# Exploiting Structure in Coordinating Multiple Decision Makers

# EXPLOITING STRUCTURE IN COORDINATING MULTIPLE DECISION MAKERS

A Dissertation Presented

by

HALA MOSTAFA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2011

Department of Computer Science

# EXPLOITING STRUCTURE IN COORDINATING MULTIPLE DECISION MAKERS

A Dissertation Presented

by

HALA MOSTAFA

Approved as to style and content by:

Victor Lesser, Chair

Shlomo Zilberstein, Member

Gerome Miklau, Member

Ana Muriel, Member

Andrew G. Barto, Department Chair
Department of Computer Science

# ACKNOWLEDGMENTS

One thing that so delayed my finishing this thesis is writing the acknowledgements. I just cannot begin to thank the following persons, because I don't think I can ever thank them enough. But here goes my modest attempt anyway.

I would like to thank my advisor and mentor Victor Lesser. Victor's eye for analogies between seemingly very different areas or approaches has been invaluable to my research and has opened up many an interesting research direction. His high standards of integrity and innovation will stay with me to the end of my career. Besides research advice, Victor has infinite capacity for caring about his students; I don't remember anything I was grappling with for which Victor didn't have a good piece of advice.

I would also like to thank Shlomo Zilberstein and Dan Corkill with whom I have been lucky to discuss my work and whose comments were useful to me in defining my ideas and making them more concrete. I am also grateful for Anita Raja's career advice and numerous emails reminding me to enjoy myself even in stressful situations.

I owe a lot to the liveliness and friendliness of the CS Department atmosphere (never mind that the heater and the AC were always competing), and I couldn't have picked a better place to spend so many years of my life. I want to thank my friends and colleagues in the MAS Lab: Bo An, Chongjie Zhang, Yoonheui Kim and Huzaifa Zafar for many interesting and useful discussions. I was very lucky to more or less belong to the RBR lab as well, where I could walk and chat for arbitrarily long periods of time about everything under the sun, whether or not it had 'MDP' in it. To Alan Carlin, Marek Petrik, Akshat Kumar, Siddarth Srivastava, William Yeoh and Chris Amato, thank you for the extremely enjoyable talks, dinners and hikes and for sharing stories of everything from fantasy baseball to mountain biking.

Michele Roberts made everything administrative just glide without us having to think about it twice. Thank you for your amazing efficiency, and for the air of cheerfulness you invariably brought into our labs.

Finally, a mere attempt at thanking my parents, brother and husband for their unfaltering support. If it was not for my family, I would never have survived this program beyond the first few months. My daily conversations with my parents sustained me and formed my main link to the 'real world'. My brother was always willing to answer my linear algebra questions. My husband's love and unshakable faith in my abilities is something I can always count on.

# ABSTRACT

## EXPLOITING STRUCTURE IN COORDINATING MULTIPLE DECISION MAKERS

SEPTEMBER 2011

HALA MOSTAFA

B.Sc., CAIRO UNIVERSITY, EGYPT

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Victor Lesser

This thesis is concerned with sequential decision making by multiple agents, whether they are acting cooperatively to maximize team reward or selfishly trying to maximize their individual rewards. The practical intractability of this general problem led to efforts in identifying special cases that admit efficient computation, yet still represent a wide enough range of problems. In our work, we identify the class of problems with structured interactions, where actions of one agent can have non-local effects on the transitions and/or rewards of another agent. We addressed the following research questions: 1) How can we compactly represent this class of problems? 2) How can we efficiently calculate agent policies that maximize team reward (for cooperative agents) or achieve equilibrium (self-interested agents)? 3) How can we exploit structured interactions to make reasoning about communication offline tractable?

For representing our class of problems, we developed a new decision-theoretic model, Event-Driven Interactions with Complex Rewards (EDI-CR), that explicitly represents structured interactions. EDI-CR is a compact yet general representation capable of capturing problems where the degree of coupling among agents ranges from complete independence to complete dependence.

For calculating agent policies, we draw on several techniques from the field of mathematical optimization and adapt them to exploit the special structure in EDI-CR. We developed a Mixed Integer Linear Program formulation of EDI-CR with cooperative agents that results in programs much more compact and faster to solve than formulations ignoring structure. We also investigated the use of homotopy methods as an optimization technique, as well as formulation of self-interested EDI-CR as a system of non-linear equations.

We looked at the issue of communication in both cooperative and self-interested settings. For the cooperative setting, we developed heuristics that assess the impact of potential communication points and add the ones with highest impact to the agents' decision problems. Our heuristics successfully pick communication points that improve team reward while keeping problem size manageable. Also, by controlling the amount of communication introduced by a heuristic, our approach allows us to control the tradeoff between solution quality and problem size.

For self-interested agents, we look at an example setting where communication is an integral part of problem solving, but where the self-interested agents have a reason to be reticent (e.g. privacy concerns). We formulate this problem as a game of incomplete information and present a general algorithm for calculating approximate equilibrium profile in this class of games.

# TABLE OF CONTENTS

# LIST OF TABLES

**Table** **Page**

# LIST OF FIGURES

**Figure** **Page**

# CHAPTER 1

# INTRODUCTION

## 1.1 Overview

In the early days of artificial intelligence, one of the main concerns was to develop an agent - decision maker - that makes optimal decisions in response to changes in a closed environment that does not contain any other agents. However, agents are increasingly being deployed in environments containing other agents, whether software agents or physical situated agents like robots. Our decision maker therefore affects these other agents and is affected by them. As a result, it can no longer reason about its decisions in isolation; it must consider other decision makers and their effects on the environment and each other. This relatively new requirement fueled the need for ways to represent and reason about the challenging problem of multi-agent decision making. Adding a sequential aspect, whereby agents take sequences of decisions over time, further complicates the problem.

In this thesis, we are concerned with problems in multi-agent sequential decision making in which agents operate in an environment fraught with uncertainty and make sequences of decisions with long- and short-term stochastic effects. We focus on situations where there is some degree of independence among the agents' sub-problems. Specifically, we focus on situations where structured interactions among agents arise due to (relatively few) actions having non-local effects on rewards and transitions of other agents. In addition to being a characteristic of many real-world problems (e.g. in sensor networks and robotics), this loose coupling of the agents' decision processes gives us traction over what would otherwise be an intractable problem. Indeed, identifying and exploiting special structure is

a widely used approach to dealing with the prohibitive complexity of the general problem where agents' decision processes are very tightly coupled.

In our work, we address situations with both cooperative and self-interested agents. For each of these, we investigate three main issues:

1. **Representation:** because existing decision- and game-theoretic models are inadequate for representing multi-agent sequential problems with structured interactions, we developed a new model that can capture the characteristics of our problems without giving up expressive power.

2. **Solution**: we use available optimization packages to calculate optimal (for cooperative agents) or equilibrium (for self-interested agents) courses of actions. Depending on the formulation and the solver, the solution can be an approximation of the optimal/equilibrium policy. To use optimization techniques, we developed mathematical formulations that exploit the special structure of our problems to reduce the number of variables involved, resulting in formulations that are faster to solve.

3. **Communication**: we develop heuristics that make it tractable to reason about communication offline. This kind of reasoning is notorious for its difficulty, mainly due to the explosion in the size of the problem that results from considering when to communicate and what. By analyzing interactions among agents, we are able to include communication possibilities only where the agents are likely to benefit from them. The resulting problem is much smaller than a problem with communication available everywhere, but still includes enough communication to reach the level of coordination necessary for a high quality solution.

In the following sections, we give a brief background of the research areas we touch upon in our work, followed by a list of the contributions made in this thesis and the thesis outline.

## 1.2 Multi-Agent Sequential Decision Making

In this section, we give a high-level description of multi-agent sequential decision making (MSDM) and briefly describe the general models developed for this problem. We then discuss some models that attempt to circumvent the high complexity by focusing on special cases of the general problem. Finally, we discuss the issue of communication among decision makers in terms of how it is represented and reasoned about.

### 1.2.1 MSDM: Process and domains

In MSDM, there is a number of decision makers, each with a set of actions. The actions affect the environment in which the agents are deployed and can directly affect other agents as well. Actions are taken over a number of steps, or decision epochs, which may be finite or infinite. The goal ranges from being a single over-arching goal for the entire set of agents, to being defined per agent, with different agents striving to achieve different goals.

MSDM, in its most general sense, is a problem complicated by the following factors:

- Multiple decision makers. The fact that there are multiple agents that affect the same environment means that no agent can reason about its decisions in isolation from the others. Additionally, realistic situations usually involve restrictions on sensing and communication, so no one agent has a full view of the problem during execution. It then becomes important to make sure that a decision that looks good to an agent locally is also a good decision globally.

- Uncertainty. Real-life situations are rife with uncertainty, whether it is sensing uncertainty that prevents an agent from knowing exactly where it is in its environment, or action uncertainty where actions do not always have the intended outcomes. Consequently, an agent cannot just follow a linear plan. The plan needs to tell the agent what to do in every situation it can face. Another consequence of reasoning under uncertainty is that even if there is a centralized entity that produced plans for all agents

offline and each agent knows the plans of all others, during execution each agent is unsure about where the others are and what they will do.

- Sequences of actions. An action at one stage affects the set of available actions at all subsequent stages. As a result, an agent cannot be myopic and just choose the action that looks best immediately, it needs to reason about the long-term effects of an action. As the number of decision epochs increases, an agent's decision problem becomes more difficult.

- Non-local effects. As mentioned earlier, an agent's action can affect other agents in the environment, and an agent may not be free to communicate details of its progress to others (due to communication costs or availability). Therefore, to reason about its own decisions and prospects in the future, an agent needs to keep track of where it believes the others to be and thus what they will do in the future. This requirement greatly complicates an agent's reasoning process.

Recent years have witnessed a surge in research on multi-agent decision making. Self-interested agents in competitive situations are mostly of interest to game theoreticians and economists, fueled by rapid developments in areas like online auctions and agent negotiation. This branch of research is concerned with both optimal (and near-optimal) decision-making in competitive settings and with designing mechanisms to ensure that the agents do not game the system or reach undesirable equilibria [27]. Research on cooperative agents is of interest to the Artificial Intelligence community, whether working with embodied agents (e.g. robots, planetary rovers) or disembodied ones (e.g. meeting schedulers [76], agent-coordinated human teams [79], wireless sensor networks [78]).

### 1.2.2 General models

#### 1.2.2.1 Cooperative agents

There is a large body of literature on models for representing MSDM problems. For the cases where agents are cooperative, researchers have leveraged the success of the Markov

Decision Process (MDP) as a single-agent decision-theoretic model. A number of multi-agent variants of MDP have been proposed, among the earliest and most general of which are the Decentralized Partially Observable MDP (DEC-POMDP) and Decentralized MDP (DEC-MDP) [15]. Each of these models is defined by the set of world states and agent actions, the rewards associated with taking each action in each state, and the transition function for the actions. These models assume that an agent cannot find out the world state with certainty, but does receive an observation correlated with the state it is in. The models differ in that whereas DEC-MDP assumes that pooling the observations of all agents is enough to determine the world state, DEC-POMDP does not. The Multi-agent Team Decision Problem (MTDP) [70] is a model whose equivalence to DEC-POMDP was proved, under the *perfect recall* assumption [75]. Perfect recall means that a state can only be reached by a unique sequence of actions. Another model is the Multi-agent MDP [23] which assumes a global state fully observable to all agents. However, this is a very strong assumption that renders the model unrealistic.

The goal in cooperative settings is to calculate a set of policies, one per agent, that maximize the total reward of all agents. Depending on the problem, the quantity to maximize can be the sum of rewards over the finite horizon of the problem, or in the case of infinite horizons, the average or discounted reward.

#### 1.2.2.2 Self-interested agents

The field of game theory focuses on situations where self-interested players/agents make decisions that affect each other and affect a common environment. Each agent tries to respond make decisions in a way that maximizes its own reward given strategies of the other agents. Games can be divided along several axes. Perfect recall (vs. imperfect recall) games involve players who never forget actions, whether theirs or others', once they observe them. In games of incomplete information (vs. perfect information), a player does not know what moves have already been played by other players, resulting in uncertainty

about the current state of the world and multiple game situations being indistinguishable to that player. Games can also be classified by the number of stages (decision-making points) they contain; 1-stage games involve only one stage of decision making while in sequential games, players take moves after observing moves of chance (e.g., a roll of a die) and moves of the other players. The most general representation for this kind of games is as normal form Games (NFGs). Sequential games can be represented as extensive form games (EFGs) where there are multiple stages, each of which is a game. Actions taken at a stage affect the game that will be played at the next stage, thereby making it necessary to think about long-term consequences of actions.

A strategy profile is a set of strategies, one per player. A strategy prescribes a probability distribution over actions to take in each possible situation. The goal in competitive settings is typically to calculate an equilibrium strategy profile; one from which no player is motivated to deviate. An equilibrium tells the designer of a system of agents what the system will converge to in the long run. But some equilibria are more desirable than others; equilibria can vary in their stability and the social welfare they achieve, among other things.

There are different kinds of equilibria, the most basic of which is the Nash equilibrium. For sequential games, however, the Nash equilibrium may not be rational because it may involve a player responding to an incredible threat from its opponent; i.e. a move that seems like a threat but is really not so because it would not make sense for the other to make. Shortcomings of the Nash equilibrium are addressed by solution concepts known as equilibrium *refinements*. For example, subgame perfect equilibria is a solution concept developed for characterizing equilibria in sequential games.

Without making any assumptions about the game, it is computationally expensive to calculate a Nash equilibrium even for a two-player NFG. The Lemke-Howson algorithm [53], one of the best known, has exponential running time on some instances. Even answering questions concerning the best equilibrium (according to some criterion), or whether a given

pure strategy is possible under some equilibrium, is an NP-hard problem [34]. Furthermore, Conitzer and Sandholm show inapproximability results; not even an equilibrium that is approximately optimal can be found in polynomial time [30].

In addition to EFG, another very general representation for self-interested agents is Partially Observable Stochastic Games (POSG) [41], which are defined exactly like DEC-POMDPs, but with a reward function per agent rather than one reward function for the entire team. An agent's reward function, however, is still in terms of joint states and actions, so no independence assumptions are made.

### 1.2.3 Specialized models

The previous sub-section clearly shows that without making any assumptions about the underlying structure of a problem, calculating the optimal policy (in the case of cooperative agents), or an equilibrium profile (for self-interested agents), is computationally very demanding. One way of dealing with this prohibitive complexity is identifying sub-classes of the general problem that are more tractable to solve, but still of practical interest. Different models cater to problems with different kinds of structure Some of these have inherently lower complexity, while others, in spite of being in the same complexity class as the more general models, are easier to solve in practice.

One way in which the general models are specialized is by assuming that interactions among agents have a certain structure whereby the agents are largely independent except for some actions that have non-local effects (e.g. Transition-Independent DEC-MDP [14], Event-Driven Interaction DEC-MDP [12]), or some states where rewards and transitions depend on actions of all agents (e.g. Interaction-driven Markov game [82], Distributed POMDPs with Coordination Locales [87]). This kind of structured interaction arises in domains like robotic search and rescue where robots typically have different goals and ways of accomplishing them, but they still affect other robots when doing certain actions.

Rather than making assumptions about *how* agents interact, another way of specializing the general models is to make assumptions about *which* agents interact. Typically, real life situations exhibit some kind of locality of interaction where an agent only interacts with a subset of other agents. This has been leveraged in models like Networked Distributed POMDP [67]. Locality of interaction arises in domains like distributed sensor networks where each sensor interacts with only a limited number of neighboring sensors.

The above models are for cooperative agents, but similar approaches have been taken with game theoretic models. Locality of interaction is again an important characteristic in many situations and has been exploited in graphical games and variants thereof [48, 52, 19, 47]. Various kinds of conditional independence among agents' decisions have given rise to models descending from Influence Diagrams [51, 46].

From this brief survey of the state of the art in models for representing MSDM problems with special structure, it is clear that there is currently no decision- or game-theoretic model that can cleanly represent the kind of problems we are interested in; problems where agents are largely independent and interaction arises because some actions have non-local effects on the rewards *and* transitions of other agents. Without a model that specifically caters to such situations, they can only be represented in extensive form or using DEC-POMDP, leading to problem instances that are much larger than they need to be. Besides being representationally inefficient, such representations obscure the structured interaction among agents, making it hard to exploit to efficiently find an equilibrium or an optimal policy.

### 1.2.4 Communication

As we tackle more complex problems requiring tighter coordination, we cannot ignore the possibility of, and oftentimes need for, communication among the decision makers, even self-interested ones. Communication allows the agents to coordinate their actions and overcome uncertainties in action outcomes and the environment. For example, robots

conducting a rescue operation need to coordinate in the face of uncertainty regarding action durations and success, even if the initial plan is common knowledge.

The concern about complexity discussed earlier is exacerbated when communication is brought into the picture. Reasoning about communication results in an explosion in the size of the state space and the number of decisions that need to be made (e.g. who should an agent communicate with, how often, and what should the content of the message be). We use the term *computational cost* to refer to the cost of solving the larger and harder problem we get when communication is involved. There have been numerous efforts to address this kind of cost. One approach is to reason about communication online. This has the advantage that at any given state, the agent only has to reason about states reachable from that state, thereby reducing the overall computational effort. Typically, during offline reasoning about domain actions, the agents either assume they will always be able to communicate [96, 71] or will never be able to communicate [13]. For these offline assumptions, online reasoning then decides where communication can be skipped or introduced, respectively. The problem with this kind of reasoning is that communication decisions made this way are typically myopic (involving limited lookahead into the future) and are usually made under further assumptions regarding the availability of communication in the future [38, 13], resulting in the possibility of over- or under-communication.

The research that has agents reasoning about communication offline takes different approaches to reducing the computational cost. Instead of simultaneously computing domain action and communication policies, the agents can optimize the communication policy and action policy iteratively with respect to each other [80]. This has the disadvantage of possibly converging to sub-optimal policies, as when, for example, the agents avoid doing an action that needs communication because communication for this action is not part of the communication policy, and the communication policy does not include this desirable communication because the action is not part of the action policy. The only way around this circularity is to reason about domain and communication actions in an integrated fashion.

Note that this does not rule out decomposition-based approaches; it just mandates that the sub-problems of finding communication and action policies should provide feedback to each other in a systematic way that does not miss the global optimal.

Another approach that was developed for general DEC-POMDPs requires agents to communicate at least every $K$ time steps, thereby only considering messages that encode observation histories up to length $K$. Since in DEC-POMDP a policy is a mapping from observation histories to actions, this approach makes policy computation less intractable. However, this approach does not weigh the costs and benefits of communication, so the agents can potentially communicate when they do not need to, or be unable to communicate when they do.

The various independence assumptions that form the bases of the special decision-theoretic models discussed earlier make reasoning about communication a prime candidate for approaches that benefit from these assumptions. If agents are not tightly coupled, their need for communication is very different from agents in a DEC-POMDP, for example. However, so far, there has been no work that tries to make offline reasoning about communication tractable by exploiting the special character of a given special model.

### 1.2.5 Solution approaches

One of the concerns raised against decision-theoretic models is their need for an accurate and, often, very detailed model of the environment. This concern is addressed by approaches that learn the environment model , as in multi-agent reinforcement learning [99], as well as algorithms that can reason over approximately-specified models. Another, more important, concern is the very high complexity of the problem of calculating optimal policies (the general DEC-POMDP and DEC-MDP models are NEXP-complete [15]).

One family of algorithms that attempt to overcome this complexity is the Joint Equilibrium-based Search For Policies (JESP) family. JESP-style algorithms solve general DEC-POMDPs by iteratively optimizing one agent's policy with respect to all other policies until no fur-

ther improvement in rewards is possible [65]. This approach reaches a local optimum, and if the best policy calculated so far is always retained, the algorithm is guaranteed to terminate. The JESP approach has been implemented in a dynamic programming algorithm (DP-JESP) and has been improved to increase parallelism by exploiting locality of interaction [67, 49].

Another large family of algorithms for DEC-POMDPs is based on the Dynamic Programming (DP) optimal algorithm proposed by Hansen et. al [41] . The Memory-Bounded Dynamic Programming (MBDP) algorithm started a line of work that attempts to make DP tractable by keeping track of a limited number of policy trees [74]. Improved MBDP (IMBDP) is motivated by the fact that only a small set of observations is possible for a given belief state and action choice [73]. IMBDP therefore only retains the most likely observations, thereby preventing the exponential growth of tree size with the number of observations. MBDP with Observation Compression (MBDP-OC) [26] addresses the same concern but in a more informed way.

Ideas from solving single-agent models have sometimes been borrowed for multi-agent planning. For example, Point-based dynamic programming for DEC-POMDPs [85] is a mix of the traditional DP approach and point-based approximations that have been used for single agent POMDPs. Solutions to a relaxed version of the original probelm (for example, a version with full observability or with free communication) can be used as heuristics for approximating the value of the best joint action (e.g., the MDP heuristic is used in MBDP).

Oftentimes, a proposed model is accompanied by an algorithm that solves this particular model by capitalizing on the special features in it (e.g. the OC-DEC-MDP model has an accompanying algorithm [17] and the Coverage Set Algorithm [14] was developed to solve TI-DEC-MDP and EDI-DEC-MDP, later generalized and improved in the Multi-agent Planning Bilinear Program algorithm [69]).

Exploiting problem structure for efficient representation and computation, together with the use of heuristic approaches, is making it possible to scale up to tens, and sometimes

hundreds, of agents. However, this ability to scale up comes at the expense of significant restrictions on the nature and amount of interactions among agents. Moreover, the fact that algorithms exploit the character of their respective models typically makes them of little use when trying to solve a new model.

Studying state of the art algorithms for sequential decision making, we found only very few approaches that rely on optimization techniques [9, 69, 95, 8] although the availability of industrial-grade optimization packages make this an attractive approach. Using optimization is a direction we will take for most of the work presented in this thesis.

## 1.3 Contributions

### 1.3.1 Problem statement

In our work, we have identified an interesting class of problems where the decision processes of multiple agents are tied together by interactions stemming from actions that have non-local effects. Our study of this class of problems addresses the following research questions:

1. How can we represent problems where agents are largely independent except for some structured interactions among them? The representation should exploit the loose coupling and avoid unnecessary verbosity, but still be able to cope with problems where interaction is arbitrarily strong.

2. Can we leverage available optimization packages and solvers? Industrial-grade packages whose performance has been optimized over many years and uses are now widely available. Can we formulate our problem of calculating a set of policies with maximum reward or stability in a way such that we can use these solvers? Moreover, can we exploit structured interactions to develop "good" formulations for which the available industrial-grade solvers not just work, but work efficiently?

3. Can we exploit structured interactions to make reasoning about communication offline tractable? In the kind of problems we address, the agents are not very tightly coupled, so they do not need to communicate all the details of their progress. Intuitively, there are only a few situations where the agents need to communicate. Can we analyze agent interactions to guess what these situations are and make communication available only at these points, thus eliminating most of the computational overhead associated with reasoning about communication? This would allow us to reason about domain and communication actions in an integrated yet efficient manner.

### 1.3.2 List of contributions

The contributions of this thesis are in the area of multi-agent decision making in settings where agents are largely independent except for some structured interactions among their decision processes. I considered both cooperative and self-interested decision makers. I studied the problems of representing situations with structured interactions, formulating and solving the decision making problem as an optimization problem, and reasoning about communication. The following is a list of my contributions:

- **Representation for cooperative and self-interested agents**: I developed Event-Driven Interaction with Complex Rewards (EDI-CR), a decision-theoretic model for representing structured transition and reward interactions. EDI-CR has the same expressive power as DEC-MDP with factored state and local observability.

- **Solving cooperative EDI-CR using optimization techniques:** I exploited structured interaction to develop a compact Mixed Integer Linear Program formulation of EDI-CR instances. I also formulated the problem of policy calculation as a continuum of problems with varying strengths of agent interactions and studied the use of homotopy methods to solve this continuum.

- **Solving self-interested EDI-CR using optimization techniques**: For an existing formulation of calculating equilibrium as a bilinear program, I studied the effect of changing the amount of agent interaction on the size of the formulation and the speed of solving it. As an alternative solution approach, I investigated an existing formulation of finding equilibria as a system of nonlinear equations and the possibility of adapting and using it to solve self-interested EDI-CR.

- **Communication among cooperative agents**: I exploited the structure explicitly represented by EDI-CR to make offline reasoning about communication tractable. I devised heuristics that strategically choose communication decision points to add to zero-communication version of the problem. The resulting problem achieves the benefits of better coordination through communication at a small fraction of the computational cost typically incurred when planning for communication.

- **Communication among self-interested agents:** I studied the problem of self-interested agents deciding whether to communicate information when doing so is necessary to accomplish a collective task, but incurs individual costs. I modeled this situation as a sequential game of incomplete information and developed a hill-climbing approach to find an approximate Nash equilibrium.

All the empirical evaluation for the above contributions was done using the Mars rovers domain. In this domain, rovers are tasked with collecting data from Mars. The rovers can be cooperative or self-interested, depending on which approach is being evaluated. Each rovers has a set of sites that it can collect data from and needs to decide which sites to visit and in what order. Collected pieces of data can be redundant, complementary, or independent of each other. In addition, a visit to a given site by one rover can make another rover's visit to another site easier or harder. More details about this domain are given in Section 2.2.1.3.

## 1.4 Thesis Outline

The body of this thesis is structured as follows: In Chapter 2, we survey general decision- and game-theoretic models for representing multi-agent sequential decision making, both for cooperative and self-interested agents. We then focus on problems with structured agent interactions, detailing the characteristics of such problems and presenting some motivating examples of them. We present our new model, Event-Driven Interaction with Complex Rewards (EDI-CR), that we developed to fill a gap in existing decision- and game-theoretic models. We conclude the chapter with a brief survey of specialized models and compare them to our new model, where appropriate.

In Chapter 3, we try to leverage the power of available industrial grade optimization packages in solving instances of EDI-CR with cooperative. To this end, we presents two formulations for our problems; as a mixed integer linear program and as a system of non-linear equations. We start the chapter with existing mathematical formulations of a more general class of problems. We then show how we exploit the special character of our problems to develop compact formulations that are easier to solve, both for the cases with 2 and more than 2 agents. The chapter ends with an overview of homotopy methods and an investigation of their use to solve EDI-CR instances.

The issue of communication among cooperative (resp. self-interested) decision makers is investigated in Chapter 4 (resp. Chapter 5). For cooperative agents, we start the chapter with a survey of ways of representing and reasoning about communication in decision-theoretic models. We then analyze the computational costs of reasoning about communication and present our approach for addressing this kind of cost. We present heuristics for limiting communication in the uni- and bi-directional interaction cases. We give experimental results showing the efficiency of our heuristics in limiting the computational cost while still allowing for high-quality solutions.

Chapter 5 addresses the issue of communication among self-interested agents in the context of a problem from the field of database management. In the view maintenance

problem, database managers need to disclose information to keep a database view up to date (collecting a reward for doing so), but incur individual costs for disclosing information. We formulate the view maintenance problem as a game of incomplete information and present an anytime hill-climbing algorithm for solving this kind of games.

Chapter 6 deals with the problem of calculating equilibrium profiles for instances of EDI-CR involving self-interested agents. We investigate the effect of changing the amount of interaction among agents on the size of EDI-CR instances formulated as bilinear programs and the speed of solving these programs. We vary the amount of interaction by varying the amount of communication the agents can have. The chapter ends with a discussion of continuation methods and their use for solving the problem of calculating equilibrium profiles in general 1- and multi-stage games, then discuss their possible use for EDI-CR instances.

Finally, Chapter 7 summarizes our contributions and discusses possible directions for future work.

# CHAPTER 2

# REPRESENTATION FOR STRUCTURED AGENT INTERACTIONS

Finding the right representation for a problem is the first step towards solving it. A suitable representation should be able to express the particularities of the kind of problems it aims to model. An explicit representation of the special structure of the problem makes it easier to design efficient solution algorithms that exploit this special structure. On the other hand, the representation should not be so tailored to the problem as to give up all generality. Ideally, it should be able to represent a wide spectrum of problems, with the benefits of using it depending on how close a given problem is to the class of problems targeted by the representation.

Another concern in discussing the merits of a representation is compactness. Again, we can use the special character of the problem to avoid duplicate or unnecessarily verbose specifications. However, this compactness should not be all lost when we start to actually operate on the representation. If a compact representation does not come with an algorithm that manipulates it directly, without having to first expand it or 'roll it out' , then the actual benefit of the representation is questionable.

When it comes to multi-agent sequential decision making, a large number of decision- and game-theoretic models have been proposed in the literature. In addition to general models, numerous specializations were proposed, together with their associated solution algorithms, in an attempt to circumvent the high complexity of solving the general models. However, there is currently no decision- or game-theoretic model that can cleanly represent the kind of problems we are interested in; problems where agents are largely independent

and interaction arises because some actions have non-local effects on the rewards *and* transitions of other agents.

We propose a new model, Event-Driven Interactions with Complex Rewards (EDI-CR), that allows us to better exploit structured interactions among the sub-problems of different agents. EDI-CR is a compact yet general representation, capable of capturing problems where the degree of coupling among agents ranges from complete independence, as in MDPs, to complete dependence, as in decentralized MDPs.

We begin in Section 2.1 by on overview of some of the most general models. In Section 2.2, we give some motivating examples that demonstrate the need for a new model. We then present EDI-CR and discuss its expressive power. In Section 2.3, we give a survey of specializations of the general representations and compare them to EDI-CR, where appropriate.

## 2.1 General Representations

In this section, we overview general decision- and game-theoretic models for representing sequential multi-agent decision-making under uncertainty.

### 2.1.1 Cooperative agents: DEC-POMDPs and DEC-MDPs

#### 2.1.1.1 DEC-POMDP

Being very useful in the single-agent case, decision-theoretic models and tools are proving similarly useful when there are multiple decision-makers involved. The family of models based on the Markov Decision Process (MDP) is large and still growing. One of the most general models in the literature is the Decentralized Partially Observable MDP (DEC-POMDP) proposed by Bernstein et al. [15]. DEC-POMDP is a model suitable for domains where the agents do not know the global state with certainty, but do receive observations from which they can form some belief over what the global state is.

**Definition 1** *An n-agent DEC-POMDP is a tuple* $< \mathcal{A}, S, A, P, R, \Omega, O >$ *where*

- $\mathcal{A}$ *is the set of* $n$ *agents*
- $S$ *is a finite set of world states, with a distinguished initial state* $s_0$
- $A = A_1 \times A_2 \times ... \times A_n$ *is a finite set of joint actions.* $A_i$ *is the set of actions that can be taken by agent* $i$
- $P : S \times A \times S \rightarrow [0, 1]$ *is the transition function.* $P(s'|s, a)$ *is the probability of the outcome state* $s'$ *when the joint action* $a$ *is taken in state* $s$
- $R : S \times A \times S \rightarrow \mathbb{R}$ *is the reward function.* $R(s, a, s')$ *is the reward obtained from taking joint action* $a$ *in state* $s$ *and transitioning to state* $s'$
- $\Omega = \Omega_1 \times \Omega_2 \times ... \times \Omega_n$ *is a finite set of joint observations.* $\Omega_i$ *is the set of observations of agent* $i$
- $O : S \times A \times \Omega \rightarrow [0, 1]$ *is the observation function.* $O(s', a, o)$ *is the probability of agents 1 through* $n$ *seeing observations* $o_1$ *through* $o_n$ *after joint action* $a$ *transitions to* $s'$
- *Joint partial observability: the n-tuple of observations made by the agents together does not (necessarily) fully determine the current state*

DEC-POMDP is a very general model in that it does not make any assumptions about the nature of interaction among the different agents. But it is this same lack of assumptions that makes DEC-POMDPs very difficult to solve (NEXP-Complete [15]). It is not possible to break down the joint problem into smaller sub-problems because every aspect of the problem dynamics tightly couples the sub-problems of individual agents. Rewards, transitions and observations are all defined over *joint* actions and states. An added difficulty is presented by the property of joint partial observability; even if the agents could piece together all their local observations, they would still be unable to determine the global state with certainty.

#### 2.1.1.2 DEC-MDP

One way of simplifying the DEC-POMDP model is to assume *joint full observability*, rather than the general *joint partial observability*, whereby the set of observations together do identify the current world state. This results in the Decentralized MDP model.

**Definition 2** *A decentralized Markov decision process (Dec-MDP) is a Dec-POMDP that is jointly fully observable, i.e. there is a mapping $J$ that identifies the global state from the observations of all agents. $J : \Omega_1 \times ... \times \Omega_n \rightarrow S$ such that $O(s, a, o_1..o_n) > 0$ if and only if $J(o_1..o_n) = s$.*

The difference between the DEC-MDP and the DEC-POMDP becomes obvious if we allow free communication among the agents; a DEC-MDP with free communication reduces to an MDP whereas a DEC-POMDP reduces to a POMDP.

In both DEC-POMDP and DEC-MDP, because an agent does not observe the global state, it can only derive a *belief* over the global state from the observation history it has seen so far. An agent's policy therefore maps each belief state to a probability distribution over actions. Because there is always a deterministic optimal policy, we can take a policy to be mapping from belief states to actions. The change from partial to full joint observability does not affect the theoretical complexity of the model; both DEC-POMDPs and DEC-MDPs are NEXP-complete [15].

Having replaced the joint partial observability assumption with full observability, the next logical step in limiting the generality of DEC-POMDPs in return for tractability is to attack one or more of the ties linking different agents together; the reward, transition and observation functions. The first step in doing so is to consider a *factored* state space where we designate each agent's part of the state individually. Note that this does *not* mean an agent's part of the state is entirely under its control.

**Definition 3** *A* factored*, n-agent DEC-MDP is a DEC-MDP where the world state can be factored into* $n+1$ *components* $S = S_0 \times S_1 \times ... \times S_n$ *where* $S_0$ *are the external features and* $S_i$ *is the set of features belonging to agent* $i$*.*

Note that a factored state space does not mean each agent has full control over its state; an agent's state can be affected by actions of other agents. Consequently, an agent's optimal decision depends on the current uncontrollable features $s_0$ and the history of its local features. As with DEC-POMDP, instead of keeping track of history, an agent can form a belief state, in which case the policy is again a mapping from belief states to actions.

For a factored DEC-MDP, we can consider further sub-classes with independence in one or more of the following aspects: rewards, transitions and observations. Basically, independence in an aspect means that for an agent $i$, the aspect in question is not affected by the actions of other agents, and only depends on the state and action of $i$. We will see some models featuring different kinds of independence in Section 2.3.1.

### 2.1.2 Self-interested agents: EFG

Extensive form game (EFG) is a general representation for self-interested multi-agent decision making that is close to factored DEC-MDPs with observable local states. EFG is a tree capturing the order in which agents take actions, what they know when they take each action, and the probabilistic outcomes of actions.

