-specific independence* if, for one possible value of a parent variable x^2, x^1 has conditional independence of another parent variable x^3, but is not independent of x^3 for other values of x^2. When a domain contains many instances of context-specific independence, the conditional probability tables for its variables can be represented more compactly as trees (Boutilier et al., 1996).

Consider the network shown in Figure 6.2. This network specifies the Bayes net corresponding to the joint action $\langle \text{NORTH}, \text{NORTH} \rangle$ in a two-agent Meeting-Under-Uncertainty domain. In addition to the state variables $\mathcal{X}_0$, $\mathcal{Y}_0$, $\mathcal{X}_1$, and $\mathcal{Y}_1$ described in Section 6.1, this domain has three state variables, END_0, END_1, and ABS.[1] ABS indicates that at least one

[1] We needed to define these additional variables, END_0, END_1, and ABS, so that the reward function could be defined independently of the joint action. An action-independent reward function was necessary because

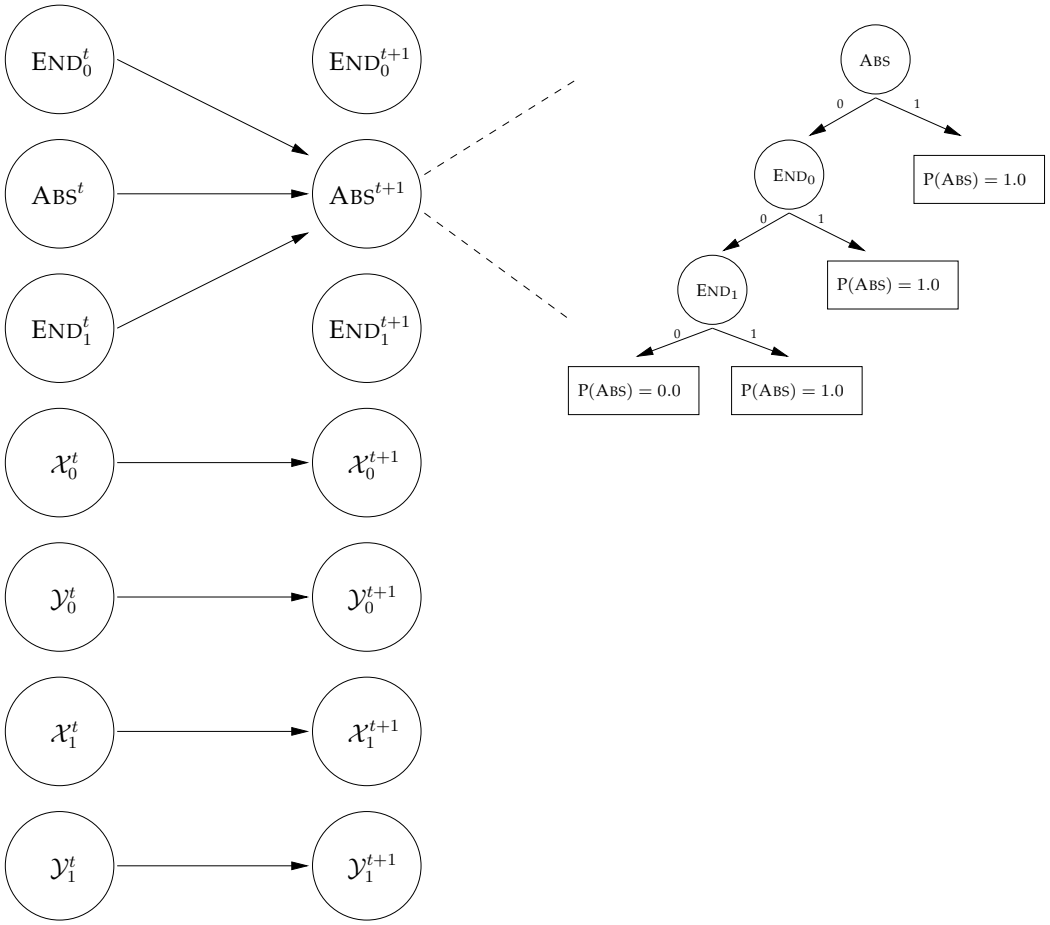


Figure 6.2. Dynamic Decision Network for the joint action $\langle$NORTH, NORTH$\rangle$ in the two-agent, 3-by-3 Meeting-Under-Uncertainty domain. The conditional probability tree for the variable ABS is shown.

agent performed a SIGNAL action at some past timestep, transitioning the problem to an absorbing state. The variables $\{\text{END}_i\}_{i\in\alpha}$ indicate that an agent i performed SIGNAL in the immediately previous timestep. The example shows the conditional probability tree for ABS. In this example, ABS^{t+1} depends on END_1^t when $\text{END}_0^t = 0$ and $\text{ABS}^t = 0$, but is independent of END_1^t when either END_0^t or ABS^t equals 1. One could, therefore, say that ABS has *context-specific independence* from END_0 and END_1.

We claim that many multi-agent domains exhibit a large amount of context-specific independence, allowing them to be represented compactly using factored representations. Context-specific independence can also be used to identify portions of the space in which agents can choose actions independently of the actions and observations of their teammates. In the following sections, we discuss how factored representations of state can be

we used Structured Policy Iteration (see Section 6.3.1.1) to generate a policy for the Meeting-Under-Uncertainty domain.

used to construct *factored policies*. We show how a joint factored policy for the underlying MDP of a Dec-MDP can be transformed into individual factored policies for each of the teammate agents, and how communication can be used to facilitate the decentralized execution of factored policies.

6.3. Tree-Structured Policies

To exploit context-specific independence and enable agents to reason about which state features should be communicated, we must construct *individual factored policies* for each agent in a multi-agent team. In particular, the types of policies that we require are *decision tree-structured policies*, which branch over state variable values and resolve into actions at the leaves. We consider two types of tree-structured policies:

- *joint factored policies*, that have *joint actions* at the leaf nodes, and
- *individual factored policies*, that have *individual actions* at the leaf nodes.

Figure 6.3 shows part of a joint factored policy for the Meeting-Under-Uncertainty domain. The complete tree-structured policy is too large to display.

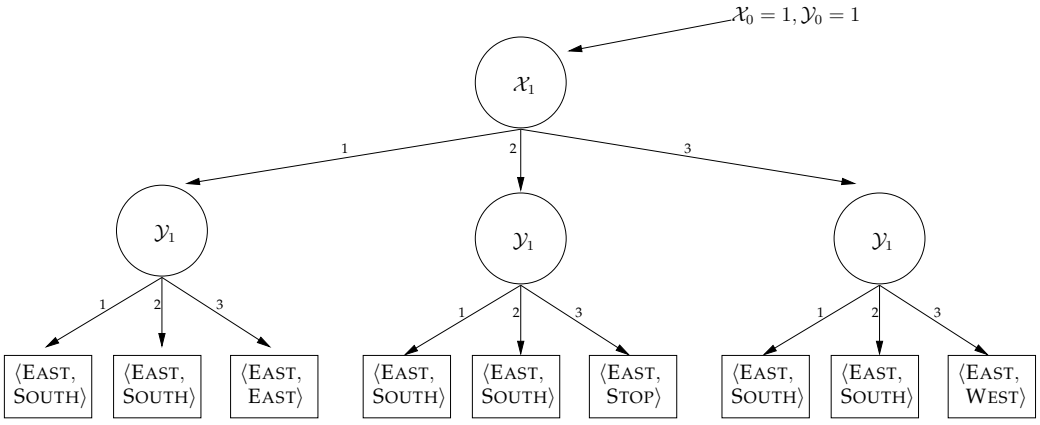


Figure 6.3. A portion of a joint factored policy for the two-agent, 3-by-3 Meeting-Under-Uncertainty domain. This is the subtree in which $\mathcal{X}_0 = 1$ and $\mathcal{Y}_0 = 1$.

Figure 6.4 shows one possible individual factored policy for agent 0. Note that an individual factored policy for a particular agent may contain state variables at its decision nodes that the agent does not directly observe. For example, in the policy below, agent 0 may need to know the values of $\mathcal{X}_1$ and $\mathcal{Y}_1$ that store the position of its teammate, agent 1. However, agent 0 does not observe these variables. In Section 6.4, we discuss how communication can be used to enable agents to execute policies that reference variables observed by their teammates.

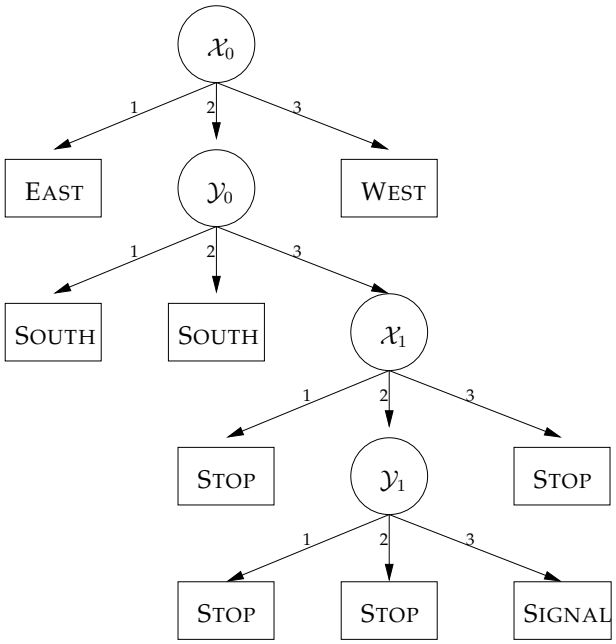


Figure 6.4. A possible individual factored policy for agent 0 in the two-agent, 3-by-3 Meeting-Under-Uncertainty domain.

6.3.1. Constructing Joint Factored Policies

In this section, we discuss two methods for constructing tree-structured joint factored policies: Structured Policy Iteration, and decision tree learning. Both methods generate policies for single-agent MDPs. To use either algorithm to generate a joint policy for a Dec-MDPs, we suppose that the state is made fully observable through free communication and treat the joint actions as single atomic actions.

6.3.1.1. Structured Policy Iteration . Structured Policy Iteration (SPI) is an algorithm that, given tree-structured conditional probability tables and for a factored MDP, computes an optimal decision-tree structured policy for that problem (Boutilier et al., 2000). SPI also requires a tree-structured reward function that, at every timestep, is independent of the chosen action. SPI relies on three helper functions defined over decision trees, SIMPLIFY, APPEND, and MERGE:

- SIMPLIFY removes redundant nodes or subtrees from a decision tree.
- APPEND combines two decision trees. APPEND(T_1, l, T_2) appends the complete tree T_2 to a tree T_1 at its leaf l. The leaves of T_2 are combined with the leaf l either by adding the value stored at l to the values of T_2's leaves, or by selecting the maximum value, depending on the context in which APPEND is used. APPEND(T_1, T_2) appends T_2 to *every* leaf of T_1.

- MERGE combines a set of decision trees by recursively APPENDing the next tree in the set to the result of the previous MERGE operation:

$$\text{MERGE}(\{T_n, \ldots, , T_1\}) = \text{APPEND}(T_n, \text{MERGE}(\{T_{n-1}, \ldots, T_1\})) \tag{6.1}$$

Structured Policy Iteration uses dynamic programming to estimate the value of and incrementally improve decision tree-structured policies. SPI is guaranteed to converge to an optimal policy. The speed of convergence depends on the amount of context-specific independence in a given domain.

6.3.1.2. Decision Tree Learning . Alternately, a tree-structured policy can be learned from a standard, unfactored MDP policy that maps every state to an action. A tree-structured policy can be learned by treating the state-action pairs as the input to a classification problem. Each state is a set of value assignments to a set of features, and the actions are class labels. A decision tree learning algorithm can then be applied to the data set (Quinlan, 1993). One benefit of using decision tree learning to construct a tree-structured policy is that the centralized MDP policy that forms the input need not be optimal. In multi-agent domains where the centralized MDP is very large, generating an optimal policy may be infeasible. If a sub-optimal centralized policy is provided, a decision tree learner can transform it into a tree-structured policy with identical expected reward.

6.3.2. Preserving Ties

Consider the case, in the two-agent 3-by-3 Meeting-Under-Uncertainty domain, in which both agents start in the upper left corner of the grid world ($\mathcal{X}_0 = 1, \mathcal{Y}_0 = 1, \mathcal{X}_1 = 1, \mathcal{Y}_1 = 2$). The joint policy shown in Figure 6.3 indicates that, in this state, the best joint action is $\langle \text{EAST}, \text{SOUTH} \rangle$. However, there are three other, equally valuable, joint actions that could have been used in the policy: $\langle \text{EAST}, \text{EAST} \rangle$, $\langle \text{SOUTH}, \text{EAST} \rangle$, and $\langle \text{SOUTH}, \text{SOUTH} \rangle$. We introduce the concept of a *tie* to describe situations such as this. We consider there to be a TIE among two or more actions at a particular state (or set of states) if the differences between the expected values of the actions are within some bound ϵ:

$$\text{TIE}(a^i, a^j, s^k) \leftarrow |\mathcal{Q}_{a^i}(s^k) - \mathcal{Q}_{a^j}(s^k)| < \epsilon \tag{6.2}$$

Detecting and preserving ties between joint actions in a policy can assist in the discovery of context-specific independence among agents. Figure 6.5 shows the same subtree of the joint policy for the Meeting-Under-Uncertainty domain shown in Figure 6.3, in which $\mathcal{X}_0$ and $\mathcal{Y}_0$ equal 1, expanded to preserve all ties among joint actions. Observe that, when $\mathcal{X}_1 = 1$, the joint actions $\langle \text{SOUTH}, \text{EAST} \rangle$ and $\langle \text{EAST}, \text{EAST} \rangle$ are optimal for all values of $\mathcal{Y}_1$. Because the expected values of $\langle \text{SOUTH}, \text{EAST} \rangle$ and $\langle \text{EAST}, \text{EAST} \rangle$ are independent of $\mathcal{Y}_1$ when $\mathcal{X}_0$, $\mathcal{Y}_0$,

and $\mathcal{X}_1$ equal 1, $\mathcal{Y}_1$ can potentially be removed from the policy at that subtree. Removing $\mathcal{Y}_1$ at that point in the policy increases the amount of context-specific independence for agent 0. In Section 6.5, we explore further how ties can be used to discover context-specific independence between agents.

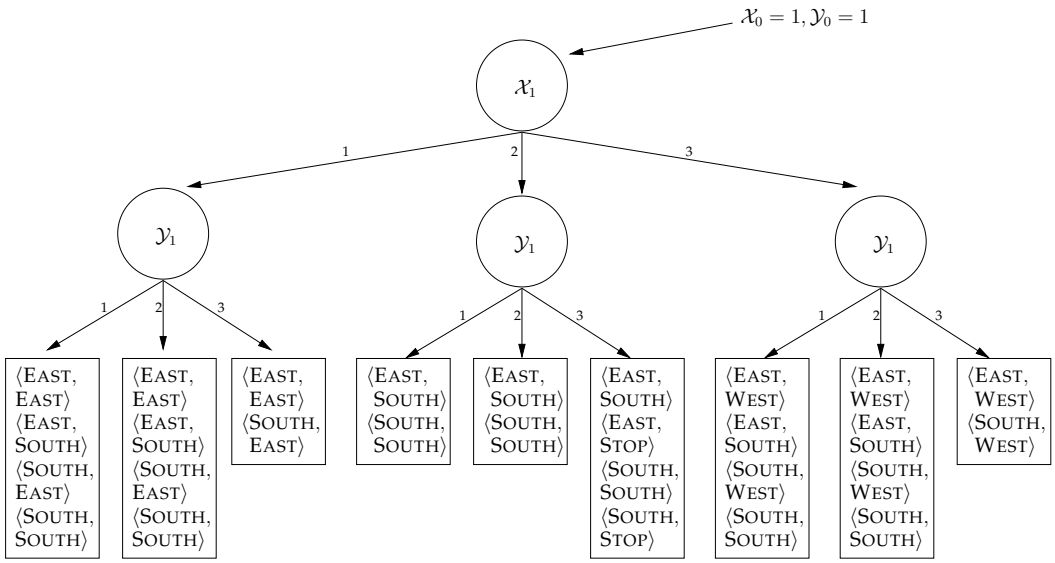


Figure 6.5. A portion of a joint factored policy that preserves all ties among joint actions, for the two-agent, 3-by-3 Meeting-Under-Uncertainty domain. This is the subtree in which $\mathcal{X}_0 = 1$ and $\mathcal{Y}_0 = 1$.