**Definition 4** *An EFG is a tuple* $< I, V, E, P, H, u, p >$ *where:*

- $I$ *is the set of players*
- *The pair* $(V, E)$ *is a finite directed tree with nodes* $V$ *and edges* $E$ *and* $Z$ *is the set of terminal nodes*
- $Player : V \setminus Z \rightarrow I$ *determines which player moves at each decision node.* $Player$ *induces a partition over* $V \setminus Z$ *and* $Player_i = \{x \in V \setminus Z | Player(x) = i\}$ *is the set of nodes at which player* $i$ *moves*

- $H = \{H_0, ..., H_n\}$ *is the set of information sets, one for each player.* $H$ *is a partition over* $Player_i$*. The information set of a node* $x$ *is denoted as* $h(x)$
- $A_i(h)$ *is the set of actions available at information set* $h$ *belonging to agent* $i$
- $u_i : Z \rightarrow \mathbb{R}$ *is agent* $i$*'s utility function defined over the set of terminal nodes. For* $x \in Z$*,* $u_i(x)$ *is the payoff to player* $i$ *if the game ends at node* $x$
- $p$ *is the transition probability of chance moves*

In a game with imperfect information, an agent does not know with certainty the state of the other agent (and thus the game played by the agents at any particular stage), but does have a probability distribution over it, much like the concept of belief in DEC-POMDP. In such games, an information set can contain more than one node, which the agent cannot tell apart. A policy should therefore make the same decision across all nodes belonging to the same information set (similar to a DEC-POMDP policy that maps belief states to actions). In situations with self-interested agents, the goal is usually to find some kind of equilibrium set of policies, one per agent, where no one agent is motivated to deviate from its prescribed policy. This equilibrium state is often desirable so that the designer can make statements about the long-term state of the multi-agent system. But a game can have multiple equilibria, and sometimes the goal is to calculate the equilibrium that maximizes social utility or some other quantity of interest. Unlike the case with cooperative agents, we are not guaranteed to find a deterministic equilibrium, and therefore a policy is a mapping from information sets to *probability distributions* over actions.

Representing a game as an EFG is justified when modeling tightly coupled games where indeed all actions participate in determining the game's next state and agents' individual rewards. However, this representation is overly verbose in the case of loosely coupled games with relatively few structured interactions.

In addition to EFG, another very general representation for self-interested agents is Partially Observable Stochastic Games (POSG) [41], which are defined exactly like DEC-

POMDPs, but with a reward function per agent rather than one reward function for the entire team. An agent's reward function, however, is still in terms of joint states and actions, so no independence assumptions are made. Hansen et. all write that a finite-horizon POSG can be viewed as a type of extensive game with imperfect information [41].

## 2.2 New Model: EDI-CR

In this section, we propose a new model, Event-Driven Interactions with Complex Rewards (EDI-CR), that allows us to better exploit structure in interactions among the subproblems of different agents. Our model explicitly captures interactions among agents and their effects, and thus makes it easier to reason about them than using general models. It is also more general than existing structured models. First, we give some motivating examples that demonstrate the need for a new model. We then present EDI-CR and discuss its expressiveness compared to DEC-MDPs with observable local state.

### 2.2.1 Motivating examples

#### 2.2.1.1 Robotic rescue

Consider a robotic team dealing with a building on fire. One agent is in charge of putting out the fire, another locates and evacuates survivors and a third delivers first aid to the injured. Most of an agent's actions affect only itself; the first agent's decision of how to attack the fire and what kind of extinguisher to use mainly affect its own progress in fire fighting. Likewise, the paramedic agent's choice of the kind of first aid care to give to the injured mainly affects its own progress towards getting them out of critical conditions. However, the fire-fighting agent's decision of when to secure a given area affects how easy it will be for the rescue agent to locate survivors in that area. Competitions like RoboCup Rescue [1] involve developing intelligent agents that are given the capabilities of the main actors in a disaster response scenario.

#### 2.2.1.2 Cleaning robots

Consider a set of cleaning robots which, between them, manage the cleanliness of a building. Each robot is responsible for a set of halls, but corridors are joint responsibility and the robots can get extra reward if they correctly coordinate their cleaning of this shared space. Other interactions stem from sharing the waste bins and potentially getting into each other's way in shared areas like corridors and elevators. Another source of interaction among the robots is that if one robot needs to move a heavy piece of furniture, it would be easier to do if it gets help from another robot.

#### 2.2.1.3 Modified Mars rovers

Consider a variant of the Mars rovers domain (used in [14]) where there are multiple rovers, each with a set of sites to visit. Probabilistically, visiting a site can be *slow* or *fast* and each outcome has an associated probability, duration and reward. In addition, each site has an earliest start time before which it cannot be visited. A site can also have a list of pre-requisites; sites that must be visited before the one in question. There is transition dependence in that a rover's visiting a site can affect the outcome probability of another rover visiting some other site. For example, if rover $j$ visits a certain site before rover $i$, $j$ can take some measurements which make processing some other site easier for agent $i$. Reward interaction takes the form of additional reward (for complementary sites) or penalty (for redundant sites) collected when the rovers visit certain combinations of sites. For an agent $j$, a *critical action* is visiting a site that affects $i$'s outcome or is associated with additional reward. Clearly, the degree of coupling of the agents' sub-problems depends on the number of transition and reward interactions.

The decision problem for each agent is what subset of sites it should visit and in what order. Note that this is different from the decision problem addressed by Becker [14] where the order of the sites is fixed and agents can only choose to visit a site or skip it, which significantly reduces the size of the problem. The local state of an agent is composed

of the sequence of actions done so far (in chronological order), together with the resulting outcomes (fast or slow). There is no need to explicitly store the current time in a state, since it can be calculated by adding up the durations of the particular action outcomes obtained.

Communication can be introduced into this setting to better coordinate the agents and reduce the uncertainty caused by probabilistic action outcomes. One communication language, for example, can be to exchange local states, leaving the agents fully coordinated. Another language can exchange only certain aspects of the local state (e.g., the last action done). For the chosen communication language and an associated communication cost function, the decision problem becomes choosing the subset of sites to visit and their order, in addition to deciding what pieces of information to communicate during execution in order to maximize the difference between rewards and communication costs.

### 2.2.2 Problem characteristics

The above examples share some fundamental characteristics:

- Each agent has its own local state and actions (e.g. the cleaning robots have different locations and different grabbing actions) and actions have probabilistic outcomes.
- Agents are generally unaware of each other's states and actions, unless some form of communication (whether deliberate or part of the setting) is specified.
- Most of an agent's action outcomes and rewards are independent of the other agent.
- Interactions among agents are relatively few, compared to the number of actions they can take. They are also structured, meaning that an action can affect the other agent in a specific way. In a reward interaction, a certain action of an agent affects the reward of a certain actions of the other agent. Transition interactions are where certain actions of an agent affect the outcome probabilities of certain actions of the other.

- Interactions are not only between actions taken at the same time. An action can be affected by something that happened in the past (e.g. a robot dumping a large object in a given bin will affect the outcome of another robot's use of the bin at any later point in time). The fact that the affecting action happened is not necessarily encoded in an agent's state.

Traditionally, the examples we described would be represented in extensive form or using DEC-POMDP, depending on whether the agents are cooperative. This requires specifying an agent's rewards and next state for each of its actions and the other agents' actions. Clearly, this representations is overly verbose, since in most cases, the rewards and new states are independent of the actions of other agents. Ignoring this fact results in instances that are much larger than they need to be. Besides being representationally inefficient, such a representation obscures the structured interaction among agents, making it hard to exploit to efficiently find an equilibrium or an optimal policy.

### 2.2.3 EDI-CR: The model

Event-Driven Interactions with Complex Rewards (EDI-CR) is a hybrid of the Transition-Independent DEC-MDP (TI-DEC-MDP) [14] and the Event-Driven Interaction (EDI-DEC-MDP) [12] models, discussed in more detail in Section 2.3.1. The state space is factored and an agent can fully observe its local state, but not the states of others. From TI-DEC-MDP, EDI-CR inherits complex rewards where in addition to local reward functions, certain combinations of actions have an additional cost/reward. From EDI-DEC-MDP, EDI-CR inherits structured transition dependence where in addition to local transition functions, certain actions of an agent can affect the transitions of another.

**Definition 5** *An n-agent EDI-CR is a tuple* $< \mathcal{A}, S, A, P_{1..n}, R_{1..n}, \rho, \tau, T >$ *where:*

- $\mathcal{A}$ *is the set of* $n$ *agents*
- $S = S_1 \times S_2 \times ... \times S_n$ *is the set of factored world states*

- $A = A_1 \times A_2 \times ... \times A_n$ *is the set of joint actions*
- $P_i : S_i \times A_i \times S_i \rightarrow [0, 1]$ *is $i$'s local transition function*
- $R_i : S_i \times A_i \times S_i \rightarrow \mathbb{R}$ *is $i$'s local reward function*
- $\rho = \{< (s_{k_1}, a_{k_1}), ..., (s_{k_m}, a_{k_m}), r_k >_{k=1..|\rho|}\}$ *is the set of reward interactions. Each interaction involves any subset of agents and lists the state-action pairs and the reward/penalty when the agents take these actions in these states.*
- $\tau = \{< (s_{k_1}, a_{k_1}), ..., (s_{k_m}, a_{k_m}), p_k >_{k=1..|\tau|}\}$ *is the set of transition interactions. The $k^{th}$ entry specifies the new transition probability $p_k$ of the state-action pair of the affected agent $k_m$ when agents $k_1$ to $k_{m-1}$ do the specified state-action pairs before $k_m$ makes its transition.*

Depending on the problem, it may be more natural to drop the states from the specification of entries in $\rho$ and $\tau$. For example, if a domain has outcomes associated with actions, and a transition dependence means that action $a_i$ affects the probability distribution over action $a_j$'s outcomes, a compact and natural way to represent this is to have an entry in $\tau$ that specifies the new probability distribution over $a_j$'s outcome, regardless of the particular states of agents $i$ and $j$.

EDI-CR can capture a wider range of problems than either of its parents. It provides us with a model that is more general than either TI-DEC-MDP or EDI-DEC-MDP but still has inherent structure that can be exploited in a tractable solution algorithm. The complexity of EDI-CR is clearly NEXP; EDI-CR is a subset of full-fledged DEC-MDP, which is NEXP, and a superset of EDI-DEC-MDP, which is also NEXP.

### 2.2.4 EDI-CR vs. factored DEC-MDP

We discuss the expressiveness of EDI-CR by considering how an EDI-CR instance can be mapped to an instance of factored DEC-MDP with observable local state and vice versa.

In the definition of EDI-CR, note that $\rho$ and $\tau$ do *not* stipulate that agents involved in a given entry do the specified actions at the same time to get the additional reward or affect another agent, i.e. we can associate a reward with a joint action whose individual components are done at different times. This kind of reward (referred to as extended reward structure in [69]) is more general than a DEC-MDP's reward function which defines rewards for transitions made *simultaneously* by all agents. Capturing the semantics of EDI-CR's reward and transition functions in a DEC-MDP would require the DEC-MDP's state to remember all previous state-action pairs, resulting in a state space exponentially larger than that of the EDI-CR's instance. Also, in DEC-MDP, the reward and transition functions are defined over joint states and actions, which makes the number of entries they have exponentially larger than that of the individual functions in EDI-CR.

We now show how an EDI-CR instance $< S_{1..n}, A_{1..n}, P_{1..n}, R_{1..n}, \rho, \tau, T >$ is mapped to a DEC-MDP instance $< \hat{S}_{1..n}, A_{1..n}, P^D, R^D, T >$. We assume that a given state cannot be encountered multiple times during a given execution, which is true if the agents know what stage they are at (e.g. when the state includes time). The need for the assumption will become apparent in the course of the mapping:

- $A_{1..n}$ and $T$ are the same
- $\hat{S}_i = S_i \times \bigcup_{t=0}^{T-1} (S_i \times A_i)^t$. The DEC-MDP's individual state spaces are the EDI-CR spaces fitted with state and action histories. We use $\hat{s}_i.current$ to refer to the current state stored in $\hat{s}_i$, without the history.
- Every entry $R^D(\hat{s}_1...\hat{s}_n, a_1...a_n)$ is calculated as follows: we add up the individual rewards $\sum_i R_i(\hat{s}_i.current, a_i)$. We then account for any additional rewards: we add the reward of the $k^{th}$ entry in $\rho$ if the state-action pairs in this entry are present in $(\hat{s}_1...\hat{s}_n, a_1...a_n)$. However, we only do this if $\exists i \; s.t. \; (s_i^k, a_i^k) = (\hat{s}_i.current, a_i)$, i.e., one of the state-action pairs in the $\rho$ entry has just been finished. This, together with

the assumption that a state is encountered at most once during an execution, avoids giving out reward for the same entry multiple times.

- Every entry $P^D(\hat{s}_1...\hat{s}_n, a_1...a_n, \hat{s}'_1...\hat{s}'_n) = v$ is calculated as follows: if, for some $i$, the history in $\hat{s}'_i$ is inconsistent with the state-action pair $(\hat{s}_i, a_i)$, then $v = 0$. Otherwise, the distribution over $i$'s next state is obtained from the $k^{th}$ entry in $\tau$ if the state-action pairs of the affecting agents in $k$ match the histories in $(\hat{s}_1...\hat{s}_n, a_1...a_n)$. Otherwise, the distribution given by $P_i$ is used. The individual distributions are then multiplied to give a distribution over the joint state.

It is worth noting that if the solution algorithm we are using to calculate a policy for the DEC-MDP operates on entire histories of actions of all agents, we may not need to factor the transition and reward interactions given by $\tau$ and $\rho$ into the DEC-MDP transition and reward functions, in which case the DEC-MDP state does not need to keep track of all transitions made so far. For example, an algorithm operating on the sequence form representation of a policy (see Section 3.1) can calculate the joint reward of a given tuple of histories and the transition probability of each agent's history given the others.

An instance of a factored DEC-MDP can be mapped to an EDI-CR instance without an exponential increase in size. This is done as follows:

- $S_{1..n}$, $A_{1..n}$ and $T$ are the same
- $\forall i R_i(s_i, a_i) = 0$ for all actions and states, i.e. individual rewards are not used
- Every entry $R^D(s_1..s_n, a_1..a_n) = v$ has a corresponding entry $< n, (s_1, a_1), ..., (s_n, a_n), v >$ in $\rho$.
- Local transition functions $P_i$ can be set to anything. They will not be used.
- Every entry $P^D(s_1..s_n, a_1..a_n, s'_1..s'_n)$ has $n$ corresponding entries in $\tau$, the $i^{th}$ of which is $< (s_j, a_j)_{j \neq i}, (s_i, a_i), p_i >$ where $p_i$ is obtained by marginalizing the next state distribution given by $P^D$ to obtain a distribution over $i$'s local state only.

## 2.3 Survey Of Special Representations

In this section we give a brief survey of some of the models proposed in the decision-theory and game-theory literature. These models attempt to circumvent the high complexity of solving the general models by focusing on special cases of the general problem.

### 2.3.1 Cooperative agents

In this subsection, we discuss some of the many specializations of DEC-POMDPs, the special structure they aim to exploit and, where appropriate, how they differ from our model EDI-CR.

#### 2.3.1.1 TI-DEC-MDP

Transition-Independent DEC-MDP is a special case of factored DEC-MDP with independent transitions and observable local states [14]. The only interaction among the agents' sub-problems is through the reward function. Agents have their individual reward functions defined over their local states and actions, but, in addition, there is a *complex reward structure* whereby certain combinations of actions have an additional cost/reward.

First, we re-state some definitions related to this model.

**Definition 6** *A factored, n-agent DEC-MDP is* transition independent *if the joint transition function $P$ can be separated into $n$ separate transition functions $P_1, .., P_n$ such that*

$$P(s_i'|(s_0...s_n),(a_1...a_n),(s_0'...s_{i-1}',s_{i+1}'...s_n')) = \begin{cases} P_0(s_0'|s_0) & i = 0 \\ P_i(s_i'|\hat{s}_i,a_i,s_0') & 1 \leq i \leq n \end{cases}$$

For agent $i$, $\hat{s}_i \in S_i \times S_0$ is the local state. The external features $S_0$ are included because they do affect the agent. Transition independence means that the new local state of each agent depends only on its previous local state, its local action, and the external features. External features change based only on the previous external features. The probability of

the new joint state is therefore just the product of the probabilities of the new individual local states.

**Definition 7** *A factored, n-agent DEC-MDP is* observation independent *if the joint observation function $O$ can be separated into $n$ separate observation functions $O_1, .., O_n$ where, for any local observation $o_i \in \Omega_i$*

$$O(o_i|(s'_0...s'_n), (a_1...a_n), (o_1...o_{i-1}, o_{i+1}...o_n)) = O_i(o_i|\hat{s}'_i, a_i)$$

That is, the observation an agent sees depends only on its local state, the external state, and the agent's action.

**Definition 8** *A factored, n-agent DEC-MDP is* locally fully observable *if* $\forall o_i \exists \hat{s}_i : P(\hat{s}_i|o_i) = 1$.

When a DEC-MDP has both local full observability and observation independence, we can get rid of $O$ and $\Omega$ in the definition, and in this case the policies become mappings from local states, rather than observation histories, to actions.

The complex reward structure in TI-DEC-MDP is defined in terms of *events*; occurrences of $(s_i, a_i, s'_i)$ tuples in agent $i$'s history. The reward structure can be viewed as a list of constraints that describe how interactions among the agents' local trajectories through the state space (which are determined by their policies and chance outcomes) affect the global value of the system. A constraint specifies a set of events, one for each involved agent, together with the additional reward obtained by the team if each agent satisfies his part of the constraint, which happens if the agent's event occurs somewhere in its history.

The complexity of TI-DEC-MDP is analyzed in [5] where it is proved that solving DEC-MDPs with independent transitions and observations and a joint reward structure is NP-Complete.

#### 2.3.1.2 EDI-DEC-MDP

Event-driven interaction DEC-MDP (EDI-DEC-MDP) is another sub-class of factored DEC-MDP that has observable local states and reward independence but has transition dependence [12]. The latter is, however, structured, with certain actions in one agent affecting the transition probabilities of certain actions of another agent; i.e., they affect the probability distribution over the next states obtained after doing the affected action. For example, an action done by agent $j$ can facilitate an action done by $i$ (making it finish faster), thereby increasing the probability of transitioning to a next state with an earlier timestamp.

As in TI-DEC-MDP, the notion of *events* is used; a dependency consists of an event of the affecting agent $j$ and a set of unique state-action pairs of the affected agent $i$. If the event happens in $j$'s history, then if $i$ later encounters one of the specified state-action pairs, $i$'s transition probability for this pair is affected. This effect is encoded in a modified version of $i$'s transition function, which takes the form $P_i(\hat{s}'_i|\hat{s}_i, a_i, b_{\hat{s}_i a_i})$. For an affected state-action pair of agent $i$, the Boolean variable $b_{\hat{s}_i a_i}$ is true if the dependency affecting this pair is satisfied and false if it is not satisfied or there is no related dependency.

Note how this model assumes that when an agent takes an action that can be influenced by a dependency, it finds out whether or not that dependency was satisfied, which explains why $b_{\hat{s}_i a_i}$ is part of the transition function. This forces us to to include the dependency history in an agent's local state to ensure the Markov property. Intuitively, when agent $i$ learns the status of a dependency, it changes its belief about the state of agent $j$ and its $j$'s probability of doing future dependencies. This, in turn, affects $i$'s transition probability, and so $i$ needs to remember the status of each dependency each time an affected action is attempted. In the worst case, $i$'s state needs to store the last time a dependency was not satisfied as well as the first time it discovered it was satisfied. For the affecting agent, the state stores the time at which it satisfies each dependency. In some cases, it may not be necessary to store these pieces of information for every single dependency if the satisfaction of a dependency implies the satisfaction of all earlier dependencies related to the same

interrelation. For more on the difference between a dependency and an interrelation, please see [10].

From the above, it is clear that EDI-CR treats transition dependencies differently from EDI-DEC-MDP in that an affected agent does not automatically know when/whether it was actually affected, as opposed to EDI-DEC-MDP where affecting an agent is accompanied by a kind of implicit "communication" (for it does transmit information about the dependency) However, the behavior of EDI-DEC-MDP can be mimicked in EDI-CR by having a state space with additional pieces of information for the boolean variables and timestamps, and an appropriate definition of the transition dependencies in $\tau$. We therefore leave it to the modeler to decide whether he wants this kind of communication, and unlike Becker's work, make it possible to have a truly zero-communication setting.

Becker points out that EDI-DEC-MDP has an upper-bound deterministic complexity that is exponential in the size of the state space and doubly exponential in the number of defined interactions (which suggests non-deterministic exponential time complexity as an upper bound). The complexity is more formally analyzed in [5] where it is proved to be NEXP-Complete.

#### 2.3.1.3 IDMG

Interaction-driven Markov game (IDMG) [82] is a sub-class of factored DEC-MDP with observable local states. It aims to describe problems in which interaction among agents is a local phenomenon specific to a set of *interaction states* where the agents are coupled through their reward and transition functions. A set of interaction states consists of adjacent joint states (in terms of transition probabilities).

**Definition 9** *An 2-agent IDMG is a tuple* $(\mathcal{M}_1, \mathcal{M}_2, \{^i\mathcal{M}_I, i = 1..n\})$ *where* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *are the 2 independent MDPs of the 2 agents and each* $^i\mathcal{M}_I$ *is an interaction game; a two-agent, fully cooperative Markov game* $^i\mathcal{M}_I = (^i\mathcal{X}_I, A, ^i\mathcal{P}_I, ^i r_I)$ *where*

- $^i\mathcal{X}_I$ *set of interaction states of game* $i$

- *$A$ is the set of joint actions*
- *${}^i\mathcal{P}_I$ is a transition probability function for states in ${}^i\mathcal{X}_I$ and joint actions*
- *${}^i r_I$ is the joint interaction reward function defined over states in ${}^i\mathcal{X}_I$ and joint actions*

A motivating example is where two mobile robots have to navigate in a common environment, each trying to reach its own goal state and the actions of one robot do not affect the movement of the other. The robots' problems are independent except for the fact that there is a single door in the environment and in states close to the door, the actions of a robot affect the rewards or transition of the other (e.g. trying to pass through the door at the same time incurs a penalty for all participants).

The IDMG model assumes that each interaction game describes a situation where the agents should coordinate. In the corresponding interaction states, each agent explicitly communicates all information useful to the decision process, where communication is assumed to be unlimited and noise-free. Therefore IDMG assumes that agents can observe their local states in the parts of the state space where they do not interact, and can observe the global state in the interaction states.

Clearly, IDMG caters to problems with a different kind of structure than the ones addressed by EDI-CR. In IDMG, we cannot (easily) represent situations where the action of one agent affects the transition/reward of another agent wherever it takes an affected action. That is because in IDMG, the agents are assumed to be tightly coupled in the interaction states, so *all* of an agent's actions taken in those states affect the other agent. Conversely, if the size of the union of all sets of interaction states is large, representing an IDMG using EDI-CR would result in large sets of reward and transition interactions $\rho$ and $\tau$, which is logical because in this case, we are modeling a situation close to an unstructured DEC-MDP.

Another difference is that in EDI-CR, we do not assume free, involuntary communication as in IDMG. This kind of communication is necessary in solving IDMG because the solution method assumes knowledge of the joint state during each Markov game.

#### 2.3.1.4 DPCL

Distributed POMDPs with Coordination Locales (DPCL) [87] is a sub-class of DEC-POMDPs where agents act independently except in certain *coordination locales*, somewhat like the interaction states in IDMG. DPCL does not assume transition or reward independence, but assumes observation independence. Again, the agents' decision processes are expressed in terms of local transition, reward and observation functions, with interactions expressed as coordination locales (CLs) that are either same-time or future-time coordination locales. As explained in [88], same-time CLs represent situations where the effect of simultaneous execution of actions by a subset of agents cannot be described by the local transition and reward functions of these agents. Future-time CLs identify situations where actions of one agent impact actions of others in the future.

DPCL differs from IDMG is that the latter assumes observable local state while the former does not. Also, IDMG assumes free communication in the interaction states, while DPCL does not.

#### 2.3.1.5 ND-POMDP

Networked Distributed POMDP is a sub-class of DEC-POMDP with factored state that aims to exploit locality of interactions [67]. ND-POMDP is motivated by domains such as distributed sensor nets where the sensors need to decide which direction to scan in. To track a target and obtain associated reward, two sensors with overlapping scanning areas must coordinate by scanning the same area simultaneously. The target's movement is uncertain and unaffected by the sensors. Based on the area it is scanning, each sensor receives observations that can have false positives and false negatives. This domain has locality of

interaction because each sensor interacts with only a limited number of neighboring sensors.

In ND-POMDP, agents have independent transition and observation functions, but have reward interactions. The reward function is defined as $R(s,a) = \sum_l R_l(s_{l1}, s_{l2}, .., s_{lk}, s_u, a_{l1}, a_{l2}, .., a_{lk}$ where each $l$ can refer to any subset of agents and $s_u$ is the part of the state that is not under an agent's control (e.g. the position of the target in the wireless sensor example). Therefore, we can decompose the set of agents into subsets, where an agent's reward only depends on agents belonging to its subset(s). The reward function induces an *interaction graph* where agents are nodes and edges connect agents in the same $l$.

Besides the fact that ND-POMDP assumes transition independence and EDI-CR does not, the difference between these two models is that in the former, agents belonging to the same subset $l$ are assumed to have very tight reward interaction; there is a single reward function per subset, and it is defined over joint actions and states of agents in the subset. This can be seen as a coarse-grained kind of independence where agents either have reward interactions involving all their actions or none at all. EDI-CR captures a more fine-grained kind of interaction where *specific* actions affect, or are affected by, what another agent does. However, we can still represent ND-POMDP instances using EDI-CR: the set of transition interactions $\tau$ will be empty while the set of reward interactions $\rho$ will have one entry for each entry in each $R_l$ function.

#### 2.3.1.6 TD-POMDP

Transition-Decoupled POMDP [94] is another sub-class of DEC-POMDPs with factored state space. In TD-POMDP, agent $i$'s state $< u_i, l_i, n_i >$ has 3 kinds of features:

- $u_i$: features uncontrollable by any agent
- $l_i$: features under agent $i$'s control
- $n_i$: features controlled by some other agent $j$ but whose values impact $i$'s local transitions (such features would then appear as locally controlled features of agent $j$)

The reward function is the sum of local rewards defined over local states and actions. Because of the decomposition of an agent's state into the above 3 sets of features, agent $i$'s local transition function can be written as

$$P(s_i^{t+1}|s^t, a) = P(u_i^{t+1}|u_i^t).P(l_i^{t+1}|s_i^t, a_i).P(n_i^{t+1}|s^t - l_i^t, a_{\neq i})$$

Non-locally controlled features allow for reward and transition interactions among agents. However there are two points to note. First, in TD-POMDP an agent cannot affect another's transition probability immediately because if $i$ affects $j$, $i$'s action will first set a non-local feature in $n_j$ then, in the next time step, $j$'s transitions can start getting affected by this feature in $n_j$. The second point is that reward interaction is modeled as a kind of transition interaction. Again, an agent can set a feature in another's state (transition dependence), and the reward of the latter can depend on whether this feature was set.

The approach for solving TD-POMDP, like the Coverage Set Algorithm for solving TI-DEC-MDP on which it is based, aims to consider fewer points in the policy space by realizing that due to the loose coupling of agents' processes, many policies of one agent have the same effect on another agent. This can be seen as grouping or *binning* policies. In Chapter 3, we use the idea of binning to obtain compact mathematical formulations of EDI-CR that can be solved efficiently. In our case, we bin sequences of actions of one agent that have the same effect on a given sequence of actions of another agent.

The work on TD-POMDP was published in 2010, and has some differences from EDI-CR (published in 2009). Although both models address loosely-coupled decision processes, EDI-CR assumes observable local states, while TD-POMDP does not. Another difference is that EDI-CR specifies interactions as first class entities, listing in the sets $\rho$ and $\tau$ the effects an agent's action has on the reward/transitions of other agents. On the other hand, TD-POMDP models interactions through the non-local features. Besides disallowing immediate effects, another consequence of this representation is that a problem with only reward, but no transition, interactions will appear in TD-POMDP as a problem with tran-

sition interactions. In general, reasoning about transition interactions is considerably more difficult than reasoning about reward interactions alone; Transition Independent DEC-MDP (2.3.1.1) is NP-Complete whereas Event Driven Interaction-DEC-MDP (2.3.1.2) is NEXP-Complete. We therefore feel that modeling reward interactions as transition interactions may add unnecessary complexity.

### 2.3.2 Self-interested agents

There have been several efforts to exploit special structure in games to achieve representational and computational savings. Most of the special models in the literature are restricted to 1-stage games and have the goal of scaling up with the number of agents, rather than the number of actions per agent [48, 47, 52, 19, 89]. In this section, we briefly review some of the game-theoretic representations that exploit special structures that would otherwise be obscured in an EFG.

#### 2.3.2.1 MAID

Multi-agent influence diagrams (MAIDs) [51, 21] are representations that have their origins in influence diagrams [45]. Like all alternatives to EFG, MAIDs try to explicitly capture a structural property of a game that would otherwise be obscured in extensive form. In the case of MAIDs, this property is that not all decision variables in a game are interdependent.

A MAID defines a directed acyclic graph in which nodes correspond to random variables of three types. For each agent $i$, there is a set of

- Decision variables, $\mathcal{D}_i$, whose domains are available actions and are represented as rectangles
- Chance variables, $\chi_i$, whose values are chosen by nature and are represented as ovals
- Utility variables, $\mathcal{U}_i$, which represent the agent's payoffs and are drawn as diamonds

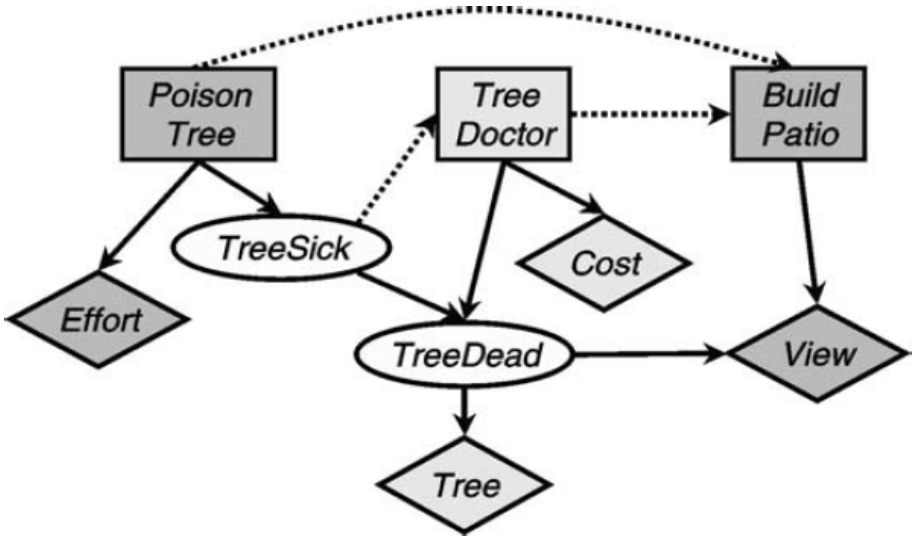



**Figure 2.1.** Multi-Agent Influence Diagram representation of the Tree Killer problem [51]

Figure 2.1 shows an example MAID. A conditional probability table (CPT) specifies the conditional probability of a node's variable given an instantiation of its parents, $P(x|Pa_x)$.

A strategy profile for agent $i$ is a set of decision rules, one for each node in $\mathcal{D}_i$. A decision rule specifies the probability of making a certain decision given values of its parents. It simply sets the CPTs of decision nodes. To represent *perfect recall* (an agent does not "forget" decisions it made in the past), all earlier decisions and their parents are among the parents of a later decision node.

Initial work on MAIDs looked at using the MAID representation as a guide for decomposing a game into interacting fragments, and provided an algorithm that finds equilibria for these smaller games in a way that is guaranteed to produce a global equilibrium for the entire game [51]. Specifically, Koller and Milch construct from MAID a *strategic relevance* graph in which the maximal strongly connected components are found and a tree is build whose nodes are these components. The components are then ordered topologically and solved in this order by changing each component back to a game tree and solving it using McKelvey and McLennan's algorithm [56].

Later work on MAIDs addresses the issue that for most realistic games, the relevance graph consists of a single strongly connected component, in which case the above algorithm degenerates into converting the game back to the original tree. Blum et. al. address this by exploiting finer-grained structure in MAIDs to improve the efficiency of a certain family of algorithms called continuation algorithms [21].

#### 2.3.2.2 TAGG

Temporal action graph games (TAGG) is a graphical representation of imperfect-information extensive form games that can be much more compact than MAIDs for games with certain special structures; namely anonymity and context-specific utility independencies [46]. TAGGs are an extension of action graph games (AGGs) to represent games taking place over multiple stages. Both representations can model anonymity, where a player's payoffs depend on *how many* players took a certain action, rather than *exactly who* these players are. AGGs and TAGGs represent a game as a graph where nodes are actions and an agent's utility depends only on the node it chose and the action counts on the neighbors of the chosen node. We can therefore specify a utility function for each action node that maps the set of configurations over the node's neighbors to a utility value. Clearly, this structure can be exploited for computational savings.

In TAGGs, Jiang et. al. [46] extend AGGs with chance and decision nodes like the ones in MAID. In addition, they introduce time-dependent counters in action nodes that keep track of how many times the given action has been chosen up to a given time point. The utility functions are also made time-dependent; $U_A^t$ specifies the utility of action $A$ at each time step $t$. An agent can therefore receive payoffs for taking an action soon after it does so and later on. Play can be seen as a sequence of AGGs played over time. At each time step $t$, players with decisions at time $t$ participate in a simultaneous-move AGG on the set of action nodes, whose action counts are initialized to be the counts at $t-1$.

To define an agent's expected utility, Jiang et. al. create the *induced Bayesian Network* (induced BN) of the TAGG. This BN has the TAGG's decision and chance variables in addition to 1) an action count variable for each action and each time step; 2)a utility variable for each time-dependent utility function $U_A^t$ and each $A$ and $t$ and 3) decision-payoff variables representing the utilities of decisions received at each of their payoff time points. Decision-payoff variables are essentially multiplexers that choose which utility function to use based on which action was taken. An agent's expected payoff from a strategy profile $\sigma$ is then the sum of the expected values of all its decision-payoff variables in the induced BN, where the CPTs of decision variables are dictated by $\sigma$.

The work also introduces the notion of an induced MAID, which is the same as the induced BN except that decision and utility variables in the latter are decision and utility nodes in the former. Jiang et. al show that a TAGG can be exponentially more compact than the corresponding induced MAID. However, one criticism that we have of this work is that it compares the size of a TAGG to that of a *naiive* MAID representation, one whose nodes have a much larger in-degree than they need to. Even though the paper later uses obvious structures in the CPTs of the induced BN/MAID to yield a representation with much smaller CPTs (e.g. the CPTs for action counts are counting functions that can benefit from intermediate values), the reduction in size is measured relative to the large naiive MAID, rather than the more compact one. In fact, the paper shows that simple manipulation of the MAID nodes and creation of intermediate ones results in a MAID whose size is only polynomial in the size of the TAGG.

In addition to proposing a new representation, Jiang et. al propose ways in which they can speed up the calculation of expected utility (EU) by exploiting the special structure in their representation. Calculation of EU is essentially doing inference to determine the marginal probability distributions over utility variables. Instead of using general BN inference techniques, Jiang et. al exploit the structure in the induced BN to speed up the calculation. For example, the induced BN has many 'counter' nodes whose CPTs are structured

counting functions. Another exploitable characteristic of the induced BN is that variables in it can be grouped by time step. This grouping, together with the introduction of some dummy variables that results in a BN satisfying the Markov property, allows Jiang et. al to perform efficient variable elimination.

From the above, it is clear that in order to do any processing on a TAGG we need to first construct the corresponding BN/MAID. For calculating the expected payoffs, the authors operate on the BN, and for calculating Nash equilibria, they apply existing MAID algorithms to the MAID. In our opinion, these facts put the purpose of having a TAGG representation in question, since it is never actually used.

#### 2.3.2.3 Succinct EFG

For some games, the game trees expressed in extensive form are too large to be stored in memory explicitly. To overcome this, Dudik and Gordon propose an implicit representation called *succinct EFG* [32]. A representation is succinct if it has enough information to support certain queries that make it possible to simulate play in a game through sampling, thus avoiding the need to explicitly represent all possible paths through a game. As such, MAIDs are themselves examples of succinct EFGs. One drawback of MAIDs is that they cannot represent context-specific independence (e.g. allowing different decision nodes to have different available actions). The second problem is that MAID algorithms rely on clique tree representations which can have high space and time complexities. Dudik and Gordon propose an algorithm that finds an extension of correlated equilibria to sequential games, with the advantage that it allows control over characteristics of the target equilibrium, e.g., we can ask for an equilibrium with high social welfare.

#### 2.3.2.4 Other representations

A number of representations have been proposed in the literature for efficiently representing 1-stage games exhibiting special structure. For example, graphical games [48], Game nets (G-nets) [52] and action-graph games [19] address games whose special struc-

ture is the locality of interactions among agents, i.e. an agent is only affected by a a subset of other agents whose size is small relative to the total number of agents. Some approaches specifically address games with a certain kind of interaction graph (e.g. a tree [47]).

The work on poker (e.g. [35]) tries to exploit structure in sequential games. This line of work is primarily concerned with the issue of scaling to larger games and provides automatic abstractions that produce much smaller games whose solutions can be converted to solutions of the original games. The problem is that with the assumptions they make, it is not clear that the techniques developed in this line of work can be used in general.

The work of Eitan et. al [7] is similar to ours in that it exploits structure in the form of transition and/or reward independence. They consider a zero-sum stochastic game in which each player has a constrained Markov Decision Process whose transitions depend only on this player's actions. A player's reward, however, depends on the states and actions of both players. We differ in that we assume that some transitions can depend on the other player's actions, whereas in their case, the transitions are independent.

## 2.4 Summary

In this chapter, we presented a brief overview of some general models of multi-agent sequential decision making, both decision- and game-theoretic. We provided some motivating examples that demonstrate the need for a new model for situations where the decision processes of the agents are largely independent, yet there are some reward and transition interactions among them. We also gave a brief survey of some of the specialized models proposed in the decision-theory and game-theory literature.

We presented our new decision-theoretic model, EDI-CR, for representing structured transition and reward interactions. EDI-CR specifies the decision problem of each agent then lists the ways in which one agent can affect the processes of others. Although there are existing models that take the same approach to representation, none of them address situations with fine-grained dependencies that we are interested in. EDI-CR can represent

problems ranging from complete independence (a group of MDPs) to complete dependence (a DEC-MDP with observable local state). The representational savings obtained from using EDI-CR depend on the number of interactions among agents.

In the next chapters, we see how we can calculate policies for EDI-CR.

# CHAPTER 3

# DECISION-THEORETIC MODELS AND OPTIMIZATION

In this chapter, we discuss the use of optimization techniques to compute policies for EDI-CR multi-agent decision problems involving cooperative agents.

For settings involving cooperative agents, finding the optimal joint policy can be formulated as an optimization problem. The objective is to maximize the expected rewards of the agent team subject to constraints guaranteeing that a feasible solution to the optimization problem represents a set of legal policies. The formulation should also factor in the interactions among agents and their effects.

There are existing mathematical formulations, both for general and specialized models. However, when applied to EDI-CR, these formulations are too verbose and/or lack global optimality guarantees. We propose mathematical formulations that exploit the special structure in EDI-CR to achieve compactness (i.e. reduce the number of variables in the formulation) and efficient computation. For the 2-agent case, our formulation is exact and its solution is guaranteed to be optimal. For cases with more than 2 agents, our formulation involves a relaxation of the original problem, and thus optimality of the resulting solution is not guaranteed.

We begin by explaining the policy representation that will be used in all the formulations in this chapter, followed by a brief discussion of existing formulations in Section 3.2. We then present our Mixed Integer Linear Program (MILP) formulations for EDI-CR and give experimental results in Section 3.3. Finally, we discuss our formulation of finding the optimal policy as a system of non-linear equations and the possible use of homotopy methods as an alternative optimization technique in Section 3.4.

## 3.1 Sequence Form Policy Representation

We use a policy representation that was independently, and later jointly, devised by Koller et. al [50] and von Stengel [83] to represent games. It has been used in settings with self-interested [61] and cooperative agents [8]. In the context of game trees, the idea behind this representation is that a policy can be characterized by the probability distribution it induces over the leaves of the tree. If two policies induce the same distribution, then they result in the same reward.

For models with local observability, a *sequence* of agent $i$, $s_1.a_1..s_t.a_t$, consists of $i$'s actions and local states. Following the nomenclature of Aras and Dutech [8], we use the term *history* instead of sequence. Concatenating a state $s$ and action $a$ to a history $h$ produces a new history $(h.s.a)$ that is called an *extension* of history $h$. A history containing $T$ (the problem's time horizon) actions is a *terminal* history. For agent $i$, the set of all histories is denoted by $\mathcal{H}_i$, terminal histories by $\mathcal{Z}_i$, and non-terminal histories by $\mathcal{N}_i$. A *joint history* $h \in \mathcal{H}$ is a tuple containing one history per agent.

An agent's policy induces a probability distribution over its histories. The *realization weight* of a history $s_1.a_1..s_t.a_t$ under a policy is the probability that the policy assigns to taking actions $a_{1..t}$ given that the states $s_{1..t}$ are encountered. A history's realization weight therefore does not include chance outcome probabilities. We will have separate terms that reflect these probabilities. The vector of all realization weights will be denoted as $x$ and the weight of history $h_i$ by $x(h_i)$.

A *pure* policy deterministically chooses one action at each decision making point. In cooperative settings there is at least one optimal pure joint policy, so we restrict our attention to pure policies. But even a pure policy will have multiple *terminal* histories with non-zero weights, because it must specify an action to take at each state reachable under the policy. Because we do not include transition probabilities in a history's weight, the realization weight can only be 0 or 1. The set of $i$'s terminal histories with weight 1 under

**Table 3.1.** Symbols used in the mathematical formulations

| Symbol | Meaning |
|---|---|
| $\mathcal{H}$ | Set of all histories |
| $\mathcal{Z}$ | Set of terminal histories |
| $\mathcal{N}$ | Set of non-terminal histories |
| $\sigma_i$ | Support set of agent $i$ |
| $\|\sigma_i\|$ | Support size of agent $i$ |
| $Q_i$ | Quantity belonging to agent $i$ |
| $Q_{-i}$ | Joint quantity of all agents but $i$ |
| $x(h_i)$ | Realization weight of $i$'s history $h_i$ |
| $i$, $j$, $k$ | Specific agents |
| $g$ | An arbitrary agent |

a policy is called *support set*, denoted by $\sigma_i$, and its size is the *support size* $\|\sigma_i\|$. Table 3.1 summarizes the symbols we use in this chapter.

## 3.2 Existing Mathematical Formulations Of DEC-MDPs

In this section, we review existing mathematical formulations of a DEC-MDP with local observability as a Non-Linear Program and as a Mixed Integer Linear Program. We will adapt some of the ideas behind these formulations for EDI-CR in the next section.

### 3.2.1 DEC-MDP NLP

The formulation of DEC-MDP with local observability as a Non-Linear Program (NLP) is given in Table 3.2. In the objective function, $\mathcal{R}(h) = \beta(h)r(h)$ is the expected reward of terminal joint history $h$, where $\beta(h)$ is the probability of encountering the joint states in $h$ given the actions in $h$

$$\beta(h) = \prod_{t=1}^{T-1} P(s_{t+1}|s_t, a_t)$$

$r(h)$ is the sum of rewards of states and actions along the history. The constraints in the NLP are called *policy constraints* and guarantee that a solution to the NLP represents a legal policy where the sum of an agent's action probabilities in any state is 1. The first

**Table 3.2.** DEC-MDP as an NLP

$$
\begin{aligned}
\max \quad & \sum_{h \in \mathcal{Z}} \mathcal{R}(h) \prod_{g \in \mathcal{A}} x(h_g) \\
\text{s.t.} \quad & \sum_{a \in A_g} x(s_{g_0}.a) = 1 \qquad g \in \mathcal{A} \\
& \sum_{a \in A_g} x(h_g.s.a) = x(h_g) \quad g \in \mathcal{A}, s \in S_g, h_g \in \mathcal{N}_g \\
& x \in [0, 1]
\end{aligned}
$$

set of constraints in Table 3.2 ensures that for any agent, the sum of action probabilities at its start state is 1. The second set of constraints in Table 3.2 ensures that the realization weights of a history's extensions add up to that history's weight.

The problem with the NLP formulation is that it results in a non-concave objective function for which no methods guarantee finding a globally optimal solution.

### 3.2.2 DEC-MDP MILP

Aras and Dutech [8] developed a formulation for DEC-POMDPs as a MILP, thereby guaranteeing that a locally optimal solution is also globally optimal. We modify their formulation for the case of DEC-MDP with local observability. For ease of explanation, Table 3.3 is for the case with only 2 agents $i$ and $j$. Because the difficulty of solving a MILP increases with the number of integer variables, Aras only restricts weights of terminal histories to be integer (in fact binary). The constraints force the other variables to be integers as well.

As in the NLP formulation, $\mathcal{R}(h, h')$ in the objective function (3.1) already accounts for the transition probabilities of both agents, so realization weights are either 0 or 1.

To linearize the objective function, Aras introduces a *compound* variable $z_{h_i,h_j}$ for each pair of terminal histories. The variable is related to the existing $x$ variables by the identity

$$z_{h_i,h_j} = x(h_i)x(h_j)$$

**Table 3.3.** DEC-MDP as a MILP

$$\max \quad \sum_{h_i \in \mathcal{Z}_i, h_j \in \mathcal{Z}_j} \mathcal{R}(h_i, h_j) z_{h_i,h_j} \tag{3.1}$$

s.t. policy constraints and

$$\sum_{h_{-g} \in \mathcal{Z}_{-g}} z_{h_g,h_{-g}} = x(h_g)\|\sigma_{-g}\| \quad g \in \mathcal{A}, h_g \in \mathcal{Z}_g \tag{3.2}$$

$$\sum_{h_i \in \mathcal{Z}_i, h_j \in \mathcal{Z}_j} z_{h_i,h_j} = \|\sigma_i\|\|\sigma_j\| \tag{3.3}$$

$$x, z \in [0,1] \quad x(h_g) \in \{0,1\} \quad g \in \mathcal{A}, h_g \in \mathcal{Z}_g$$

The question now is how to enforce the identity using a set of linear constraints. To do this, Aras uses combinatorics (knowing $\|\sigma_i\|$ and $\|\sigma_j\|$) and treats the $z$ variables as counters. Constraint (3.2) guarantees that if $h_g$ is part of some agent $g$'s support set ($x(h) = 1$), enough compound variables involving $h_g$ are set to 1, otherwise all compound variables involving $h_g$ should be 0. Constraint (3.3) limits the number of compound variables that can be simultaneously set to 1.

## 3.3 MILP Formulation Of EDI-CR

In this section, we propose a compact Mixed Integer Linear Program formulation of EDI-CR instances [62]. The key insight we use is that due to structured interactions, most action sequences of a group of agents have the same effect on a given agent. This allows us to treat these sequences similarly and use fewer variables in the formulation. We present experiments showing that our formulation is more compact and leads to faster solution times and better solutions than formulations ignoring the structure of interactions. We begin with a formulation for 2-agents then generalize to one for 3 or more agents.

### 3.3.1 Formulation of 2-agent EDI-CR

#### 3.3.1.1 Binning histories

For the 2-agent case, the NLP in Table 3.2 is a Quadratic Program (QP) whose objective function has the form $x^{\mathsf{T}}Qx$ where $Q$ is the reward matrix. $Q(h_i, h_j) = \mathcal{R}(h_i, h_j)$ if $h_i$ and $h_j$ are terminal histories, and is 0 otherwise. The MILP in Table 3.3 "flattens" this matrix, multiplying each matrix entry by a compound variable created for that entry. This approach makes sense for DEC-MDPs, because agents' decision processes are tightly coupled and the rewards/transitions of one agent strongly depend on the actions taken by another. For a given history $h_i$, $\mathcal{R}(h_i, h_j)$ can vary widely depending on $h_j$, and a given row or column in $Q$ contains many distinct values, thus justifying the need for a variable per entry in $Q$.

The situation can be very different in the presence of structured interactions. An agent is only affected by the those actions of another agent that are involved in reward and transition interactions in $\rho$ and $\tau$. So for a given $h_i$, the rewards and transition along $h_i$ do not depend on the exact actions in the history of another agent. Suppose $\tau$ specifies that action $a_3$ of agent $j$ affects the transition probability of $a_7$ of $i$. Now if $h_i$ involves doing $a_7$ at time 6, all histories $h_j$ that do $a_3$ before time 6 have the same effect on $h_i$'s transitions.

In the matrix view of the objective function, because in EDI-CR agents have their local reward functions, we can express $Q$ as the sum of reward matrices of the 2 agents $Q_i + Q_j$. Note that this does not assume that rewards are independent; each entry in these matrices can depend on the histories of both agents. The rows (resp. columns) in $Q_i$ (resp. $Q_j$) will contain many duplicate entries, reflecting the fact that an agent is oblivious to many details of the other agent's history.

The main idea in our formulation is that for a history $h_g$, we group all histories of the other agent that have the same effect on the transitions and rewards in $h_g$ into a single ***bin***. For each history $h_g$ of some agent $g$, the set of bins it induces, $B_{h_g}$, is a partition over the set of terminal histories of the other agent.

**Table 3.4.** 2-agent EDI-CR as a MILP

$$\max \sum_{b \in B_{h_g}} \mathcal{R}_g(h_g, b_{h_g}) z_{h_g,b} \qquad g \in \mathcal{A}, h_g \in \mathcal{Z}_g,$$

s.t. policy constraints and

$$\sum_{b \in B_{h_g}} z_{h_g,b} = x(h_g) \qquad g \in \mathcal{A}, h_g \in \mathcal{Z}_g$$

$$z_{h_g,b} \leq \sum_{h_{-g} \in b} \beta(h_{-g}|h_g) x(h_{-g}) \quad g \in \mathcal{A}, h_g \in \mathcal{Z}_g, b \in B_{h_g}$$

$$x, z \in [0,1] \quad x(h_g) \in \{0,1\} \qquad g \in \mathcal{A}, h_g \in \mathcal{Z}_g$$

Instead of creating a variable for every pair of terminal histories, we introduce a single variable $z_{h_g,b}$ for every history $h_g$ and every bin $b \in B_{h_g}$ associated with it. In the matrix view, we create a variable for each *distinct* entry in $Q_i$ and $Q_j$. Because structured interaction results in many duplicate entries, binning can significantly reduce the number of compound variables we introduce. Our MILP for EDI-CR is given in Table 3.4.

In the objective function, we fold into $\mathcal{R}_g(h_g, b)$ those quantities of $h_g$ that are oblivious to which history in $b$ is played, namely $h_g$'s transition probabilities and rewards. We therefore have

$$\mathcal{R}_g(h_g, b) = r_g(h_g|b)\beta(h_g|b)$$

The factors on the right can be calculated using *any* history $h_{-g} \in b$

$$r_g(h_g|b) = \sum_{t=1}^{T-1} R_g(s_t, a_t, s_{t+1}) + r_\rho(h_g, h_{-g})/2$$

where $r_\rho(h, h_j)$ represents rewards (if any) that depend on actions of both agents, as specified in $\rho$. Dividing by 2 avoids double counting reward interactions. The transition probability is given by

$$\beta(h|b) = \prod_{t=1}^{T-1} P_\tau(s_{t+1}|s_t, a_t, \{a'_1..a'_t\})$$

$P_\tau$ depends on the local transition function $P_g$ and, for transitions involved in $\tau$, actions in $h_{-g}$ done up to time $t$, $\{a'_1..a'_t\}$.

We fold into $z_{h_g,b}$ quantities that depend on the particular $h_{-g}$ in the bin, namely the transition probabilities along $h_{-g}$, given history $h_g$ ($\beta(h_{-g}|h_g)$). The identity defining a compound variable is therefore

$$z_{h_g,b} = x(h_g) \sum_{h_{-g} \in b} \beta(h_{-g}|h_g) x(h_{-g}) \tag{3.4}$$

$z_{h_g,b}$ is therefore the probability that $g$ plays $h_g$, multiplied by the probability that the other agent plays a history in $b$.

The effect of the number of interactions on the size of the formulation is clear. As the number of interactions increases, we need more bins (thus more compound variables), since each bin represents a unique way in which an agent affects another. In the extreme case of a general DEC-MDP, an agent's history needs a separate bin for each of the other's histories, essentially creating a compound variable for every pair of histories as in the DEC-MDP MILP.

#### 3.3.1.2 Enforcing the identity

We need to enforce identity (3.4) by linear constraints. This is more challenging than in the DEC-MDP case where the binary nature of the compound variables allows the use of combinatorics to devise the constraints. In our formulation, the compound variables are no longer binary, and we must resort to other properties of, and relations among, the variables to derive constraints equivalent to the identity.

Summing both sides of (3.4) over all bins of $h_g$ gives

$$\sum_{b \in B_{h_g}} z_{h_g,b} = x(h_g) \sum_{b \in B_{h_g}} \sum_{h_{-g} \in b} \beta(h_{-g}|h_g) x(h_{-g})$$

Since $B_{h_g}$ partitions $\mathcal{Z}_{-g}$, the double sum reduces to a sum over all histories of the other agent, giving

$$\sum_{b \in B_{h_g}} z_{h_g,b} = x(h_g) \sum_{h_{-g} \in \mathcal{Z}_{-g}} \beta(h_{-g}|h_g) x(h_{-g}) \tag{3.5}$$

A legal policy prescribes an action at each state reachable by a non-terminal history whose realization weight is non-zero. As a result, histories in the support set cover all possible transitions of actions along their parents. This means that the sum of probabilities of transitions along histories in the support set must be 1, i.e., $\sum_{h_{-g} \in \sigma_{-g}} \beta(h_{-g}|h_g) = 1$. It follows that

$$\sum_{h_{-g} \in \mathcal{Z}_{-g}} \beta(h_{-g}|h_g) x(h_{-g}) = 1 \tag{3.6}$$

because only the $x$s of histories in $\sigma_{-g}$ are 1, so the left side is the sum of their corresponding $\beta$s. From (3.5) and (3.6), we have the following set of constraints, one per terminal history of each agent

$$\sum_{b \in B_{h_g}} z_{h_g,b} = x(h_g) \tag{3.7}$$

This constraint simply guarantees that if $h_g$ is not part of the support, all the compound variables involving $h_g$ and its bins should be 0. If $h_g$ is part of the support, it guarantees there is enough contribution from the compound variables associated with all bins of $h_g$.

However, the above constraint does not prevent one compound variable from taking too high a value at the expense of another. We can use identity (3.4) itself as a source of

upper bounds on compound variables. Because in (3.4) $x(h_g)$ is either 0 or 1, we have the following set of constraints, one per history per bin associated with this history:

$$z_{h_g,b} \leq \sum_{h_{-g} \in b} \beta(h_{-g}|h_g) x(h_{-g}) \tag{3.8}$$

Together, constraints (3.7) and (3.8) strictly enforce the identity. One advantage of our constraints over the combinatorics-based constraints in the DEC-MDP formulation is that ours do not involve the size of the support set, which Aras calculates from the parameters of the problem by assuming that the number of states a state-action pair transitions to is constant. But in settings where this number depends on the particular action taken, determining the support size requires carefully looking at an agent's decision tree and the paths in it, which is non-trivial for large problems.

As for the number of constraints, the set of constraints in (3.7) has the same size as the constraints in the DEC-MDP MILP; there is one constraint per terminal history of each agent. The difference is that we have fewer terms in the summation on the left hand side than in the DEC-MDP MILP.

The set in (3.8), however, is larger, because it has a constraint for each bin of each terminal history of each agent, as opposed to a constraint for each terminal history. So, for example, if from the perspective of each history of each agent all the histories of the other agents fall into 1 of 3 bins, our formulation will have 3 times as many constraints of type (3.8) as there are in the DEC-MDP MILP. But as will be seen in Section 3.3.3, this does not prevent us from obtaining computational advantage over the DEC-MDP formulation.

### 3.3.2 MILP for 3 or more agents

The idea of binning histories extends beyond 2 agents. With $n$ agents, an agent's bins contains history tuples, where each tuple consists of histories of the $n-1$ other agents. The compound variable associated with a history $h_g$ and one of its bins $b$ is given by the identity

$$z_{h_g,b} = x(h_g) \sum_{h_{-g} \in b} \prod_{h_f \in h_{-g}} \beta(h_f|h, h_{-g}) x(h_f) \tag{3.9}$$

As in the 2-agent case, the set of bins associated with $h_g$ is a partition over $\mathcal{Z}_{-g}$, so we can use constraint (3.7). The greater challenge is devising linear constraints that impose upper bounds on the $z$ variables, similar to constraint (3.8). With 2-agents, we simply obtained linear constraints by dropping the leading $x$ in the identity. But with 3 or more agents, doing so would result in a non-linear constraint.

In the following, we use properties of legal policies and structured interactions to derive 2 sets of linear constraints (in addition to constraint (3.7)) that attempt, but are not guaranteed, to enforce the identity. Note that even if the identity is violated, any solution to the MILP still forms a legal set of policies, since the legality is guaranteed by the policy constraints.

For ease of exposition, we show the derivation of the constraints associated with a history $h_i$ of agent $i$ when $\mathcal{A}$ contains three agents $i$, $j$ and $k$.

If we assume that an action of agent $i$ can be affected by at most one other agent, we can decompose $b$ into a bin for each affecting agent, $b_j$ and $b_k$ ($b = b_j \times b_k$). Dropping the leading $x$ in (3.9) and using the decomposition of $b$ to break down the summation gives

$$z_{h_i,b} \leq \sum_{h_j \in b_j} x(h_j) \sum_{h_k \in b_k} x(h_k) \beta(h_j|h_i, h_k) \beta(h_k|h_i, h_j) \tag{3.10}$$

We can obtain two linear upper bounds from the above by setting all $x(h_j)$ (resp. $x(h_k)$) to 1. But these bounds would be too loose; for a feasible solution $< x_s, z_s >$ to the MILP, $z_s$ can be very different from the $z$ calculated by applying the identity to $x_s$. In other words, the solver has too much freedom to violate the identity and set some $z$s higher than the identity allows if this improves the value of the objective function. The reward reported by the solver (the value of the objective function at $z_s$) is therefore higher than the true reward obtained when the agents follow the policies prescribed by $x_s$. The solver is optimizing a

relaxed version of the problem whose optimal solution does not necessarily correspond to an optimal of the original problem. We need to tighten the upper bound so that the relaxed problem corresponds more faithfully to the original problem.

Consider the coefficient of some $x(h_j)$ in the non-linearized constraint (3.10):

$$\sum_{h_k \in b_k} x(h_k)\beta(h_j|h_i,h_k)\beta(h_k|h_i,h_j) \tag{3.11}$$

Setting all $x(h_k) = 1 \forall h_k \in b_k$ allows this coefficient to be higher than it can be under a legal policy. Regarding the coefficient as a sum over the contributions of each $h_k \in b_k$, we can decrease the coefficient by limiting the contributions of the $h_k$s. To do this, we decompose the sum in (3.11) into a series of sums over bins of $k$'s histories constructed from the perspective of $j$'s history $h_j$. We denote the bins of $k$'s histories induced by $h_j$ as $b_{h_jk}$ ($\bigcup b_{h_jk} = b_k$). Because $j$'s transition probability is the same under all $h_k$ in the same bin, we can factor this probability out. The coefficient can then be re-written as

$$\sum_{b_{h_jk}} \beta(h_j|h_i,b_{h_jk}) \sum_{h_k \in b_{h_jk}} x(h_k)\beta(h_k|h_i,h_j)$$

Now we can use the same reasoning behind constraint (3.7) to replace the second summation involving $x(h_k)$ with an upper bound on it

$$\left\lfloor \sum_{h_k \in b_{h_jk}} \beta(h_k|h_i,h_j) \right\rfloor \tag{3.12}$$

where $\lfloor x \rfloor$ denotes min($x$,1). We can therefore bound the factor multiplying each $\beta(h_j|h_i,b_{h_jk})$ to be at most 1. The coefficient of $x(h_j)$ is restricted to be

$$\sum_{b_{h_jk}} \beta(h_j|h_i,b_{h_jk}) \left\lfloor \sum_{h_k \in b_{h_jk}} \beta(h_k|h_i,h_j) \right\rfloor$$

We can obtain a coefficient for each $x(h_k)$ in a similar fashion. Using these restricted coefficients, and the fact that a coefficient cannot exceed 1, we approximately enforce identity (3.9) using constraint (3.7) and the following bounds

$$\begin{aligned} z_{h_i,b} &\leq \sum_{h_j \in b_j} x(h_j) \left\lfloor \sum_{b_{h_j k}} \beta(h_j|h_i, b_{h_j k}) \left\lfloor \sum_{h_k \in b_{h_j k}} \beta(h_k|h_i, h_j) \right\rfloor \right\rfloor \\ z_{h_i,b} &\leq \sum_{h_k \in b_k} x(h_k) \left\lfloor \sum_{b_{h_k j}} \beta(h_k|h_i, b_{h_k j}) \left\lfloor \sum_{h_j \in b_{h_k j}} \beta(h_j|h_i, h_k) \right\rfloor \right\rfloor \end{aligned} \tag{3.13}$$

The quest for tight linear relaxations for non-linear functions is common in the optimization literature. A trilinear term of the form $xyz$ where $x$, $y$ and $z$ are between 0 and 1 can be replaced by a new variable $w$ and the following set of constraints:

$$\begin{aligned} w &\leq x \\ w &\leq y \\ w &\leq z \\ w &\geq x + y + z - 2 \end{aligned} \tag{3.14}$$

The above is a a linear relaxation of the term's convex envelope and guarantees that $w$ is within a certain amount of the product $xyz$ [55]. Although these constraints are somewhat similar to the constraints our formulation generates for the 3-agent case, there is a problem in using them directly. The identity (3.9) does not define $z_{h_g,b}$ 'cleanly' as just the product of 3 variables; there are summations and constants involved. Using the upper bounds in (3.12) to obtain a clean trilinear term would interfere with the last inequality in (3.14) because the right-hand side may no longer be a lower bound on $z$.

Even if we could use the above linear relaxation, there is no guarantee that it is tighter than the relaxation we developed using properties of legal policies.

The idea of further binning histories within a given bin to bound the values of coefficients can be used with any number of agents. For $n$ agents, this would result in $n-1$ upper bounds per $z$ variable[1].

### 3.3.3 Experimental results

We now present experimental results of our two formulations applied to the Mars rovers problem in Section 2.2.1.3.

#### 3.3.3.1 Results of 2-agent formulations

We compare 3 formulations of EDI-CR instances: 1) the NLP formulation in Table 3.2, but restricted to 2 agents, 2) the DEC-MDP MILP formulation in Table 3.3 and 3) the EDI-CR MILP formulation in Table 3.4. All 3 formulations were solved using IBM ILOG Cplex [2] under the default parameters; the first using Cplex Mixed Integer QP solver, and the other two using Cplex MILP solver. We experimented with 22 instances of the modified Mars rovers problem. The number of interactions ranges from 4 to 7.

We note that the time to *generate* the 3 formulations is almost the same; constructing the bins and objective function for the EDI-CR MILP is not more expensive than constructing the reward matrix for the QP or the objective function for the DEC-MDP MILP. In all 3 cases, we iterate over every pair of histories of the 2 agents to calculate their rewards and probabilities.

**Optimality:** First, we look at the behavior of the 3 formulations with respect to optimality. Note that even after obtaining a solution that we know is optimal (by comparing to a known optimal solution), Cplex may spend a long time verifying optimality. We therefore have 5 possible outcomes of a run:

1. Optimal solution found and verified

[1] Higher order terms can be relaxed by repeated application of relaxations of lower order terms [24].

**Table 3.5.** Optimality of 2-agent formulations

| | QP | DEC-MDP MILP | EDI-CR MILP |
|---|---|---|---|
| 1) Optimal, Verified | 5 | 9 | 17 |
| 2) Optimal, Not verified | 9 | 5 | $x$ |
| 3) Local optimal | 5 | - | - |
| 4) Suboptimal | 3 | 6 | 5-$x$ |
| 5) No solution | 0 | 2 | 0 |

2. Optimal solution found but not verified before time out[2]

3. Locally optimal solution found (only possible in solving the QP)

4. Optimal solution not found before time out, but a suboptimal solution was found

5. No solution found before time out

Of the 22 instances, Table 3.5 compares how many fall in each category for each formulation. Because a MILP solver would never report solution that is only locally optimal, the corresponding entries are marked by '-'. Our formulation resulted in a provably optimal solution in 17/22 instances. In the remaining instances, we obtained higher rewards than the other formulations, but cannot say with certainty that our solution was optimal, so each of the remaining 5 instances falls into category 2 or 4. QP and DEC-MDP MILP were equally good at finding optimal solutions, although DEC-MDP MILP was better at verifying optimality. The table shows that the non-concavity of the QP can often lead the solver to just report a locally optimal solution. It also shows that in some cases, the number of compound variables introduced in the DEC-MILP is too large to allow the solver to find *any* solution before time out (row 5).

**Formulation size:** Next, we look at the size of the MILP with and without exploiting structured interactions. We break down our 22 instances into 3 groups G1, G2 and G3 containing 5, 9 and 8 instances, respectively. Table 3.6 shows the number of terminal

[2]Time out is 60 seconds for small instances and 600 seconds for larger ones.

**Table 3.6.** Size of 2-agent formulations

| | $\mathcal{Z}_i$ | $\mathcal{Z}_j$ | $z_{EDI}$ | $z_{DEC}$ | $C_{EDI}$ | $C_{DEC}$ |
|---|---|---|---|---|---|---|
| G1 | 81 | 46 | 254 | 3,762 | 381 | 127 |
| G2 | 162 | 112 | 608 | 18,062 | 882 | 274 |
| G3 | 941 | 781 | 3,793 | 596,950 | 5,515 | 1,722 |

histories for each agent $|\mathcal{Z}_i|$ and $|\mathcal{Z}_j|$, the number of compound variables introduced in the DEC-MDP formulation$|z|_{DEC}$ and our EDI-CR formulation $|z|_{EDI}$, and the number of constraints (besides the policy constraints). Results were averaged over instances in each group. Clearly, the DEC-MDP formulation introduces many more compound variables than our formulation which only create as many variables as needed to distinguish between bins induced by a given history. The difference in the number of variables becomes more pronounced as the problem size increases. Although our formulation has more constraints than the DEC-MDP MILP, we next show that the increased number of constraints is offset by the large reduction in the number of variables, resulting in MILPs that are overall easier to solve.

**Solution time:** Table 3.7 shows the results of comparing both the time needed to find the optimal solution (reported as 'Find'), and the time needed to verify that the solution is indeed optimal (reported as 'Verify'). The times are in seconds, averaged over instances in each group. For groups where some solutions were not found/verified within reasonable time, the number of instances over which the averaging was done is indicated in brackets. In general, solving the EDI-CR MILP formulation is significantly faster than solving the other 2 formulations. There is also a large difference in the time needed to verify optimality. In the Small group, only 3 instances could be solved provably optimally within 60 seconds using the DEC-MDP MILP and QP formulations. In the Medium group, the differences in time to verify optimality among the different formulations is even more pronounced. In the Large group, Cplex found solutions for all the instances of the EDI-CR MILP formulation, but could not verify optimality. A solution with the same quality could not be found with any of the other formulations.

**Table 3.7.** Solution time (in seconds) of 2-agent formulations

| | Find EDI | Find DEC | Find QP | Verify EDI | Verify DEC | Verify QP |
|---|---|---|---|---|---|---|
| G1 | 0.29 | 8.68 | 0.57 | 0.12(3) | 3.5(3) | 0.58(3) |
| G2 | 0.59 | 10.72 | 6.4 | 0.35(6) | 21.6(6) | $> 60$ |
| G3 | 83 | N/A | N/A | N/A | N/A | N/A |

**Table 3.8.** Comparison of 3-agent formulations (size)

| | $\mathcal{Z}_i$ | $\mathcal{Z}_j$ | $\mathcal{Z}_k$ | $z_{EDI}$ |
|---|---|---|---|---|
| G1 | 116 | 114 | 102 | 507 |
| G2 | 163 | 170 | 220 | 898 |
| G3 | 285 | 184 | 158 | 1265 |
| G4 | 263 | 267 | 654 | 1567 |

#### 3.3.3.2 Results of 3-agent formulations

The 3-agent case exacerbates the problem of the DEC-MDP MILP formulation which introduces hundreds of thousands to several millions variables in our test cases. Because Cplex was unable to solve (and usually even load) the DEC-MDP MILP of our instances, we omit this formulation from further discussion.