Henceforth, we allow the leaves of policy trees to contain sets of actions. Extending SPI to generate policy trees that preserve ties among joint actions is straightforward. In their original forms, APPEND and MERGE pick one action at random when they encounters a tie. They can just as easily concatenate multiple actions into a leaf node, if the actions are ϵ-similar. To preserve ties among joint actions when learning a decision tree from an MDP policy, one must have available a Q-value table that gives expected rewards for every state-action pair. Instead of learning a decision tree with single joint actions at the leaves, it is possible, for each state, to group all of the actions with ϵ-similar expected rewards, and learn a decision tree over the sets with the maximum expected reward for each state. Increasing ϵ reduces the expected reward of a joint policy, and therefore the derived individual policies, but can increase the amount of context-specific independence.

6.4. Executing Tree-Structured Policies

In this section, we explain how agents can execute factored policies. We begin by showing how agents can execute a joint factored policy in the presence of free communication. Then, we discuss how communication usage can be reduced by providing each agent

with an individual factored policy. As in previous chapters, we guarantee that agents will **A**void **C**oordination **E**rrors (ACE) during execution.

6.4.1. Centralized Execution

To ensure that agents will **A**void **C**oordination **E**rrors during execution of a **J**oint **F**actored **P**olicy (ACE-JFP), we provide all of the agents with an identical joint policy. If any leaves in the joint policy contain more than one joint action, we solve the problem of equilibrium selection by choosing one action according to some canonical action ordering. The overall procedure for executing a tree-structured policy, shown in Table 6.1, is straightforward. At the beginning of each timestep, each agent broadcasts the values of the state features that it observes to all of its teammates. The agents then traverse the policy tree, choosing branches according to the value assignments of the state variables that they encounter, until they reach an action at a leaf node.

ACE-JFP($policy$, $features_{OWN}$, $features_{ALL}$)

1. **if** $features_{ALL} = \emptyset$
2. broadcast $features_{OWN}$ to all teammates
3. $features_{ALL} \leftarrow features_{OWN} \cup$ features received from teammates
4. **if** $policy$ is a leaf
5. **return** $policy.action$
6. $x \leftarrow features_{ALL}[policy.variable]$
7. **return** ACE-JFP($policy.child_x$, $features_{OWN}$, $features_{ALL}$)

Table 6.1. The algorithm to recursively execute a joint factored policy. At the beginning of each timestep, every agent broadcasts its local features, $features_{OWN}$, to all of its teammates.

As it is shown in Table 6.1, each agent executing ACE-JFP communicates one message per timestep. That message contains the values of all of the state features that the agent observes. Therefore, the total number of messages per timestep is α, the number of agents. As more than one agent may observe each state feature, the upper bound on the number of features communicated per timestep is $\alpha \cdot |\mathcal{X}|$. Some communication savings may be achieved by dividing up the redundantly-observed variables among the agents, such that each variable is communicated exactly once per timestep. In that case, the number of features communicated is $|\mathcal{X}|$. The agents may still be communicating unnecessarily. In the Meeting-Under-Uncertainty domain, each agent observes 4 state features: ABS, END$_i$,, $\mathcal{X}_i$, and $\mathcal{Y}_i$. If communication of ABS was assigned to agent 0, the total number of features communicated per timestep would be $1 + 3 \cdot \alpha$. However, the joint policy does not contain

the variables ABS or $\{\text{END}_i\}_{i \in \alpha}$; it depends only on the $\mathcal{X}$ and $\mathcal{Y}$ positions of the agents. If, instead, the agents communicated only those state features appearing in the policy, in this domain, they would communicate $2 \cdot \alpha$ features per timestep.

Finally, a tradeoff can be made between the number of *messages* and the number of *features* communicated. In some domains, it may be preferable to communicate many small messages, rather than a few unnecessarily large ones. Table 6.2 shows the ACE-JFP-SEQUENTIAL algorithm, in which agents communicate the value of a state variable only upon reaching it in the policy. By dividing up communication of redundantly-observed variables among the agents, the upper bound on the number of features and messages communicated per timestep can be reduced to the depth of the policy tree. The actual number of features communicated varies over the course of execution, as different portions of the policy tree may be deeper than others.

ACE-JFP-SEQUENTIAL($policy$, $features_{OWN}$)

1. **if** $policy$ is a leaf
2. **return** $policy.action$
3. **if** $policy.variable \in features_{OWN}$
4. communicate $features_{OWN}[policy.variable]$ to all teammates
5. $x \leftarrow features_{OWN}[policy.variable]$
6. **else**
7. $x \leftarrow$ receive communication from teammates
8. **return** ACE-JFP-SEQUENTIAL($policy.child_x$, $features_{OWN}$)

Table 6.2. The algorithm to recursively execute a joint factored policy, communicating state features in the order that they appear the in policy.

6.4.2. Decentralized Execution

ACE-JFP-Sequential shows how executing a factored policy, even a joint factored policy, can lead to communication savings over communicating all state features at every timestep. However, teams can achieve even greater savings by equipping each agent with an *individual factored policy*. In Section 6.5, we show how to transform a joint factored policy into individual factored policies for each agent, without introducing coordination errors. Here, we present an algorithm for **A**voiding **C**oordination **E**rrors during execution of an **I**ndividual **F**actored **P**olicy (ACE-IFP). As in ACE-JFP, agents traverse their policy trees, choosing branches according to the values of the state features that they encounter. Recall, however, that an agent's individual policy may contain decision nodes that reference state variables that the agent does not itself directly observe. Agents executing individual

factored policies must reason about **when** and **what** to communicate. Execution steps in which an agent traverses its individual policy tree and reaches an action, having evaluated only nodes concerned with that agent's local variables, indicate portions of the state space in which that agent has *context-specific independence* from its teammates. On the other hand, when an agent encounters a variable that it does not directly observe, this identifies an instance in which the agent depends on the knowledge of one of its teammates, indicating the need for communication.

The type of communication used in this section differs from that used in ACE-JFP and the two previous chapters. In general, multi-agent communication policies can be categorized as belonging to one of three paradigms:

- *tell*, in which each agent reasons about its local knowledge and decides whether to send information to its teammates
- *query*, in which agents ask their teammates for information when they believe it is needed
- and *sync*, short for *synchronize*, in which if at least one agent initiates communication, all of the teammates broadcast their full observation histories.

Previous work on communication in Dec-MDPs and Dec-POMDPs has utilized the *tell* paradigm, as discussed in Chapters 4 and 5 (Roth et al., 2005, 2006), or the *sync* paradigm (Nair et al., 2004; Goldman and Zilberstein, 2003). In this chapter, we consider a different option, *query* communication. Instead of trying to predict whether its information will be useful to its teammates, each agent asks its teammates for information as needed.

In order to *query*, agents must reason about **when** communication is necessary, as well as **what** information they require. The decision-tree structured policies we have created provide answers to both of these questions. Table 6.3 shows how communication enables the distributed execution of a factored policy and how, simultaneously, the factored policy directs the agent's decisions about **when** and **what** to communicate. ACE-IFP takes as input an individual tree-structured policy for an agent i, and $features$, a list of the current values that agent i has observed for those state variables visible to itself. It assumes that each agent possesses a list of which state features are observed by which of its teammates.

The procedure for **A**voiding **C**oordination **E**rrors during execution of an **I**ndividual **F**actored **P**olicy is straightforward. If $policy$ is a leaf, the agent executes the action stored at that leaf (lines 1-2).[2] If there is a variable at $policy$'s root node, the agent must determine the current value of that variable. If this variable is observed directly by the agent, its value

[2]If the leaf contains a set of actions because ties among independent actions remain once the individual policy has been constructed, the agent uses a helper function, CHOOSEACTION, to choose one of them at random. They all have equal value and, as we show in Section 6.5, are guaranteed not to cause mis-coordination.

ACE-IFP($policy$, $features$)

1. **if** $policy$ is a leaf
2. **return** CHOOSEACTION($policy.action$)
3. **if** $policy.variable \in features$
4. $x \leftarrow features[policy.variable]$
5. **else**
6. $x \leftarrow$ QUERY correct teammate about $policy.variable$
7. **return** ACE-IFP($policy.child_x$, $features$)

Table 6.3. The algorithm to recursively execute an individual factored policy, communicating when necessary.

will be contained in $features$ (lines 3-4). However, if the variable is not part of the agent's feature set, the agent knows that it must communicate. Note that unlike other approaches, which rely on broadcast communication, *query* communication as implemented in this approach is peer-to-peer. When an agent reaches a variable in its policy tree that it does not itself observe, it now knows not only exactly **when** and **what** to communicate (namely, "now" and "the variable currently in question"), but also **to whom** it must communicate. After retrieving the value of the variable at hand (line 6), the agent continues traversing its policy tree by taking the branch associated with that value (line 7).

The amount of communication needed during execution corresponds to the amount of context-specific independence among the agents in a given domain. Whereas agents executing in domains in which their actions are frequently independent of the actions and observations of their teammates will need to communicate only a little, agents in domains requiring a great deal of tight coordination will find that they must communicate often.

As described above, an agent may communicate several times in a single timestep, each time asking for the value of a single state feature. By asking for information only when it is immediately necessary, the agents minimize the total number of state features communicated over the course of execution. However, there may be domains in which, rather than reducing the number of bits communicated, it is desirable to reduce the total number of messages. In that case, once an agent makes the decision to communicate, it should also attempt to predict what other information it may need to know to make a decision at that timestep. Table 6.4 shows ACE-IFP-PREDICTIVE, which extends ACE-IFP to enable an agent to communicate at most once to each of its teammates.

ACE-IFP-PREDICTIVE($policy$, $features$, $teammate\_features$)

1. **if** $policy$ is a leaf
2. **return** $policy.action$
3. **if** $policy.variable \in features$
4. $x \leftarrow features[policy.variable]$
5. **else**
6. $j \leftarrow$ the teammate that observes $policy.variable$
7. **if** $teammate\_features_j = \emptyset$
8. $\overrightarrow{v} \leftarrow$ find all variables observed by j in $policy$
9. $teammate\_features_j \leftarrow$ QUERY agent j about $\overrightarrow{v}^j$
10. $x \leftarrow teammate\_features_j[policy.variable]$
11. **return** ACE-IFP-PREDICTIVE($policy.child_x$, $features$, $teammate\_features$)

Table 6.4. The algorithm to recursively execute a factored policy, communicating when necessary, and minimizing the number of messages communicated by predicting the state variables that are potentially useful.

ACE-IFP-PREDICTIVE has an additional argument, $teammate\_features$, a vector that starts out empty. During execution, if the agent comes to a variable that it does not observe (lines 5-9), it finds the teammate j that observes that feature. The agent then fills $teammate\_features_j$ by collecting all of the variables observed by agent j contained in the subtree of the policy rooted at this decision point (lines 8-9) and QUERYing agent j for their values. $teammate\_features$ is then maintained for the rest of this decision step, so that no further communication is needed between this agent and agent j. In some cases, the agent will have requested information that it will later discover to be irrelevant. Therefore, when deciding how to execute a tree-structured policy, once must know whether it is preferable to communicate extra messages or extra features and choose an algorithm accordingly.

6.5. Transforming Joint Factored Policies Into Individual Factored Policies

By using ACE-IFP to execute individual factored policies, agents can exploit context-specific independence present in the domain to allow them to choose actions independently, communicating only to avoid coordination errors. However, the amount of independent execution that is possible for a particular agent depends on the structure of its individual factored policy. Consider the individual policy shown in in Figure 6.4 for agent 0 in a two-agent 3-by-3 Meeting-Under-Uncertainty domain. Agent 0 is able to execute that policy independently until it reaches the goal location because its individual policy is constructed to maximize its context-specific independence from agent 1. In this section, we present an algorithm, GENERATEINDIVIDUALPOLICY, for transforming a joint factored

policy into an individual factored policy an agent i. The goal of this transformation is two-fold. First, it ensures that the team of agents executing the resulting individual policies will achieve the same level of performance as a team executing a joint policy with full communication. The individual policies will not only ensure that the agents avoid all coordination errors not present in the input MDP policy, they will preserve the same expected reward of the joint policy. Secondly, our algorithm attempts to maximize the portion of each individual policy tree in which the agent can execute independently of the variables observed by its teammates.

The first step in the process of transforming a joint factored policy into an individual factored policy for an agent i is to reorder the variables in the policy so that the variables local to agent i are at the root of the tree. Intuitively, this increases the likelihood that, during execution, agent i will be able to traverse portions of its policy tree without encountering a variable that it cannot observe directly. Note that changing the variable ordering may cause the policy tree to become larger. However, as we demonstrate in our experiments (see Section 6.6.2), a correctly variable-ordered individual factored policy may be much larger than even the joint factored policy, yet still have significantly more context-specific independence. Choosing the correct ordering of the agent's variables in the policy is an open problem, and can potentially have a dramatic impact on the amount of context-specific independence that is discovered.

In Section 6.3.2, we explained how joint policy trees can be enhanced to preserve ties among joint actions. We define two additional tree operations applicable to factored policies in which the leaves store sets of actions:

- INDEPENDENT, which operates over individual leaves of a policy tree and finds, for an agent i, those actions that it can perform independently of the action choices of its teammates, and
- INTERSECT, which extends the SIMPLIFY operator to remove redundant subtrees from a policy tree by intersecting the action sets at the subtrees' leaves.

The INDEPENDENT and INTERSECT operators are key components of the GENERATEINDIVIDUALPOLICY algorithm.

An individual action is considered INDEPENDENT for an agent i in a given leaf of a joint policy tree if the action is optimal when paired with any of the other actions that its teammates could potentially choose at that leaf. Consider the leaf shown in Figure 6.6. It is a leaf from the two-agent, 3-by-3 Meeting-Under-Uncertainty domain, containing the optimal joint actions when $\mathcal{X}_0$, $\mathcal{Y}_0$, $\mathcal{X}_1$, and $\mathcal{Y}_1$ equal 1. Agent 0 has two possible individual

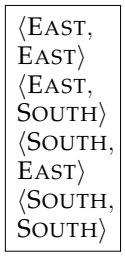


Figure 6.6. The leaf at $\langle \mathcal{X}_0 = 1, \mathcal{Y}_0 = 1, \mathcal{X}_1 = 1, \mathcal{Y}_1 = 1 \rangle$ from a joint policy for the two-agent, 3-by-3 Meeting-Under-Uncertainty domain.

actions: EAST and SOUTH. Both actions are optimal when paired with either of the individual actions that agent 1 could potentially choose at that leaf. Therefore, EAST and SOUTH are said to be INDEPENDENT actions for agent 0 at this leaf of the joint policy tree.

Table 6.5 details the procedure for finding the INDEPENDENT actions for an agent i in a given policy tree leaf. INDEPENDENT takes $\overrightarrow{a}$, the set of optimal joint actions at a particular leaf of a joint policy tree, and an agent i, and builds $\overrightarrow{ind}$, agent i's INDEPENDENT actions at that leaf. For every joint action a in $\overrightarrow{a}$, INDEPENDENT checks if a^i, agent i's component of that action, can be paired with any of the combined actions that i's teammates may choose. $\{\overrightarrow{a}\}^{-i}$, defined in line 2, is the set of agent i's teammates' joint actions. The comparison at line 4 asks whether, for every combined action a^{-i} in the leaf, the joint action constructed by pairing a^i with a^{-i} exists in the set of optimal actions at this leaf. If a joint action exists for every combined action that could be taken by agent i's teammates, then a^i is independent and can be added to $\overrightarrow{ind}$ (lines 5-6).

INDEPENDENT($\overrightarrow{a}$, i)

1. $\overrightarrow{ind} \leftarrow \emptyset$
2. $\{\overrightarrow{a}\}^{-i} \leftarrow$ the possible combined actions of i's teammates
3. **for** each $a \in \overrightarrow{a}$
4. **if** $\langle a^i a^{-i} \rangle \in \overrightarrow{a}, \forall a^{-i} \in \{\overrightarrow{a}\}^{-i}$
5. $\overrightarrow{ind} \overset{\cup}{\leftarrow} a^i$
6. **return** $\overrightarrow{ind}$

Table 6.5. The INDEPENDENT operation finds the individual actions that agent i can perform independently of its teammates' action choices in a given policy tree leaf.