We compare the NLP in Table 3.2, solved using Knitro [3], and EDI-CR 3-agent MILP from Section 3.3.2 solved using Cplex [2] under the default parameters. Left to automatically determine the most suitable algorithm, Knitro chose the active-set algorithm. We used 25 instances of the Mars Rovers problem broken down into 4 groups G1 to G4 containing increasingly larger instances. G1 contains 10 instances and each of the other groups contains 5 instances. The number of interactions ranges from 4 to 10.

**Table 3.9.** Comparison of 3-agent formulations (time in seconds)

| | Reward Calculation | Dom. Removal | Binning | MILP | NLP | MILP Total | NLP Total |
|---|---|---|---|---|---|---|---|
| G1 | 192 | 78 | 6.6 | 1.2 | 27.4 | 278 | 297.7 |
| G2 | 1296 | 353 | 56.2 | 7.22 | 284.9 | 1712 | 1933 |
| G3 | 2063 | 802 | 119 | 38 | 588 | 3022 | 3453 |
| G4 | 3376 | 480 | 478 | 72 | 1800 | 4406 | 5656 |

**Table 3.10.** Comparison of 3-agent formulations (reward as % of maximum)

| | MILP | NLP |
|---|---|---|
| G1 | 88 | 86 |
| G2 | 85.7 | 88.3 |
| G3 | 88.7 | 87.2 |
| G4 | 88.6 | 43.6 |

Tables 3.8,3.9 and 3.10 summarize our experimental results averaged over instances in each group. Table 3.8 shows the sizes of the instances we tested on and the number of compound variables our formulation creates. The number of non-policy constraints introduced by our formulation can be calculated as $|\mathcal{Z}_i| + |\mathcal{Z}_j| + |\mathcal{Z}_g| + 2|z|$. The first 3 terms are due to constraint (3.7) and the last term is due to the upper bounds in (3.13). It is important to note that the constraint matrix, although large, is fairly sparse; the constraint of a terminal history only involves the $z$s of this history's bins, and the constraint of a $z_{h,b}$ only involves histories in $b$ of 1 affecting agent. As will be shown presently, Cplex solves EDI-CR MILPs in very little time.

To generate both the MILP and NLP, we need to calculate rewards for each triplet $< h_i, h_j, h_k >$ (time indicated in column 'Reward Calculation') and remove dominated histories up front (indicated in column 'Dom. Removal')[3], overhead that was insignificant in the 2-agent case. Column 'Bin' shows the time to construct bins and calculate their associated constraint matrices for the MILP. Although this step is rather expensive, it results in a MILP that is solved at least an order of magnitude faster than the NLP. Actual solution times taken by Cplex are given in columns 'MILP' and 'NLP'. We also give the total time to generate and solve the MILP (reward calculation + dominated histories removal + bin construction + solver time) in column 'MILP Total' and for NLP (reward calculation + dominated histories removal + solver time) in 'NLP Total' We note that the dominated histories removal step can be sped up by optimizing our implementation of it. Even with our

[3]Without this pre-processing step (details in [8]) none of the formulation would fit in memory. The reported numbers are those of undominated histories.

current implementation, the time to solve the NLP far exceeds the bin constraints construction time and the MILP solving time combined. The difference becomes more pronounced with larger instances. Indeed, for instances in G4, we timed out Knitro and reported the reward it obtained after 30 minutes.

Finally, Table 3.10 compares the rewards obtained by the policies from the NLP and MILP solutions. As explained in Section 3.3.2, our MILP is a relaxation of the original problem where some $z$s can be higher than their values under identity (3.9). The solution reported by the MILP is therefore an over-estimate of the optimal reward. The table shows the reward of the MILP and NLP policies as a percentage of this over-estimate. For smaller instances, MILP and NLP give comparable rewards, but on larger ones, Knitro is unable to produce good policies within 30 minutes.

Whereas we do not know of a way to improve the quality of the NLP solution, that of the MILP can be improved by having a more faithful correspondence between the MILP and the original problem, i.e. making the upper bounds on $z$s tighter and/or obtaining lower bounds on $z$. This would result in a space of feasible solutions that better resembles that of the original problem, and would prevent the solver from pursuing solutions that appear to be good, but are sub-optimal in the original space. This represents an interested area for future research.

### 3.3.4 Related work

Formulating decision problems as mathematical programs has been done by other researchers, with the aim of making use of available industrial-grade solvers like Cplex. Aras and Dutech proposed two MILP formulations for general DEC-POMDPs [8]. One of these formulations was given in Table 3.3. The other uses game-theoretic concepts to linearize the objective function. This latter formulation is out-performed by the one we reviewed and built upon.

Petrik and Zilberstein [69] developed formulations of decision problems of cooperative and self-interested agents as separable bilinear programs and presented an algorithm for solving this class of programs. The QP discussed in this paper is itself a bilinear program, because realization weights of one agent are only multiplied by weights of the other agent, so the objective function is linear if the weights of one agent are fixed. Our previous work on the EDI-CR model [60] used the bilinear formulation and solution algorithm.

Aras et. al [9] give a mathematical formulation for a special case of DEC-POMDP called Network Distributed POMDP [67] (ND-POMDP) where agents have independent transition and observation functions, but have reward interactions. In ND-POMDP, we can decompose the set of agents into subsets, where an agent's reward only depends on agents belonging to its subset(s). Because they only consider problems where each subset contains 2 agents (i.e. binary interactions), Aras et. al were able to formulate this problem as a QP. They present a linearization of the QP to a compact MILP that avoids having a compound variable for each joint terminal history. However, they report that the compactness of their formulation does not translate to savings in the time needed to solve the resulting MILP, compared to a simple formulation with one variable per joint history. One explanation they provide is that the compact MILP has a constraint matrix that is not sparse, making it hard for Cplex to deal with it efficiently.

Besides the fact that ND-POMDP assumes transition independence and EDI-CR does not, the difference between these two models is that in the former, agents belonging to the same subset are assumed to have very tight reward interaction; there is a single reward function per subset, and it is defined over joint actions and states of agents in the subset. We can see this as a coarse-grained kind of independence where agents either have reward interactions involving all their actions or none at all. EDI-CR captures a more fine-grained kind of interaction where *specific* actions affect, or are affected by, what another agent does. As a result of this difference, formulations of ND-POMDP would not be very useful, if at all, when directly applied to EDI-CR, since they cannot capture and exploit fine-grained

interactions. The general ideas and techniques for linearizing a high order expression can, however, be useful across models and formulations.

Another MILP formulation of Transition-Independent DEC-MDP [14] is given by Wu and Durfee [95]. Their formulation is approximate and is the result of discretization and piecewise linear approximation of non-linear constraints. This work therefore finds exact, optimal solutions to an inexact model. The errors introduced by the discretization and linearization can be controlled at the expense of introducing more variables into the formulation. Because Transition-Independent DEC-MDP is a sub-class of EDI-CR, we cannot use its MILP formulation for our model.

One way of decomposing a large mathematical program that encodes the decision problems of all agents is to break down this problem into 1) a search for optimal commitments regarding each agent's outgoing influences and 2) a search for optimal local policies that respect the commitments decided upon. This is the approach taken by Witwicki et. al [92]. For a given set of commitments, they add constraints to the traditional Linear Program formulation of MDPs to guarantee that a feasible policy respects the commitments. Each agent can then solve its linear program separately.

## 3.4 Solving EDI-CR Using Homotopy Methods

In this section, we explore a different way of solving EDI-CR instances, and more generally, DEC-MDPs. In previous sections, we formulated the problem of finding the optimal policy as a mathematical program. We now show a different way of approaching the same problem. We formulate the problem of finding the optimal policy as a system of nonlinear equations and discuss the use of homotopy methods for solving this system.

We start by giving a background on homotopy methods then discuss the use of homotopy for EDI-CR and DEC-MDPs.

### 3.4.1 Introduction to homotopy

Continuation methods have been widely used in numerical analysis to solve systems of nonlinear equations $F(x) = 0$. The system of equations can encode a constrainted or unconstrained optimization problem, a problem of finding the zeros of a function, finding the fixed point of a function, or tracking the curve of a function. The words *continuation* and *homotopy* are often used interchangeably, but there are subtle distinctions between them (which are beyond the scope of this thesis). Probability-one homotopy methods [90, 29, 91] are variants of homotopy methods that guarantee the convergence to a solution with probability 1.

The Encyclopedia of Optimization [33] defines homotopy as a continuous map from the interval [0,1] into a function space, where the continuity is with respect to the topology of the function space. A homotopy $\rho(\lambda)$ continuously deforms the function $\rho(0) = g$ into the function $\rho(1) = f$ as $\lambda$ goes from 0 to 1. In this case, $f$ and $g$ are said to be *homotopic*. It is then clear how continuation methods got their name; they move along a continuum between solving an easy variant of the original problem $g$ and solving the original problem $f$ itself. Essentially, these methods find a solution to the easy variant, then try to calculate the change that needs to be made to the previous solution to make it solve a new problem that is a little closer to the original problem, eventually terminating with a solution to the original problem.

We start by giving some notation used by Watson in [90] and will be used in this section. Let $E^n$ denote $n$-dimensional Euclidean space and $E^{m\times n}$ the set of real $m \times n$ matrices. The gradient of a differentiable function $f : E^n \to E$ is the row vector $(\frac{\partial f}{\partial x_1}(x), ..., \frac{\partial f}{\partial x_n}(x))$. The Jacobian matrix of $F : E^n \to E^m$ is

$$\nabla F(x) = \begin{pmatrix} \nabla F_1(x) \\ \vdots \\ \nabla F_m(x) \end{pmatrix}$$

A *homotopy map* $\rho$ is a mapping $\rho : E^m \times [0,1) \times E^n \rightarrow E^n$. Simply put, $\rho$ is a system of $n$ functions operating on vectors $x \in E^n$. The functions are parameterized by a vector $a \in E^m$ and the homotopy parameter $\lambda \in [0,1)$. The shorthand $\rho_a(\lambda, x)$ denotes $\rho(a, \lambda, x)$.

**Definition 10** *A map is said to be* transversal to 0 *if its Jacobian* $\nabla\rho$ *has full rank on* $\rho^{-1}(0)$.

The following Lemma from [90] states the conditions $\rho$ must meet to guarantee convergence to a solution of the original problem.

**Lemma 1** *Suppose that* $\rho$ *is transversal to 0 and for each* $a \in E^m$ *the system* $\rho_a(0, x) = 0$ *has a unique nonsingular solution* $x^{(0)}$. *Then for almost all* $a \in E^m$ *there is a smooth* zero curve $\gamma \subset [0,1) \times E^n$ *of* $\rho_a(\lambda, x)$, *emanating from* $(0, x^{(0)})$, *along which the Jacobian matrix* $\nabla\rho_a(\lambda, x)$ *has rank* $n$. $\gamma$ *does not intersect itself or any other zero curves of* $\rho_a$, *does not bifurcate, has finite arc length and either goes to infinity or reaches the hyperplane* $\lambda = 1$ *at point* $(1, x^*)$. *If rank* $\nabla\rho_a(\lambda, x) = n$, *then* $\gamma$ *has finite arc length.*

The zero curve $\gamma$ traces the change in the solution $x$ as $\lambda$ changes. For a given point $(\hat{\lambda}, \hat{x})$ on $\gamma$, we have that $\rho_a(\hat{\lambda}, \hat{x}) = 0$. In other words, $\hat{x}$ is the solution to the system of equations when it is deformed by $\hat{\lambda}$. $x^{(0)}$ is therefore the solution to the initial system of equations and $x^*$ is the solution to the original system. The condition on the Jacobian $\nabla\rho_a(\lambda, x)$ guarantees that there is only one way to proceed along the zero curve, thus avoiding bifurcation.

The lemma summarizes the improvement of probability one homotopy methods over the original continuation methods which can fail to terminate because the curve can spiral, return to $\lambda = 0$ or bifurcate, among other problems.

The high-level steps involved in solving a problem using homotopy methods are:

1. Formulate the problem as a system of $n$ equations in $n$ unknowns $f(x) = 0$.

2. Construct a homotopy map $\rho(\lambda, x)$ satisfying the conditions of the homotopy method.

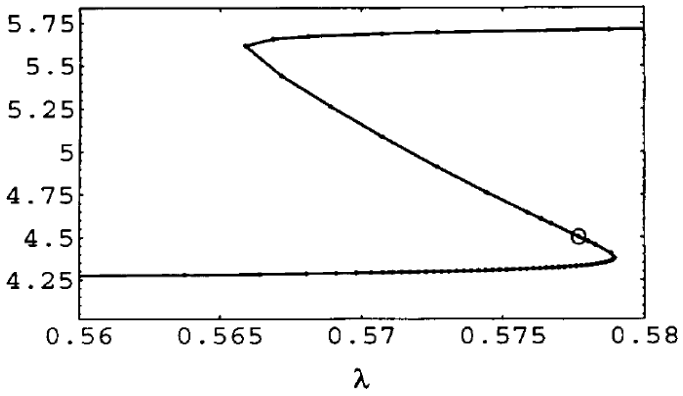



**Figure 3.1.** An example zero curve of a homotopy map. The x-axis is $\lambda$ and the y-axis is $x$.

3. Track the zero curve of $\rho$ from the point $(0, x^{(0)})$ to $(1, x^*)$.

There are several ways of constructing the homotopy map $\rho$, and choosing the map that will perform best is more of an art than an exact science.

#### 3.4.1.1 Natural vs. 'artificial' parameter $\lambda$

Sometimes $\lambda$ is a physical parameter in the original function $f$ and we are interested in how the solution of the deformed $f(x; \lambda)$ changes as we change this physical parameter, i.e., we are interested in the solution behavior as we change the natural parameter. But it may also be the case that $\lambda$ is an artificial parameter that just allows us to transition smoothly from $g$ to $f$, and solutions to $\rho(\lambda, x) = 0$ have no physical meaning for $\lambda < 1$. One commonly used artificial parameter map is just a convex combination of the functions $g$ and $f$, which gives rise to the 'canonical' map

$$\rho(\lambda, x) = \lambda f(x) + (1 - \lambda)g(x) \tag{3.15}$$

Note that unlike in continuation methods, in homotopy methods $\lambda$ is not necessarily monotonically increasing, as in Figure 3.1. The progress along $\gamma$ is therefore not expressed

in terms of $\lambda$, but in terms of arc length. Let $p$ denote the function that maps arc length, denoted by $s$, to a point on $\gamma$, i.e.

$$p(s) = (\lambda(s), x(s))$$

#### 3.4.1.2 Tracking the zero curve

Tracking the $\gamma$ curve is a complicated matter that has been the subject of research in numerical analysis. HOMPACK90 [91] is a software package for this kind of tracking that has been used widely. For the rest of this section, we will be referring to a tracking technique implemented in HOMPACK90 called the *normal flow algorithm*. This algorithm has 3 phases repeated until $\lambda$ reaches 1:

1. **Prediction**: For an estimate $h$ of the optimal step size (in arc length) to take along $\gamma$, and given 2 previous points on $\gamma$, $P^1 = (\lambda(s_1), x(s_1))$ and $P^2 = (\lambda(s_2), x(s_2))$, predict the next point on the curve, $Z^{(0)} = p(s_2 + h)$. The prediction is done using Hermite cubic interpolation. The predicted point will typically not be on the $\gamma$ curve.

2. **Correction**: Starting from the predicted point $Z^{(0)}$, apply the following Newton corrector iteration until convergence (i.e. the change in $Z$ is small enough)

$$Z^{(k+1)} = Z^{(k)} - [\nabla \rho_a(Z^{(k)})]^\dagger \ \ \rho_a(Z^{(k)})$$

   where $[\nabla \rho_a(Z^{(k)})]^\dagger$ is the Moore-Penrose inverse of the Jacobian of $\rho$ at $Z^{(k)}$. The goal of the corrector is to bring back the predicted point to the zero curve. When the corrector iteration converges, the $Z$ from the last iteration is accepted as the next point on $\gamma$.

3. **Step size estimation**: this phase calculates an estimate $h$ of the optimal size of the next step taken on the zero curve. A large step size makes faster progress along the

curve, but may result in very inaccurate prediction that then requires many corrector iterations to bring back to the zero curve. A smaller step size results in accurate predictions but requires many iterations of the 3 phases. The details of this step are such that $h$ increases if the corrector phase needed a few iterations to converge and decreases otherwise. Basically, if the previous prediction step resulted in a point far off $\gamma$, the algorithm becomes more 'cautious' and takes a smaller step in the future.

### 3.4.2 Homotopy for linearly constrained optimization

Consider the following optimization problem where $A \in E^{m \times n}$ and $b \in E^m$

$$\begin{aligned} &\text{Maximize} \quad f(x) \\ &\text{subject to} \quad g(x) = Ax - b \leq 0 \end{aligned} \tag{3.16}$$

The Karush-Kuhn-Tucker (KKT) conditions for this problem are

$$\begin{gathered} (\nabla f(x))^{\mathsf{T}} + A^{\mathsf{T}} u = 0 \\ u \geq 0 \\ Ax - b \leq 0 \\ u(Ax - b) = 0 \end{gathered}$$

where $u$ are the Lagrange dual variables for inequality constraints. The last 3 constraints are called complementarity constraints. They can be expressed as a system of nonlinear equations $K(u, x) = 0$ defined by

$$K_i(u, x) = -|b_i - A_i x - u_i|^3 + (b_i - A_i x)^3 + u_i^3 \qquad 1 \leq i \leq m$$

A simple map for the above optimization problem is

$$\rho_a(\lambda, x, u) = \lambda \begin{pmatrix} (\nabla f(x))^{\mathsf{T}} + A^{\mathsf{T}} u \\ K(u, x) \end{pmatrix} + (1 - \lambda) \begin{pmatrix} x - x^0 \\ u - u^0 \end{pmatrix} \tag{3.17}$$

where $a = (x^0, u^0)$. However, the above map can result in an unbounded zero curve. The fix suggested by Watson [90] is to replace the constraint in (3.16) by

$$Ax - b - (1 - \lambda)b^0 \leq 0$$

Basically, the above is a relaxation of the original constraints that allows $Ax - b$ to be greater than 0 if the constant $b^0$ is non-zero. Consequently, $K$ is now a function of $\lambda, x, u, b^0$. A similar relaxation is introduced to allow $K(\lambda, x, u)$ to be greater than 0 if a parameter $c^0$ is positive. As a result of these changes, the original $\lambda K(x, u) + (1 - \lambda)(u - u^0)$ is replaced by

$$K_i(\lambda, x, u, b^0, c^0) = -|(1 - \lambda)b_i^0 + b_i - A_i x - u_i|^3 + ((1 - \lambda)b_i^0 + b_i - A_i x)^3 + u_i^3 - (1 - \lambda)c_i^0$$

and the map becomes

$$\rho_a(\lambda, x, u) = \begin{pmatrix} \lambda[(\nabla f(x))^\mathsf{T} + A^\mathsf{T} u] + (1 - \lambda)(x - x^0) \\ K(\lambda, x, u, b^0, c^0) \end{pmatrix} \tag{3.18}$$

where $a = (x^0, b^0, c^0)$.

Watson [90] discussed the use of maps like the one above when the optimization problem (3.16) is non-convex, in which case the KKT conditions are satisfied by stationary, as well as optimal, points. Nonetheless, Watson states that we can still use maps like the one above because for the convergence proofs he gives, convexity is a sufficient, but not necessary condition.

### 3.4.3 Solving DEC-MDPs using homotopy

Solving a 2-agent DEC-MDP can be formulated as solving the following bilinear program [69]

$$\begin{aligned} \text{Maximize} \quad & f(x) = x_i^{\mathsf{T}} C x_j \\ \text{subject to} \quad & A_i x_i = b_i \\ & A_j x_j = b_j \\ & x_i, x_j \geq 0 \end{aligned} \tag{3.19}$$

$x_i$ and $x_j$ are the policies of the 2 agents and the matrix $C$ encodes the rewards that depend on actions of both agents. The constraints guarantee that $x_i$ and $x_j$ represent legal policies.

Clearly, if it was not for the bilinear term $x_i^{\mathsf{T}} C x_j$, the problem would degenerate into 2 independent optimization probelms that can be easily solved. This simple observation suggests homotopy methods as a way of moving from a solution of the easy independent problems (MDPs) to a solution of the original coupled problem (DEC-MDP).

The canonical map (3.15) does not involve the homotopy parameter $\lambda$ in the function $f$. But in our case, we would like to do just that. We would like to involve $\lambda$ in the objective function of (3.19) to transition from a problem that does not care about interactions to one that takes all interactions into consideration. Involving $\lambda$ in the problem is a called *embedding*, a technique that has been used to solve problem which could not be solved using the standard map (e.g., the Mixed Complementarity Problem [4]).

Using embedding, we want to optimize the following family of objective functions

$$f(x, \lambda) = (1 - \lambda) x_i^{\mathsf{T}} r_i + \lambda x_i^{\mathsf{T}} C x_j + (1 - \lambda) x_j^{\mathsf{T}} r_j \tag{3.20}$$

where $r_i$ and $r_j$ are the rewards from the agents' individual MDPs. Note that because the agents affect each other, the transition and reward functions of one agent's MDP are under-specified in the absence of any assumptions about what the other agent will do. So in order to construct an MDP for agent $i$, for example, we assumed that agent $j$ will act in a way that is best for $i$. At $\lambda = 0$, the policies that maximize the above objective function are the optimal policies of the individual MDPs.

Moving $\lambda$ from 0 to 1 increases the importance of the bilinear term and therefore requires the solver to consider the other agent's actions when optimizing one agent's problem. In other words, the problem smoothly transitions from complete independence to a problem with tighter interactions that requires more coordination.

We experimented with several maps that encode our optimization problem as a system of equations that is 0 at the optimal policy. We started with a map similar to (3.18) but adapted to deal with the equality constraints representing the policy constraints in (3.19).

$$\rho_a(\lambda, x, u, v) = \begin{pmatrix} (\nabla f(x, \lambda))^{\mathsf{T}} - u + E^{\mathsf{T}} v \\ K(\lambda, x, u, b^0, c^0) \\ Ex - b \end{pmatrix} \tag{3.21}$$

where $E = \begin{pmatrix} A_i & 0 \\ 0 & A_j \end{pmatrix}$ is the constraint matrix of the policy constraints in (3.19), and $b = [b_i; b_j]$. $\nabla g(x) = -I$. In the above, $\rho_a : \mathbb{R}^N + 1 \rightarrow \mathbb{R}^N$ where $N = 2(|\mathcal{H}_i| + |\mathcal{H}_j|) + |Con_i| + |Con_j|$ and $Con$ is the number of constraints. The top part of the map has an equation per history per agent and $K$ has an equation for each non-negativity constraints. The bottom part has an equality per policy constraint.

The problem with the above map is that it results in a Jacobian $\nabla\rho$ that is of deficient rank at $\lambda = 0$; some $x$s will have identical columns in $\nabla\rho$. To see how this can happen, consider 2 terminal histories of agent $i$, $h_1$ and $h_2$. If these histories are extensions of the same parent history, the variables of their corresponding realization weights, $x_{h_1}$ and $x_{h_2}$, will have identical columns in the constraint matrix $A_i$; having no children, they are both involved in only the one constraint with the parent history. For variables in $x_i$, the relevant sub-matrix of $\nabla\rho$ is (dimensions indicated):

$$\frac{\partial \rho}{\partial x_i} = \begin{pmatrix} 0 & |\mathcal{H}_i| \times |\mathcal{H}_i| \\ \lambda C^{\mathsf{T}} & |\mathcal{H}_j| \times |\mathcal{H}_i| \\ \frac{\partial K_{1..|\mathcal{H}_i|}}{\partial x_i} & |\mathcal{H}_i| \times |\mathcal{H}_i| \\ 0 & |\mathcal{H}_j| \times |\mathcal{H}_i| \\ A_i & |Con_i| \times |\mathcal{H}_i| \\ 0 & |Con_j| \times |\mathcal{H}_i| \end{pmatrix}_{N \times |\mathcal{H}_i|}$$

At $\lambda = 0$, $\frac{\partial K_{1..|\mathcal{H}_i|}}{\partial x_i} = 0$ and the columns belonging to $x_{h_1}$ and $x_{h_2}$ become identical, violating the requirement of $\rho$ being transversal to 0 in Lemma (1).

To remedy the above problem, we need to have a term in the first part of the map that involves $x$ and results in distinct columns in $\nabla\rho$. One way of doing this is to use the following map

$$\rho_a(\lambda, x, u, v) = \begin{pmatrix} (\nabla f(x, \lambda))^{\mathsf{T}} - u + E^{\mathsf{T}} v + (1 - \lambda) G(x - x^0) \\ K(\lambda, x, u, b^0, c^0) \\ Ex - b \end{pmatrix} \tag{3.22}$$

where $a = (x^0, b^0, c^0)$ and $x^0 = [x_i^0; x_j^0]$ is the solutions to the 2 independent MDPs. $G$ is a diagonal matrix that can be used for scaling, but here we just set it to $I$.

In our discussion so far, we have not used anything that is specific to EDI-CR; the optimization problem in (3.19) represents a general DEC-MDP and the above map is therefore valid for general DEC-MDPs. What makes homotopy methods particularly suited to solving EDI-CR instances is that in the latter, interaction among agents is relatively sparse. Consequently, the starting point representing solutions to the MDPs under the assumption of independence should be a good point. In a general DEC-MDP, the solution of the actual coupled problem may be completely different from the solution to the easy variant. In EDI-CR, we hope that the solution of the MDPs is fairly close to the solution of the original problem.

### 3.4.4 Challenges in using homotopy

We have encoded the above map in the homotopy software package HOMPACK [91]. We tried to solve simple instances of EDI-CR but faced many challenges in doing so, which we discuss below. This discussion contains some very technical material, but we feel that it is important to document these technicalities so that future researchers interested in using homotopy methods in general, and HOMPACK in particular, can learn from our experience. We hope the following discussion can serve as a kind of trouble-shooting guide, suggesting possible culprits in the homotopy code and framework if the approach does not work out of the box.

The first challenge was to get familiar with and understand the HOMPACK code, which is in FORTRAN90 and does not come with very clear documentation. To start using HOMPACK, the user must first implement routines that encode her homotopy map and its Jacobian. When the simple problems were taking too long to converge, we also needed to understand the detailed linear algebra routines enough to find out which routines were responsible for this and modify some of their internal parameters.

#### 3.4.4.1 Sparse linear algebra

Because our problems involve more than a few hundred variables, we needed to use the sparse version of the linear algebra routines (for example, in one problem the Jacobian has 63,000 elements, only 2585 of which are non-zero). A major challenge in using HOMPACK concerns the stability of the sparse linear solver. The same problem was reported by Borkovsky et. al [22] who cautioned

> "Both Layne Watson (the principal author of HOMPACK90) and Ken Judd (an authority on numerical methods in economics) acknowledge that the GMRES method can and does fail for some problems. There is no guidance as to which problems are susceptible, but we strongly suspect problems with extremely sparse Jacobians."

We spent a lot of time trying to squeeze all the structural zeros out of our system of equations, thereby reducing the size of the Jacobian of $\rho$, which strongly affects the efficiency of the homotopy method. On the other hand, Borkovsky recommends reducing the size and sparsity of the Jacobian by eliminating variables. It is not clear whether we should strive for more or less sparsity, but if we are to take Borkovsky's advice, we need to completely re-formulate the problem to use fewer variables.

Upon deeper inspection, we found that one routine that was taking very long to converge is called during the corrector phase. Part of the corrector phase in the normal flow algorithm (detailed in Section 3.4.1) involves solving a system of linear equations to calculate the kernel of the Jacobian $\nabla\rho$. This is done using the Generalized minimal residual method (GMRES, [72]). GMRES is a Krylov subspace method; i.e. it approximates the solution of the system of equations by the vector in a Krylov subspace with minimal residual. In iteration $i$, GMRES considers the $i^{th}$ Krylov subspace. For a matrix of rank $m$, the $m^{th}$ subspace is the entire space and GMRES arrives at the exact solution, but the idea is that after a few iterations, the solution obtained from the $k < m$ subspace is a good approximation. GMRES is often used with restarts; in GMRES($k$),after the $k^{th}$ (inner) iteration, accumulated data are cleared and the intermediate results are used as the initial data for the next $k$ iterations. These outer iterations continue until convergence. The difficulty is in choosing an appropriate value for $k$. If $k$ is too small, GMRES($k$) may be slow to converge, or fail to converge entirely. A larger $k$ involves excessive work (and uses more storage).

In our experiments, we had to increase $k$ from its default value, a recommendation that was also made in [22]. We also increased the tolerance associated with the test for convergence.

Another reason the GMRES routine was terminating with an error flag is that at some point, the system it was solving was ill-conditioned. For a reason we could not find out, GMRES was operating on a matrix with a very large condition number. We found a flag in

the code (called `STRONG_VERSION`) which if set to False, stops GMRES from complaining about large condition numbers. The root of the problem, however, is still unknown.

#### 3.4.4.2 Small steps in prediction

As mentioned in Section 3.4.1, when the corrector phase of one iteration takes many iterations to converge to a point on the $\gamma$ curve, the step size $h$ of the next iteration is reduced. The predicted point in the new iteration depends both on $h$ and on the previous 2 points on the curve and their tangents. One problem we faced was that the $\lambda$ of a new point was almost the same as that of the previous point, i.e. the predictor phase takes very small steps along the $\lambda$ dimension. In one case, the rate of change of $\lambda$ with respect to the arc length $s$ was calculated to be 0.00695, which means there is virtually no progress. Given that the predictor phase is what actually moves us along the $\gamma$ curve, the algorithm as a whole was not making much progress beyond the initial point.

We tried to force the algorithm to be less cautious by not penalizing $h$ very much when the corrector phase took a long time to converge. But the result was that the next prediction did take a large step, but ended up so far from the curve that correction took even longer. So it seems that caution is indeed justified, which suggests that the curve is highly winding. The questions is whether this is a problem with the particular homotopy map we used, or a more fundamental problem with using homotopy methods to solve our kind of problems.

#### 3.4.4.3 Possible alternatives

As a final remark regarding our attempts with the homotopy method, we would like to stress that even though homotopy methods did not work out-of-the-box for solving EDI-CR instances, we believe there is a lot of potential to the general idea of gradually transitioning from an easy to a hard problem, especially for EDI-CR where loose coupling usually implies that a solution obtained assuming total independence may be somewhat close to that of the original problem that takes all interactions into consideration.

We mentioned that one easy problem to start with is to ignore all interactions and pretened the agents' problems are completely separate. But other possibilities exist. We can ignore *some* interactions while taking others into considerations. The choice of which interactions to initially ignore is itself an interesting research question. We can also change the fashion in which we transition from the easy to the hard problem. We discuss this and other possible directions for future work in Section 7.2.

## 3.5 Summary

This chapter presented ways in which EDI-CR instances can be mathematically formulated so that we can leverage available optimization packages. The first formulation we give is as a MILP, both for 2-agent and more than 2 agent cases. We base our formulation on the insight that most action histories of a group of agents have the same effect on a given agent, thereby allowing us to treat these histories similarly and use fewer variables in the formulation. We compare our formulation to 2 others: a nonlinear program and a formulation devised for tightly coupled DEC-MDP. The first is expensive to solve and can result in sub-optimal solutions. The second has the problem of resulting in programs of prohibitive size.

Experiments show that our MILP is more compact and leads to faster solution times and generally better solutions than formulations ignoring the structure of interactions. Our formulation therefore allows us to solve larger problems which would otherwise be intractable.

This chapter also presented our formulation of finding the optimal policy as a system of non-linear equations. We discussed the use of homotopy methods as an alternative optimization technique and the challenges we faced in doing so.

# CHAPTER 4

# COMMUNICATION IN COOPERATIVE EDI-CR

In this chapter, we return to settings with cooperative agents and discuss the important issue of communication in problems with structured interactions. As we tackle more complex problems requiring tighter coordination, we cannot ignore the possibility of, and oftentimes need for, communication among the decision makers. Even when agents know each other's initial policies, uncertainty about action outcomes can create ambiguity about what states the other agents are in and, consequently, what they will do in the future. This ambiguity introduces the need for coordination during the execution of a distributed policy. Communication raises a host of interesting and challenging problems which we believe can particularly benefit from consideration in the structured interaction setting.

In Chapter 3, we discussed how the problem of finding optimal policies for a set of cooperative agents can be formulated and solved as an optimization problem. The same solution approaches can be used when the problem involves communication, as well as domain, actions. But the main impediment to reasoning about communication offline is the very large problems sizes that result from following each domain action with a communication action. Even if such problems can be represented, they are usually too large to solve. Our approach to making offline planning for communication tractable [60] is to construct a smaller problem whose optimal action and communication policies are the same as, or close to, those of the original problem.

We start this chapter with a survey of work on communication in decision theoretic models. In Section 4.2, we discuss the different kinds of communication costs and provide an analysis that highlights the potential gains of restricting the number of points where

agents can communicate. Section 4.3 presents our heuristics for limiting the amount of communication in situations with uni-directional transition interactions. Experimental results are given in Section 4.4. We extend our work to settings with bi-directional interactions in Section 4.5, using Bayesian networks to calculate belief estimates.

In spite of several attempts to get around the complexity of offline reasoning, ours is the first work to focus on making it more tractable by restricting the problem size in a way that has little or no effect on solution quality, thereby making it possible to reason about long-term consequences of communication without incurring the prohibitive costs typically associated with doing so.

## 4.1 Related work: Communication in Decision Theoretic Models

In this section we compare and discuss some of the work that has been done on communication in decision theoretic models. We try to highlight the similarities and differences among the different works in a way that makes it easier to understand how they stand in relation to each other. The following are the criteria we use for comparison:

- **Reasoning time:** reasoning about communication can be done offline or online.
- **Observability:** models vary in what they assume an agent can observe. Observability ranges from joint partial observability where combining all agents' partial views does not tell the agents the global state, to locally observable states.
- **Independence assumptions:** transition independence (T.I.) and/or observation independence (O.I.).
- **Communication language**: at one end of the spectrum, the agents can be verbose and exchange their entire states (or their observation histories in case of partial observability) after each action. Where the interaction among agents is limited, a summarization of an agent's state often goes a long way. For example, in TI-DEC-MDP,

a fair degree of coordination can be obtained by exchanging a single bit of information about each shared task indicating whether it was done or not [14]. Another language can be the language of probabilities of doing actions that affect the other agent. For example, an agent can inform another of the probability of doing a certain critical action in the future, with drastic changes in this probability triggering communication [96]. Generally, an agent can communicate any information that helps the recipients refine their beliefs over what the sender will do/has done.

- **Initial centralized policy:** can make different assumptions about whether and how much communication will be available during execution.

- **Calculating the value of communication:** refers to how an agent assesses the worth of a communication action. One approach considers the effect of communication myopically, i.e., considering only the immediate effects of communication without regard to the long-term effects. Another approach considers the expected value of the state resulting from communication. A different class of approaches does not calculate a value for a communication action. Rather, communication is triggered when a certain condition is met, e.g., when the actions taken by the team of agents would change if they communicate.

- **Digesting messages:** different approaches react to receiving a message differently. Some approaches make the agents re-plan in the light of the newly-received information. Others use received information to refine beliefs. If the communicated information is complete enough, the agents can purge the histories of observations they have seen so far and proceed from a known global state. If there is partial observability, the agents can still purge their histories, but they will start from a new belief over the global state. For approaches where communication happens if a certain condition is satisfied, one round of communication can trigger a second round and so on until the most recent communication results in a state that does not invite further communi-

cation. If communication was planned for offline, the recipient simply follows the path dictated by his state and the received information. In this case, the effect of the message is "pre-compiled" in the agent's plan.

Tables 4.1 and 4.2 compare some related work, as well as our suggested work, along the above criteria. The following paragraphs give more details.

Reasoning about communication at execution time has been approached in a variety of ways. Becker et. al [13, 11] assume at planning time that communication is not possible and obtain a 0-communication policy. During execution, their agents myopically calculate the value of synchronizing their states based on 1-step problem dynamics. Because the approach is online, therefore restricted to reachable states, and the model is transition-independent, the calculation is fast and simple. However, the communication decisions in this case are only optimal assuming further communication is not possible, which sometimes results in over-communication. Improvements on the value-of-communication calculation which vary the degree of myopia of an agent, as well as take the other agent into consideration, are given in [11]. However, as the agent becomes less myopic, the computational cost of calculating the value of communication increases because the agent needs to look further ahead when deciding whether to communicate.

The work of Xuan [97] moves in the opposite direction, assuming free communication at planning time and deciding where it is possible to skip communication during execution by considering the utility of communication. Roth et. al [71] tackle Dec-POMDPs and also assume free communication at planning time, thereby obtaining a single-agent POMDP whose policy is then used as an approximate solution for the Dec-POMDP. During execution, an agent communicates its observation history if doing so benefits team performance. So the decision to communicate does not consider a cost for communication. Upon receiving an observation history, an agent prunes beliefs that disagree with the received histories, as a result of which it may find that communicating its *own* history is useful, thereby triggering a new round of communication.

Goldman and Zilberstein [38] introduce the Dec-POMDP-Com model that explicitly represents communication actions and messages.[1] The setting considered throughout that work is a *goal-oriented* one where agents are trying to meet in a grid. Two approaches are presented; one involves agents communicating upon reaching sub-goals, thereby setting new goals. The other involves agents act myopically optimizing the choice of when to send a message, assuming no additional communication is possible.

To get around the complexity of offline reasoning, Communicative JESP [65] requires agents to communicate at least every $K$ time steps, thereby only considering messages that encode observation histories up to length $K$. However, this restriction means agents communicate not to improve performance but to make policy computation tractable. They do, however, address the general Dec-POMDP case and make no independence assumptions.

The work of Spaan [80] also does offline reasoning and is close to ours in that both works treat communication actions as just another kind of actions and have messages appear to the recipient as observations. Their work, however, deals with DEC-POMDPs while our agents are assumed to observe their local states. The major difference is that their work optimizes the communication policy and action policy iteratively with respect to each other, whereas we solve for the optimal policy (containing both domain and communication actions) as a whole. We suspect that Spaan's iterative mutual improvement of action and communication policies is an attempt to avoid the complexity of optimizing both policies simultaneously. As will be detailed in Sections4.3 and 4.5, we propose heuristics to reduce the size of the problem, thereby allowing us to do the desired simultaneous optimization.

Another work that reasons about communication offline is that of Beynier et. al [18] where communication is added to the Opportunity Cost DEC-MDP (OC-DEC-MDP) model that was originally proposed in [17]. OC-DEC-MDP handles temporal and precedence constraints on agent tasks with complexity polynomial in $|S|$. It has local full observability

[1] Shen et. al[77] proved that this explicit representation does not increase the expressiveness of the model; communication actions are just special types of actions and messages are special types of observations.

**Table 4.1.** Related work

| | Becker IAT'05 [13] | Nair et al. AAMAS'04 [66] | Spaan et al. AAMAS'06 [80] |
|---|---|---|---|
| Reasoning time | Online | Offline | Offline |
| Language | Local state | History of observations since last *sync* | No explicitly defined language |
| Independence assumptions | T.I. and O.I. | None | Domain T.I. and O.I. Messages modeled as observations so communication introduces dependence |
| Observability | Local full obs. | Joint partial obs. | Joint partial obs. |
| Centralized policy | Assumes no communication | Assume synchronizing is possible. One variant enforces a *sync* every $k$ time units | Mutual optimization of action and communication policies w.r.t. each other until convergence |
| Solution Method | Coverage Set Algorithm (CSA) for centralized policy | JESP-like iterations | JESP-like iterations to optimize domain policy w.r.t communication policy. Heuristics for calculating communication policy. |
| VoC | Myopically calculated online based on expected increase in utility after doing 1 *sync* given belief over other agent's state | Value of a *sync* calculated offline as an expected value over the belief states that can result from the *sync* | Information content of a message measured by entropy over the external uncontrolled state $s_0$ |
| Digesting messages | Re-run CSA from new global state | Discard observation history and adopt new belief state. Further reactions are compiled into policy | Update belief over $s_0$ |

and observation independence. The task graph (where tasks are nodes and arcs are predecessor relations) is acyclic. Each task has a time window and a reward. The time and resources taken by a task's execution are probabilistic. Each agent $i$ individually constructs its MDP. The agents start from some initial set of policies from which they calculate the

**Table 4.2.** Related work (cont.)

| | Roth et al. AAMAS'05 [71] | Goldman et al. AAMAS'03 [37] | This work |
|---|---|---|---|
| Reasoning time | Online | Online | Offline |
| Language | History of observations | Local state | Signaling action completion |
| Independence assumptions | None | T.I. and O.I. (goal oriented) | Structured transition & reward dependence |
| Observability | Joint partial obs. | Joint full obs. but local state not known | Local full obs. |
| Centralized policy | Assumes free communication which transforms problem to 1 large POMDP | Assumes no communication | Assumes communication is possible (with restrictions) but does not enforce it |
| Solution Method | Q-POMDP heuristic approximates best joint action for the large POMDP. It calculates distribution over joint beliefs. Choose action with the best expected value. | Greedy myopic approach. For each possible distance between 2 robots, there is a communication policy | Reason about communication offline. Formulate problem as mathematical program and solve for communication and action policy |
| VoC | Myopic online calculation. Communication if sharing local observation history would results in a different joint action. | Greedy myopic online calculation | Calculated at planning time given impact on self and other agent |
| Digesting messages | Prune possible joint beliefs that disagree with received histories. Can trigger further communication | Set global state, purge observation history, set new goal | Reaction to a message already compiled into policy |

opportunity cost (OC) of their actions, given the policies of other agents. From these OCs, each agent then calculates a policy that considers both its local rewards and the effects of its actions on the team's rewards (as mirrored by the OCs). The agents then have a new set of policies which they can use for the next iteration. This whole process rests on the assumption that the task graph of each agent is linear (the task execution order is fixed), otherwise the effects of changing the starting time of a task would not propagate properly. Also, this assumption allows the state to only store the last executed task rather than all tasks and their outcomes.

The OC-DEC-MDP model is yet another sub-class of general DEC-MPDs that overlaps with ours. Both models have local full observability and transition dependence. However, whereas OC-DEC-MDP models only a specific kind of transition dependence (precedence constraints), we model arbitrary (and more general) transition dependence. We also have reward interactions, which are not in OC-DEC-MDP while the latter has resource constraints, which are not in our model.

Communicative OC-DEC-MDP allows an agent to communicate the end time of a previous successfully executed task. So both our communication models and that of Beynier use the same communication language. The latter, however, can deal with non-instantaneous communication with probabilistic cost. Planning for communication is done offline where the benefit of a communication action (increasing recipient's expected utility) is weighed against its cost (time delay and resources consumed). However, there is still a decoupling between agents' MDPs because the effect of communication on the other agent is calculated only from the sender's perspective given the recipient's policy during the last iteration, again aided by the fairly simple task graph.

The above work uses a kind of decoupling where each agent solves his MDP individually given the last known policy of other agents. This decoupling greatly facilitates the process of coordination. A different kind of decoupling is seen in the work of Witwicki et al. [92] where they decouple agents' execution by first deciding on a set of commitments

and then having each agent plan for itself assuming these commitments will be respected. The first phase, searching the commitment space, is done using a heuristic algorithm while the second phase is done using linear programming to force the committing agent to respect the commitments. This work therefore coordinates agents using up-front commitments rather than communication, which has the advantage of reducing the size of the policy space. The downside is that if a commitment fails to be met, there is no way to alert the dependent agent. An extension of this work iteratively increases the number of parameters associated with a commitment, thus allowing agents to model each other more accurately at the expense of a larger problem size [93]. However, this is only done in the context of a sub-class of Becker's Event-Driven Interaction DEC-MDPs where the agent interaction graph is acyclic and, other than time and commitment-related flags, there are no commonly observed state features.

Another work that uses commitments for coordination is that of Xuan et. al [96]. They identify sources of uncertainty inherent in commitments and discuss ways to incorporate them into the modeling of commitments, as well as mechanisms to handle these uncertainties, such as contingency analysis. The search for what commitments each agent should make is done through a negotiation process. This work has the advantage that it does not assume that commitments are fixed; it admits the possibility of commitment changes (e.g. failure).

Whereas decoupling, whether of the first or second kind shown above, makes problem solving easier, it is prone to converging to a local optimum and failing to find the global optimum that would be achieved if both sub-problems (in the first case, problems of both agents, and in the second, finding commitments and policies) were solved simultaneously.

Another difference in modeling communication relates to whether it is modeled implicitly or explicitly. In the former, one agent's actions affect observations seen by another agent. In explicit modeling, communication is a first-class action and a message vocabulary is part of the model. It has been shown for least one model that adding explicit communi-

cation does not increase the expressiveness of a model; Shen et. al show that DEC-MDP-Com as introduced in [38] is equivalent to DEC-MDP [77]. However, it is sometimes done for the sake of clarity. Another reason for explicitly modeling communication is that it makes it easier to delineate communication-related issues and parameters and add more sophistication to the model (e.g. complex communication cost, communication delay, noisy communication channels and dropped messages [81]).

## 4.2 Communication Costs

In this section, we discuss the different kinds of communication costs, with a focus on the computational cost introduced when we reason about communication. We then analyze the effect of limiting communication on problem size.

The first, and more obvious, kind of communication costs are *operational costs* due to things like transmission cost, cost in reduced battery life in the case of sensors transmitting data and privacy loss in adversarial settings or where the communication channel is not secured, to name a few examples. A modeler would typically assign such costs a quantitative value that is commensurate with whatever unit is used for rewards, thus allowing an agent to conduct a cost-benefit analysis of the effect of communicating this or that piece of communication on the team's overall reward. This is the kind of cost that has garnered the most attention in the literature (e.g., [13, 11]).

The second kind of cost is the *computational cost* associated with reasoning about communication. This cost is in terms of the increase in problem size (and the computational effort of solving it) and the size of the resulting policies. The action space increases with the introduction of communication actions while the state space increases because a state typically needs to store message (in explicit communication) or observation history (in implicit communication). The main impetus behind our work is that even if communication in a particular domain were entirely free, it would still be necessary to limit communication in order to keep the problem size tractable.

The amount of increase depends on *what* is communicated and *when*. Obviously, the more frequent the communication, the longer the message/observation histories will be. What is communicated affects the number of possible histories. A richer communication language (one with a larger vocabulary) results in a larger number of possible histories. For example, when there is local full observability, the richest language is that of local states where agents synchronize their local states, thus disclosing the global state and achieving maximum coordination. This language is very expensive because in the worse case, the number of histories of length $T$ is $O(|S|^T)$. An example at the other extreme is a very limited language with Boolean vocabulary that just tells the recipient whether a certain action was done or not.

In our work, we address both operational and computational costs. We use heuristics to limit the availability of communication, thus reducing problem size and computational cost without greatly harming solution quality. We then calculate the optimal action and communication policy of the resulting problem so that operational cost of communication is minimized while maximizing team reward.

### 4.2.1 Limiting communication

Like offline reasoning about domain actions, offline reasoning about communication attempts to consider all the ramifications of all actions/communications before execution starts. The result is a policy dictating what, if anything, should be communicated in each situation. However, reasoning about communication offline is notorious for being intractable. The number of messages agents can send can grow as large as the set of all possible observation histories (or set of states in locally fully observable domains), since every history/state may represent a different belief that an agent may wish to convey. Offline reasoning would require the enumeration of all possible messages, as well as their intended effect on the team belief.

To make offline reasoning about communication tractable, we limit the number of communication possibilities. A *communication possibility* after an action means the solution algorithm can choose whether to communicate after that action. It does not mean that communication will actually take place at that point, only that it is possible. Full-fledged communication considers all communication possibilities, and that is what makes it intractable. An important observation, however, is that very few of these possibilities are useful.

**Definition 11** *A* useful communication possibility *is one at which the optimal action is to actually communicate.*

If we can *find useful communication possibilities* in advance and only add these to our problem, we can get (near) optimal solutions with much smaller problem sizes and thereby solve problems that are intractable if all communication possibilities are considered.

### 4.2.2 Problem size analysis

In this section, we motivate the effort to limit the number of communication possibilities by analyzing the relation between the amount of communication available to agents and the problem size. This analysis is for a particular state representation (one that stores the sequence of actions taken and outcomes obtained, together with timestamped sent and received message histories), and measures the problem size in terms of a particular metric; the number of sequences in an agent's MDP.

In this analysis, and in our work on settings with uni-directional interactions in Section 4.3, we use a limited communication language that only allows an agent to signal the successful completion of an action, and we only allow this right after the action is finished. Even though a message does not tell the recipient exactly where the sender is in its state space, it does serve to refine the recipient's belief over the sender's state and, hopefully, allow him to better respond to execution-time eventualities.

We express problem size in terms of the number of actions available at every state, $A$, the number of outcomes per action, $O$, the number of messages an agent can send and receive in a given time unit, $M$ and $R$, respectively. Note that an agent actually sends/receives only one of these messages in a given time unit. For this analysis, we make the simplifying assumptions that these four quantities are the same for both agents (agent sub-problems are of equal sizes) and are the same across time units. We also assume an action takes a single time unit, so for horizon $T$ , an agent goes through $T$ act-send-receive iterations.

#### 4.2.2.1 No communication

If no communication is allowed anywhere (no communication possibilities), the number of sequences in an agent's MDP is $\sum_{k=1}^{T} A^k O^{k-1}$ which is $\mathcal{O}(A^{T+1}O^T)$.

#### 4.2.2.2 Full-fledged communication

With full-fledged communication, each of the $T$ stages consists of choosing an action, getting an outcome, choosing what message to send, and probabilistically receiving a message. At the beginning of the $k^{th}$ time unit, there are $A^{k-1}O^{k-1}M^{k-1}R^{k-1}$ states where any of $A$ actions can be taken, resulting in $A^kO^{k-1}M^{k-1}R^{k-1}$ sequences. Each of these actions has $O$ outcomes, leading to $A^kO^kM^{k-1}R^{k-1}$ states where an agent send any of the $M$ messages, giving $A^kO^kM^kR^{k-1}$ new sequences. Therefore the number of new sequences generated during the $k^{th}$ time unit is:

$$A^kO^{k-1}M^{k-1}R^{k-1} + A^kO^kM^kR^{k-1}$$

and the total number of sequences in an agent's MDP is

$$\sum_{k=1}^{T} A^kO^{k-1}M^{k-1}R^{k-1} + A^kO^kM^kR^{k-1} = \mathcal{O}(A^{T+1}O^{T+1}M^{T+1}R^T) \tag{4.1}$$

To get a feel for how large the problem is, we apply the above general formula to the Mars rovers domain. If $d$ is the number of sites that can be visited then $A = d$ and $O = 2$

for the fast and slow outcomes. $R = d+1$ because the other agent can signal the completion of any of $d$ sites or nothing. $M = 2$ because after a given action, the agent can signal it or send nothing. The total number of sequences therefore approximately (because of using $d$ instead of $(d+1)$ for simplification) reduces to:

$$\sum_{k=1}^{T} d^{2k-1}2^{2k-2} + d^{2k-1}2^{2k} = \mathcal{O}(d^{2T+1}2^{2T})$$

#### 4.2.2.3 Limited communication

We now see how restricting the amount of communication affects problem size. Specifically, we want to see the effect of the number of communication possibilities on the number of sequences in an agent's MDP. Of course, this depends on *where* the possibilities are; the earlier they are in time, the larger their effect on the MDP size (because of earlier branching). As a worst-case analysis, we assume that if we have $C$ communication possibilities, they are located at the first $C$ levels of the MDP. Note that we do not have a possibility after *each* action in the first $C$ levels; we have a possibility after a *single* action at each level. For this reason, we cannot just take the general formula and apply it for the first $C$ time units then take the no-communication formula and apply it to the remaining time units. Also for the sake of a worst-case analysis, we assume all $C$ possibilities can be encountered; i.e., they are on the same path. To see why this is the worst-case, consider the other extreme where each possibility is on a different path. In this latter case, the number of possible message histories is only $C + 1$ whereas in the former case, there are $2^C$ histories, since each possibility can actually be used or not.

During each of the first $C$ stages, and for only one action at a stage, an agent sends one of $M$ messages or nothing. The number of sequences resulting from domain actions during the first $C$ stages is

$$\sum_{i=1}^{c-1} A^i O^{i-1} M R^i (c-i) + \sum_{i=1}^{c} A^i O^{i-1} M R^{i-1}$$

Each of these stages has exactly one occasion where the agent can choose which of $M$ messages to communicate, if any. So the number of sequences resulting from communication actions is $c(M+1)$. The number of sequence from domain actions in the remaining $T-c$ stages is $Q\sum_{i=1}^{T-c} A^i O^{i-1}$ where $Q$ is the number of states at the end of the first $C$ stages and is $\mathcal{O}(A^c O^c R^c + A^{c-1}O^{c-1}MR^c)$. The second factor in the product is $\mathcal{O}(A^{T-c+1}O^{T-c})$. The total number of sequences, from domain and communication actions, is therefore of the order of

$$\begin{aligned}&\sum_{i=1}^{c-1} A^i O^{i-1} M R^i (c-i) + \sum_{i=1}^{c} A^i O^{i-1} M R^{i-1} + c(M+1) \\ &+ (A^c O^c R^c + A^{c-1}O^{c-1}MR^c)(A^{T-c+1}O^{T-c})\end{aligned} \tag{4.2}$$

which reduces to $\mathcal{O}(A^{T+1}O^T R^c + A^T O^{T-1} M R^c)$.

The reduction in problem size from full to limited communication is therefore considerable. For the Mars rover example, full communication gives MDPs whose size is $\mathcal{O}(d^{2T+1}2^{2T})$ whereas $C$ communication possibilities give $\mathcal{O}(d^{T+c+1}2^T)$.

## 4.3 Heuristics For Uni-directional Interactions

In this section, we propose three heuristics, H1 through H3, for deciding where to add communication possibilities. The heuristics assume that transition dependence and communication are uni-directional; agent $j$ affects and can send messages to agent $i$. H1 and H2 rely on analyzing the static structure of the MDPs while H3 is more sophisticated and relies on an initial policy in addition to the static structure. In Section 4.5, we show how we extended and improved H2 and H3 for the case with bi-directional interactions.

**The general plan:** For H2 and H3, we start with a base no-communication pair of MDPs and set $C$, the number of communication possibilities we allow the heuristic to add. A heuristic considers the set of communication possibilities and assigns a score to each possibility based on its perceived usefulness (*impact*). This is done by first calculating a

no-communication policy then using it as a context for evaluating the impact of potential communication points. The top $C$ communication possibilities are then added to the base MDPs (in $MDP_j$ as communication actions and in $MDP_i$ as possible received messages) and the resulting pair is solved using the techniques in Chapter 3. If a heuristic makes good decisions regarding which points to add, a solution using $C + 1$ possibilities should not be worse than one using $C$ possibilities because all points in the latter are available to the former. We can then use $C$ to control the tradeoff between problem size and solution quality.

### 4.3.1 Heuristics based on static structure

#### 4.3.1.1 H1: Add after critical actions

The simplest heuristic is to add communication possibilities only after critical actions. The intuition is that these are the actions that affect $i$, so they are the ones $i$ cares about. The problem with H1, though, is that it can make communication available when it is too late for $i$ to benefit from it, resulting in $j$ never actually using the added possibilities and following a zero-communication policy.

#### 4.3.1.2 H2: Add after actions with very different outcomes

H2 tries to analyze $MDP_j$ to determine which actions have outcomes with very different effects on $j$'s probability of starting future critical actions early enough for $i$ to benefit from them. A communication possibility considered by H2 is characterized by a sequence of finished actions and their outcomes, *done*, together with a potential next action $a_{next}$. A possibility's usefulness score is proportional to the difference between the effects entailed by $a_{next}$'s outcomes. The intuition is that if an action's outcomes have very different consequences, the particular outcome obtained during execution will greatly influence $i$ and thus $i$ will need to know about it.

For an outcome $oc$ of $a_{next}$, the effect on future critical actions is obtained by inspecting the sub-tree in $MDP_j$ rooted at the state where $oc$ is obtained after the sequence *done*. The impact of $oc$ is given by

$$\sum_{a \in criticals} w(a) * \sum_{t \in start(a)} (T - t) * P(t)$$

where *criticals* is the set of $j$'s critical actions and *start(a)* is the set of possible times $j$ can start action $a$, *(T - t)* favors earlier start times and $P(t)$ is the probability that in the subtree, $j$ starts $a$ at $t$. $w(a)$ is a way of giving more weight to more lucrative critical actions. If $a$ is part of a shared task, it is proportional to the common reward and, if $a$ affects the other agent, it is proportional to the reward of doing the affected action. According to this formula, an outcome resulting in $j$ being more likely to start future critical actions earlier is going to be associated with a larger value, assuming that the earlier $j$ satisfies a dependency, the better. For a dependency where $j$'s action affects $i$ negatively, we can use $\frac{1}{T-t}$ to favor later start times.

### 4.3.2 Heuristic based on an initial policy

The problem with basing our scores solely on an analysis of the structure of the agents' MDPs is that we do not have any indication of which actions will actually be taken by the optimal policy, therefore the communication possibilities we introduce may actually never be encountered because they are in a part of the state space that is never visited by the optimal policy. We therefore need some kind of guide as to which parts of the space will be visited. One possible guide is the zero-communication optimal policy which consists of the domain actions that would be optimal if communication were not possible. Even though this is not necessarily the optimal policy when we do introduce communication, as we will see in the experimental results section, this policy serves as a good approximation and helps focus our attention on important parts of the space.

#### 4.3.2.1 H3: Add where it causes most belief change

H3 tries to assess $i$'s "surprise" after a given communication, with the intuition that the most useful communications are those that inform the recipient of something thought very unlikely. To do this, H3 reasons about $i$'s beliefs before and after a potential communication. To calculate the former beliefs, we need some kind of initial policy, since beliefs induced solely from the agents MDPs are very loose. We therefore solve the (smaller) no-communication problem to get a pair of initial policies. The score of communication possibility $k$ is proportional to the difference between $i$'s beliefs derived from $j$'s initial policy $\pi_j$ and its beliefs if possibility $k$ is added and used. $i$'s beliefs are over the times $j$ can finish future critical actions. We use the KL-divergence as a measure of the difference in beliefs and express the score of a possibility as

$$\sum_{a \in important} KL(B(finish(a)|\pi_j) \;||\; B(finish(a)|\pi_j, k))$$

where $B$ is the probability distribution specifying $i$'s belief over $a$'s finish time. The more a communication possibility causes $i$'s beliefs to change from the base beliefs, the higher its score will be.

H3 does not consider all future actions because even if a communication drastically changes $i$'s belief over a given action $a$, this change has no consequences on $i$'s decisions if none of its future actions are affected by $a$. We therefore want to find out which of $j$'s critical actions affect $i$'s future. To reason about $i$'s future, we calculate a probability distribution over $i$'s state at the time of communication given $i$'s initial policy. For each possible state, we traverse all its descendants to determine which affected actions can be done in the future and thus which of $j$'s critical actions matter.

### 4.3.3 Evaluating communication points in context

Because H3 is concerned with measuring the belief change induced by a communication point, the top $C$ points should not be added at once because some points may provide

no new information given other points in the added set. Rather, the top point is added and subsequent points are evaluated *in the context* of previously added points. Note, however, that we are *not* solving the problem for each added point. We merely assume that the added point is indeed used and calculate the various beliefs accordingly. For in-context evaluation, the scoring formula is modified to measure the departure of $B(finish(a)|\pi_j, k, e)$ from $B(finish(a)|\pi_j, e)$ where $e$ is the set of previously added points. We may therefore have two points that individually cause a large departure from the belief induced by $\pi_j$, but one of them may become completely useless given the other if it conveys no new information about what $j$ will do in the future. By iteratively adding points, H3 avoids adding such useless points.

Taking a closer look at the set of already-added points $e$, we see that we only need to include in $e$ points that are earlier in time than point $k$, since at the time of getting the message associated with $k$, the receiver will not have received messages from later points.

Points in $e$ can either be on the path leading to $k$ or not. The points on the path are referred to as $e^+$ and, by our assumption that earlier points are indeed used, should be set to True while evaluating $k$ to reflect the assumption that the receiver already got messages from these points when it gets a message from $k$. Points that are not on the path to $k$ could not have been encountered by the sender before getting to $k$, and so the messages there could not have been received. The receiver, however, does get some information merely from *not* getting these messages that it knows would have been sent if the sender had encountered the associated points. We refer to these points as $e^-$ and set them to False in evaluating $k$. The impact of $k$ in the context of the set of points $e$ is therefore

$$\sum_{a \in important} KL(B(finish(a)|\pi_j, e^+{=}T, e^-{=}F) \;||\; B(finish(a)|\pi_j, e^+{=}T, e^-{=}F, k))$$

Clearly, if there is a point in $e^+$ that imparts the same information as $k$, $k$ will have no impact. Similarly, if the receiver knows that the sender will encounter either a certain

point in $e^-$ or $k$, and there are no other possibilities, then again $k$ has no impact because the sender already learned what information it can from not getting messages from $e^-$.

### 4.3.4 Automatically determining needed communication

As mentioned earlier, H2 and H3 accept a parameter that specifies the number of communication possibilities they should add. A different alternative is to allow these points to determine how many possibilities they need to add.

We would like to determine the number of communication possibilities needed to achieve optimal reward (what full communication would get) *a priori* rather than in retrospect after actually solving the problem. We believe that the scores calculated by H3 can guide us in this process. These scores measure the belief change induced by adding one more communication possibility in the context of previous ones. The hypothesis is that if, after adding some possibilities, the remaining possibilities all have low scores, then adding more possibilities will not increase reward because they are not actually going to be used, since the solution algorithm will not choose a communication that does not very much affect beliefs. Therefore, if a heuristic sees a significant drop in impacts of potential points in the context of $n$ previously added points, it can conclude that only $n$ communication points are needed for this instance.

We believe the number of needed points is a measure of how tightly coupled the different sub-problems in a problem instance are. It says how much coordination is needed among agents to achieve (near) optimal reward. Equally importantly, it tells us how large (and thus usually how difficult) the instance will be when we reason about communication offline; since the state keeps track of sent and received messages, the more frequently agents communicate, the larger the state space.

If a heuristic can indeed determine how much communication is needed to achieve (near) optimal reward, we can use this as a measure of problem difficulty that not only depends on the static structure of the problem (like the measure in [6]), but also takes

into consideration what actions the agents will actually take. The more we refine the way our heuristics calculate the impact of a potential communication, the more accurate our measure will be. Given the large variation in difficulty among instances of the same model, a measure of difficulty is an interesting area for future research.

## 4.4 Experimental Results: Uni-directional Heuristics

In this section, we compare the performance of our three heuristics in choosing useful communication possibilities. For H2 and H3, we see the effect on allowing a larger problem size on solution quality.

### 4.4.1 Experimental setup

Using the Mars rovers scenario introduced in Section 2.2.1.3, we conducted experiments to investigate how much our heuristics can reduce the size of the problem (given by the total number of sequences in the agents' MDPs), and the effect of this reduction on solution quality (given by the sum of the agents' rewards). The main parameters that affect an instance's size are the number of sites available to each rover, the time horizon and the number of dependencies. One notable observation is the difficulty of obtaining random scenarios whose optimal policies actually contain communication. Apparently, purely randomly generated scenarios are not tightly-coupled enough to warrant communication; knowing each other's initial policies often goes a long way in coordinating the agents.

We ran experiments on 21 instances whose composition is shown in Table 4.3. The "small" group contains 12 instances for which full communication produced a problem small enough to be solved. The "big" group contains 9 instances for which this is not possible. The "Average full size" column gives the average size (in terms of the number of terminal sequences) when full communication is allowed. All instances have uni-directional dependence and communication from rover $j$ to rover $i$. The instances we address are considerably larger than those attempted by Becker et. al using the EDI-DEC-MDP model

**Table 4.3.** Instance composition

| # instances | $i$ sites | $j$ sites | Time | # deps | Average full size |
|---|---|---|---|---|---|
| 12 | 3-9 | 3-11 | 7-10 | 2-6 | 3422 |
| 9 | 10-21 | 10-21 | 8-12 | 7-19 | 42955 |

where there are 5 sites per agents and the number of dependencies ranges between 2 and 4 [12]. Our instances are also much larger because our agents can visit the sites in any order whereas Becker has a fixed order for visiting sites.

### 4.4.2 Performance of heuristics

For the two heuristics H2 and H3 where we can control the size of the MDPs (by setting the number of communication points $c$), we investigate the effect of increasing the size on solution quality. Figure 4.1 shows how H2 and H3 behave on the small and big instances. The x-axis gives problem size as a fraction of the size obtained when full communication is allowed. We move along this axis by increasing the value of $c$. The y-axis gives solution quality as a fraction of the maximum quality we could obtain; for small problems, this is the quality with full communication but for larger problems, this is the best quality achieved by H2 or H3. Only 3 of the big instances could be solved using H1, so the figure only shows the average size and reward obtained by H1 on the small instances (which is a single data point, since H1 is not parameterizable by $c$). The curves were obtained by averaging the curves of individual instances after doing linear interpolation between an instance's points. We extrapolated an instance's curve towards the lower left corner using data obtained from solving its no-communication version (which, by definition, has minimum size and rewards) and towards the top right corner of the graph using the point $(1, 1)$ (which, by definition, holds for all instances, regardless of whether they could actually be solved using full communication).

As the figure clearly shows, as we increase the number of communication points, H3's reward rises more sharply than H2 and H1 does much worse than H2 and H3 on small instances and results in instances too large to solve on the larger instances. To see why,

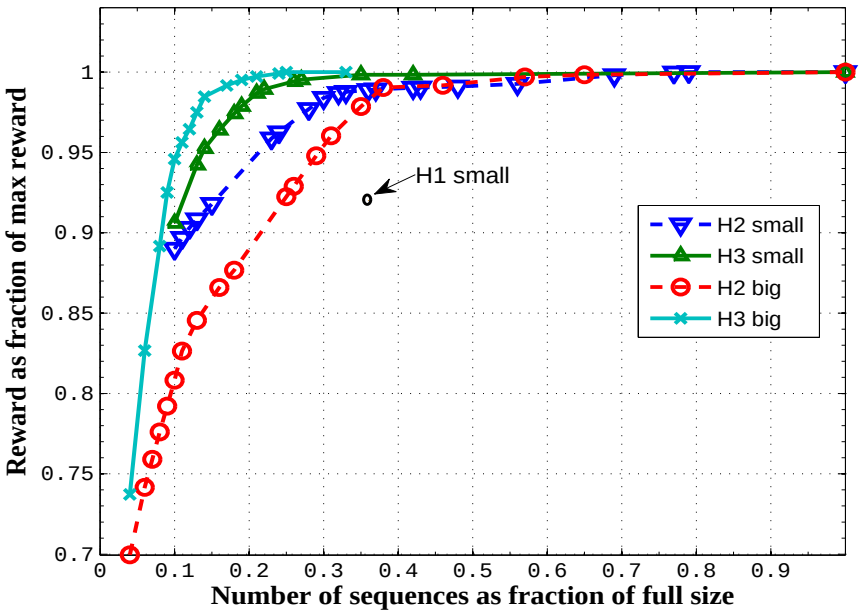



**Figure 4.1.** Effect of problem size on solution quality

remember that H3 adds points much more judiciously than H2; it 1) relies on an initial policy 2) assesses the belief change caused by a message and 3) considers $i$'s policy from the communication point onward. Besides failing to do these things, H2's problem is that even if an action's outcomes have very similar effects, it may still be important for $i$ to know about it if the action itself is very unexpected, something that H2 does not account for. Also, H2 makes a single decision for adding possibilities after *all* outcomes of an action, but in fact, the usefulness of communicating the different outcomes may vary, so deciding whether to add a communication possibility after each outcome should be done separately. H1 adds communication possibilities after all critical actions, so it adds many useless possibilities, but still fails to add possibilities that are early enough in time to be useful.

The net result is that H3 adds useful communication points earlier while H2 may add several useless points, the penalty of which is an increase in size with little or no increase in reward. The figure shows that the difference between H2 and H3 is more pronounced in

larger instances. The reason is that the increase in size itself (the penalty) depends on the how big an MDP was before adding a point.

As for the number of communication possibilities added, H3 needed to add 1-2 possibilities to the small instances to obtain optimal reward (which we know, since these instances could be solved with full communication). To achieve the same reward, H2 needed to add an average of 3 points for instances that need 1 point and 6 for those that need 2. For the large instances, H3 could add 3-4 points before the instances got too large to solve.

### 4.4.3 Effect on execution time

To demonstrate that H3's limited problem sizes do translate to solution time speedups, we averaged the time taken to solve the small instances with full communication compared to using H3 (and achieving the same optimal reward). On average, H3 took only 19% of the time taken by full communication. This means that even though H3 incurs an overhead due to building and solving the initial no-communication pair of MDPs, the overall solution time is dramatically decreased. For larger instances, we cannot calculate the speedup because a full communication solution is not possible.

In all our experiments, the time of determining the communication possibilities that should be added is insignificant compared to the times of building and solving the no-communication MDPs (in H3) and the final MDPs (in all heuristics). For the small instances, H3 took 43 msec on average to choose the possibilities to add. For the larger instances, H3 took on average 188 msec. H2 takes even less time, since its scoring calculations are less sophisticated.