Figure 6.7 shows additional examples of INDEPENDENT. In Figure 6.7 (a), agent 2 may potentially choose action x or action y. Because agent 1's individual action a is optimal when paired with either x or y, a is INDEPENDENT for agent 1 in this leaf. Likewise, the

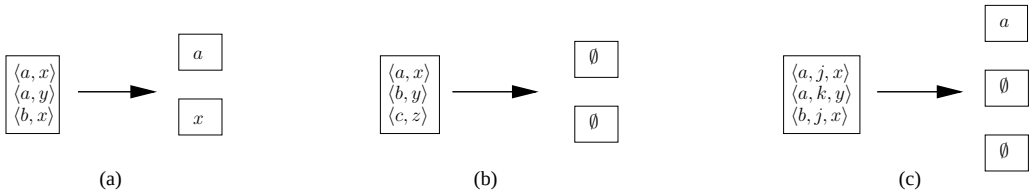


Figure 6.7. Examples of individual actions discovered by the INDEPENDENT operation.

individual action x is INDEPENDENT for agent 2 in this leaf, because it is optimal when paired with either a or b, the two possible individual actions that agent 1 could choose. By contrast, Figure 6.7 (b) shows a policy tree leaf in which, although there are several joint actions with equivalent expected values, there are no INDEPENDENT actions for either agent. In domains with more than two agents, such as in Figure 6.7 (c), an action is INDEPENDENT if it can be paired with each possible combined action that could be chosen its teammates. Therefore, action a is INDEPENDENT for agent 1, since it can be paired with $\langle j, x\rangle$ and $\langle k, y\rangle$, the possible combined actions of agents 2 and 3. However, agents 2 and 3 do not have any independent actions.

Like SIMPLIFY (Boutilier et al., 2000), INTERSECT is an operation that removes redundant leaves and subtrees from a policy tree. However, unlike SIMPLIFY, INTERSECT operates over policy trees with sets of actions at the leaves. In principle, INTERSECT can be applied to both joint and individual factored policies. In practice, we use it only on individual policy trees. Consider the policy tree shown in Figure 6.8 (a). It is an individual policy for an agent 1 that observes the feature $\mathcal{X}_1^1$ and can take the individual actions {A, B, C}. Its teammate, agent 2, observes the state features $\mathcal{X}_2^1$, $\mathcal{X}_2^2$, and $\mathcal{X}_2^3$. SIMPLIFY cannot reduce this policy tree, as it has no nodes with identical children.

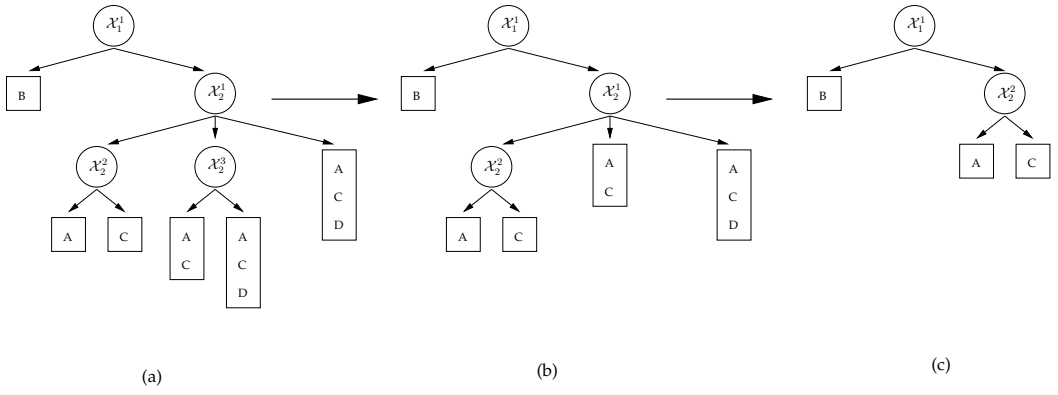


Figure 6.8. Examples of tree redundancies removed by INTERSECT.

However, additional nodes can be removed from the tree because many of the leaves contain redundant actions, all with equal expected values. In particular, the node labeled $\mathcal{X}_2^3$ in Figure 6.8 (a) can be removed from the policy tree because its children, all leaves, have a non-empty action set intersection. The actions A and C are optimal for agent 1, independent of the value of $\mathcal{X}_2^3$. Finding this intersection increases the amount of context-specific independence in the tree, removing a variable that agent 1 does not observe. The resulting policy tree is shown in Figure 6.8 (b). INTERSECT can reduce the policy tree even further. Consider the subtree of the policy in Figure 6.8 (b) whose root is labeled $\mathcal{X}_2^1$. Again, the actions A and C are present in all of $\mathcal{X}_2^1$'s children. However, $\mathcal{X}_2^1$ cannot be replaced by a leaf containing the action set $\{\text{A}, \text{C}\}$. The choice of action A versus C, while independent of the value of $\mathcal{X}_2^1$, still depends on the value of $\mathcal{X}_2^2$. Therefore, INTERSECT replaces $\mathcal{X}_2^1$ with the subtree rooted at $\mathcal{X}_2^2$, as shown in Figure 6.8 (c). Note that, while INTERSECT leads to a smaller policy tree and can potentially remove an agent's dependencies on features observed by its teammates, it also reduces the number of action choices available to the agent at each leaf in its policy. However, as the actions are all equally optimal and independent of teammate actions, this reduction in flexibility will not harm performance.

Table 6.6 details the INTERSECT operation. INTERSECT calls a helper operation, UNION, that recursively builds the action set comprised of all of the actions at the leaves of a policy subtree. INTERSECT takes as input $policy$, the root node of a policy tree, either joint or individual, and returns a reduced policy tree of the same type. If the input tree is itself a leaf node, no further simplification can be done (lines 1-2). If the policy is not a leaf, then INTERSECT is called recursively on all of $policy$'s children. INTERSECT can potentially reduce a policy tree if either all of its children are leaves or if exactly one child is not a leaf node. $\mathcal{A}_L$, defined in line 5 and constructed in lines 9-10, stores the action set composed by the intersection of the action sets at each of $policy$'s children that is a leaf node. If there is one $child$ of $policy$ that is not a leaf, $\mathcal{A}_{NL}$ (line 6) stores the UNION of all of the action sets at the leaves of that $child$ subtree (line 13), while $child_{NL}$ stores a pointer to that subtree (lines 7 and 14). However, if $policy$ has more than one non-leaf child, discovered at Lines 15-16 by reaching a non-leaf $child$ of $policy$ when $\mathcal{A}_{NL} \neq \emptyset$, then INTERSECT cannot further reduce $policy$, and it is returned as is.

If all of the children of $policy$ are leaves and the action sets of those leaves have a non-empty intersection (lines 17-19), then the root node of $policy$ is redundant and can be replaced by a new leaf containing the intersection. If exactly one $child$ of $policy$ is not a leaf, $child_{NL}$, and all of the actions at the leaves of $child_{NL}$ are present in the intersection of $policy$'s leaf-children (lines 20-21), then $child_{NL}$ can replace $policy$'s root node. If none

INTERSECT($policy$)

```
1.  if policy is a leaf
2.      return policy
3.  for each child ∈ policy
4.      INTERSECT(child)
5.  A_L ← ∅
6.  A_NL ← ∅
7.  child_NL ← ∅
8.  for each child ∈ policy
9.      if child is a leaf
10.         A_L ← A_L ⋂ child.actions
11.     else
12.         if A_NL = ∅
13.             A_NL ← UNION(child)
14.             child_NL ← child
15.         else
16.             return policy
17. if A_L ≠ ∅
18.     if A_NL = ∅
19.         return A_L
20.     if A_NL ⊂ A_L
21.         return child_NL
22. return policy
```

Table 6.6. The INTERSECT operation that simplifies a policy tree by intersecting subtrees that contain intersecting sets of actions.

of these conditions are true, then the root node is not redundant and is preserved in the policy tree.

The full process of transforming a factored joint policy into a factored individual policy for an agent i can be found in Table 6.7. It should be noted that GENERATEINDIVIDUALPOLICY is a heuristic and is not guaranteed to produce individual policies with the *maximum* possible amount of context-specific independence between agents. It does, however, guarantee that agents will avoid coordination errors and achieve the same expected reward as agents executing a joint factored policy.

The inputs to GENERATEINDIVIDUALPOLICY are a joint policy and an agent identifier i. First, the joint policy is re-written to move the state variables that agent i observes to the upper-most nodes of the tree (line 1). This reordered tree is processed to discover the INDEPENDENT actions for agent i at each leaf (lines 3-4). Leaves containing INDEPENDENT actions are replaced by those actions. However, if a given leaf has no INDEPENDENT actions, steps must be taken to carefully convert the joint actions at that leaf into a single

GENERATEINDIVIDUALPOLICY($policy$, i)

1. Move agent i's state variables to the root.
2. **for** each $leaf \in policy$
3. **if** INDEPENDENT($leaf$) $\neq \emptyset$
4. $leaf \leftarrow$ INDEPENDENT($leaf$)
5. **else**
6. Break ties among joint actions using canonical action ordering.
7. $leaf.action \leftarrow leaf.action^i$
8. **return** INTERSECT($policy$)

Table 6.7. The algorithm that transforms a tree-structured joint policy into a factored individual policy for agent i.

individual action. First, if any ties remain, meaning that the leaf contains more than one possible joint action, those ties are broken according to a predetermined canonical action ordering (line 6). This is necessary to avoid equilibrium-selection mis-coordinations such as the one shown in Figure 6.9, in which the individual components of the different agents' optimal joint actions do not always recombine to form an optimal joint action. The selected joint action is converted into an individual action by extracting agent i's component (line 7). Finally, the tree is INTERSECTed, in an attempt to remove as many variables as possible from the policy tree (line 8). The more compact the tree, the more likely it will be that agent i will be able to execute the policy that it contains without encountering a need to know the values of its teammates' variables. By finding INDEPENDENT actions for each agent at every leaf, and breaking ties according to a canonical action ordering where such INDEPENDENT actions do not exist, we ensure that agents executing the resulting individual factored policies will avoid coordination errors.

6.6. Results

We demonstrate the effectiveness of our approach in two domains, a Meeting-Under-Uncertainty problem (Xuan et al., 2001; Goldman and Zilberstein, 2003; Bernstein et al., 2005) and a simplified version of the multi-agent taxi domain (Ghavamzadeh and Mahadevan, 2004), comparing the amount of communication needed to execute individual tree-structured policies to the amount used by a centralized team.

6.6.1. Meeting-Under-Uncertainty Domain

We first applied our algorithm to two agents operating in grid worlds of various sizes. Tree-structured joint policies for the problems were generated using a version of Structured

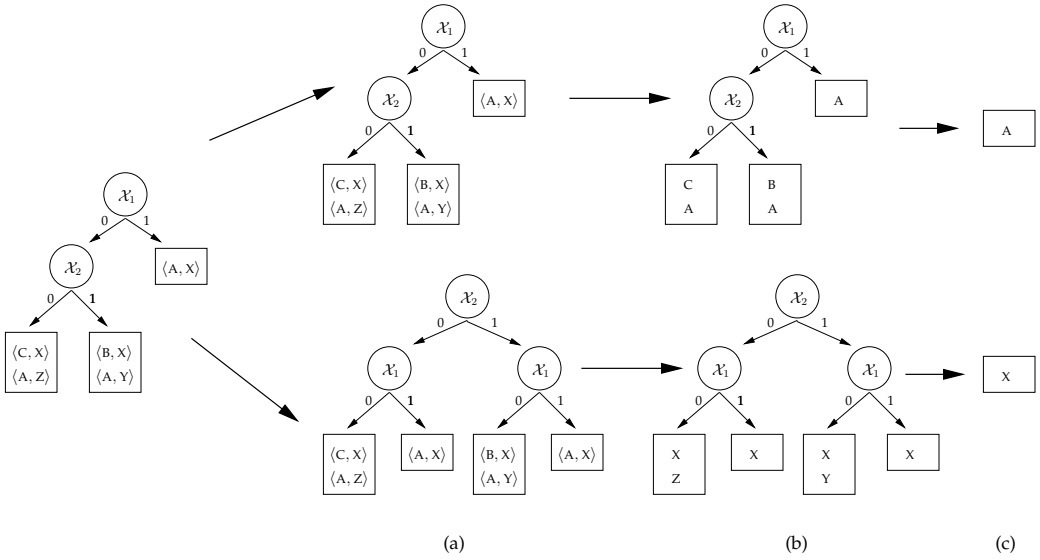


Figure 6.9. An example of mis-coordination that can occur if ties are not broken using a canonical action ordering. $\mathcal{X}_1$ is a state feature local to agent 1, that can execute individual actions $\{A, B, C\}$, and $\mathcal{X}_2$ is local to agent 2, that has individual actions $\{X, Y, Z\}$. (a) An individual policy is generated for each agent, with its local variable at the root. (b) The policies are converted into policies over individual actions, without an intermediate tie-breaking step. (c) The resulting simplified policies indicate that agent 1 should always perform action A and agent 2 should always perform X, making the joint action $\langle A, X \rangle$. This is an uncoordinated action whenever $\mathcal{X}_1 = 0$.

Policy Iteration, modified to preserve ties among joint actions (Boutilier et al., 2000). Figure 6.4 shows a factored individual policy for agent 1 operating in the 3-by-3 grid world shown in Figure 6.1, with the goal at location (2,3). A visual inspection of the policy shows that, as expected, for most of the state space, agent 1 can act independently. Agent 1 finds that it needs to communicate only when it reaches the goal location. At this point, it must ask agent 2 for its position, to determine if both agents are at the goal and can SIGNAL.

We compared our execution methods, ACE-IFP and ACE-IFP-PREDICTIVE, to a team executing a joint factored policy with full communication, both ACE-JFP and ACE-JFP-SEQUENTIAL, running 10,000 trials of each method. In each trial, we started the agents in a random location and allowed them to execute until the end of the problem, when the agents performed a SIGNAL action. The results for a 3-by-3 grid world are summarized in Table 6.8.

Both factored algorithms achieve the same average discounted reward as a team communicating at every timestep, but require communication of significantly fewer messages and variables. ACE-IFP, which enables agents to execute individual factored policies by asking for information as they see that it is needed, communicates 43.8% fewer state

	Mean Discounted Reward (σ^2)	Mean Messages Sent (σ^2)	Mean Variables Sent (σ^2)
ACE-JFP	13.23 (1.01)	6.96 (1.02)	13.92 (1.02)
ACE-JFP-SEQUENTIAL	13.23 (1.01)	11.21 (3.34)	11.21 (3.34)
ACE-IFP	13.24 (2.47)	6.30 (1.86)	6.30 (1.86)
ACE-IFP-PREDICTIVE	13.23 (2.45)	3.25 (1.01)	6.50 (1.01)

Table 6.8. Relative performances and communication usages for teams executing in the 3-by-3 Meeting-Under-Uncertainty domain.

variables than ACE-JPF-SEQUENTIAL. If the object is to minimize the number of messages, ACE-IFP-PREDICTIVE communicates slightly more than half the number of messages needed by ACE-IFP, with only a few extra state variables communicated, and 53.5% fewer messages than ACE-JFP, in which each agent communicates in every timestep. These results confirm our intuition that the Meeting-Under-Uncertainty domain admits a great deal of context-specific independence between agents. This independence increases as the problem size grows. Figure 6.10 shows the amount of communication used by a team executing a factored policy (without attempting to predict what variables will be needed), as a percentage of the number of state variables communicated by a team with full communication for grid worlds of increasing size, from 3-by-3 (82 states) up to 7-by-7 (2402 states). As the problem size increases, in this domain, the number of states in which agents can act without coordinating increases, leading to even greater communication savings over a team that uses full communication to execute a centralized policy.

One major challenge posed by multi-agent problems is the exponential growth in the number of states and joint actions as the number of agents increases. However, as the agents may be independent of their teammates for large portions of the state in some multi-agent domains, the single-agent factored policies needed to control the agents may be very compact. Table 6.9 shows that the number of leaves in each agent's factored individual policy grows linearly as the number of agents increases from two to five in a 3-by-3 Meeting-Under-Uncertainty domain.

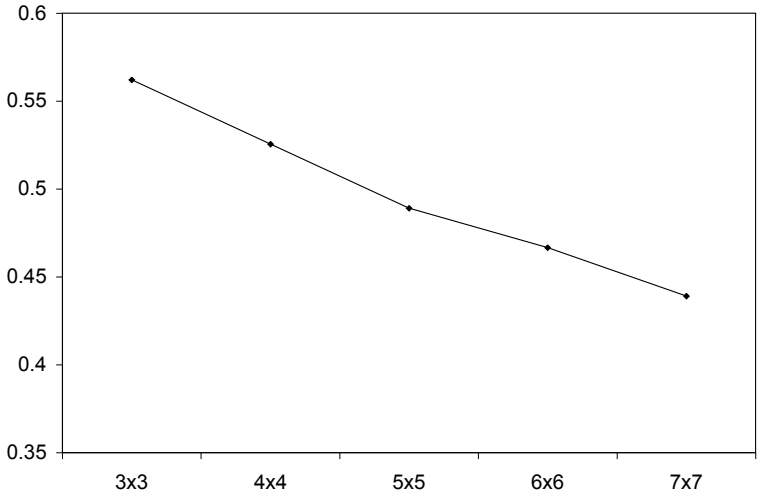


Figure 6.10. Communication usage as a function of problem size in n-by-n Meeting-Under-Uncertainty domains. The y-axis shows the ratio of variables communicated by a team using ACE-IFP to the variables communicated by a team using ACE-JFP-SEQUENTIAL.

	States	Joint Actions	Policy Tree Leaves
2 AGENTS	82	36	9
3 AGENTS	730	216	13
4 AGENTS	6562	1296	17
5 AGENTS	59050	7776	21

Table 6.9. Policy-tree size grows linearly with the number of agents in a 3-by-3 Meeting-Under-Uncertainty domain, even as the number of states and joint actions grow exponentially.

6.6.2. Multi-agent Taxi Problem

We wished to verify the effectiveness of our approach in a more dynamic, non-goal-oriented domain. The multi-agent taxi problem has previously been used as a test domain in the multi-agent reinforcement learning literature (Ghavamzadeh and Mahadevan, 2004). The domain consists of two, or more, taxi agents moving in a grid world, picking up and delivering passengers to four taxi stands. As defined originally, with the taxis moving in a 5-by-5 grid world, the underlying Markov decision process for the two-agent taxi domain has 640,000 states and 49 joint actions, making it too large to solve with Structured Policy Iteration. Additionally, we were unable to generate an exact Q-value table, which would contain over 31 million entries, for this problem. Therefore, we define a simplified version

of the multi-agent taxi domain, in which the taxi agents move directly between the four taxi stands, as shown in Figure 6.11.

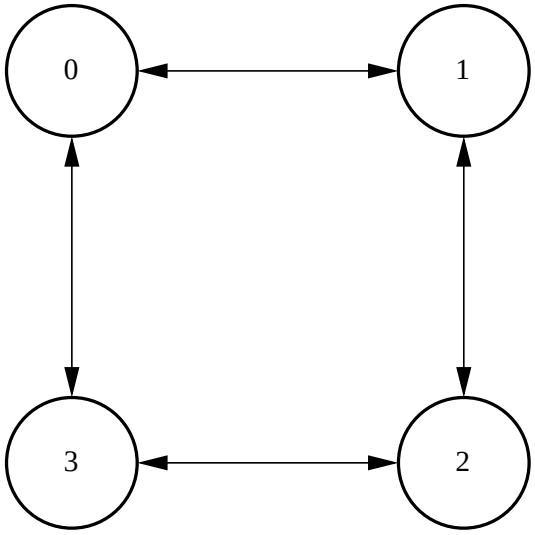


Figure 6.11. Relative positions of the four taxi stands in the simplified multi-agent taxi domain.

Each taxi i has three local variables, POS_i, its current position at one of the four taxi stands, HASP_i, a boolean variable indicating whether the taxi currently has a passenger, and GOAL_i, the destination of that passenger. In addition, there is a boolean variable for each taxi stand, indicating whether a passenger is waiting there. To make the problem collectively observable, we allow Taxi 0 to observe the passenger variables for taxi stands 0 and 1, and Taxi 1 to observe whether there is a passenger at taxi stands 2 and 3. Overall, this simplified domain has 16,384 states. The taxis may move CLOCKWISE or COUNTER-CLOCKWISE, and can PICKUP and DELIVER passengers. A new passenger may appear at a taxi stand with probability 0.25 in any timestep. Agents accumulate a reward of +20 for delivering a passenger to its desired goal location and a penalty of -100 for delivering a passenger to an incorrect taxi stand. There is a cost of -1 for each each agent in any timestep that it does not deliver a passenger. Therefore, the team as a whole accumulates the highest expected reward when the agents coordinate their movements to pick up the passengers closest to their current locations. In the event that both taxis attempt to pick up a passenger from the same taxi stand simultaneously, Taxi 0 succeeds in acquiring the passenger.

We generated a joint policy for the simplified two-agent taxi domain by solving the underlying MDP, using the ZMDP solver (Smith, 2007) to generate a Q-value table, and learning a decision tree over the pairs of states and sets of optimal actions. We then transformed this tree-structured joint policy, which has 3044 leaves, into individual factored

policies for each agent. Taxi 0's policy has 922 leaves, and Taxi 1's policy has 4154 leaves.[3] The complete policies are too large to display here. What is important to note is that, although the agents must inquire about their teammates' state variables while moving to PICKUP new passengers, each taxi may execute independently once it has picked up a passenger and is moving to DELIVER it. Figure 6.12 shows a portion of Taxi 0's individual policy for which HASP_0 is true, meaning that Taxi 0 has a passenger. A complete description of the simplified two-agent taxi domain can be found in Appendix A.5.

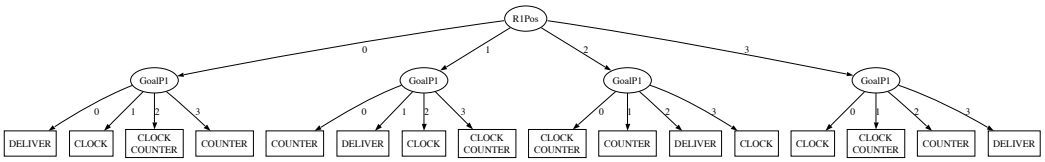


Figure 6.12. A portion of Taxi 0's individual policy. This is the subtree for which HASP_0 is true, meaning that Taxi 0 has a passenger and is moving to deliver the passenger to its desired goal location.

This context-specific independence enables the agents to execute their factored policies with substantially less communication than a team executing the joint factored policy with free communication. Table 6.10 summarizes the amount of communication, measured both in number of messages and in number of state variables communicated, used by teams executing individual factored policies with and without prediction, and a team using communication to execute a centralized joint factored policy. These results were generated over 10,000 trials for each method, where each trial was 20 timesteps long.

	Mean Discounted Reward (σ^2)	Mean Messages Sent (σ^2)	Mean Variables Sent (σ^2)
ACE-JFP	67.46 (11.53)	40.0 (0.0)	200.0 (0.0)
ACE-JFP-SEQUENTIAL	67.46 (11.53)	147.22 (6.12)	147.22 (6.12)
ACE-IFP	67.46 (11.53)	79.32 (11.27)	79.32 (11.27)
ACE-IFP-PREDICTIVE	67.46 (11.53)	23.29 (2.31)	109.29 (11.84)

Table 6.10. Relative performances and communication usages for teams executing in the simplified two-agent taxi domain.

[3]The difference in policy-size due to Taxi 0 "winning" the passenger in the event that both taxis attempt to PICKUP from the same location simultaneous. Taxi 1's policy is larger than the joint policy because of the variable ordering used to maximize its context-specific independence.

A team of agents can execute ACE-IFP by communicating 46.12% fewer state variables than a team executing a joint factored policy with free communication, using ACE-JFP-SEQUENTIAL. If the goal is to reduce the number of messages communicated, rather than the number of state variables, ACE-IFP-PREDICTIVE can enable a team to communicate 70.54% fewer messages than a team using the simple ACE-IFP algorithm, but at a price of communicating 27.42% more (unnecessary) state variables. ACE-IFP-PREDICTIVE executes 41.78% fewer messages than ACE-JFP.

6.7. Summary

This chapter introduced a method by which factored representations, specifically decision tree-structured policies, can be used to identify and exploit context-specific independence among teammates. We defined the INDEPENDENT and INTERSECT operations over policy trees, and provided an algorithm that uses these operators to transform a centralized tree-structured joint policy into individual policies for each agent. These individual policies prevent mis-coordination and attempt to maximize context-specific independence. We then showed how communication can be used to enable a decentralized team to execute tree-structured individual policies. Each agent's policy identifies contexts in which that agent can choose actions independently of its teammates' observations and action choices. In contexts where an agent does depend on its teammates, the agent's policy instructs it to *query* its teammates for the information that it needs to choose an action. In this way, individual factored policies enable agents to answer the questions of **when** and **what** to communicate. We demonstrated the effectiveness of this approach in two domains, the n-by-n Meeting-Under-Uncertainty domain, and the simplified multi-agent taxi domain. In both cases, we showed that agents executing individual factored policies benefit from significant communication savings over teams executing centralized policies.

It is important to point out that work discussed in this chapter does not address potential costs of communication. Whenever agents discover that communication is necessary, they communicate. The agents have no way of reasoning about the marginal benefit of communication, so that they may trade it off against the cost. Further work is required to integrate reasoning over the cost of communication into this approach. One possibility is that factored policies could be used to identify situations in which communication is definitely **not** needed, and another approach could be used to reason about whether to communicate in those cases where communication may have some benefit. For now, this work is primarily useful for characterizing multi-agent domains by the amount of context-specific independence present. Domains in which agents require very little communication

to execute individual factored policies are domains in which the agents have a great deal of independence from one another. When agents must communicate frequently to execute their individual policies, it indicates that they are in a domain with many potential coordination problems and may benefit most from a centralized controller or an algorithm like ACE-PJB-COMM in which agents synchronize their choice of joint action.

CHAPTER 7

Conclusions and Future Work

This thesis investigated and contributed support to the claim that:

> Reasoning about communication decisions at execution-time provides a more tractable means for coordinating teams of agents under partial observability than including all communication decisions in a comprehensive policy that is computed off-line, and can be done without sacrificing too much in terms of team performance.

In this chapter, we enumerate contributions made in this thesis in support of this hypothesis. We close by proposing several directions for future research.

7.1. Contributions

The contributions made by this thesis achieve two overarching goals:

1. We enable a team of agents to **A**void **C**oordination **E**rrors (ACE) by providing algorithms that allow a team of decentralized agents to execute a centralized policy.
2. We provide algorithms that, at execution-time, answer the questions of **when** and **what** a team of cooperative agents should communicate.

There are four main contributions in this thesis:

Reasoning about possible joint beliefs. In Chapter 4, we explore the problem of how a team of decentralized agents can **A**void **C**oordination **E**rrors by executing a centralized policy. We contribute the ACE-PJB algorithm which enables agents to choose actions by reasoning over a *distribution of* **P***ossible* **J***oint* **B***eliefs* (Section 4.3). By ensuring that all agents maintain the same distribution of possible beliefs, and requiring that agents choose actions based only on this distribution, we guarantee that the team will avoid mis-coordination.

The ACE-PJB algorithm is composed of two subroutines: GROWTREE (Section 4.3.1), which calculates the distribution of possible joint beliefs, and Q-POMDP (Section 4.3.2), a heuristic for selecting a joint action based on this distribution. Because the number of possible joint beliefs grows exponentially in time, we show how a particle filter can be used to represent the distribution of possible joint beliefs using a constant amount of memory (Section 4.6).

Deciding when to communicate. The ACE-PJB algorithm provides a means by which a team of decentralized agents can execute a centralized policy without communicating while avoiding coordination errors. However, because the agents are not allowed to choose actions based on their local observations, the overall reward achieved by such a team will generally be low. We therefore show how communication can be used by agents to integrate their local information into the joint belief of the team. We present the ACE-PJB-COMM algorithm that answers the question of **when** an agent should communicate its observation history to its teammates (Section 4.4). ACE-PJB-COMM is a myopic algorithm that instructs agents to communicate when they determine that communicating their observation histories will change the joint action selected by the team as a whole in the current timestep, leading to a gain in expected reward of more than the cost of communication. Our experiments show that ACE-PJB-COMM enables agents to communicate significantly less than a team executing a centralized policy with free communication, reducing the number of observations communicated by between 47.7% and 66.6% in the domains tested. These communication savings are balanced in some domains by a small decrease in expected reward, corresponding to approximately one extra LISTEN action before the agents open a door in the Tiger domains. In the MABC domain, there is no difference in expected reward between a team executing ACE-PJB-COMM and a team with free communication.

Deciding what to communicate. When an agent chooses to communicate, the basic ACE-PJB-COMM algorithm requires it to broadcast all of the observations that it has received since the last time it communicated. This requirement rests on an assumption that communication has a fixed cost per message. In Chapter 5, we identify two additional paradigms of communication restrictions, domains in which there is a fixed cost per observation and those in which the bandwidth available is limited (Section 5.2). We introduce an extension to ACE-PJB-COMM, SELECTIVE ACE-PJB-COMM, that addresses the question of **what** to communicate by enabling an agent to select a subset of the most valuable observations from its observation history (Section 5.3). We provide SELECTIVE ACE-PJB-COMM with

a heuristic, BUILDMESSAGE (Section 5.3.1), that greedily assembles a message comprised of observations with high information value, thus avoiding the need to search exhaustively among all possible subsets of the agent's observation history. In our experiments SELECTIVE ACE-PJB-COMM is able to achieve significant communication savings over ACE-PJB-COMM (28.7% reduction in observations communicated in the two-agent tiger domain, 45.11% reduction in the Colorado/Wyoming domain), with no loss of expected reward.

Identifying context-specific independence between teammates. In Chapter 6, we explore the use of factored representations for modeling Dec-MDP teams. We show how decision tree-structured policies can be used to identify regions of context-specific independence between agents in a given domain. We contribute the ACE-IFP algorithm that enables agents to **A**void **C**oordination **E**rrors by executing **I**ndividual **F**actored **P**olicies, using communication to assist the agents when they reach a state where coordination is needed (Section 6.4). This algorithm achieves many goals. First, it enables agents to act independently, without communication, in those areas of the state space where no potential mis-coordinations exist. Secondly, executing a tree-structured policy enables an agent to identify exactly **when** communication is necessary and **what** communication is needed. Because this algorithm relies on *query* communication, the policy also instructs agents about **which** teammate to ask for information.

We show how agents can construct individual factored policies from a tree-structured joint policy (Section 6.5). We provide heuristics for maximizing the amount of context-specific independence present in these individual policies, while ensuring that agents do not introduce coordination errors into their independent execution. Our experiments show that, in domains with substantial amounts of context-specific independence, the ACE-IFP algorithm allows a team of agents to significantly reduce the amount of communication needed to execute without coordination errors. In the 3-by-3 Meeting-Under-Uncertainty domain, our agents communicate 43.8% fewer state variables than a team with free communication. In the simplified taxi domain, the communication savings is 46.1%. We discuss how measuring the amount of communication needed by a team in which each agent executes an individual factored policy quantifies the amount of inter-dependence among the agents in a particular domain.

7.2. Future Directions

Several interesting issues came up during work on this thesis that could not be addressed, due to the substantial challenges that they pose. We discuss a few below.

7.2.1. Communicating Beliefs

In this thesis, we present algorithms that allow agents to reason about communicating observations or state features. One could, however, make the argument that agents should communicate *beliefs*, as they are the most natural and compact representation of local information. However, communicating belief introduces additional problems, primarily because beliefs do not have the same independence properties as observations. The observation that an agent receives at a given timestep is dependent only on the current state and possibly the actions taken by that agent and its teammates in the previous timestep. An agent's belief about the world state, on the other hand, also depends on its belief at the previous timestep. Suppose an agent were to receive a belief communicated by one of its teammates. If the agent updated its own belief to integrate this communicated information, in and of itself a challenging problem because a single belief may correspond to many different sequences of observations, it could no longer communicate its belief to its teammates in the future. The agent's own belief would contain information from the message communicated to it, leading to a double-counting of information.

In our work on communication for the AIBO Robocup team (Roth et al., 2003), we were concerned about the possibility that robots would reinforce outdated information by communicating secondhand information to their teammates. To address this problem, we developed an approach in which each robot separately maintained an *individual* and a *shared world model*. We enforced strict boundaries between these models, to ensure that the individual model stored only information directly observed by that robot. Robots were only allowed to communicate information from their individual world models, guaranteeing that robots would not re-broadcast information received from their teammates as if it was new. A similar approach is needed to enable agents to communicate beliefs. Agents will need to maintain separate belief models to distinguish between beliefs based on only their local observations, which they *may* communicate to their teammates, and those based on both local and communicated information, which they may not. Additionally, agents will need to separate beliefs that they have already communicated from newly observed information, so that they do not repeat themselves.

7.2.2. Understanding Silence

In early work on communication in Dec-MDPs, a heuristic called "No news is good news" was proposed (Xuan et al., 2001). This heuristic suggested that agents could agree on a plan to achieve a particular goal, and that each agent should decide to communicate only if it detects that it has deviated from the plan. Thus, as long as agents do not receive communication from their teammates, they may assume that everything is going well. More broadly, we observe that there is useful information in silence. An interesting question is how to extract and make use of that information. Consider the two-agent Tiger domain discussed in previous chapters. Using Q-POMDP and ACE-PJB-COMM, agents choose to communicate when they accumulate a surplus of two consistent observations (e.g., observing HEARLEFT twice in a row). If, at the second timestep, one agent does not communicate, its teammate should be able to deduce, therefore, that the agent observed inconsistent observations and prune its distribution of possible joint beliefs accordingly. Likewise, during the execution of factored policies discussed in Chapter 6, each agent should, hypothetically, be able to compute its teammates' tree structured policies, and by observing the state variables that its teammates *don't* ask for, make predictions about the values of that agent's local state features.