### 4.4.4 Automatically determining needed communication

We conducted experiments to investigate whether we can indeed use the scores calculated by H3 to determine the number of communication points needed by the (near) optimal policy *before* actually solving the problem. We use the same test cases as in the previous sections to get the following preliminary results.

Figures 4.2 and 4.3 show H3 scores for the group of small and big instances. We divide the small instances into those that required 1 (left) and 2 (right) communication possibilities to achieve optimal reward (which we know because we could solve these instances with full communication). We divide large instances into those that become too large to solve after adding 3 (left) and 4 (right) possibilities. The y-axis shows H3 scores normalized by the maximum score obtained by any point in the context of $\pi_j$ only (i.e., when no points have been added yet), averaged over the relevant instances. The figure shows the top 2 scores obtained in each successive context; first in the context of just $\pi_j$ then in the context of $\pi_j$ and all previously added.

As can be seen, scores decrease as we add more points, since the most informative points are added first. Also, for the small instances that require $n$ (=1 or 2) points to achieve optimal reward, we see a large drop in the top score after $n$ points are added, indicating that indeed adding further points is useless. For the larger instances, we do not know the optimal reward, so we cannot be sure what the necessary number of points is. Nonetheless, the figure does support the fact that the $3^{\text{rd}}$ and $4^{\text{th}}$ points proved useful for instances in the left and right parts of the figure, respectively. In the left part, the top score obtained in the context of the first 3 added points is low, but not much lower than the previous top score, suggesting that reward may further improve if we could add a $4^{\text{th}}$ point. In the right part, however, the drop in the last top score seems significant, suggesting that a fifth point will not be useful.

## 4.5 Heuristic For Bi-directional Interactions

So far, we have been dealing with uni-directional interactions. This section presents a heuristic, H3B, whose main idea is like H3 in subsection 4.3.2, but which deals with bi-directional interactions and communication in ways that will be detailed in the following sub-sections. In each iteration of H3B, each agent assesses its potential communication

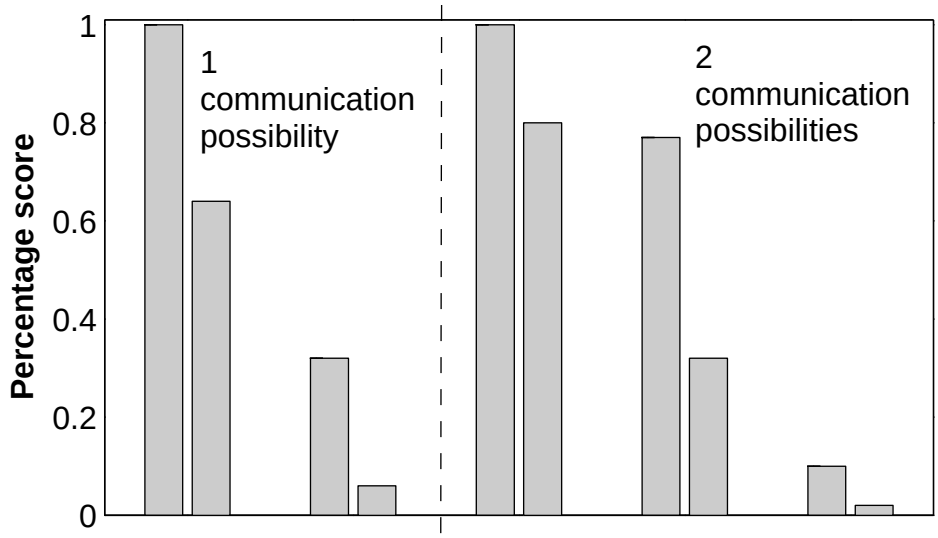



**Figure 4.2.** H3 Scores of successively added communication possibilities in small instances

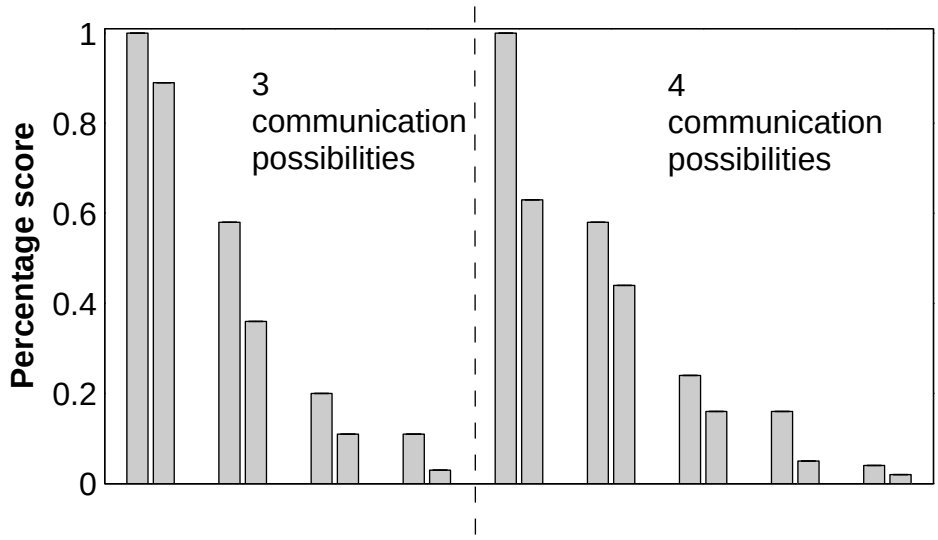



**Figure 4.3.** H3 Scores of successively added communication possibilities in large instances

points and the one with the highest impact among all agents' points is added to the problem. The process repeats until the desired number of points are added.

Another main difference between H3 and H3B is that the latter considers reward when deciding whether to include a communication possibility.

### 4.5.1 Incorporating reward

The heuristic H3 in subsection 4.3.2 measures the impact of a communication possibility as the amount of change it causes in the recipient's beliefs. Suppose agent $i$'s action $a_1$ affects the transition probability of agent $j$'s $a_2$, and that $i$ is using H3B to assess a communication possibility $k$. In general, $i$ may be interested in the change of receiver's ($j$'s) belief over any of the following:

1. Whether/when the sender will do its affecting action, i.e. $i$'s probability of doing $a_1$
2. The receiver's transition probability, i.e. $j$'s probability of transitioning to possible next states after doing $a_2$
3. The receiver's future reward, i.e. $j$'s expected future reward from the point of receiving $i$'s message onward

If we choose the first option, it may be the case that even though the message associated with $k$ implies that $i$'s probability of doing $a_1$ changed from 0.9 to 0.1, $a_1$ does not significantly change the transition probability of $a_2$, so the large change in $a_1$'s probability does not translate to a large change in $a_2$'s transitions, so agent $j$ will not find the message very useful. In other words, $j$ will take the same actions whether or not the message is received. As a result, measuring this kind of belief change does not give a good indication of the impact a communication point has.

If we choose the second option, again, even if the message implies the transition probability of $a_2$ does change a lot, it may be the case that the next states that $a_2$ can transition to (its outcomes) are more or less close in terms of the long-term consequences, so again $j$

will not find the message very useful because the change implied by the message does not really change its prospects.

We believe the third option, beliefs over future rewards, is indeed the kind of belief that we should reason about when assessing the impact of a communication possibility. An agent is ultimately only interested in maximizing reward, and as long as a change does not affect its expectations about its reward, it will not be important for the agent. Furthermore, we argue that if a message causes an agent's expectations about its future rewards to *increase*, then again, the agent will not find it useful in that it will not be motivated to act differently upon getting the message.

Based on the above intuitions, the bi-directional heuristic H3B calculates the impact of a message sent at communication possibility $k$ on recipient $j$ as

$$\max(E(r_j|\pi) - E(r_j|\pi, k), 0) \tag{4.3}$$

where $\pi$ is the initial zero-communication policy and $E(r_j)$ is $j$'s expected future reward. In the uni-directional case, belief depends on the policy of the sender only. Here, belief depends on the policies of both agents because interaction is bi-directional.

### 4.5.2 Using Bayesian networks to calculate beliefs

Now that we decided what beliefs a sender needs to calculate, we need to decide *how* to calculate them. Basically, if agent $i$ is assessing a communication possibility, $i$ wants to calculate $j$'s expected belief over $j$'s future reward. This calculation is complicated by the fact that $j$'s belief over its rewards (collected at the leaves at time $T$) depends on its belief over transitions at time $T-1$ which can be affected by the actions $i$ did prior to time $T-1$. The latter can similarly depend on $i$'s previous transitions which depend on $j$'s earlier actions. Clearly, bi-directional transition interactions make the calculation of belief very complicated.

One way around carrying these calculations ourselves is to formulate the calculation as inference in a Bayesian Network (BN). By including random variables representing reward in the BN, we can calculate the expectations in (4.3) as prior and posterior beliefs over these variables.

The idea of building a BN from a given pair of policies was used by Jiang et. al [46] who used the BN to formulated the problem of calculating the expected utility of a strategy profile as an inference problem. The resulting network is called the *induced* BN. In our case, the induced BN of a pair of policies $< \pi_i, \pi_j >$ contains four kinds of nodes (in the following, assume each action has $m$ probabilistic outcomes):

- **Action nodes**: For each agent $k$, there is an action node representing doing that action in a particular state. We do not, however, create nodes for actions done after the last critical action, because these are not affected by another agent, so their expected reward can just be folded into the reward nodes discussed below. For an agent $k$, the parents of a node $n_a^s$ associated with doing action $a$ at state $s$ are the previous action node and its associated outcome node (unless $s$ is $k$'s start state, in which case the node has no parents in the BN). Each action node $n_a^s$ has an associated binary random variable that says whether the action is actually done, which will only be true if outcomes of previous actions lead the agent to state $s$. The sum of probabilities of all the nodes associated with action $a$ being true is the probability that agent $k$ following policy $\pi_k$ takes action $a$ at some point.

- **Outcome nodes**: Each action node $n_a^s$ is the parent of an outcome node $n_o^s$ representing the probabilistic outcome of that action. The associated random variable has $m$ values. If action $a$ is affected by another agent, $n_o^s$ will also have as parents all nodes associated with the affecting action that are earlier than the current node.

- **Reward nodes**: To every leaf (action node-outcome node) pair in the BN, attach a reward node whose parents are these 2 nodes only. The random variable of the reward

node has $m + 1$ values. The first $m$ values are the the future rewards of the agent who owns the node for each of the $m$ possible outcomes of the last critical action. As mentioned above, given the outcome of the last critical action, these rewards are independent of the other agent. The last value is 0, reflecting the fact that if the states leading to this reward node are not reached, no reward should be collected.

- **Complex reward nodes**: for the $k^{th}$ reward interaction $< (s_{k_1}, a_{k_1}), ..(s_{k_n}, a_{k_n}), r_k >$ in $\rho$ in the EDI-CR definition of the problem, create a node representing the complex reward for this interaction. The parents of the node are all action nodes associated with actions $a_{k_1}..a_{k_n}$. The node's random variable has 2 values, $r_k$ and 0.

Note that a *single* BN is constructed for all agents, allowing the evidence provided by one agent (in the form of a message) to affect posteriors over nodes belonging to another agent. For 2 agents, the BN has 2 'root' nodes, one for the action taken by each agent at its start state. Figure 4.4 shows an example induced BN.

The way we construct our induced BN implies that the nodes have some kind of 'memory'. The advantage of this is that it precludes having action and reward nodes depend on every one of their ancestors, which would lead to very large conditional probability tables (CPTs). The downside is that we have many nodes in the BN. However, because the BN is for a given pair of policies, the number of nodes is still limited. In the experimental results subsection (4.5.4), we investigate the time taken to do inference on the induced BN.

Now we turn to the issue of setting the CPTs for the nodes in the BN.

- **Action nodes**: The CPT of an action node $n_a^s$ reflects the probability of taking action $a$ in state $s$ according to the policy. Because there is always an optimal joint policy that is deterministic, the CPT of an action node is deterministic. If $s$ is the start state, the node's random variable is always True. Otherwise, the variable is only True if the previous action's variable is true and its outcome leads to state $s$.

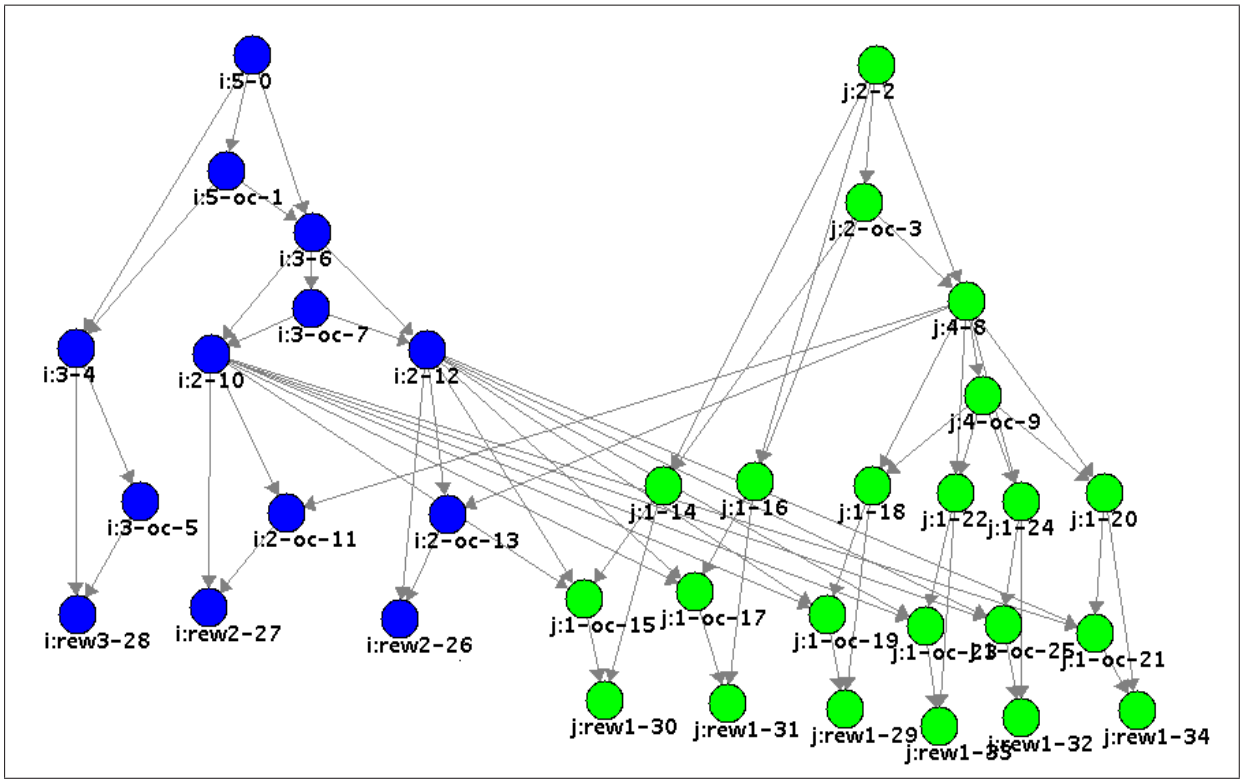



**Figure 4.4.** Induced Bayesian Network. Blue nodes belong to $i$ and green ones to $j$.

- **Outcome nodes**: The CPT of an outcome node depends on the agent's local transition function and whether the action in question is affected by a transition interaction.

- **Reward nodes**: The CPT of a reward node is deterministic. If the parents of the reward node are $n_a^s$ and $n_o^s$, the probability of the $v^{th}$ value of the associated random variable, where $v \leq m$, is 1 if $n_a^s = T$ and $n_o^s = v$. The probability of the last value (which is 0) is 1 if $n_a^s = F$.

- **Complex reward nodes**: Again, the CPT here is deterministic. For the node representing the $k^{th}$ reward interaction, the probability of the associated random variable taking on the value $r_k$ is 1 if the parent action nodes are all True. Otherwise, the variable is 0 with probability 1.

Once the induced BN is constructed, we can calculate the impact of a communication possibility. Suppose communication possibility $k$ of agent $i$ is associated with communicating after doing action $a$ in state $s$ and getting the $v^{th}$ outcome. The impact of $k$ is calculated by setting $n_a^s = T$ and $n_o^s = v$ and using an inference engine to calculate the posterior expected reward of $j$. For reward node $n_r$, $pact(n_r)$ is $n_r$'s parent action node and $poc(n_r)$ is its parent outcome node and $r[v]$ is its $v^{th}$ reward value. Agent $i$ calculates agent $j$'s posterior expected reward given initial policy $\pi$ and possibility $k$ as the sum of $j$'s individual rewards and a portion of the shared rewards from reward interactions.

$$\begin{aligned} E(r_j|\pi,k) = &\sum_{n_r} P(pact(n_r)|\pi,k) \sum_{v=1}^{m} P(poc(n_r)=v|pact(n_r),\pi,k) * r[v] \\ &+ \frac{1}{2} \sum_{n_{cr}} rew(n_{cr}) \prod_{pact(n_{cr})} P(pact(n_{cr})|\pi,k) \end{aligned} \tag{4.4}$$

For a complex reward node $n_{cr}$, $rew(n_{cr})$ is the extra reward/penalty associated with the node and $pact(n_{cr})$ iterates over the parent action nodes, i.e. the actions involved in the reward interaction represented by $n_{cr}$.

To calculate priors and posteriors, we used the inference engine in JavaBayes, a Bayesian Networks package developed by Fabio Cozman [31].

### 4.5.3 Evaluating communication points in context

As in the uni-directional case, a communication possibility may be rendered useless by the addition of another. In other words, communicating at the former gives the receiver no new information beyond what conveyed by communication at the latter. However, because in the bi-directional case both agents can send messages, calculating the impact of a communication possibility in the context of already-added ones is more complicated.

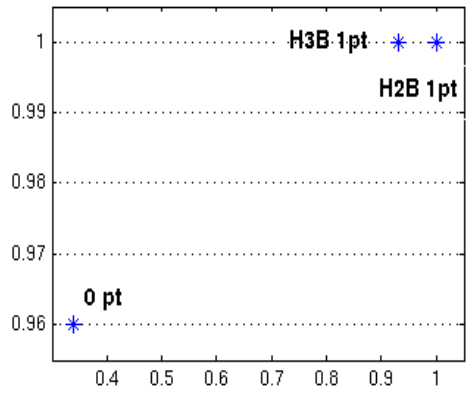
As before, a sender, for example agent $i$, needs to take into consideration the set of its own earlier points chosen so far $e = e^+ \cup e^-$. Additionally, it needs to consider communication points added so far that belong to the receiver $j$, again assuming that all previously added points are actually used. The rationale is that when $i$ is evaluating a point, $i$ is trying to measure how much $j$ already knows, and taking into consideration where $j$ can be in its state space (as indicated by $j$'s earlier points) helps $i$ refine its beliefs over $j$.

For the sender to evaluate communication point $k$ in the context of $e$ and the set of points belonging to the receiver $C$, the sender partitions $C$ into groups where points in each group are on the same path (i.e. can all be encountered in a given run). If the set of groups is $G$, we take the impact of a point $k$ to be its maximum impact in the context of any group $g \in G$:

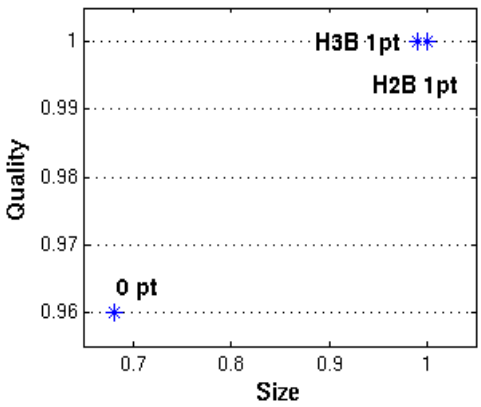
$$\max_{g \in G} E(r_j|\pi, e^+{=}T, e^-{=}F, g = T, C \setminus g = F) - E(r_j|\pi, e^+{=}T, e^-{=}F, g = T, C \setminus g = F, k)$$

### 4.5.4 Experimental results

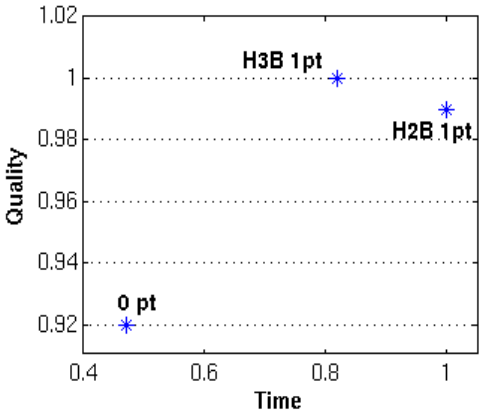
In this section, we give sample results of the heuristic for bi-directional interactions given in Section 4.5. We compare H3B to a bi-directional (and slightly improved) version of H2, which we call H2B. The improvement comes from performing the H2 scoring calcu-



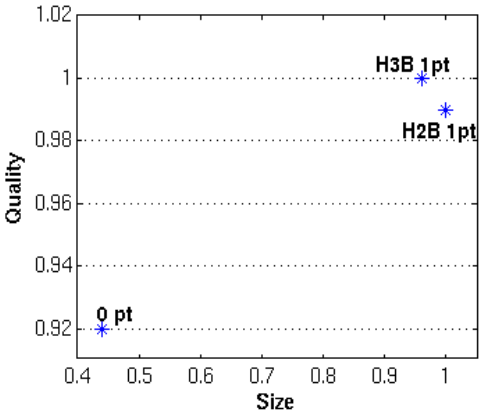

(a) Case 1



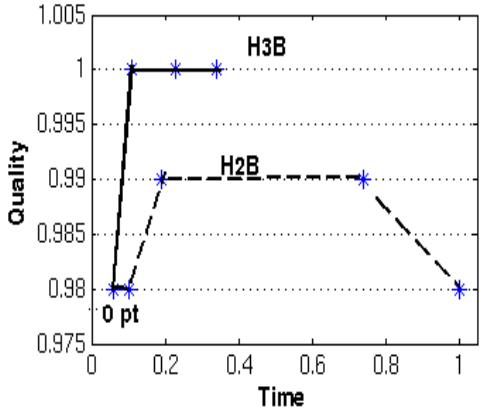

(b) Case 1



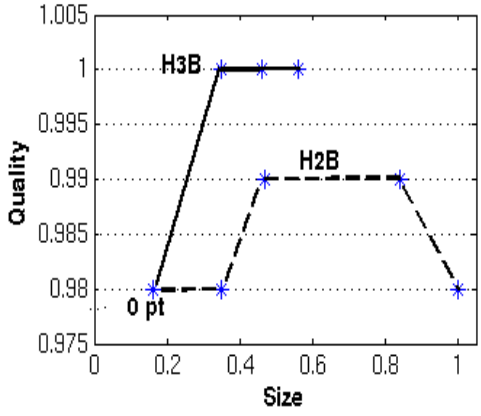

(c) Case 2




(d) Case 2

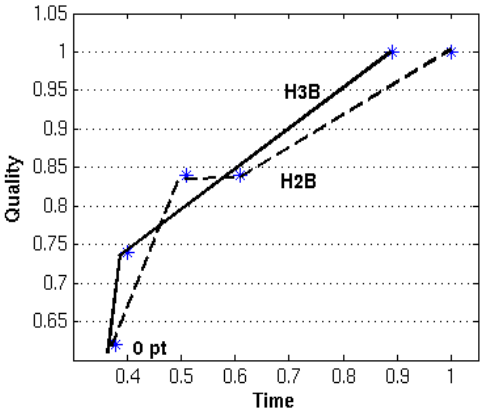



(e) Case 3

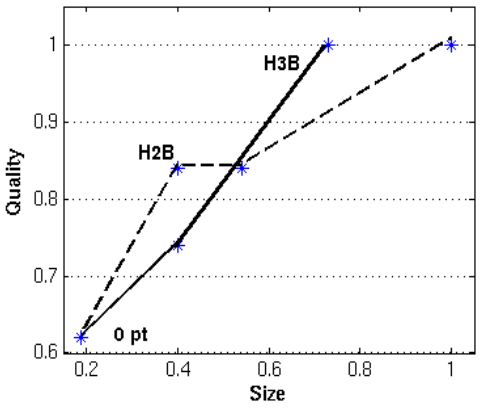



(f) Case 3

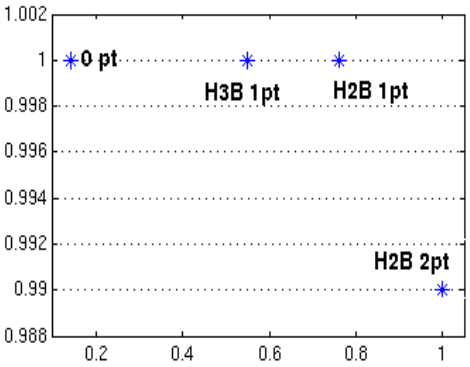
**Figure 4.5.** Time vs. Quality (left column) and Size vs. Quality (right column) of H3B vs. H2B



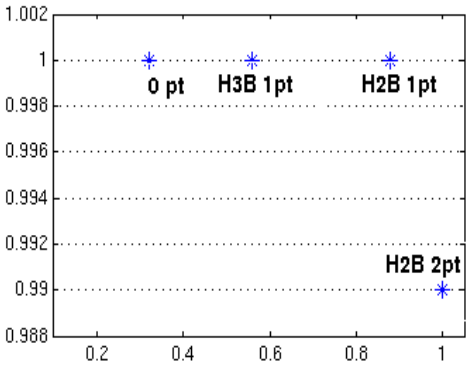

(a) Case 4



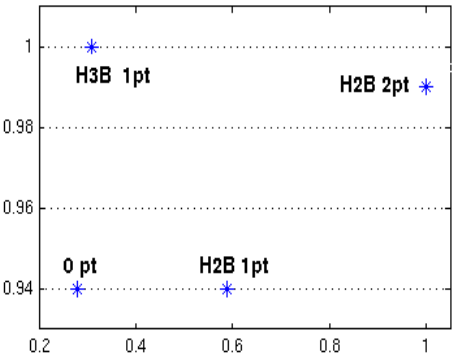

(b) Case 4



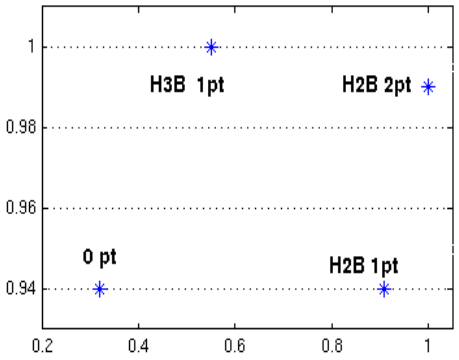

(c) Case 5




(d) Case 5




(e) Case 6




(f) Case 6

**Figure 4.6.** Time vs. Quality (left column) and Size vs. Quality (right column) of H3B vs. H2B (cont.)

lations in the context of the initial no-communication policy; the probability that a critical action starts at time $t$ is calculated based on the initial policy rather than just the transition function of the problem.

We use bi-directional scenarios from the Mars rovers domain (Section 2.2.1.3). We present and discuss results of H3B and H2B on 5 sample cases that illustrate the different ways these heuristics can behave. In our experiments, there are two issues that we want to verify:

- H3B is better than H2B at picking communication points to add to the problem. As a result, as H3B adds communication points, it achieves higher gains in solution quality than H2B.
- The time taken by the inference process in H3B to calculate beliefs does not outweight the benefits. H2B is a far simpler heuristic whose scoring calculations take almost 0 time. The question is whether the additional time taken by H3B is justified.

If the above hypotheses hold, H3B would result in steeper Time (or Size) vs. Quality curves than H2B.

In Figures 4.5 and 4.6, each row shows results from a given test case. Quality is the sum of the expected rewards of the agents under the optimal policy and Size is measured in terms of the number of compound variables added in the MILP formulation of an instance (Section 3.3). We feel that this measure may be more reflective of the size of the problem than the number of terminal histories that was used in our older results for the uni-directional case.

Each of the 3 quantities Time, Quality and Size is reported as a fraction of the maximum value attained by this quantity using any heuristic and number of communication points. It is important to note that the Time includes time taken by the heuristic to evaluate communication points, so for H3B, it includes the time to construct the Bayesian Network and run inference on it. Another thing to note about Time is that the values reported are the

first times at which the solver found the solution it terminated with, not the time it took to verify that the found solution is optimal. With bi-directional communication, problem size grows much more significantly as we add communication points, making it impossible to evaluate what full-fledged communication would result in and use that for normalization.

Successive points on the same curve are obtained by adding communication points, one at a time. For scenarios where only 1 point is added, we just show a single data point rather than a curve. In each figure, the Time, Quality and Size obtained without any communication is labeled '0 pts'.

In all the figures, the steeper the curve, the better; for the same increase in Time or Size, a heuristic with a steeper curve obtains more quality improvement than a heuristic with a flatter curve.

Policies for the instances we used in our experiments were calculated by formulating the problem as a Mixed Integer Linear Program (MILP) using the idea of binning as shown in Chapter 3 and solving the MILP using IBM ILOG Cplex [2].

### 4.5.5 Discussion

For cases 1 and 2 (first two rows in Figure 4.5), only 1 communication point was needed to get maximum quality. In both these cases, H3B decides to add only 1 point; all other points evaluated in the context of this point have no impact on the other agent's beliefs, which strongly indicates that only this point is actually needed.

In case 1, we allowed H2B to add another point and got no improved quality. This highlights that whereas H2B continues to give positive scores to successive communication points, H3B knows when to stop adding points. Both heuristics added a point each that resulted in maximum quality. Although they added different points, these points resulted in instances with very similar sizes that were solved in very similar times.

In case 2, H3B adds a point that results in a slightly higher rewards than H2B, as well as a smaller instance that is solved faster. Allowing H2B to add a second point (not shown in

the graph) still does not lead it to choose the same point that H3B added, so H2B still does not get the maximum quality. In fact, H2B does not get any quality improvement from this second point.

In case 3, H3B does not realize that additional communication points are not useful. It adds 3 points but obtains no improvement in quality beyond the first point. The question now is: When is H3B more likely to realize that no more communication points are added? In other words, when do additional points get zero score in the context of existing points? Looking closely at H3B's evaluations, we found that in cases 1 and 2, H3B added a point after state $\{0{=}1\}$ of the second agent, i.e., after it executes action 0 and gets outcome 1, the agent has the option to communicate. In case 3, however, H3B adds a communication point (call it $P_1$) after state $\{6{=}0,1{=}1,2{=}1\}$ of the second agent, i.e. after these 3 actions are done and these outcomes are obtained. So the first point added in case 3 is late in the game. When H3B is evaluating further points in the context of $P_1$, most of them are *earlier* than $P_1$ an therefore their impacts on the other agent's beliefs are the same with and without $P_1$. In contrast, in cases 1 and 2, most communication points are later than the first added point, and are therefore unlikely to contribute much beyond what was communicated at the first point. This results in these later points getting mostly zero scores and H3B deciding there is no benefit in adding further points.

H3B currently does not retract points once they are added, and the above observation does not suggest that H3B should retract $P_1$ and add an earlier point instead. One important fact to keep in mind is that making communication available early results in a larger increase in problem size than having a later communication point. Consider increasing the branching factor right after the root of the decision tree rather than right before the leaves: the latter will result in fewer states than the former.

The behavior of H2B in case 3 is also interesting. H2B starts by adding a communication point that is never used by the optimal policy, resulting in an increase in problem size with no corresponding increase in solution quality. The second point added by H2B

does increase quality, although not to the maximum. The third point causes no improvement beyond the second one. Adding a fourth point results in a much larger problem that the MILP solver can find no optimal policy for within 300 seconds (which is 15x the time required for H3B with 1 point). The reported quality is the quality obtained at that cutoff time, which is no better than the no-communication policy, even though one of the added points is actually used.

Case 3 therefore illustrates that in addition to resulting in larger problems that take longer to solve, choosing the wrong communication points to add can actually have a negative impact on quality if so many points are added that the solver fails to find a good policy for the resulting large problem.

In case 4, H2B chooses a better first point than H3B; the H3B point results in lower quality than the H2B point. The size of the resulting problems is about the same, although the time needed for the problem constructed by H3B is lower. The second point added by H3B, however, is much more useful. It achieves the maximum quality that H2B can only achieve after adding two more points. H3B decides no more points should be added. Indeed, allowing H2B to add a fourth point (not shown) does not improve quality.

In case 5, H3B decides to add a point, which is indeed used in the optimal policy, but its use causes an imperceptible improvement in reward. In this case, H3B decides that further communication points are useless. H2B adds a first point (different from the one added by H3B) which is never used. Allowing H2B to add another point results in a larger problem for which again, the solver does not find a good solution, even though the second added point is the same as the H3B point.

Case 5 illustrates a situation that we had to deal with frequently in running experiments: communication does not always make a significant improvement in quality. We found it rather hard to generate scenarios where communication makes a big difference, and where multiple communication points are needed. We believe this is partially an artifact of our random instance generator and the domain we use. It is possible that in a domain with

non-unit action durations, we can have scenarios where an action that does not get enabled takes longer to execute, possibly leaving insufficient time for remaining actions. In this case, it may be more urgent to communicate if the enabling action is not done. Another possible cause is that the time horizons we use are too short. Having bi-directional communication increases problem size, so we could not push the time horizons to the values of the uni-directional instances used in Section 4.4. Shorter horizons prevent us from having significant long-term effects for an un-enabled action. Shorter horizons also mean less uncertainty, again reducing the impact of communication.

To demonstrate that the particular rewards used make a big difference in the impact of communication, we took the scenario from case 5 (where communication made almost no difference in quality) and changed the rewards. We kept the problem structure intact, in terms of interactions and action pre-requisites. The resulting scenario is shown in case 6. Adding a communication point now improves reward from 94% to 100%. As in most previous cases, H3B decides not to add further points, and it takes H2B two communication points to get close to full reward.

## 4.6 Summary

In this chapter, we addressed the problem of the explosion of problem size when we try to reason about communication offline. We proposed the idea of restricting the points where agents have the option to communicate and analyzed the effect of this restriction on problem size, highlighting the significant reduction in size that can be obtained. We presented three heuristics for strategically choosing the set of points where communication is available when interaction among agents is uni-directional. Experimental results show that we can achieve a large fraction of the solution quality obtained from full-fledged communication at a small fraction of the computational cost.

For the bi-directional case, we presented an extention to the uni-directional heuristics that takes belief about reward into consideration and calculates belief estimates as inference

in a Bayesian network. Again, we compared this heuristic to a simple heuristic which does take the initial no-communication policy into consideration, but does not assess the impact of communication on the other agent's beliefs. As in the uni-directional case, our heuristics obtains the benefit of communication (in scenarios where communication matters), at a fraction of the problem size and computation time of the simpler heuristic.