The challenge of taking advantage of this information is to avoid brute-force search. When an agent makes a decision about whether to communicate its own observation history (or some portion thereof), it compares two possibilities: the action the team would choose if it does not communicate and the action that would be chosen if the agent communicated exactly that observation history. When considering why its teammate did not communicate, the agent must consider **all** hypothetical observation histories, asking, "Would my teammate have communicated in this case?" Additionally, to ensure that the agents maintain a synchronized distribution of possible joint beliefs, this hypothetical reasoning must be recursive. When an agent does not communicate, it must also consider all possible histories it could have observed and ask, "Would my teammate believe that I would have communicated if I had observed this?"

7.2.3. Modeling Partial Possible Joint Beliefs

When watching agents executing individual factored policies in the Meeting-Under-Uncertainty domain defined in Chapter 6, one observes behavior that seems inefficient. When an agent reaches the goal location, it asks its teammate for its current position so that it can know whether to wait at that location or, if its teammate has also reached the

goal, perform the SIGNAL action. In every subsequent timestep, this agent will continue to ask its teammate, "What is your $\mathcal{X}$ position? What is your $\mathcal{Y}$ position?" This behavior arises because, unlike in the approaches detailed in Chapters 4 and 5, agents executing individual factored policies do not keep any history about the state previously communicated to them by their teammates. It seems intuitively clear that what is missing is either some mechanism by which the agent can tell its teammate, "Inform me when a condition $\mathcal{C}$ becomes true," or a means by which the agent can estimate some component of its teammate's state over time.

The problem with the first approach is identifying the necessary condition $\mathcal{C}$. The Meeting-Under-Uncertainty domain has a clear goal. However, in a domain like the taxi problem, there are no immediately obvious variables whose values one agent would want to be appraised of at all times. Therefore, it is difficult to define a principled and generally applicable approach that would yield the desired behavior. The second possibility, in which one agent models some portion of the variables observed by its teammates, is no easier. In fact, one of the major benefits of the factored approach introduced in this thesis is that agents *never* need to model the possible beliefs of their teammates. If we wished to extend our algorithm in this way, there are many problems we would encounter. Agents using ACE-PJB-COMM are able to model the distribution of possible joint beliefs because they always know the joint action selected by the team. In our factored approach, one goal is to allow agents to select actions independently. Without knowing the actions its teammates are taking, how can an agent model the evolution of its teammates' state variables? Additionally, because communication in this approach is peer-to-peer, how can an agent take into account the information one of its teammate may be receiving from another?

7.2.4. Combining Factored Representations with Algorithms that Consider Communication Cost

The ACE-IFP algorithm detailed in Chapter 6 differs from the coordination algorithms in Chapters 4 and 5 (ACE-PJB-COMM and SELECTIVE ACE-PJB-COMM) in a significant way. ACE-PJB-COMM and SELECTIVE ACE-PJB-COMM enable agents to make decisions about **when** to communicate by trading off the cost of communication with the expected increase in reward that would result. Communication decisions made by ACE-IFP, which enables agents to execute individual factored policies, do not incorporate this kind of tradeoff. Whenever agents reach a point in their individual policies where their decision depends on a state variable observed by another agent, they *query* for that information. It may be the case, in some domains, that the cost of choosing incorrectly at one particular policy

branch is small. However, our approach does not distinguish between mis-coordinations that lead to a large negative reward and those in which the difference between the optimal joint action and some other action choice is small; we eliminate all possible coordination errors. Therefore, there is no way to reason about a tradeoff in terms of expected reward.

One might suppose that the solution to this problem is as simple as storing values for the actions in the leaves of a factored policy. However, when computing expected value, one must also know the likelihood of all possible outcomes. For the reasons detailed above, it would be very challenging for agents to attempt to estimate the values of their teammates' state variables. Therefore, it is not straightforward for an agent to attempt to estimate the potential loss of reward that would result from it choosing not to ask for information.

A potentially interesting approach would be to combine these factored policies with another algorithm such as ACE-PJB-COMM that reasons about the cost of communication. Individual factored policies identify those portions of the state space where, for a given agent, communication is never necessary. ACE-PJB-COMM could then be used to make communication decisions in the rest of the state space. Again, the challenge arises from the problem of modeling possible joint beliefs. ACE-PJB-COMM, as currently constructed, requires each agent to be able to predict the joint action. Additionally, it relies on a complete model of the distribution of possible joint beliefs. There is no immediately obvious means to hybridize these two approaches, but it is an interesting area for research.

7.2.5. Extending Factoring to Dec-POMDPs

Unlike ACE-PJB-COMM and SELECTIVE ACE-PJB-COMM, which are applicable to both Dec-MDPs and Dec-POMDPs, ACE-IFP is only applicable to Dec-MDPs. This restriction is because ACE-IFP builds on a large body of previous work on factored MDPs. There has been some preliminary work aimed at extending factored representations to POMDPs (Hansen and Feng, 2000), and this work could form a basis for extending factored approaches to Dec-POMDPs. The dynamic programming algorithm presented by Hansen and Feng generates policies for POMDPs in which the factored state is represented using algebraic decision diagrams (ADDs). The approach is applicable to POMDP domains in which the state variables are observed directly, although potentially noisily. The generated policies have the form of finite state controllers, in which nodes are actions and transitions between nodes depend on conjunctions of observed state variables.

It is possible to compute joint factored POMDP policies in which the nodes of the finite state controllers are joint actions. As in ACE-JFP, communication of observed state

variables would enable execution of this joint policy. However, reducing communication by separating these joint factored policies into individual factored policies remains a significant challenge. It may be possible, like in the GENERATEINDIVIDUALPOLICIES method that supports ACE-IFP, to separate each joint policy into a finite state controller over individual actions for each agent, and combine nodes with identical actions to eliminate the need to consider variables observed by the agent's teammates. Without some synchronization mechanism, a naive approach to generating individual factored policies seems likely to lead to a large number of coordination errors.

7.3. Concluding Remarks

This thesis explores the use of communication for the coordination of cooperative multi-agent teams. We present algorithms that enable agents to avoid coordination errors and reason about **when** and **what** to communicate. This thesis supports the claim that cooperative teams, modeled by Dec-POMDPs and Dec-MDPs, can effectively coordinate their action choices by reasoning about communication at execution time. Pushing off communication decisions to execution time assists in developing heuristic solutions for these highly intractable models, while the Dec-POMDP and Dec-MDP models provide a framework for reasoning about **when** and **what** agents should communicate.

The overall approach presented in this thesis is to generate centralized plans for decentralized teams, treating the agents as if they will be able to communicate all of their observations to all of their teammates. Then, at execution time, the agents reason about **when** and **what** to communicate so as to enable decentralized execution of these centralized policies. There are several tradeoffs inherent to this approach. One the one hand, it is computationally easier to generate policies for centralized teams than for decentralized teams. On the other, pushing off communication decisions to execution time requires agents to do a significant amount of reasoning during execution. However, making communication decisions at execution-time is easier than planning for communication prior to execution. Instead of reasoning about all possible messages that agents could construct and choose to communicate, an agent making a communication decision at execution-time needs to consider only the observation history that it has actually observed.

This thesis presents **heuristic** algorithms for coordinating agent actions and making communication decisions. Teams executing our algorithms may behave sub-optimally, choosing in some cases to communicate more than teams controlled by an optimal policy, and in some cases choosing to communicate less. Consequently, we do not guarantee that our algorithms will enable agents to make optimal action choices. We do, however,

guarantee that agents will make **coordinated** action choices. In our work, we ensure that agents will avoid all potential coordination errors, while attempting to make efficient use of the communication resources available to them.

Currently, it is computationally infeasible to apply the algorithms presented in this thesis to real-world robot or agent problems. In addition to the computational complexity of the algorithms themselves, there is a significant computational challenge involved in solving even the centralized POMDPs and MDPs that serve as input to our algorithms. We believe, however, that this thesis presents principled first steps towards solving the problem of reasoning about the effective use of limited communication resources.

BIBLIOGRAPHY

Allen, M. and Zilberstein, S. (2007). Agent influence as a predictor of difficulty for decentralized problem-solving. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI)*.

Åström, K. J. (1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*.

Becker, R., Lesser, V., and Zilberstein, S. (2005). Analyzing myopic approaches for multi-agent communication. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*.

Becker, R., Zilberstein, S., Lesser, V., and Goldman, C. V. (2003). Transition-independent decentralized Markov Decision Processes. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*.

Becker, R., Zilberstein, S., Lesser, V., and Goldman, C. V. (2004). Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of centralized control of Markov decision processes. *Mathematics of Operations Research*.

Bernstein, D. S., Hansen, E. A., and Zilberstein, S. (2005). Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*.

Bernstein, D. S., Zilberstein, S., and Immerman, N. (2000). The complexity of decentralized control of Markov Decision Processes. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Bhasin, K., Hayden, J., Agre, J. R., Clare, L. P., and Yan, T. Y. (2001). Advanced communication and networking technologies for Mars exploration. In *International Communication Satellite Systems Conference and Exhibit*.

Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*.

Boutilier, C. and Dearden, R. (1996). Approximating value trees in structured dynamic programming. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*.

Boutilier, C., Dearden, R., and Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. (1996). Context-specific independence in Bayesian networks. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Bratman, M. E. (1987). *Intention, Plans and Practical Reason*. Harvard University Press.

Cassandra, A. R. (2005). *pomdp-solve*.
http://www.cassandra.org/pomdp/code/index.shtml.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI)*.

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence Journal*.

Dias, M. B. and Stentz, A. (2000). A free market architecture for distributed control of a multirobot system. In *Proceedings of the Sixth International Conference on Intelligent Autonomous Systems (IAS)*.

Dias, M. B., Zlot, R. M., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*.

Dietterich, T. G. (1998). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*.

Doshi, P. and Gmytrasiewicz, P. (2005). A particle filtering algorithms for Interactive POMDPs. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*.

Doshi, P., Zeng, Y., and Chen, Q. (2007). Graphical models for online solutions to Interactive POMDPs. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Emery-Montemerlo, R. (2005). *Game-Theoretic Control for Robot Teams*. PhD thesis, Carnegie Mellon University.

Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings*

of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).

Fujita, M., Veloso, M., Uther, W., Asada, M., Kitanon, H., Hugel, V., Bonnin, P., Bouramoué, J.-C., and Blazevic, P. (2000). Vision, strategy, and localization using the Sony legged robots at RoboCup-98. *AI Magazine*.

Georgeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M. (1999). The belief-desire-intention model of agency. In *Proceedings of Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL)*.

Gerkey, B. and Mataric, M. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotic Research*.

Ghavamzadeh, M. and Mahadevan, S. (2004). Learning to act and communicate in cooperative multiagent systems using hierarchical reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Gmytrasiewicz, P. J. and Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*.

Goldberg, D., Cicirello, V., Dias, M. B., Simmons, R., Smith, S., and Stentz, A. (2003). Market-based multi-robot planning in a distributed layered architecture. In *Proceedings of the Multi-Robot Systems Workshop*.

Goldman, C. V. and Zilberstein, S. (2003). Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conferences on Autonomous Agents and Multi-agent Systems (AAMAS)*.

Goldman, C. V. and Zilberstein, S. (2004). Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*.

Guestrin, C. and Gordon, G. (2002). Distributed planning in hierarchical factored MDPs. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Guestrin, C., Koller, D., Parr, R., and Venkataraman, S. (2003). Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*.

Guestrin, C., Venkataraman, S., and Koller, D. (2002). Context-specific multiagent coordination and planning with factored MDPs. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*.

Gustafson, D. and Matson, E. (2003). Taxonomy of cooperative robotic systems. In *IEEE International Conference on Systems, Man and Cybernetics*.

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*.

Hansen, E. A. and Feng, Z. (2000). Dynamic programming for POMDPs using a factored state representation. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable domains. *Artificial Intelligence*.

Kalra, N., Ferguson, D., and Stentz, A. (2005). Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Koller, D. and Parr, R. (1999). Computing factored value functions for policies in structured MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*.

Lagoudakis, M. and Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*.

Lenser, S. and Veloso, M. (2000). Sensor resetting localization for poorly modeled mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Lenser, S. and Veloso, M. (2003). Visual sonar: Fast obstacle avoidance using monocular vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*.

Matarić, M., Nilsson, M., and Simsarian, K. (1995). Cooperative multi-robot box-pushing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Nair, R., Ito, T., Tambe, M., and Marsella, S. (2000). Robocup-rescue: A proposal and preliminary experiences. In *Proceedings of RoboCup-Rescue Workshop, Fourth International Conference on Multi-Agent Systems (ICMAS)*.

Nair, R., Pynadath, D., Yokoo, M., Tambe, M., and Marsella, S. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*.

Nair, R., Roth, M., Yokoo, M., and Tambe, M. (2004). Communication for improving policy computation in distributed POMDPs. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*.

Nair, R., Varakantham, P., Tambe, M., and Yokoo, M. (2005). Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*.

Oliehoek, F. A. and Vlassis, N. (2007). Q-value functions for decentralized POMDPs. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Ooi, J. M. and Wornell, G. W. (1996). Decentralized control of a multiple access broadcast channel. In *Proceedings of the 25th IEEE Conference on Decision and Control (CDC)*.

Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*.

Parker, L. (1998). ALLIANCE: An architecture of fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*.

Peshkin, L., Kim, K.-E., Meuleau, N., and Kaelbling, L. P. (2000). Learning to cooperate via policy search. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*.

Poupart, P., Ortiz, L. E., and Boutilier, C. (2001). Value-directed sampling methods for monitoring POMDPs. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Pynadath, D. V. and Tambe, M. (2002). The communicative multiagent team decision Problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rosencrantz, M., Gordon, G., and Thrun, S. (2003). Decentralized sensor fusion with distributed particle filters. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Roth, M., Simmons, R., and Veloso, M. (2005). Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*.

Roth, M., Simmons, R., and Veloso, M. (2006). What to communicate? Execution-time decision in multi-agent POMDPs. In *The Eighth International Symposium on Distributed*

Autonomous Robotic Systems (DARS).

Roth, M., Simmons, R., and Veloso, M. (2007). Exploiting factored representations for decentralized execution in multi-agent teams. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*.

Roth, M., Vail, D., and Veloso, M. (2003). A real-time world model for multi-robot teams with high-latency communication. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Sellner, B., Heger, F. W., Hiatt, L. M., Simmons, R., and Singh, S. (2006). Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE*.

Seuken, S. and Zilberstein, S. (2007). Memory-bounded dynamic programming for DEC-POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Shen, J., Lesser, V., and Carver, N. (2003). Minimizing communication cost in a Distributed Bayesian Network using a decentralized MDP. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Simmons, R. and Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*.

Smith, T. (2007). ZMDP.
http://www.cs.cmu.edu/ trey/zmdp/.

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

Szer, D. and Charpillet, F. (2006). Point-based dynamic programming for DEC-POMDPs. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.

Szer, D., Charpillet, F., and Zilberstein, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Thrun, S. (2000). Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*.

Uther, W., Lenser, S., Bruce, J., Hock, M., and Veloso, M. (2002). CM-Pack'01: Fast legged robot walking, robust localization, and team behaviors. In Birk, A., Coradeschi, S., and Tadokoro, S. (Eds.), *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag.

Vail, D. and Veloso, M. (2003). Dynamic multi-robot coordination. In *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*. Kluwer Academic Publishers.

Varakantham, P., Nair, R., Tambe, M., and Yokoo, M. (2005). Winning back the CUP for distributed POMDPs: Planning over continuous belief spaces. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Veloso, M., Uther, W., Fujita, M., Asada, M., and Kitano, H. (1998). Playing soccer with legged robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Vig, L. and Adams, J. A. (2006). Multi-robot coalition formation. *IEEE Transactions on Robotics*.

Xuan, P. and Lesser, V. (2002). Multi-agent policies: From centralized ones to decentralized ones. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*.

Xuan, P., Lesser, V., and Zilberstein, S. (2001). Communication decisions in multi-agent cooperation: model and experiments. In *Proceedings of the Fifth International Conference on Autonomous Agents*.

APPENDIX A

Experimental Domains

This section provides additional details about the domains used in the experiments described in this thesis.

A.1. Multi-agent Tiger Domain

The multi-agent tiger domain (Nair et al., 2003) was defined as an extension of the single-agent tiger domain that has served as a common POMDP benchmark problem (Cassandra et al., 1994). Several modifications of the basic tiger problem have been used as test domains, for both single-agent and multi-agent algorithms (e.g. tiger-grid (Littman et al., 1995) and persistent multi-agent tiger (Doshi et al., 2007)). The experiments performed in this thesis work were done on the basic two- and three-agent tiger problem (Nair et al., 2003), with some parameters changed in the observation function. In this work, the likelihood of a correct observation (e.g., observing HEARLEFT when the tiger is behind the left door ($s = \text{SL}$)) is set such that, in a centralized problem, one consistent joint observation, meaning that the two robots both heard the tiger behind the same door, is sufficient to prompt the team to open a door. However, in the distributed case, one observation would not be sufficient to prompt an agent to communicate to its teammate.

- $\alpha = 2$
- $\mathcal{S} = \{\text{SL}, \text{SR}\}$
- $\mathcal{A}_i = \{\text{OPENL}, \text{OPENR}, \text{LISTEN}\}$
- $\mathcal{P} =$

Action	**State**	SL	SR
$\langle \text{OPENL}, * \rangle$	*	0.5	0.5
$\langle \text{OPENR}, * \rangle$	*	0.5	0.5
$\langle *, \text{OPENL} \rangle$	*	0.5	0.5
$\langle *, \text{OPENR} \rangle$	*	0.5	0.5
$\langle \text{LISTEN}, \text{LISTEN} \rangle$	SL	1.0	0.0
$\langle \text{LISTEN}, \text{LISTEN} \rangle$	SR	0.0	1.0

- $\Omega_i = \{\text{HEARLEFT}, \text{HEARRIGHT}\}$
- $\mathcal{O}_i =$

Action	**State**	HEARLEFT	HEARRIGHT
$\langle \text{OPENL}, * \rangle$	*	0.5	0.5
$\langle \text{OPENR}, * \rangle$	*	0.5	0.5
$\langle *, \text{OPENL} \rangle$	*	0.5	0.5
$\langle *, \text{OPENR} \rangle$	*	0.5	0.5
$\langle \text{LISTEN}, \text{LISTEN} \rangle$	SL	0.7	0.3
$\langle \text{LISTEN}, \text{LISTEN} \rangle$	SR	0.3	0.7

- $\mathcal{R} =$

Action	SL	SR
$\langle \text{OPENL}, \text{OPENL} \rangle$	-50	+20
$\langle \text{OPENL}, \text{OPENR} \rangle$	-100	-100
$\langle \text{OPENL}, \text{LISTEN} \rangle$	-101	+9
$\langle \text{OPENR}, \text{OPENL} \rangle$	-100	-100
$\langle \text{OPENR}, \text{OPENR} \rangle$	+20	-50
$\langle \text{OPENR}, \text{LISTEN} \rangle$	+9	-101
$\langle \text{LISTEN}, \text{OPENL} \rangle$	-101	+9
$\langle \text{LISTEN}, \text{OPENR} \rangle$	+9	-101
$\langle \text{LISTEN}, \text{LISTEN} \rangle$	-2	-2

The three-agent tiger problem differs from the two-agent problem only in the observation function.

- $\alpha = 3$
- $\mathcal{O}_i =$

Action	**State**	HEARLEFT	HEARRIGHT
$\langle \text{OPENL}, *, * \rangle$	*	0.5	0.5
$\langle \text{OPENR}, *, * \rangle$	*	0.5	0.5
$\langle *, \text{OPENL}, * \rangle$	*	0.5	0.5
$\langle *, \text{OPENR}, * \rangle$	*	0.5	0.5
$\langle *, *, \text{OPENL} \rangle$	*	0.5	0.5
$\langle *, *, \text{OPENR} \rangle$	*	0.5	0.5
$\langle \text{LISTEN}, \text{LISTEN}, \text{LISTEN} \rangle$	SL	0.65	0.35
$\langle \text{LISTEN}, \text{LISTEN}, \text{LISTEN} \rangle$	SR	0.35	0.65

A.2. Multi-access Broadcast Channel Domain

The multi-access broadcast channel problem (MABC) (Ooi and Wornell, 1996) was formulated as a Dec-POMDP domain (Hansen et al., 2004), and has since been used as a common benchmark domain (Bernstein et al., 2005; Szer and Charpillet, 2006; Seuken and Zilberstein, 2007). In this domain, two agents are trying to broadcast messages to each other across a shared channel. Only one message can be broadcast at a time. Each agent may be in one of two states: EMPTY, meaning that its message buffer is empty, and FULL, meaning that the agent has a message in its buffer to send. New messages appear in the agents' buffers with different probabilities: 0.9 for agent 1, and 0.1 for agent 2.

Agents have two possible actions: SEND and DONTSEND. SENDing a message fails if both agents' buffers are FULL and they attempt to SEND simultaneously. Otherwise, SEND succeeds, earning the team a reward of +1. Agents noisily observe collisions. If both agents attempt to SEND messages simultaneously, with probability 0.9, they will observe CONFLICT, indicating a need to coordinate message-SENDing in the next timestep.

- $\alpha = 2$
- $\mathcal{S}_i = \{\text{EMPTY}, \text{FULL}\} \quad \rightarrow \quad \mathcal{S} = \{EE, EF, FE, FF\}$
- $\mathcal{A}_i = \{\text{DONTSEND}, \text{SEND}\}$
- $\mathcal{P} =$

Action	**State**	EE	EF	FE	FF
*	EE	0.09	0.01	0.81	0.09
$\langle *, \text{DONTSEND} \rangle$	EF	0.0	0.1	0.0	0.9
$\langle *, \text{SEND} \rangle$	EF	0.09	0.01	0.81	0.09
$\langle \text{DONTSEND}, * \rangle$	FE	0.0	0.0	0.9	0.1
$\langle \text{SEND}, * \rangle$	FE	0.09	0.01	0.81	0.09
$\langle \text{DONTSEND}, \text{DONTSEND} \rangle$	FF	0.0	0.0	0.0	1.0
$\langle \text{DONTSEND}, \text{SEND} \rangle$	FF	0.0	0.0	0.9	0.1
$\langle \text{SEND}, \text{DONTSEND} \rangle$	FF	0.0	0.1	0.0	0.9
$\langle \text{SEND}, \text{SEND} \rangle$	FF	0.0	0.0	0.0	1.0

- $\Omega_i = \{\text{NOCONFLICT}, \text{CONFLICT}\}$
- $\mathcal{O}_i =$

Action	**State**	NOCONFLICT	CONFLICT
*	EE	1.0	0.0
*	EF	1.0	0.0
*	FE	1.0	0.0
$\langle \text{DONTSEND}, * \rangle$	FF	1.0	0.0
$\langle *, \text{DONTSEND} \rangle$	FF	1.0	0.0
$\langle \text{SEND}, \text{SEND} \rangle$	FF	0.1	0.9

- $\mathcal{R} =$

Action	EE	EF	FE	FF
⟨DONTSEND, DONTSEND⟩	0	0	0	0
⟨DONTSEND, SEND⟩	0	+1	0	+1
⟨SEND, DONTSEND⟩	0	0	+1	+1
⟨SEND, SEND⟩	0	+1	+1	0

A.3. Colorado/Wyoming Domain

We introduced the Colorado/Wyoming domain (Roth et al., 2006) to serve both as a larger test domain than the two-agent tiger problem and also because it has observations of varying information qualities. The presence of some observations that are more valuable than others allows us to use the Colorado/Wyoming domain to demonstrate the effectiveness of the SELECTIVE DEC-COMM algorithm.

The overall state in the Colorado/Wyoming domain can be divided into two components, the fully observable or known component, k, which includes the position of the two agents within the 5-by-5 grid world, and a partially observable component, p, namely whether the agents are in Colorado or Wyoming. For compactness, the domain is described here via a factored representation. For our experiments, however, we used Q-MDP (Littman et al., 1995) to estimate the POMDP value function needed by Q-POMDP and SELECTIVE ACE-PJB-COMM. In this domain, agents' movement actions succeed deterministically. If either agent SIGNALs, the problem resets to a new, random state.

- $\alpha = 2$
- $\mathcal{S} = \{\langle p, k \rangle\}$

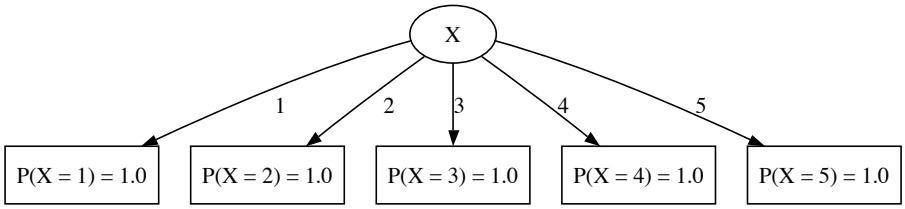
  $p = \{\text{COLORADO}, \text{WYOMING}\}$

  $k = \langle \mathcal{X}_0, \mathcal{Y}_0, \mathcal{X}_1, \mathcal{Y}_1 \rangle, \mathcal{X}_0, \mathcal{Y}_0, \mathcal{X}_1, \mathcal{Y}_1 \in [1, 5]$
- $\mathcal{A}_i = \{\text{NORTH}, \text{SOUTH}, \text{EAST}, \text{WEST}, \text{STOP}, \text{SIGNAL}\}$
- $\mathcal{P} =$

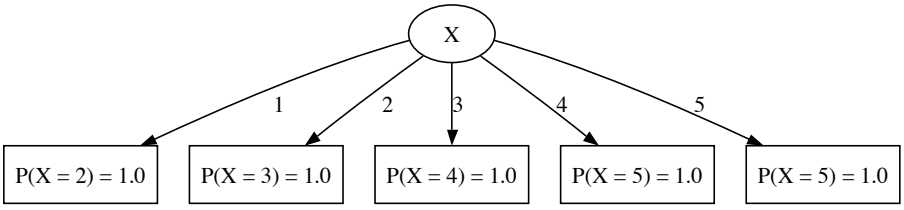
Action	p	COLORADO	WYOMING
*	COLORADO	1.0	0.0
*	WYOMING	0.0	1.0
$\langle$SIGNAL, $*\rangle$	*	0.5	0.5
$\langle *,$ SIGNAL$\rangle$	*	0.5	0.5

  - $\mathcal{X}_i$:

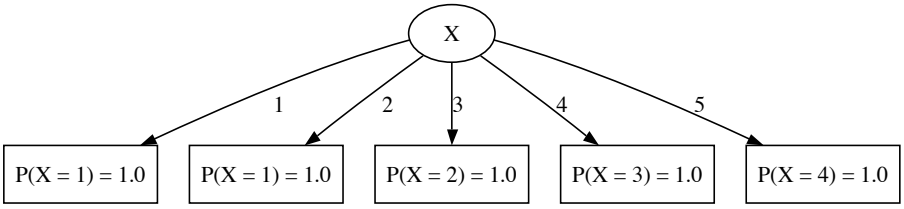
    $*$ $a_i \in \{\text{NORTH}, \text{SOUTH}, \text{STOP}\}, a_{1-i} \neq \text{SIGNAL}$:



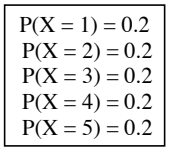
* $a_i = \text{EAST}, a_{1-i} \neq \text{SIGNAL}$:



* $a_i = \text{WEST}, a_{1-i} \neq \text{SIGNAL}$:

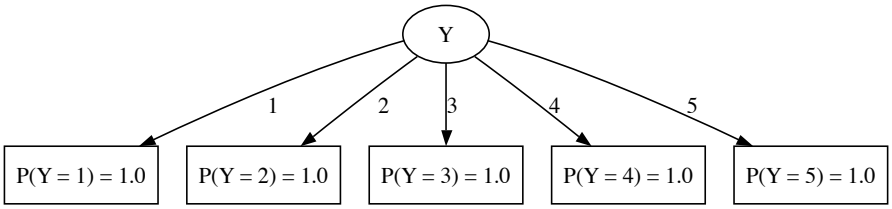


* $a_i = \text{SIGNAL}$:

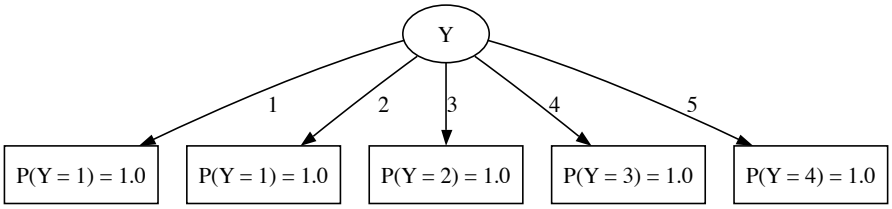


– $\mathcal{Y}^i$:

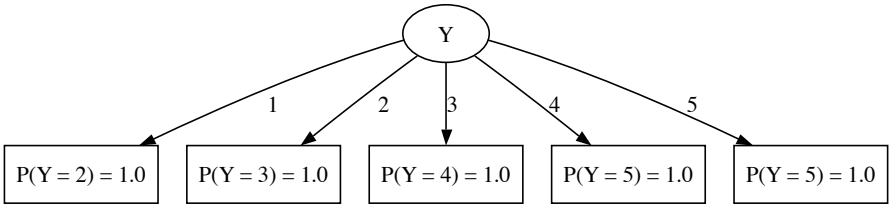
* $a_i \in \{\text{EAST}, \text{WEST}, \text{STOP}\}, a_{1-i} \neq \text{SIGNAL}$:



* $a_i = \text{NORTH}, a_{1-i} \neq \text{SIGNAL}$:



* $a_i = \text{SOUTH}, a_{1-i} \neq \text{SIGNAL}$:



$*$ $a_i = *, a_{1-i} =$ SIGNAL:

P(Y = 1) = 0.2
P(Y = 2) = 0.2
P(Y = 3) = 0.2
P(Y = 4) = 0.2
P(Y = 5) = 0.2

- $\Omega_i = \{$MOUNTAIN, PIKESPEAK, PLAIN, OLDFAITHFUL$\}$
- $\mathcal{O}_i =$

Action	p	MOUNTAIN	PIKESPEAK	PLAIN	OLDFAITHFUL
$\langle$SIGNAL, $*\rangle$	*	0.25	0.25	0.25	0.25
$\langle *,$ SIGNAL$\rangle$	*	0.25	0.25	0.25	0.25
$a_0, a_1 \neq$ SIGNAL	COLORADO	0.5	0.15	0.3	0.05
$a_0, a_1 \neq$ SIGNAL	WYOMING	0.3	0.05	0.5	0.15

- $\mathcal{R} =$

Action	**State**	**Reward**
$a_0, a_1 \neq$ SIGNAL $a_0, a_1 \neq$ STOP	*	-2
$a_0, a_1 \neq$ SIGNAL $a_0 =$ STOP$, a_1 \neq$ STOP	*	-1
$a_0, a_1 \neq$ SIGNAL $a_0 \neq$ STOP$, a_1 =$ STOP	*	-1
$a_0, a_1 =$ STOP	*	0
$a_0 =$ SIGNAL$, a_1 \neq$ SIGNAL	*	-100
$a_0 \neq$ SIGNAL$, a_1 =$ SIGNAL	*	-100
$a_0, a_1 =$ SIGNAL	$p =$ COLORADO $\mathcal{X}_0, \mathcal{X}_1 = 4 \wedge \mathcal{Y}_0, \mathcal{Y}_1 = 2$	+20
$a_0, a_1 =$ SIGNAL	$p =$ COLORADO $\mathcal{X}_0, \mathcal{X}_1 \neq 4 \vee \mathcal{Y}_0, \mathcal{Y}_1 \neq 2$	-50
$a_0, a_1 =$ SIGNAL	$p =$ WYOMING $\mathcal{X}_0, \mathcal{X}_1 = 5 \wedge \mathcal{Y}_0, \mathcal{Y}_1 = 5$	+20
$a_0, a_1 =$ SIGNAL	$p =$ WYOMING $\mathcal{X}_0, \mathcal{X}_1 \neq 5 \vee \mathcal{Y}_0, \mathcal{Y}_1 \neq 5$	-50

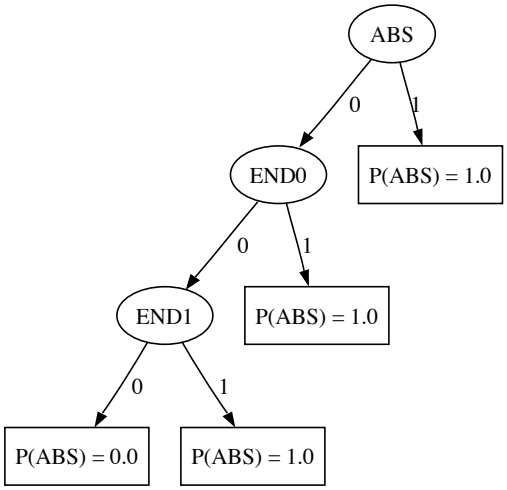
A.4. Meeting-Under-Uncertainty Domain

A number of different Meeting-Under-Uncertainty problems have served as experimental domains in the Dec-POMDP literature (Xuan et al., 2001; Goldman and Zilberstein, 2003; Bernstein et al., 2005). We introduced this variant to serve as a simple example of a multi-agent domain with a great deal of context-specific independence between agents (Roth et al., 2007) . Below, we show the details for a two-agent, 3-by-3 Meeting-Under-Uncertainty domain, but the structure remains the same for problems with more agents or in larger grid worlds.