We believe that our approach for limiting the computational cost of reasoning about communication offline is an important step towards the goal of being able to reason about domain and communication actions simultaneously. The particular scoring rules we gave in this chapter are for the EDI-CR model, but similar scoring rules that assess the impact of a communication point can be crafted for other models using their particular interaction structures. Applying the idea of introducing limited communication possibilities to improve coordination in a model with structured interactions (e.g. DPCL [87]) can be an interesting area of future work.

# CHAPTER 5

# COMMUNICATION AMONG SELFISH AGENTS

In this chapter, we present work that we did prior to the idea of structured interaction. This work still fits in the larger picture of the thesis in that like all the previous chapters, it is concerned with multi-agent sequential decision-making, but with self-interested agents.

We study the problem of multiple self-interested agents deciding whether to communicate information when doing so is necessary to accomplish a collective task, but incurs individual costs. As an example of such settings, we present the view maintenance problem with self-interested database managers (Section 5.1) . In this problem, database managers need to disclose information to keep a database view updated (and thus collect rewards), but they incur individual costs for disclosing information.

We give a brief background about games of incomplete information and their solutions in Section 5.2. We then formulate the view maintenance problem as a game of incomplete information in Section 5.3. Section 5.4 presents our general anytime algorithm for solving games of incomplete information. When used to solve games derived from instances of the view maintenance problem, the algorithm tells each selfish agent what to communicate, and when, in order to maximize its net reward (profit associated with problem solving, minus communication costs) with respect to the strategies of other agents.

Our algorithm has three novel features: it collapses the game tree as a pre-processing step, resulting in more tractable trees; it generates local measures that guide the search by indicating which parts of a strategy profile are least stable; and it proposes a global measure of the stability of a profile as a whole by calculating upper bounds on players' regrets when playing this profile. To do the search, our algorithm uses hill-climbing to

find strategy profiles with lower regrets and thus higher stability (an equilibrium profiles has zero regret). Section 5.5 gives experimental results on both random game trees and game trees derived from the view maintenance problem. We compare our algorithm to the Quantal Response Equilibria (QRE) algorithm [86] that is part of the software package Gambit [57]. Our algorithm can reduce the level of regret to 5% faster, and on a larger fraction of test cases, than QRE.

## 5.1 The View Maintenance Problem

A database view is a dynamic, virtual table composed of the result set of a query executed over one or more data sources. The view maintenance problem [20, 25, 28, 40, 54] concerns how a view is refreshed when its underlying data sources are updated. This problem has been studied in settings where view refreshing is expensive due to factors like the communication cost of transferring large amounts of data.

We study the view maintenance problem when *self-interested* database managers from different institutions are involved [63]. Ideally, whenever any of the underlying data sources is modified, the change will be reflected in the view. However, because the database managers operate on behalf of different self-interested institutions, privacy is a concern, so a database manager may not always be willing to disclose information about changes made it its database. However, some level of cooperation among the managers is needed to ensure the view is somewhat maintained. Each piece of information has an associated cost incurred by the manager disclosing it and a reward distributed equally among all managers. The reward depends on the disclosed information as well as previously disclosed information, creating a reward interaction among the different database managers. A database manager has to decide how much it contributes to refreshing the view, and consequently how much privacy loss it suffers. Because a manager's final payoff also depends on the actions of other managers, each manager needs to reason about the nature and number of

updates at other databases, what they can reveal in the future and the probability of their revealing it.

In our setting, the database managers (DBMs) disclose some information about their database updates in order to provide the view holder (VH) with a more up-to-date view. In return, VH gives the *coalition* of DBMs a reward that depends on how much information about their updates they disclosed and how much is still hidden. The reward is divided equally among the DBMs; VH does not care about the individual contributions of the DBMs.

The updates made to the base relations are processed in batches; the process of refreshing the view happens at intervals rather than continuously as updates are made. These intervals can be fixed in length or can depend on the number of updates made. At the end of an interval, the updates made since the last refresh make up the input to the refresh process. We assume that the VH gives the DBMs $T$ time steps to (partially) update the view. There are therefore $T$ decision points for each DBM.

An episode of the view maintenance problem starts with each manager having a set of changes (insertions, modifications and deletions) known only to itself. The managers are given a fixed number of stages where at each stage, each manager decides what kind of change to disclose, if anything.

## 5.2 Games Of Incomplete Information

In this section, we give some background material on a class of games called *games of incomplete information*. We also discuss a characterization of a strategy as a point in space and what an approximate equilibrium means.

### 5.2.1 Background

Games of incomplete information are used to model situations where each player has private information, his *type*, that affects his own payoffs but is unknown to the other play-

ers. However, the prior probability distribution over agents' types is common knowledge. Such game of *incomplete* information, where an agent is missing some information about one or more aspects of the other agents, is transformed to a game of *imperfect* information where the the agent knows some probability distribution over the missing information, but does not have perfect knowledge of what it is exactly [42]. This transformation is effected by adding random moves of Nature assigning a type for each player according to the prior distribution. The rules of the game may stipulate that certain actions by other players are not observable by this player. As a result, a player may not be able to distinguish among a set of nodes in the game tree if all these nodes have the same observable history from this player's perspective. An *information set* is a set of nodes (*members* of the information set) indistinguishable to a player. Consequently, a player makes its decision as a function of the information set, rather than the particular node, it is at.

In incomplete information games, the first $n$ levels of the game tree represent chance nodes where at level $i$, Nature assigns player $i$'s type with probability specified by the commonly known probability distribution over $i$'s type space. A *strategy* $\sigma$ for player $i$ is a complete plan covering all possible contingencies for every possible type. For each information set $h \in H_i$, a *behavior strategy* is $\sigma_i(h) \in \Delta(A_i(h))$ where $\Delta(A_i(h))$ is the set of all probability distributions over actions available at information set $h$.

A *Bayesian Nash* equilibrium (BNE) of a game with incomplete information $\Gamma$ corresponds to the Nash equilibrium of the normal form game derived from $\Gamma$. BNE is defined as a strategy profile and beliefs specified for each player about the types of the other players. Each player maximizes its expected payoff given its beliefs about the other players' types and the strategies they play. Note that in this solution concept, players do not update their beliefs about each other as the game progresses.

For sequential games, BNE suffers from the same problems in imperfect information settings as Nash equilibrium in perfect information settings. When using BNE or NE, the players may reach an unrealistic equilibrium that does not make sense. The reason is a

phenomenon known as *incredible threats* where a player $i$ tries to avoid a threat made by another player $j$ but the threat is implausible in that $j$ would not carry out the threat if it is playing rationally.

To remedy the incredible threats problem of NEs and BNEs, we need to ensure that players make a rational decision even at nodes off the equilibrium path. In other words, players should play rationally in every *subgame*; a part of the game tree that does not cut across any information set. In games of complete information, every node is the root of a subgame and this equilibrium refinement is called *subgame perfect equilibrium*. In games of incomplete information, however, a game generally has only one subgame, which is the game itself. *Perfect Bayesian equilibrium* is a refinement that specifies a belief-strategy pair that satisfies the following condition: the beliefs are consistent with the strategy and the strategy is rational given the beliefs. Equilibrium refinements are beyond the scope of this thesis.

### 5.2.2 A strategy as a point in multi-dimensional space

As mentioned earlier, at each $h \in H_i$, $\sigma$ specifies a probability distribution over actions available at information set $h$. It is therefore straightforward to think of a strategy profile as a point in multi-dimensional space. The dimensionality of the space is $\sum_{i=1}^{n} \sum_{h \in H_i} (|A_i(h)| - 1)$, where each dimension extends from 0 to 1. For each player $i$, and each of his information sets $h$, there is a dimension for each action available to $i$ at $h$, except the last action which is assigned the probability left over from the other actions. Because probabilities of actions at an information set must add up to 1, not all points in the space correspond to valid strategy profiles. The search for a BNE is a search in this multi-dimensional space for a point that satisfies the equilibrium condition: given the other player's part of the profile represented by the point, no player would like to deviate from its strategy.

### 5.2.3 Approximate equilibria

A point in the above multi-dimensional space is a BNE if it satisfies certain constraints which guarantee that at each information set of each player, the player's strategy is rational. In other words, if there is a single action with maximum expected value, that action is played with probability 1. If there are several such actions, the probability mass is divided among them such that the same rationality holds for the other player. Thus no player is tempted to deviate from the prescribed strategy. Stated more formally, the following condition should hold at each information set $h$:

$$\sum_{a \in A_i(h)} \sigma_i(h, a) * E(\text{Payoff}_i(a)) = max_a(E(\text{Payoff}_i(a))) \tag{5.1}$$

where $\sigma_i(h, a)$ is the probability that strategy $\sigma$ assigns to taking action $a$ at $h$ and $E$ is the expected value, to player $i$, of taking action $a$. This expected value is calculated in terms of the payoffs of the leaf nodes that $i$ can end up in when taking action $a$ and the probabilities of actions along the branches from the root to these leaves passing through $a$.

Now, consider a situation where some of these constraints are violated. For example, at an information set $h$, the above equation does not hold; the right-hand side is greater than the left-hand side by 0.5. This means that, holding the other player's strategy fixed, this player gains 0.5 if he switches to the action that maximizes the right-hand side. We refer to the amount by which a constraint $c$ is violated as $\delta_c$, known in the literature as *regret*.

A search for an exact equilibrium corresponds to a Constraint Satisfaction Problem. The search for an approximate equilibrium where some $\delta$s are non-zero can be thought of as a Constraint Optimization Problem (COP). In both cases, the variables are the probabilities assigned to actions by strategies and the constraints are as described above. In this work, we try to find an approximate equilibrium by solving a COP.

## 5.3 View Maintenance As A Game

In this section we detail how the view maintenance problem is formulated as a sequential game of incomplete information. We start by presenting the abstraction we will be using, then give a formal definition of the view maintenance game.

### 5.3.1 Problem abstraction

Consider 2 base relations; *Authors* and *Books* with DBMs $DBM_A$ and $DBM_B$. Consider a view whose query is "`SELECT Title, Author FROM Books, Authors WHERE Books.Pages > 600 AND Authors.City = Manhattan`" displaying the titles of all books with more than 600 pages whose authors live in Manhattan. Denoting insertion by $i$ and deletion by $d$, the elements of the vector $v^j_{all} =< i^j_{all}, d^j_{all} >$ represent the number of $i$ and $d$ updates made to relation $R_j$ since the last maintenance process. While $v^j_{all}$ shows the counts of *all* the changes made, $v^j_{pr} =< i^j_{pr}, d^j_{pr} >$ shows counts for only those tuples that are judged by $DBM_j$ to be *potentially relevant* (PR) to the view, i.e. tuples that meet the selection filter specified by the view query for $R_j$. In our example, a tuple in the books relation is potentially relevant if the book has more than 600 pages. Depending on whether the tuple(s) from other relation(s) that a tuple joins with (which we henceforth refer to as *complementary tuples*) meet their respective filters, the update may or may not actually be relevant to the view.

We believe elements of the vector $v^j_{pr}$ represent strategic information that $DBM_j$ would not like to reveal. The importance of this information is if $DBM_j$ has many PR tuples, it may want to withhold this fact and wait for some $DBM_i$ to disclose tuples rather than go ahead and disclose tuples itself. In this case, $DBM_i$, not knowing exactly how many PR tuples $DBM_j$ has, may worry that not enough reward is being accumulated and thus choose to disclose its own tuples. This situation is clearly to $DBM_j$'s advantage.

### 5.3.2 The view maintenance game

Our view maintenance problem can be formulated as a sequential game of incomplete information. Let $n$ be the number of relations and assume each DBM is responsible for exactly 1 relation. Let $c \in \{i, d\}$ denote a change made to a relation, which can be insertion or deletion. Let $p_c^k$ be the probability that a relation has $k$ changes of type $c \in \{i, d\}$. For simplicity, we assume this probability is independent of the particular relation in question. The view maintenance game therefore has the following components [1]:

- $I = \{DBM_1, ..., DBM_n\}$
- $A_j(h)$ is the set of pieces of information that player $j$ possesses but has not revealed on the path from the root to members of the information set $h$
- The type space of player $j$ is $T_j = \{v_{pr}^j \mid 0 \leq v_{pr}^j[c] \leq v_{all}^j[c] \ \forall c \in \{i, d\}\}$; each type corresponds to a pair of possible counts of PR tuples for the 2 kinds of change. If there are $m$ tuples as a whole affected by a given kind of change, the number of PR tuples is anywhere in [0,$m$]. The size of the type space is therefore $|T| = \Pi_{c \in \{i,d\}}(v_{all}^j[c] + 1)$ [2]
- The transition probability of the chance move assigning player $j$'s type is $p \in \Delta(T_j)$ where $\Delta(T_j)$ is the set of all probability distributions over $T_j$. Assuming the numbers of $i$ and $d$ changes are independent, the probability of a type is $p(v_{pr}^j) = \Pi_{c \in \{i,d\}} p_c^{v_{pr}^j[c]}$
- The payoff $u(z)$ at a terminal node $z$ is determined by the sequence of actions taken on the path from the root to $z$. We need to specify, for each action, the cost to the

[1] We assume that moves are sequential rather than simultaneous; a player taking an action can observe all earlier actions.

[2] To see how $v_{pr}^i$ affects the reward function, we follow the argument in [64] whereby information affecting the set of actions available to a player can be thought of as affecting the player's reward function. We can think of all the actions being available all the time, with the resulting payoffs depending on the private information.

player disclosing the information and the common reward that all players get when this information is disclosed

- $T$ specifies the number of stages in the game

We assume that initially, each $DBM_j$ discloses its $v^j_{all}$. Alternatively, this information can be obtained from statistics about how many changes of each type are made to the database, on average. The probability of a given $v^j_{pr}$ can be estimated from historical statistics.

As for the reward, we base the reward for a piece of information on 3 factors: 1) the type of change ($i$ or $d$); 2) the base relation affected by the change and 3) whether the information represents a main tuple or the complementary of an already disclosed tuple. The rationale is that user preferences can be such that one type of change is more important than the other and some relations need to be more up-to-date than others. The third factor allows the VH to express different preferences for knowing different kinds of information. As in the case of rewards, disclosing different pieces of information incurs different amounts of privacy, communication and other kinds of costs. The incurred cost can also depend on what has been revealed so far (e.g. privacy costs can be sub- or super-additive).

## 5.4 Anytime Algorithm for Computing Approximate BNE

The algorithm we propose operates on the tree of an Extensive Form Game. For example, it can operate on the game tree representing an instance of the view maintenance problem. The algorithm first collapses the game tree by making "obvious" decisions and backing up values wherever possible. This backing up eliminates parts of the tree that will obviously never be reached, resulting in a collapsed tree of smaller size. The algorithm then tries to satisfy constraints derived from the collapsed game tree (of the form given in Equation 5.1) *as much as possible* in a hill-climbing manner. It generates a initial random strategy profile and iteratively improves it until either the profile becomes "stable", or no

further improvement is possible. In the latter case, the profile is randomly perturbed and the process repeats. We discuss a range of stability measures that we can use in assessing a profile. The following paragraphs elaborate on these steps.

### 5.4.1 Collapsing the game tree

Our experiments in building game trees from instances of the view maintenance problem show that the size of the *raw* game tree (the tree before any collapsing) tends to be very large. Examining raw trees shows that there are some nodes at which decision making is not complicated by the incompleteness of information. These are nodes where a player would choose to reveal the same piece of information regardless of the type of the other player. We therefore collapse the raw tree using the following simple algorithm. Initially, all nodes are assumed to be roots of collapsible subtrees. We work from the leaves of the tree upward, determining which nodes are indeed roots of collapsible subtrees. For each such node, we collapse its subtree using simple backups, excising the collapsed subtree. The node becomes a terminal node whose payoffs reflect backed up values. Algorithm 5.4.1 shows how this is done.

**for all** $level$ such that $0 \leq level \leq 2T$ **do**
  collapsible[$level$] = non-terminal nodes at depth $level$
**end for**
**for all** $level$ such that $0 \leq level \leq 2T$ **do**
  **for all** $node$ in collapsible[$level$] **do**
    $i$ = Player($node$)
    **if** best action is the same across $h(node) \vee (|h(node)| == 1) \vee (|A_i(h(node))|$==1)
    **then**
      children = $node$.children
      $node$.payoff$_i$ = $\max_{c \in children}$ $c$.payoff$_i$
      delete $node$.children
    **else**
      remove all ancestors of $node$ from their respective collapsible[$level$] arrays
    **end if**
  **end for**
**end for**

**Algorithm 1:** Simple algorithm for collapsing trees

The algorithm starts by assuming all nodes are collapsible. For each node, it checks if at least one of three conditions holds: 1) incompleteness of information does not affect the player's decision, so the best action is the same regardless of which particular node the player is at; 2) $h(node)$, the node's information set, contains only this node, so the best action is just chosen; 3) a node belongs to an information set with a single available action. Figure 5.1 shows these three situations. Action nodes are in circles enclosing the index of the acting player. Terminal nodes are shown in black circles with a pair of numbers specifying the associated payoff for each player. A dotted box encloses nodes in the same information set. Because we work from the leaves upward, a node eligible for collapsing always has terminal children. This simple collapsing algorithm is very effective for game trees derived from the view maintenance problem. In Section 5.5.1, we give supporting experimental results and discuss why collapsing is effective.

### 5.4.2 Iteratively improving a point

To iteratively improve a point (strategy profile), the following 3 issues need to be addressed:

1. Which component(s) of the point should we improve? Should we focus on improving individual constraints or the profile as a whole?

2. How should we explore the space? How do we generate neighboring points to which we can move?

3. How do we assess a point? What measure of a point indicates the algorithm is moving in the right direction in the multi-dimensional space?

**What should we improve?** As mentioned in earlier, an equilibrium point/profile must satisfy certain constraints. Improving individual constraints or the profile as a whole amounts to making local or global changes to a profile, respectively. A local change tries to improve a constraint associated with some information set $h \in H_i$ to reduce the regret of

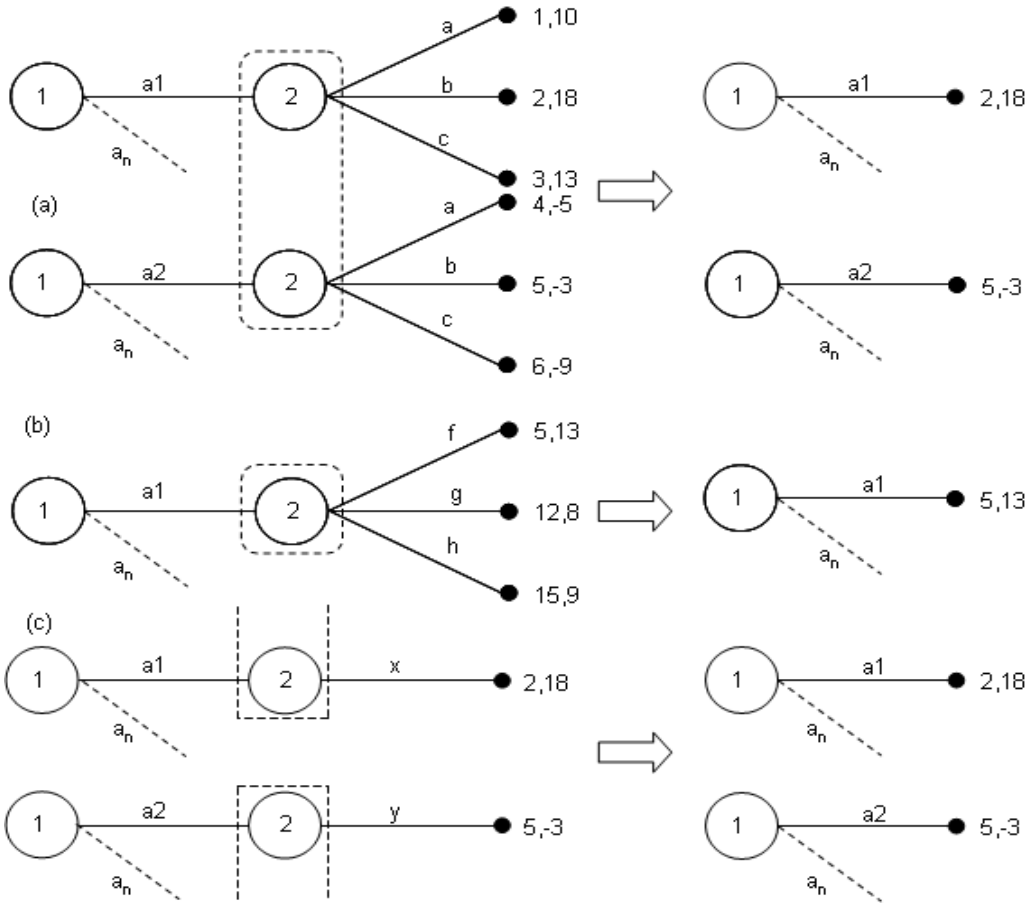



**Figure 5.1.** Collapsible subtrees: (a) action b is the best across the information set (b) singleton information set (c) single available action

player $i$ at $h$. A global change completely overhauls one or both players' strategies to get to a more "stable" point; one at which the players' motivations to deviate is lower. Owing to the complexity of overhauling a profile, we improve individual constraints with the hope of effecting a global improvement through local changes. Because it is not easy to determine which local changes produce the largest global improvement, we use the local regret, $\delta$, as a heuristic to decide which parts of a strategy profile are more important to improve. Constraints with high $\delta$s are associated with information sets with high regrets.We greedily attempt to improve first. Empirical observations indicate that this heuristic is indeed useful; improving constraints with high $\delta$s results in more stable points. We quantify the notion of stability later in the text.

**Generating potential next points** For each variable (action probability) $v$ involved in the constraint $c$ with the maximum $\delta_c$, we calculate the required change in $v$ to bring $\delta_c$ down to 0, assuming all other variables are unchanged. We assess the impact of changing $v$ on the $\delta$s of other constraints by evaluating the partial derivatives of affected constraints w.r.t. $v$. If changing $v$ results in a point with greater than or equal stability than the current point, the new point is added to the list of Potential Next Points (PNPs).

The approach described above is one of two ways of decreasing a given $\delta_c$. Instead of changing the probabilities of actions involved in $c$ as done above, we can switch the player's preference for two actions $a$ and $b$ by switching their probabilities. We generate points from such reversals and, as with the first approach, we assess the broader impact of the change and decide whether to admit the points to PNP.

**Assessing a point** Now that we have a set of PNPs, we need to move to the most stable PNP. Even though $\delta$s determine which part of the profile to improve first, these local measures do not provide good basis for comparing the stability of different points. The problem is that each $\delta$ specifies the additional reward a player gets if it deviates at a single information set. This says nothing about the player's potential gains if it deviates at multi-

ple information sets, nor about the change in the regret of the other agent resulting from the deviation. We therefore need a global measure that specifies a player's overall motivation to deviate from (or completely overhaul) its strategy.

Following the notion of $\epsilon$-BNE, we consider a profile stable if no player stands to make more than $\epsilon\%$ more reward by changing its prescribed strategy. We define a global measure called *Maximum Overall Motivation* (MOM) to deviate. MOM($\sigma$) is the maximum, over all players, upper bound on motivation to deviate from $\sigma$, assuming strategies of other players are held constant. MOM is therefore an upper bound on $\epsilon$. The lower the MOM, the more stable $\sigma$ is. Approximating an equilibrium this way makes sense because practically, a player will not want to take on the difficult task of calculating its best response strategy if it knows that it stands to increase its payoff by no more than $\epsilon\%$.

We propose a simple way of calculating MOM. To calculate the upper bound on the motivation of player $i$ under the strategy profile $\sigma$, we build a modified game $\Gamma_{revealed}$ from the original game $\Gamma$. $\Gamma_{revealed}$ is a single-player perfect information game where $i$ plays with Nature. We construct it as follows. Each node $n \in h$ where $h \in H_{j \neq i}$ is changed to a chance node where the probability of Nature playing action $a$ is $\sigma_j(h, a)$. In addition, the information sets in the original game tree are revealed, i.e. $i$ is granted full access to the history of play including the moves of Nature that determined players' types, thereby removing $i$'s uncertainty about where it is within a given information set. $\Gamma_{revealed}$, being a perfect information game, can be solved by doing simple backups. $i$'s payoff in $\Gamma_{revealed}$ is an upper bound on the payoff of its best response strategy in $\Gamma$, since $i$ can do no better than having perfect information. Because of the simplicity of doing backups, we can quickly evaluate MOMs for a large number of PNPs.

To summarize, we use a local measure ($\delta$) to generate PNPs and a global measure (MOM) to assess and compare points. The global measure indicates how stable a point is, but does not give indication of how it should be improved. The local measure indicates where it may be effective to try to improve.

**A range of approximations:** Examining the MOM landscape in some of the experiments we conducted showed that sometimes the upper bound provided by MOM is very loose; e.g., the MOM landscape is everywhere higher than 20%. The reason is that $\Gamma_{revealed}$ is *too easy* compared to $\Gamma$. We can obtain a tighter upper bound if we calculate payoffs in a game that is harder than $\Gamma_{revealed}$ but still easier than $\Gamma$. In fact, there is a whole spectrum of such games with varying levels of difficulty. At one end of the spectrum is $\Gamma_{revealed}$ where *all* information sets are disclosed. These games are very easy to solve but provide very loose upper bounds. At the other end is $\Gamma$ where *no* information sets are disclosed.

To illustrate the kind of bound we get from a slightly harder game than $\Gamma_{revealed}$, consider the game $\Gamma_{LI}$ which differs from $\Gamma_{revealed}$ only in that all information sets except the highest-level sets are revealed. Clearly, the payoff in this game is at least as high as that obtained in $\Gamma$ but no higher than in $\Gamma_{revealed}$. Solving this game is still easy; do regular backups from the bottom of the game tree upwards, and, on reaching the highest-level information sets, choose the action that maximizes reward in expectation over the turn player's beliefs about where it is within this information set. We call the maximum motivation to deviate from $\sigma$ in this game *MOM-LessInformation*(MOM-LI).

To illustrate the different possible payoffs with an example, Table 5.1 shows the payoffs achieved by player $i$'s different strategies when its opponent plays its part of the strategy profile $\sigma$ in different variants of an example game. *Payoff($\sigma$)* is $i$'s payoff from playing $\sigma_i$ in the original game $\Gamma$. *PBR* is $i$'s payoff from its best response to $\sigma_j$ in $\Gamma$. It is not easy to compute and requires calculating $i$'s payoff in a transformed game where $j$'s nodes are changed to chance nodes with action probabilities as dictated by $\sigma_j$. *PPI* is $i$'s payoff from its best response to $\sigma_j$ in the easy perfect information game $\Gamma_{revealed}$. *PLI* is $i$'s payoff from its best response to $\sigma_j$ in the slightly harder game $\Gamma_{LI}$ with less-than-perfect information. Since $Payoff(\sigma) \leq PBR \leq PLI \leq PPI$, the overall motivations calculated using $PLI$ and $PPI$ (MOM-LI and MOM, respectively) are upper bounds on a player's actual regret.

**Table 5.1.** Calculating MOM With Different Amounts of Information

| Quantity | Player 1 | Player 2 |
|---|---|---|
| Payoff($\sigma$) | 8.97 | 7.88 |
| Payoff of B.R.in $\Gamma$ (PBR) (unknown) | 10.39 | 8.48 |
| Payoff in $\Gamma_{revealed}$ (PPI) | 12 | 9.09 |
| Payoff in $\Gamma_{LI}$ (PLI) | 11 | 9 |
| Overall Motivation (PPI-Payoff($\sigma$))/PPI * 100% | 25.25% | 15.26% |
| MOM | 25.25% | |
| Overall Motivation LessInfo (PLI-Payoff($\sigma$))/PLI * 100% | 18.45% | 12.4% |
| MOM-LI | 18.45% | |

In Table 5.1, we get a much tighter bound on regret when using $PLI$ rather than $PPI$ (18.45% vs 25.25%) at the cost of a slightly more involved computation.

## 5.5 Experimental Results

In this section, we present results of our efforts to solve games of incomplete information, whether they are general games or games derived from instances of the view maintenance problem introduced in Chapter 5.1. We start by analyzing the efficacy of our pre-processing step which tries to losslessly collapse the game tree. We then compare the performance of our anytime hill-climbing algorithm to that of the Quantal Response Equilibria (QRE) algorithm [86].

### 5.5.1 The effect of collapsing

The first set of experiments we conducted investigates the efficacy of our collapsing algorithm for trees from random instances of the view maintenance problem (henceforth called VM trees) as well as general trees. Table 5.2 shows the result of collapsing VM trees. Both players have the same type space ($v_{all}$). Unless indicated otherwise in brackets, we generated 10 random instances per configuration, for a total of 65 instances. As can be seen, the size of the collapsed tree is roughly an order of magnitude smaller than the raw

**Table 5.2.** Collapsing VM trees

| T | $v_{all}$ | Raw Size | Avg % Reduction |
|---|---|---|---|
| 2 | $< 1,1 >$ | 716 | 84.3 |
| | $< 1,2 >$ | 2253 | 89.5 |
| | $< 2,1 >$ | 2253 | 88.8 |
| | $< 2,2 >$ | 6847 | 84.7 |
| 3 | $< 1,1 >$ | 3608 | 91.2 |
| | $< 2,1 >$ | 15423 | 88.9 |
| | $< 3,1 >$(5) | 36232 | 92.9 |

tree. This pre-processing step is therefore very useful for providing our anytime algorithm with tractable input.

To see how much general game trees collapse, we generated trees where both players have the same number of types and the same number of actions is available at each information set. We generated 10 random trees for each configuration $< T$,*numTypes,numActions*$>$ where $1 \leq T \leq 4$ and both $numTypes$ and $numActions$ are 2 or 3. Payoffs were generated randomly in the range [0,15]. Table 5.3 shows the raw tree size and average percentage reduction for these configurations. N/A entries were too large to generate. Collapsing general trees yields a reduction in the number of nodes in a tree that ranges from 4 to 27%.

Clearly, trees derived from the view maintenance problem are much more susceptible to collapsing. To understand why this is the case, we need to remember the source of uncertainty faced by a player in a VM tree. With imperfect information about player $j$'s type, player $i$ is uncertain about the number and nature of tuples yet undisclosed by $j$. However, there is *no uncertainty* regarding the *payoffs* of actions. This results in the lowest level of the tree always collapsing, making it more likely that levels higher up in the tree collapse as well (a node is eligible for collapsing only if its children are terminals).

### 5.5.2 Performance of the search algorithm

We compared our anytime algorithm to the Quantal Response Equilibria (QRE) algorithm [86] as implemented in Gambit [57]. QRE has the advantage of being an anytime

**Table 5.3.** Collapsing general trees with 2 (top) and 3 (bottom) types per player

| | #Actions=2 | | #Actions=3 | |
|---|---|---|---|---|
| $T$ | Raw Size | Avg % Reduction | Raw Size | Avg % Reduction |
| 1 | 34 | 23.5 | 58 | 15.5 |
| 2 | 130 | 25.5 | 490 | 21.9 |
| 3 | 514 | 27.3 | 4378 | 22.3 |
| 4 | 2050 | 24.9 | N/A | N/A |
| 1 | 70 | 11.1 | 124 | 4.4 |
| 2 | 286 | 11.7 | 1096 | 7.5 |
| 3 | 1150 | 13.5 | 9844 | 7.9 |
| 4 | 4606 | 12.6 | N/A | N/A |

algorithm, so we can calculate regret values for its intermediate results and compare them to intermediate results from our algorithm.

We ran the two algorithms (anytime search and QRE), on 2 tree types (VM and general trees) using MOM and MOM-LI for a total of 8 sets of experiments. In all our results, we are interested in the average time, in seconds, needed to reduce regret (MOM or MOM-LI) to 5%. We bin results by tree size and show the percentage of trees in each size bin for which the algorithm could reach the desired regret within the indicated time range. Note that the reported tree size is the size of the collapsed, rather than the original, tree. We omit from our tables time or size bins that were found to be empty. Because results using MOM-LI are always better than using MOM, we only present the former.

Table 5.4 compares the percentage of VM trees in a given size range that were solved within the indicated time by our algorithm and QRE when using MOM-LI. We randomly generated costs and rewards, which sometimes result in a tree that collapses to an empty game. This happens if, for example, it is always lucrative to disclose all information regardless of any uncertainty. Out of the 65 VM trees reported in Table 5.2, 52 collapse to non-empty games. For each of these 52 trees, we ran our search algorithm 3 times starting from different random points. As can be seen in Table 5.4, for most of the trees in any given tree size bin, our algorithm reaches the required level of regret within 100 seconds. Our

**Table 5.4.** Percentage of VM trees solved by our algorithm (top) vs. QRE (bottom)

| Tree Size | $\leq 20$ sec | 21-100 sec | 101-500 sec | 501-1000 sec | $> 1000$ sec |
|---|---|---|---|---|---|
| 0-200 | 100<br>88.9 | _<br>_ | _<br>_ | _<br>_ | _<br>11.1 |
| 200-400 | 96.1<br>94.1 | _<br>5.9 | 3.9<br>_ | _<br>_ | _<br>_ |
| 400-1000 | 66.7<br>50 | 20.8<br>25 | 8.3<br>12.5 | _<br>_ | 4.2<br>12.5 |
| 1000-2000 | 28.5<br>_ | 47.6<br>_ | 14.3<br>14.3 | 4.8<br>28.6 | 4.8<br>57.1 |
| 2000-3600 | _<br>_ | 44.4<br>_ | 48.1<br>22.2 | 3.7<br>_ | 3.7<br>77.8 |

**Table 5.5.** Percentage of general trees solved by our algorithm (top) vs. QRE (bottom)

| Tree Size | $\leq 20$ sec | 21-100 sec | 101-500 sec | 501-1000 sec | $> 1400$ sec |
|---|---|---|---|---|---|
| 0-200 | 96.2<br>97.1 | 2.9<br>_ | _<br>_ | _<br>_ | 0.9<br>2.9 |
| 200-400 | 85.3<br>40 | 12<br>56 | 2.7<br>_ | _<br>_ | _<br>4 |
| 400-600 | 44.4<br>_ | 55.6<br>100 | _<br>_ | _<br>_ | _<br>_ |
| 800-1100 | _<br>_ | 20<br>_ | 40<br>80 | 20<br>10 | 20<br>10 |

anytime search algorithm performs better than QRE on smaller trees and much better than it on larger trees. QRE fails to finish within the allocated time on a much higher fraction of larger trees than our algorithm.

Table 5.5 compares the performance of our algorithm on general trees to QRE. We generated 73 random trees, none of which collapsed to an empty tree Again, on each tree we ran our search algorithm 3 times starting from different random points. General trees proved to be more challenging than VM trees. Our algorithm performs better than QRE on smaller trees and is comparable to it on larger ones.