In principle, the state space of this Meeting-Under-Uncertainty domain consists only of the $\mathcal{X}$ and $\mathcal{Y}$ positions of the agents and an additional variable, ABS, indicating that at least one agent performed a SIGNAL action, transitioning the problem to an absorbing state. However, because we used Structured Policy Iteration to generate a tree-structured MDP policy for the domain, and SPI requires domains in which the reward function depends only on the state and not on the state and current action (Boutilier et al., 2000), we augment the domain with an additional state variable for each agent. This variable, END_i, indicates that agent i SIGNALed in the immediately previous timestep. Motion actions in this domain are non-deterministic. Movement attempts succeed with probability 0.9, but fail, leaving the agent in its original position, with 0.1 probability.

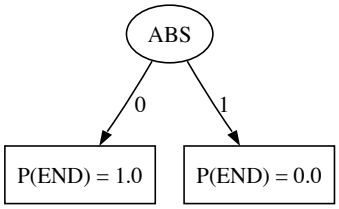
- $\alpha = 2$
- $\mathcal{S} = \{\langle \text{ABS}, \text{END}_0, \text{END}_1, \mathcal{X}_0, \mathcal{X}_1, \mathcal{Y}_0, \mathcal{Y}_1 \rangle\}$
  $\text{ABS} \in [0,1], \text{END}_i \in [0,1], \mathcal{X}_i \in [0,2], \mathcal{Y}_i \in [0,2]$
- $\mathcal{A}_i = \{\text{NORTH}, \text{SOUTH}, \text{EAST}, \text{WEST}, \text{STOP}, \text{SIGNAL}\}$
- $\mathcal{P} =$
  - ABS:

* $a = *$:

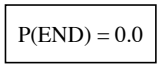


– END_i:

* $a_i = \text{SIGNAL}$:

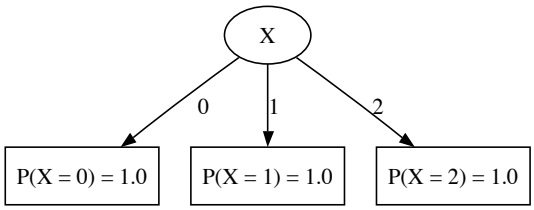


* $a_i \neq \text{SIGNAL}$:

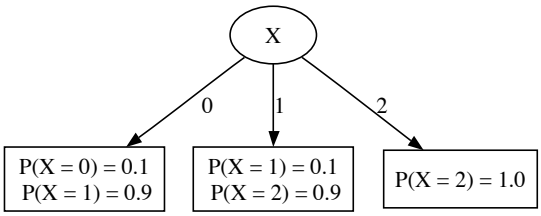


– $\mathcal{X}_i$:

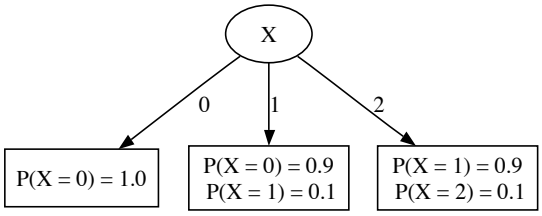
* $a_i \in \{\text{NORTH}, \text{SOUTH}, \text{STOP}, \text{SIGNAL}\}$:



* $a_i = \text{EAST}$:

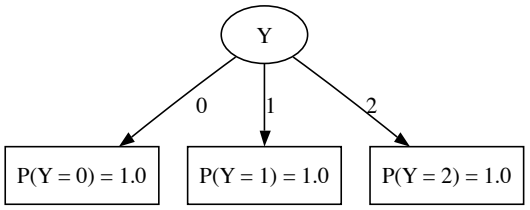


* $a_i =$ WEST:

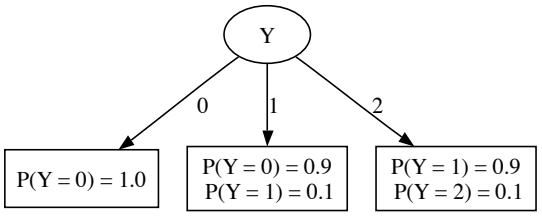


– $\mathcal{Y}_i$:

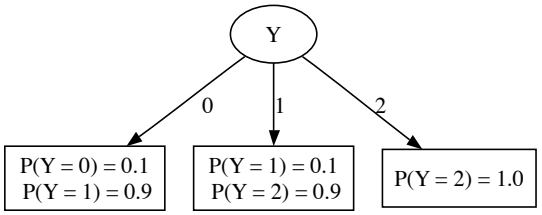
* $a_i \in \{$EAST, WEST, STOP, SIGNAL$\}$:



* $a_i =$ NORTH:

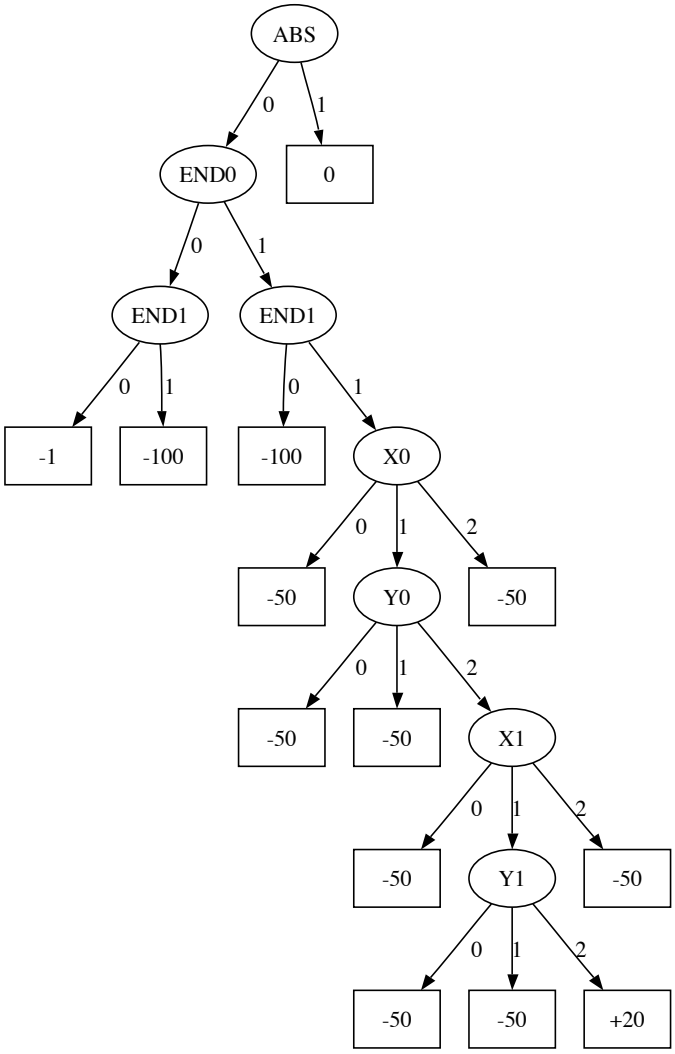


* $a_i =$ SOUTH:



- $\mathcal{O} =$ each agent i observes $\{$ABS, END$_i$, $\mathcal{X}_i$, $\mathcal{Y}_i\}$

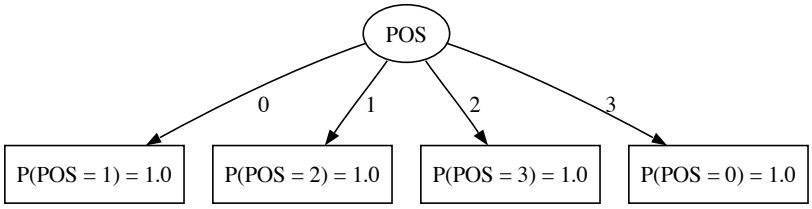
- $\mathcal{R} =$


ABS
0
1
END0
0
0
1
END1
END1
0
1
0
1
-1
-100
-100
X0
0
1
2
-50
Y0
-50
0
1
2
-50
-50
X1
0
1
2
-50
Y1
-50
0
1
2
-50
-50
+20

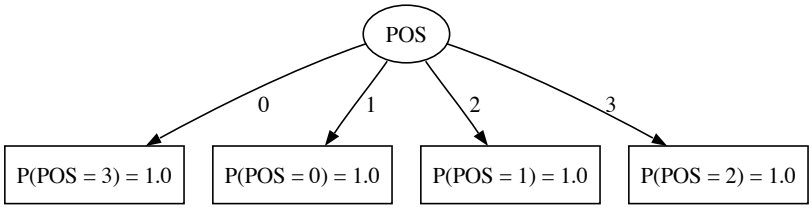
A.5. Simplified Taxi Domain

In this domain, two taxi agents are moving between four base stations. Taxis can move CLOCKWISE or COUNTERCLOCKWISE, with movements succeeding deterministically. New passengers appear at each base station with probability 0.25 in each timestep. The taxis must PICKUP passengers and deliver them to their GOAL base stations. The GOAL of a passenger is assigned randomly when it is picked up. If both taxis attempt to PICKUP the same passenger simultaneously, Taxi 1 will succeed. The taxis accumulate a reward of +20 for successfully delivering a passenger to its desired GOAL. There is a penalty of -100 for delivering a passenger to an incorrect GOAL location. All other movements and actions have a cost of -1.

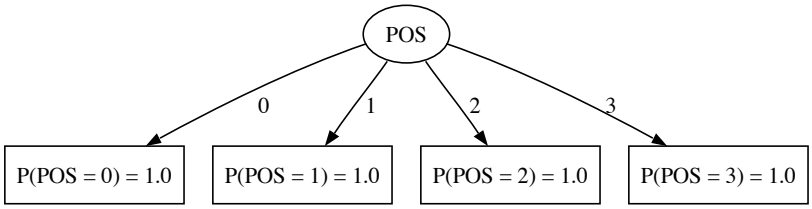
- $\alpha = 2$
- $\mathcal{S} = \{\langle \text{POS}_0, \text{POS}_1, \text{HASP}_0, \text{HASP}_1, \text{GOAL}_0, \text{GOAL}_1, \text{BASE0}, \text{BASE1}, \text{BASE2}, \text{BASE3} \rangle\}$
  $\text{POS}_i \in [0,3], \text{HASP}_i \in [0,1], \text{GOAL}_i \in [0,3], \text{BASE}n \in [0,1]$
- $\mathcal{A}_i = \{\text{CLOCKWISE}, \text{COUNTERCLOCKWISE}, \text{PICKUP}, \text{DELIVER}\}$
- $\mathcal{P} =$
  - POS_i:
    - $a_i = \text{CLOCKWISE}$:



    - $a_i = \text{COUNTERCLOCKWISE}$:

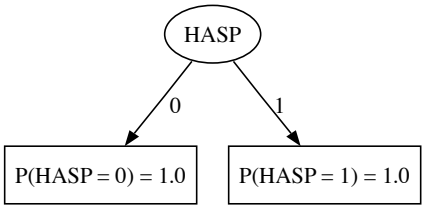


    - $a_i \in \{\text{PICKUP}, \text{DELIVER}\}$:

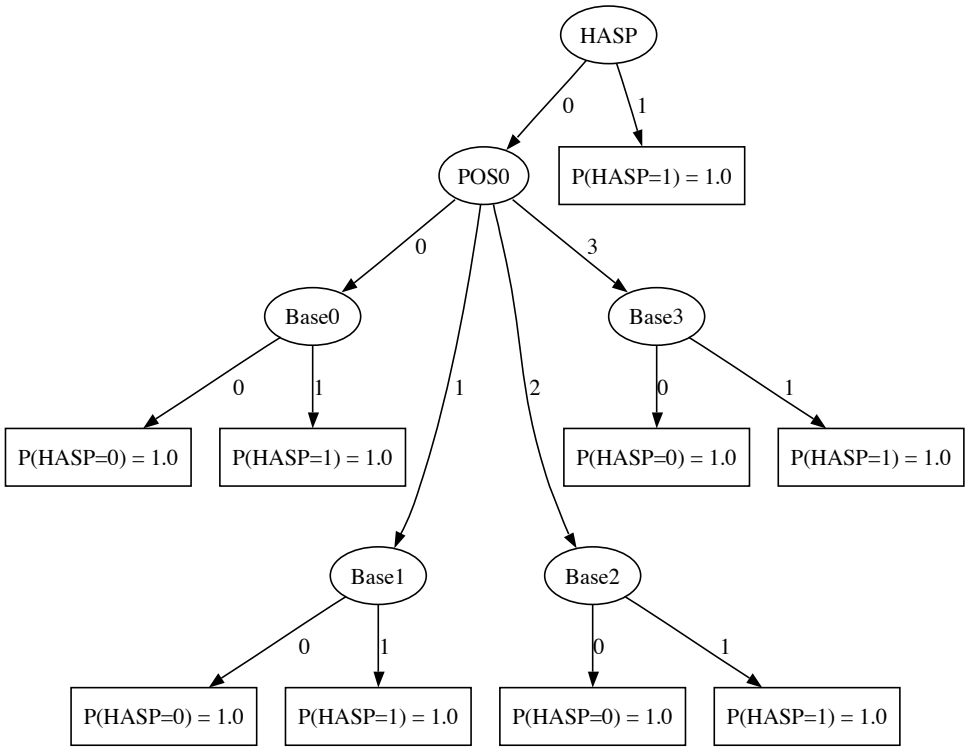


  - HASP_0:

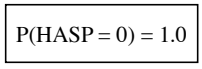
* $a_0 \in \{\text{CLOCKWISE}, \text{COUNTERCLOCKWISE}\}$:



* $a_0 = \text{PICKUP}$:

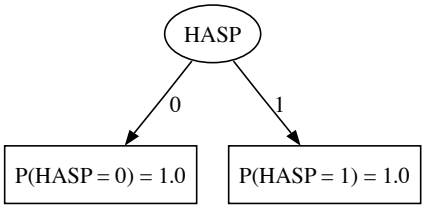


* $a_0 = \text{DELIVER}$:

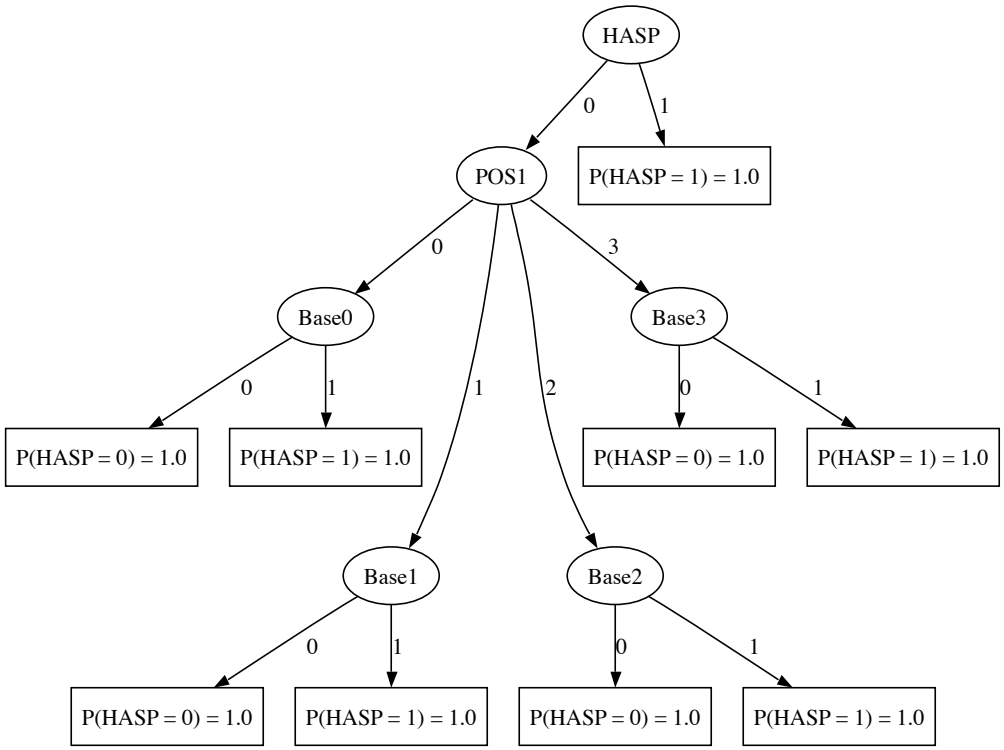


– HASP_1:

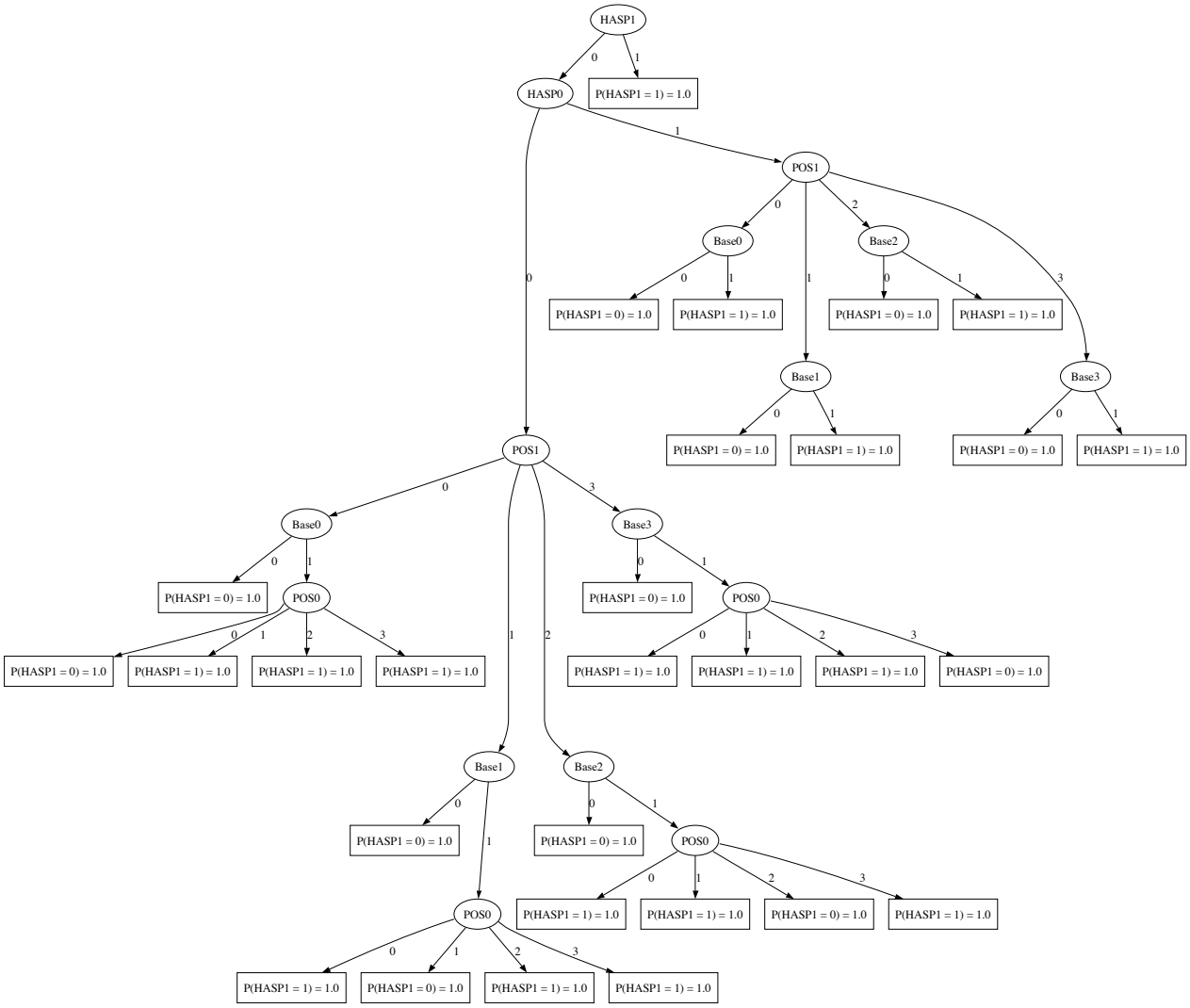
* $a_1 \in \{\text{CLOCKWISE}, \text{COUNTERCLOCKWISE}\}$:



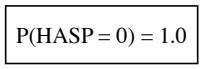
$*$ $a_0 \neq$ PICKUP, $a_1 =$ PICKUP:



∗ $a_0 = \text{PICKUP}, a_1 = \text{PICKUP}$:

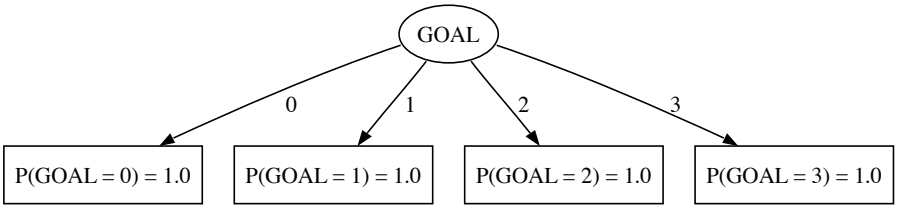


∗ $a_1 = \text{DELIVER}$:

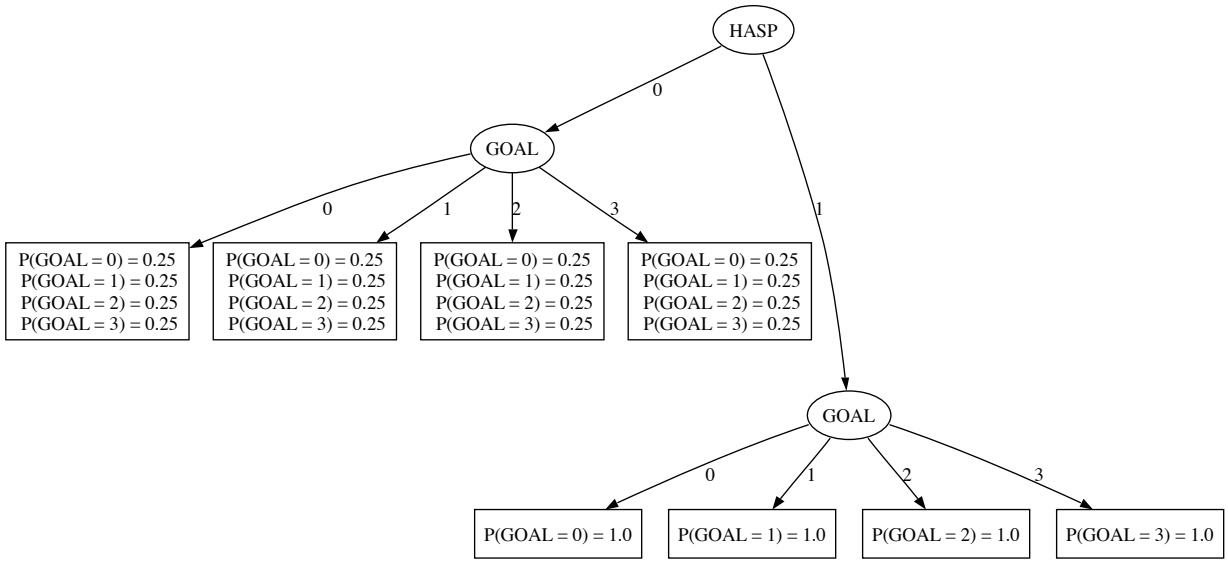


– GOAL_i:

∗ $a_i \in \{\text{CLOCKWISE}, \text{COUNTERCLOCKWISE}, \text{DELIVER}\}$:

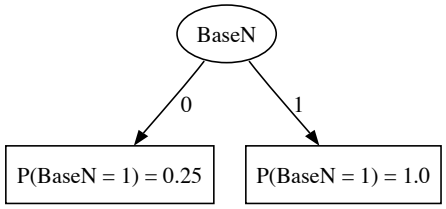


* $a_i =$ PICKUP:

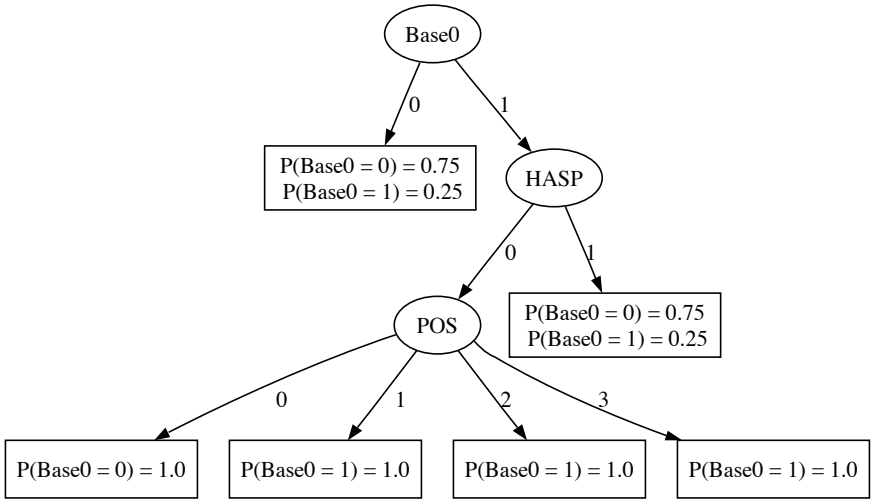


– BASEn:

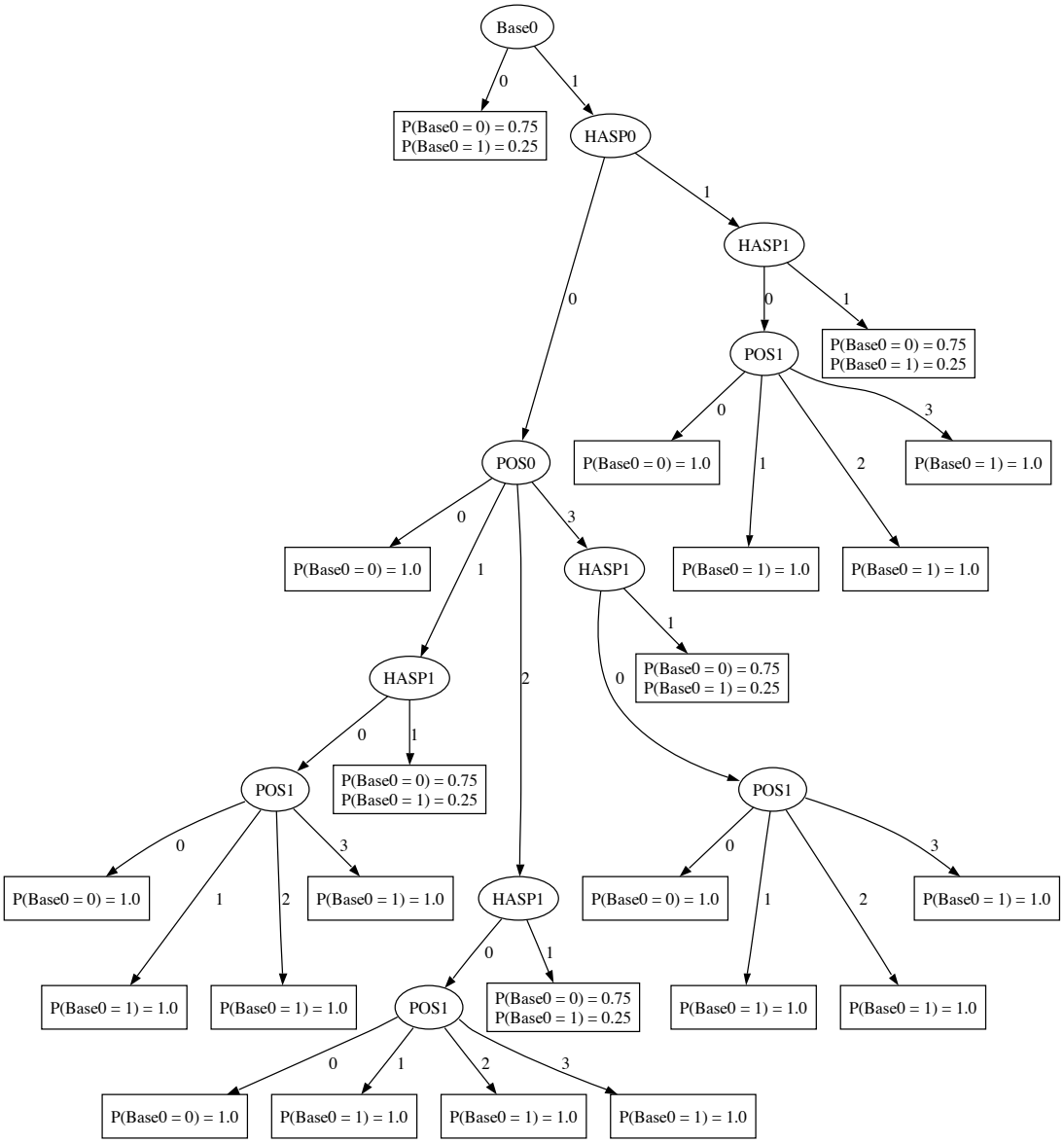
* $a_i \neq$ PICKUP $\forall i$:



* $a_i =$ PICKUP, $a_{1-i} \neq$ PICKUP:



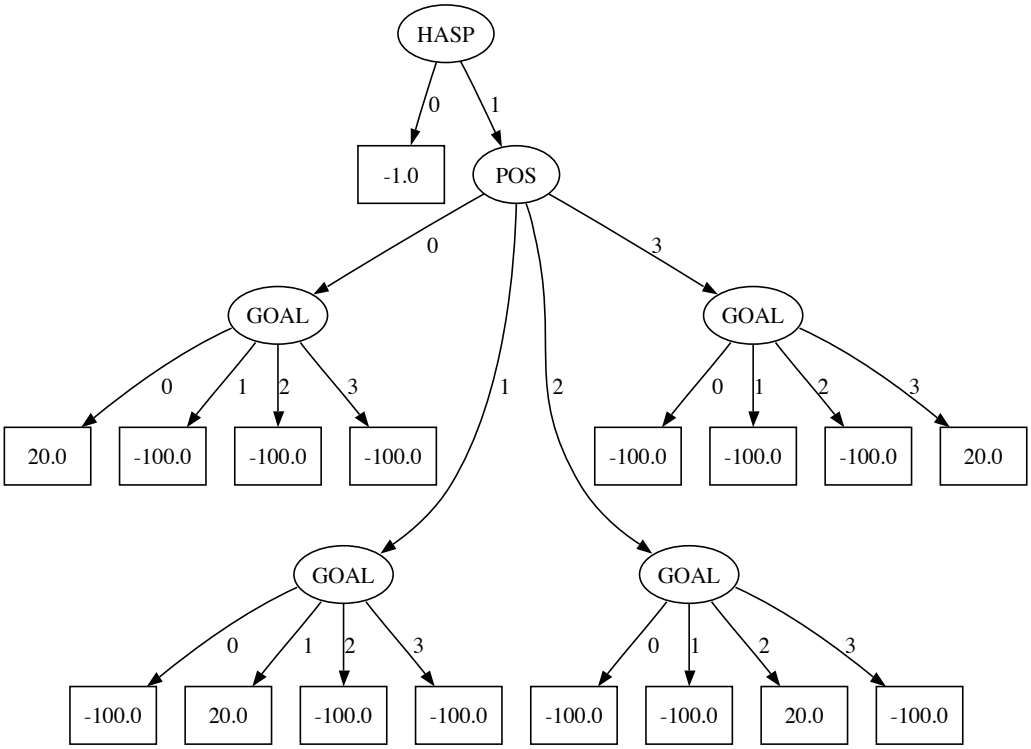
$*$ $a_0 = \text{PICKUP}, a_1 = \text{PICKUP}$:



- $\mathcal{O} =$, each agent i observes $\{\text{POS}_i, \text{HASP}_i, \text{GOAL}_i\}$. Agent 0 observes BASE0 and BASE1. Agent 1 observes BASE2 and BASE3.
- $\mathcal{R}_i =$
  - $a_i \in \{\text{CLOCKWISE}, \text{COUNTERCLOCKWISE}, \text{PICKUP}\}$:

-1.0

– $a_i = \text{DELIVER}$:



- $\mathcal{R} = \mathcal{R}_0 + \mathcal{R}_1$