Some remarks about our results are in order. First, it should be noted that there are many possibilities for fine-tuning the search algorithm (e.g. random restarts and changing

the magnitudes of random perturbations as the search proceeds), but we leave this for future work. Second, it is important to remember that a strategy profile provides players with a plan of action for every type with non-zero probability in the game definition. Therefore we only need to run the search algorithm when the players' type spaces, or the probability distributions over them, change. In the view maintenance problem, database managers can continue using a strategy as long as the number of potentially relevant tuples and the probability distributions over them are unchanged. So the time taken to calculate a strategy profile is amortized over all the view maintenance episodes for which the profile is valid.

## 5.6 Summary

In this chapter, we studied communication among self-interested decision makers whose goal is to maximize their individual payoffs. As an example of such settings, we presented the view maintenance problem with self-interested database managers. We formulated this problem as a game of incomplete information and presented a general anytime hill-climbing algorithm for solving this class of games, i.e. for finding (approximate) equilibrium profiles. To aid the hill-climbing search, we developed local and global profile stability measures.

We experimentally investigated two aspects of our work. First, we verified the effectiveness of a pre-processing step that we developed to losslessly collapse game trees whenever possible. This step proved very effective when applied to games derived from the view maintenance problem, and somewhat effective on general trees. Second, we compared the time taken by our algorithm to reduce regret (and thus increase stability) compared to an existing algorithm. Our algorithm reduces regret to 5% faster, and on a larger fraction of test cases, than the Quantal Response Equilibria algorithm in Gambit.

# CHAPTER 6

# GAME-THEORETIC MODELS AND OPTIMIZATION

In previous chapters, we were concerned with structured interaction among cooperative agents. Now we turn our attention to *self-interested* agents whose goal is to maximize their individual rewards, rather than team rewards. Interaction among self-interested agents takes place in the context of a *game*. Our model Event-Driven Interactions with Complex Rewards (EDI-CR) can be used to represent *loosely coupled stochastic games*, which have the same characteristics as their cooperative counterpart in Section 2.2.2. As in the cooperative case, generically representing this kind of games in extensive form without regard to their special structure results in very large problems.

In this chapter, we discuss the use of optimization techniques to find equilibrium policies for our class of games. We give a brief background on games and equilibria in Section 6.1. In Section 6.2 we re-state an existing formulation of finding an equilibrium profile as a bilinear program and compare this approach to representing our games in extensive form and solving them using a game-theoretic software package. We provide analytical and experimental results to show the representational and computational savings we obtain compared to extensive form in settings with different amounts of interaction. Noting that self-interest does not preclude communication, we use communication to vary the amount of interaction among agents. So to create settings with different amounts of interaction, we experiment with different communication schemes.

In Section 6.3, we discuss a different approach where finding an equilibrium is formulated as a problem of solving a system of non-linear equations. The system of equations can be solved using continuation methods which are discussed in this section, with references

to related work. Although this work is in its early stages, the resulting formulation has the advantage of allowing for more than two agents.

## 6.1 Background

The field of game theory focuses on situations where self-interested players make decisions that affect each other and/or affect a common environment. Each agent tries to respond to the decision-making strategies of the others in a way that maximizes his own reward.

Games can be categorized along several axes. Perfect recall (vs. imperfect recall) games involve players who never forget actions, whether theirs or others', once they observe them. In games of incomplete information (vs. perfect information), a player does not know what moves have already been played by other players, resulting in uncertainty about the current state of the world and multiple game situations being indistinguishable to that player. Games can also be classified by the number of stages (decision-making points) they contain; 1-stage games involve only one stage of decision making. A *sequential stochastic game* describes a situation where agents interact over a number of stages. Each stage begins with the game at some state. Agents take actions simultaneously and, in general, receive rewards based on the actions taken by all agents and the particular state the game was in. The game then probabilistically transitions to a new state based on the previous state and joint actions. In a general game, the agents are tightly coupled; each action of each agent affects the rewards and next games of all others.

A *strategy profile* is a set of strategies $\sigma = (\sigma_1, ..., \sigma_n)$, one per player. $\sigma_{-i}$ denotes the set of strategies of all players except $i$. The goal in competitive settings is typically to find an *equilibrium* strategy profile from which no player has motivation to deviate. In games of perfect information, the Nash equilibrium is a commonly used solution concept. A strategy profile $\sigma$ is a Nash equilibrium if, for every player $i$, $u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i})$ for all $\sigma'_i$, where $u_i$ is $i$'s utility.

## 6.2 EDI-CR With Varying Communication As A Bilinear Program

In this section, we review an existing formulation of a stochastic game as a bilinear program (BLP) derived by Petrik and Zilberstein [69]. We then introduce varying levels of communication and investigate the effect this has on the representational and computational savings obtained using EDI-CR and BLP versus representing and solving the problem as an extensive form game (EFG, see Section 2.1.2).

The formulation of extensive form games as BLP is as follows:

$$\begin{aligned}
\text{Maximize} \quad & x_i^\mathsf{T} r_i + x_i^\mathsf{T}(C_i + C_j)x_j + x_j^\mathsf{T} r_j - \lambda_i^\mathsf{T} b_i - \lambda_j^\mathsf{T} b_j \\
\text{subject to} \quad & A_i x_i = b_i \quad A_j x_j = b_j \\
& r_i + C_i x_j - A_i^\mathsf{T} \lambda_i \leq 0 \\
& r_j + x_i^\mathsf{T} C_j - A_j^\mathsf{T} \lambda_j \leq 0 \\
& x_i, x_j \geq 0
\end{aligned} \tag{6.1}$$

As in Chapter 3, agents' policies are represented using sequence form [50]. $x_i$ and $x_j$ are vectors of realization weights of agent $i$ and $j$'s action sequences, respectively. $r_i$ (resp. $r_j$) is a vector representing the individual rewards of agent $i$ (resp. $j$); those rewards that do not depend on what the other agent does. $C_i$ and $C_j$ represent rewards of $i$ and $j$ whose attainment depends on what *both* agents did. $A_i$, $A_j$, $b_i$ and $b_j$ form the policy constraints. $\lambda_i$ and $\lambda_j$ are the variables of each agent's dual optimization problem. Their presence in the objective function reflects our interest, not in a solution that maximizes the sum of rewards, but in one that is an equilibrium. The above bilinear program is solved using the Multi-agent Planning Bilinear Program algorithm [69] to find a Nash equilibrium.

We believe the idea of binning can be used to linearize the above bilinear program, as we did in the cooperative case in Section 3.3. The first part of the objective function in (6.1) is the same as in the cooperative case, and can be linearized in the same way. An advantage of our binning technique for linearization is that it saves us from dealing with the large reward matrices by representing only distinct entries in these matrices. But in

(6.1), unlike in the cooperative case, the reward matrices also appear in the constraints. We need to find a way to collapse them in the constraints. We leave the details of using binning in the self-interested case for future work.

### 6.2.1 Analytical and experimental setup

We now investigate how the degree of coupling affects the relative savings of using EFG and EDI-CR. One simple way to change the degree of coupling is by introducing different amounts of communication among agents. Communication is a special kind of transition interaction; sending a message makes the recipient transition to a state where the message is observed, thereby affecting its transition probability. As such, communication can be handled by any solution method used in the no-communication case.

We present experimental results from 8 instances of a self-interested variant of the Mars rovers domain (Section 2.2.1.3) where the rovers belong to different countries and each one has its own reward function. As in the cooperative case, a rover's action can affect the reward and/or transition of another rover. Although self-interested, a rover can benefit from communicating its progress to another rover. For example, consider 2 rovers $i$ and $j$ where $i$'s visiting site 1 accomplishes some initial work that makes it useful for $j$ to visit site 2, which would otherwise not be worthwhile. Suppose that $j$ can only visit site 3 after visiting site 2, and that $j$'s visit to site 3 also sets the stage for $i$'s visit to site 4. In this case, even if communication has a cost, $i$ may choose to tell $j$ that it visited site 1 to tempt $j$ to visit site 2 and then 3, eventually setting the stage for $i$'s visit to site 4. The decision whether to communication can be reasoned about in a decision-theoretic manner similar to reasoning about domain actions.

Communication among self-interested agents raises many issues that are of no concern in the cooperative case, like truthfulness, issues of trust and privacy concerns. In this chapter, we assume truthful communication.

We present three communication schemes; no, mandatory and optional communication. For each, we analyze the effect on the size of an instance when represented using EDI-CR and EFG. For EFG and EDI-CR, we measure size as the number of states in the joint game tree and in each agent's decision process, respectively. We express these quantities in terms of $A$ actions, $O$ outcomes per action, $T$ time steps, $k$ reward interactions and $m$ transition interactions, with $k + m \leq A$. The variables $k$ and $m$ allow us to investigate how the number of interactions and their nature affect the size of a representation. We stress that our analysis is not specific to the Mars rover example. It applies to any loosely coupled game that fits the characterization we give in Section 2.2.2. To simplify the analysis, we assume that an action takes one time unit and that actions can repeat.

We also look at the effect of communication on the time to find the first Nash equilibrium[1]. EFGs are solved using the `logit` solver in the game theoretic package Gambit [57] and reported as "Gambit" (more details about this algorithm in Section 6.3.4). EDI-CR is solved as a bilinear program reported as "BLP". We time out a solver after 30 minutes and report "N/A".

In our Mars rovers instances, $T \in [6, 8]$ and the number of actions is 5 or 6 (unlike the analytical model, an action here can take more than one time unit). To avoid generating very large games that would not fit in memory regardless of the representation, we specify restrictions over actions by having earliest start times before which they cannot proceed.

### 6.2.2 No communication

We first look at the case where communication is not allowed. An agent makes decisions based only on its local state, which keeps track of what actions have been done so far and the outcomes obtained for them. With EFG, each stage consists of actions and outcomes for both agents. The number of nodes is therefore $\mathcal{O}(A^{2T}O^{2T})$.

[1]Because it is hard to compare solution qualities in selfish settings, we are concerned with finding *any* equilibrium

**Table 6.1.** Size and performance comparison for the no-communication case (times in seconds)

| EFG infosets | EFG size | EDI-CR size | % reduction | Gambit time | BLP time |
|---|---|---|---|---|---|
| 49 | 1301 | 132 | 89.85% | 9 | 2 |
| 68 | 1618 | 140 | 91.35% | 18 | 9 |
| 100 | 3195 | 216 | 93.24% | 63 | 2.4 |
| 151 | 7987 | 303 | 96.21% | 306 | 2.5 |
| 173 | 11.2k | 348 | 96.89% | 1080 | 3 |
| 296 | 25.1k | 610 | 97.57% | N/A | 2.5 |
| 333 | 44.4k | 695 | 98.43% | N/A | 2.6 |
| 841 | 473k | 2079 | 99.56% | N/A | 8 |

In EDI-CR, each stage in an agent's decision process consists of an action and outcome for this agent only, resulting in $\mathcal{O}(A^T O^T)$ states per agent. Because there are transition interactions, an agent needs to remember the outcome of affected actions, but our state representation remembers all outcomes, so states space size is independent of $m$.

While theoretically the sizes of EFG and EDI-CR are both exponential in the time horizon $T$, Table 6.1 shows that in practice, doubling the exponent results in game trees that are too large to build and/or solve.

### 6.2.3 Mandatory communication

We now model situations where communication is inherently part of the setting, rather than a conscious decision on the part of the agents. An agent $i$ doing its part of a reward or transition interaction *involuntarily leaves a trace* that it has done this action. Consequently, the other agent $j$ will see this trace upon doing *its* part of the action. An agent does not suffer a cost for this implicit communication, but cannot avoid it either. To allow an agent to make decisions based on the traces it sees, an agent's state keeps track of a flag denoting whether a trace was seen upon doing each reward or transition interaction.

In EFG, even though the state now stores the actions, outcomes and $k + m$ flags of each agent, the number of states is *not* $\mathcal{O}(A^{2T} O^{2T} 2^{2k+2m})$. The reason is that the values of an agent's flags are fully determined by earlier actions of the other agent, so when an

agent does an ineteraction, there is no branching over whether it will see a trace there. In fact, there is no more branching in this communication scheme than in the case without communication, and the number of nodes in the EFG tree is still $\mathcal{O}(A^{2T}O^{2T})$.

Even though they are of the same size, the EFG representation of the no communication case and the mandatory case are not the same. To see why, note that because of the additional flags, an agent has more information available to make its decisions when there is communication. This translates into the game tree having more information sets per agent; nodes that were indistinguishable in the absence of communication can now be told apart. Comparing Tables 6.1 and 6.2 shows how much the number of information sets in an EFG increased. Since a policy specifies a distribution over actions for each information set, mandatory communication increases the size of the policy space and makes finding a Nash equilibrium more difficult. Table 6.2 indeed shows that even though the size of EFG did not change, the solution time generally increased.

As for EDI-CR with mandatory communication, there *is* probabilistic branching in an agent's decision process over whether or not it sees a trace upon doing an interaction, since that depends on what the other agent has done. The size of each agent's process is therefore $\mathcal{O}(A^T O^T 2^{k+m})$.

It is interesting to note the effect of mandatory communication on the size gap between EFG and EDI-CR. Compared to no-communication, mandatory communication results in EDI-CR achieving less reduction in size. The increased coupling introduced by communication makes EFG less inadequate, compared to EDI-CR, although the savings obtained by EDI-CR are still quite significant. If we increase the frequency and language of communication so that the agents communicate their entire states at every state, the decision processes will be so tightly coupled that EDI-CR's advantage of representing them separately will be lost.

**Table 6.2.** Size and performance comparison for the mandatory communication case (times in seconds)

| EFG infosets | EFG size | EDI-CR size | % reduction | Gambit time | BLP time |
|---|---|---|---|---|---|
| 82 | 1301 | 1107 | 14.91% | 21 | 2.7 |
| 83 | 1618 | 377 | 76.70% | 29 | 6 |
| 135 | 3195 | 600 | 81.22% | 120 | 6.5 |
| 204 | 7987 | 1481 | 81.46% | 460 | 3 |
| 201 | 11.2k | 2600 | 76.80% | 900 | 4 |
| 574 | 25.1k | 3475 | 86.17% | N/A | 5 |
| 630 | 44.4k | 3000 | 93.24% | N/A | 14.3 |
| 1438 | 473k | 7823 | 98.35% | N/A | 5.6 |

### 6.2.4 Optional communication

We now look at optional communication where an agent can choose whether to leave a trace upon doing its part of an interaction. Even though communication here does have a cost, an agent may still decide to communicate if it knows that communication will cause the other agent to do something beneficial to it. For example, in the Mars rovers scenario, if rover $j$ knows from $i$'s policy that if $i$ visits site $s_i$, then $i$ will visit $s_j$, and if $s_j$ has a much higher value if visited by both rovers, then rover $i$ will choose to leave a trace at $s_i$ as an inducement for $j$ to go there too.

To represent optional communication in EFG, in addition to actions and outcomes for each agent, there is an action node with two branches (leave trace or not) after every decision to do part of an interaction. A state keeps track of the actions and outcomes of both agents, as well as at most $k + m$ binary communication decisions per agent, for a total of $\mathcal{O}(A^{2T}O^{2T}2^{2(k+m)})$ states. Note that even though in this scheme an agent can potentially have the same information to make its decisions as in the mandatory case, the number of decisions itself is much larger, because of the communication decisions, resulting in a larger number of information sets.

In EDI-CR, again, there are communication decision nodes, in addition to branching over whether an agent will see a marker upon visiting a site. The number of states is $\mathcal{O}(A^{T}O^{T}2^{2(k+m)})$.

**Table 6.3.** Size and performance comparison for the optional communication case (times in seconds)

| EFG infosets | EFG size | EDI-CR size | % reduction | Gambit time | BLP time |
|---|---|---|---|---|---|
| 547 | 21.1k | 6213 | 70.58% | N/A | 8.6 |
| 136 | 3777 | 671 | 82.23% | N/A | 3 |
| 190 | 7511 | 1093 | 85.45% | N/A | 2.8 |
| 602 | 51.6k | 5651 | 89.06% | N/A | 214 |
| 589 | 68.3k | 5766 | 91.57% | N/A | 11 |
| 2668 | 295k | 13.9k | 95.29% | N/A | 35 |
| 2004 | 316k | 10.2k | 96.76% | N/A | 32 |
| N/A | 2200k | 21.8k | 99.01% | N/A | 195 |

Table 6.3 shows that having communication decisions results in huge EFG trees, making it impossible for Gambit to solve them within a reasonable amount of time. However, the $4^{th}$ instance shows that solution time and size are not always correlated, which can be explained by the fact that we are searching for the *first* equilibrium we can find, and the time this takes depends on both the size of the problem and the structure of the search space.

The results in this section show that even as we increase the amount of interaction among agents by introducing communication, EDI-CR is still much more compact, and allows the use of faster solution algorithms, than a general representation like extensive form games. In fact, many of the instances we obtained after introducing communication are too large to represent using EFG, let alone solve.

## 6.3 Finding Equilibria As A System Of Non-linear Equations

In Section 3.4, we formulated the problem of finding the optimal policy for cooperative agents as a problem of solving a system of non-linear equations. The same can be done for the problem of finding an equilibrium strategy profile for a group of self-interested agents. This system can then be solved using continuation methods or using a general-purpose solver.

The formulations as a system of equations discussed in this section have the key advantage of allowing for more than two agents, unlike the bilinear program formulation from the previous section. For this reason, we believe it is important to include them in this thesis, even though this line of work is still in its early stages. We start an investigation into how existing formulations as a system of equations can be adapted to exploit structured interactions in EDI-CR, in which case we will have a formulation that can both be efficiently solved and admit more than two agents. The initial investigation in this section raises many interesting questions for future research.

We start with a background on the use of continuation methods for normal and extensive form games (NFG, EFG) in the work of Govindan and Wilson [39] which was reviewed in the work of Blum et. al [51]. We then go into some detail for the case of EFG and discuss the possibility of adapting the EFG formulation to EDI-CR problems. In Section 6.3.4, we give a brief survey of related work that demonstrates that continuation, and later homotopy, methods have a long history of application in game theory.

### 6.3.1 Continuation for NFG

As with homotopy methods, in continuation methods we formulate the problem at hand as a system of non-linear equations $F(x, \lambda) = 0$ characterized by a parameter $\lambda$. As opposed to homotopy methods, $\lambda$ moves from 1 to 0 and is interpreted as the magnitude of the perturbation. At $\lambda = 1$, the initial system is maximally perturbed and this perturbed system has an easy solution. Continuation methods trace the solution as $\lambda$ moves to 0. If we have a solution $(x, \lambda)$, to arrive at a nearby solution with a slightly lower $\lambda$, the differential changes in $x$ and $\lambda$ must cancel out so that $F$ remains 0. We therefore need to solve the following equation:

$$[\nabla_x F \quad \nabla_\lambda F] \begin{bmatrix} dx \\ d\lambda \end{bmatrix} = 0 \tag{6.2}$$

When applied to games, the perturbation is a vector of bonus rewards that transforms the original game into one that is easy to solve. Each agent is given a fixed bonus, scaled by $\lambda$, for each of its actions, irrespective of what the other agents play. If the bonuses are large enough, the best response of each agent is easy to find and is independent of other agents, giving a unique pure strategy equilibrium to the perturbed game.

For normal form games (NFGs), the perturbation bonus vector $b$ contains an element for each action of each agent. Following the convention in [51], we denote a game $G$ perturbed by payoff vector $b$ as $G \oplus b$. In this game, for each action $a$ of agent $n$, $a \in A_n$, and set of actions $t$ of all other agents, $t \in A_{-n}$, the payoff to agent $n$ is $(G \oplus b)_n(a,t) = G_n(a,t) + b_a$. If $b$ is large enough, the game has a unique equilbiruim where each agent plays the action for which $b_a$ is maximum.

To formulate the problem of finding an equilibrium as a system of equations, Govindan and Wilson introduced a vector function, called *deviation function*, which Blum et. al call $V^G$. For a game with $m$ actions, $V^G$ maps a strategy profile $\sigma$, of length $m$, to a vector, also of length $m$. For an action $a$ of agent $n$, $V^G_a(\sigma)$ is the payoff to $n$ of playing $a$ while all other agents follow $\sigma$. In other words, it is $n$'s payoff for deviating from $\sigma$ to play the pure strategy $a$.

$$V^G_a(\sigma) = \sum_{t \in A_{-n}} G_n(a,t) \prod_{k \in N \setminus \{n\}} \sigma_{t_k} \tag{6.3}$$

where $N$ is the set of all agents.

The literature also defines a retraction operator $R$ that retracts a point in $\mathbb{R}^m$ to the space of legal profiles $\Sigma$. Govindan and Wilson use the following lemma by Gul et. al [36].

**Lemma 2** *If $\sigma$ is a strategy profile of $G$, then $\sigma = R(V^G(\sigma) + \sigma)$ iff $\sigma$ is an equilibrium.*

Using this lemma, an equilibrium can be defined as the solution to the following system of equations

$$F(x, \lambda) = x - R(x) - V^G(R(x)) - \lambda b \tag{6.4}$$

$V^G + \lambda b$ is the deviation function of the perturbed game $G \oplus \lambda b$, so $F(x, \lambda) = 0$ if and only if $x$ is an equilibrium of $G \oplus \lambda b$.

The expensive step is the calculation of the Jacobian $\nabla_x F$ for equation (6.2). This involves calculating the $m \times m$ Jacobian $\nabla V^G$. For actions $a \in A_n$ and $a' \in A_{n'}$,

$$\nabla V^G_{a,a'}(\sigma) = \sum_{t \in A_{-n,n'}} G_n(a, a', t) \prod_{k \in N \setminus \{n,n'\}} \sigma_{t_k}$$

The entry for $(a, a')$ is the expected payoff to agent $n$ when it deviates to $a$ and $n'$ deviates to $a'$ while all other agents follow $\sigma$. As can be seen, the summation in the above computation involves a number of terms that is exponential in the number of agents.

### 6.3.2 Continuation for EFG

In extensive form games, the bonus vector, and the deviation function, have one entry per *history* of actions, rather than per action. Another difference from the NFG formulation is that the payoff function is defined over *leaves*. Therefore, in the perturbed game, for each leaf $z \in Z$, $(G \oplus b)_n(z) = G_n(z) + b_{H_n(Z)}$, where $H_n(z)$ is the action history of agent $n$ that leads to leaf $z$.

The deviation function in EFG is defined as follows:

$$V^G_h(\sigma) = \sum_{z \in Z_h} G_n(z) \prod_{k \in N \setminus \{n\}} \sigma_k(z)$$

where $Z_h$ is the set of leaves reachable by playing actions in $h$. $V^G_h(\sigma)$ is the portion of agent $n$'s payoffs for playing history $h$, unscaled by $n$'s probability of playing $h$. The

Jacobian is along the lines of that of the NFG case. For histories $h$ of $n$ and $h'$ of agent $n'$, we have

$$\nabla V^G_{h,h'}(\sigma) = \sum_{z \in Z_{n,n'}} G_n(z) \prod_{k \in N \setminus \{n,n'\}} \sigma_k(z) \tag{6.5}$$

where $Z_{h,h'}$ is the set of leaves reachable by playing actions in $h$ and $h'$. The sum is over the leaves of the tree, which may be exponential in the number of agents.

### 6.3.3 Continuation for EDI-CR

For NFGs, Blum et. al provide an informal proof of Gul's lemma (Lemma 2) for the special case where the strategy profile is perfectly mixed, i.e. each pure strategy has a non-zero probability. Let $V^G_n(\sigma)$ be agent $n$'s portion of the vector $V^G(\sigma)$, and let $\Sigma_n$ be the simplex of legal profiles of $n$. $\Sigma_n$ is defined by $1'x = 1$. For a perfectly mixed profile, $V^G_n(\sigma)$ must be a scalar multiple, $c$, of the all ones vector, since at equilibrium, no pure strategy can have a higher payoff than others in the support. We believe there is a mis-statement in the informal proof given by Blum et. al whereby $V^G_n(\sigma)$ is said to be orthogonal to ${\Sigma_n}^2$. It is clear that for any vector $x \in \Sigma_n$, $V^G_n(\sigma).x = c$. To understand the origins of the proof, and to see how it extends to EFG, we went back to the original paper of Gul et. al [36] referenced by Govindan and Wilson, as well as Blum et. al. Gul dicusses the more general case where the support of a mixed strategy is allowed to be smaller than the set of all pure strategies. For this more general case, the following lemma is given:

**Lemma 3** *$\sigma$ is an equilibrium profile $\sigma_1, ..., \sigma_{|N|}$ with value $v = (v_1, ..., v_{|N|})$ iff for each player $n$ there exists $x_n \in \mathbb{R}^{m_n}$ such that*

$$V^G_n(\sigma) + x_n = v_n \mathbf{1}, \quad x_n \geq \mathbf{0}, \quad \textit{and} \quad \sigma_n.x_n = \mathbf{0}$$

[2]The first author, Ben Blum, could not be reached for verification.

where $m_n$ is the dimensionality of agent $n$'s strategy space.

To see why the above conditions are necessary and sufficient, consider an agent $n$ with 3 actions. Suppose that for a strategy profile $\sigma$, $V_n^G(\sigma) = (3, 3, 1)^\mathsf{T}$. One vector $x_n$ that satisfies the lemma is $x_n = (0, 0, 2)^\mathsf{T}$, with $v_n = 3$. If similar vectors can be found for all other agents in the game, then $\sigma$ is an equilibrium. For agent $n$, the lemma simply says that all pure strategies in the support should have equal payoffs, and any strategy that has a lower payoff, hence a non-zero entry in $x_n$, cannot be in the support, which is guaranteed by the requirement that the dot product of $x_n$ and $\sigma_n$ be 0, i.e. the corresponding entry for the inferior strategy in $\sigma_n$ must be 0. We see that if $\sigma$ is an equilibrium, $x_n$ is orthogonal to $\sigma_n$, rather than $V_n^G(\sigma)$ being orthogonal to the space $\Sigma_n$.

For the case of sequential games like games in extensive form or EDI-CR, we cannot require that for a given agent, every history with non-zero realization weight have the same payoff. We should instead require that *competing* histories have the same payoff. Two histories are competing if they cannot both be part of the support of a pure policy. As we saw in Section 3.1, because of chance outcomes, even a pure policy will have multiple histories with non-zero weight in its support. So the requirement is that if $h$ and $h'$ are both in agent $n$'s support, and they dictate different actions in a given information set, then they should have the same expected reward. Therefore, in the sequential case, instead of having one scalar value $v_n$ associated with each agent $n$ as in Lemma 3, we need multiple scalars per agent. In their work on using continuation methods to solve MAIDs [21], Blum et. al use the NFG characterization of equilibria (Lemma (2)), so we assume that they were able to devise a variant of Lemma (3) for the sequential case.

The main deciding issue in whether continuation methods are efficient for a given problem is whether there is an efficient way of computing the Jacobian in equation (6.5). Note that this calculation is done in the context of a given strategy profile. In the case of MAIDs, fixing the strategy profile results in a Bayesian network (the induced BN), and the calculations required for the Jacobian amount to doing inference on this BN. Blum et. al use the

special structure of the induced BN (e.g., a given decision node only depends on a subset of earlier decision nodes) to avoid duplicate calculations of certain probability distributions that are part of the Jacobian calculations.

In previous work, we cast EDI-CR instances as MAID [61]. However, the MAID representation of our problems has some drawbacks. The structure in our loosely coupled games (the independence of most actions' rewards and transitions) is obscured because a MAID decision node does not branch over the possible decisions, so we cannot isolate a single action and represent its dependence on another. Second, MAIDs do not naturally capture dependencies that are temporally nonlocalized, forcing us to resort to constructs that "remember" actions done in the past and allow them to affect future actions without having the latter depend on all previous decisions. A MAID representation is essentially stateless, and trying to capture a game in which agents have local state that is affected by previous actions and affects the choice of future actions is problematic.

Because of the above problems with MAID, we believe that it is much better to work on improving the Jacobian calculation while operating directly on the EDI-CR representation. Looking at equation (6.5) and how each entry is for a tuple of histories, it is clear that structured interaction implies many duplicate entries in the Jacobian, and using the idea of binning, we can avoid duplicating these calculations. Moreover, in calculating a single entry, we suspect that structured interaction can again allow us to reduce the amount of computation required.

### 6.3.4 Related Work

#### 6.3.4.1 Tracing procedure

The linear tracing procedure of Harsanyi and Selten [43] adjusts arbitrary prior beliefs into equilibrium beliefs. First, the players play best responses against identical prior beliefs concerning the play of the other players. Next, they observe that their beliefs are not met

and subsequently update their beliefs and play the best response to the new beliefs. This continues until equilibrium beliefs for the game have been found.

This general tracing procedure was implemented using homotopy methods to solve $n$-person games in normal form by Herings and van den Elzen [44] where the homotopy transforms the problem of playing a best response to the initial prior beliefs to the problem of playing a best response to opponents' actual play to form an equilibrium. Each point on the path is an equilibrium of a restricted game where the prior is played with some probability that is initially one and decreases to zero.

This work was extended by von Stengel [84] et. al to solve extensive form 2-player games. They generate a piece-wise linear path in strategy space, representing using the sequence form. The starting point again represents a prior and players adjust their behaviors based on information about the strategies that are actually being played.

#### 6.3.4.2 Logit equilibria

The tracing procedure discussed above is similar to another procedure for finding a game's equilibrium strategy profile. McKelvey and Palfrey [58, 59] define a kind of equilibrium called *Quantal Response Equilibrium* (QRE). QRE is the fixed point of the process of players choosing among strategies based on expected utility, but make choices based on a quantal choice model. This choice model accounts for variations in the choice of a player by assuming the player maximizes his utility, but the observed utility is distorted by a random additive *error*. McKelvey and Palfrey interpret mixed strategy profiles as the observed distribution of strategy choices when players' choices are modeled using the quantal choice model. If the errors are drawn independently from an extreme value distribution with parameter $\lambda$ (larger $\lambda$ mean smaller error), the form of the rule determining quantal response equilibrium choice probabilities is logistic. McKelvey and Palfrey refer to QREs arising from this kind of error distribution as *logit equilibria*.

The set of logit equilibria can be viewed as a correspondence from $\lambda$ to the set of mixed strategy profiles. At $\lambda = 0$ maps to the mixed strategy giving equal probabilities to all pure strategies. As $\lambda$ approaches infinity, the correspondence is structured to converge to a unique Nash equilibrium. McKelvey and Palfrey use homotopy to reach the logit solution.

#### 6.3.4.3 Easy initial game

Govindan and Wilson [39] modify the payoffs of the original game sufficiently so that the perturbed game has a unique equilibrium, then trace back to the original game. They establishes the conditions for application of the homotopy method for reversing the deformation of the original game to one with a unique equilibrium. They show how homotopy can be implemented with the global Newton method.

Turcoy [86] presents a homotopy approach to tracing a branch of the logit equilibrium correspondence, with application to the problem of computing a single Nash equilibrium. The logit equilibria are expressed as the zeroes of a system of equations parameterized by $\lambda$. Turcoy's algorithm is implemented in the `logit` solver in the game-theoretic software package Gambit [57].

Blum et. al solve games represented as Multi-Agent Influence Diagrams (MAIDs) [51], discussed in Section 2.3.2.1, using continuation methods. A large perturbation is applied to the rewards in the form of a bonus vector that rewards an agent for its actions regardless of anything else that happens in the game. If large enough, these bonuses dominate the original game rewards and simply determine what the optimal strategies are. The continuation method traces a path from the solution of the deformed game to that of the original. Blum et. al exploit the special structure in MAIDs to improve the efficiency of a key step in the algorithm, that of calculating the Jacobian of a certain function.

## 6.4 Summary

In this chapter, we addressed the decision-making of self-interested agents whose goal is to maximize their individual rewards, rather than team rewards. We used our model, Event-Driven Interactions with Complex Rewards (EDI-CR), to represent the loosely coupled stochastic games that arise due to structured interactions among selfish agents. As in the cooperative case, generically representing this kind of games in extensive form without regard to their special structure results in very large problems.

We discussed the use of optimization techniques to find equilibrium policies for EDI-CR games. We used an existing formulation of finding an equilibrium profile as a bilinear program and compared this approach to representing our games in extensive form and solving them using a game-theoretic software package. Analytical and experimental results show the representational and computational savings we achieved in settings with different amounts of interaction, which we obtained by introducing different communication schemes to the problem.

Finally, we discussed a second optimization-based approach where finding an equilibrium is formulated as a problem of solving a system of non-linear equations. The system of equations can be solved using continuation methods and has the advantages of allowing more than two agents. This line of work is in its early stages and raises many interesting questions for future research.

# CHAPTER 7

# CONCLUSIONS

In this chapter, we conclude the thesis with a summary of the contributions presented therein, followed by a discussion of possible directions for future work.

## 7.1 Thesis contributions

The contributions of this thesis are in the area of multi-agent decision making in settings where agents are largely independent except for some structured interactions among their decision processes. These interactions arise due to some actions of an agent having non-local effects on rewards and transitions of other agents. I consider both cooperative and self-interested decision makers. I studied the problem of representing situations with structured interactions, formulating and solving the decision making problem as an optimization problem, and reasoning about communication. The following is a list of my contributions:

**Representation for cooperative and self-interested agents**: I developed Event-Driven Interaction with Complex Rewards (EDI-CR), a decision-theoretic model for representing structured transition and reward interactions. This model advances the state of the art by filling a gap in both decision- and game-theoretic models; most existing models address different kinds of structured interactions. Those that address non-local action effects either cannot capture both reward and transition interactions, or represent independence at a coarse level where agents either fully interact or not at all. Using EDI-CR, we can represent instances that are too large to represent using existing models. EDI-CR's representational savings do not come at the expense of expressiveness; EDI-CR has the same expressive

power as DEC-MDP with factored state and local observability. Depending on the amount of interaction, experimental results show 1-2 order of magnitude reduction in problem size when using EDI-CR compared to general unstructured models.

**Solving cooperative EDI-CR using optimization techniques:** I exploited structured interaction to develop compact Mixed Integer Linear Program (MILP) formulations of EDI-CR instances. The key insight I used is that because agent interactions are structured, most action sequences of a group of agents have the same effect on a given agent. This allows us to *bin* these sequences and thus use fewer variables in the formulation. For the case of 2 agents, my formulation solves the exact problem while for 3 or more agents, the formulation represents a relaxation of the original problem. Experimental results comparing our EDI-CR MILP formulation to a formulation that ignores structure shows a significant reduction in the number of variables introduced, which translates to faster solution times. We are able to find the optimal solution, and verify its optimality, in a larger fraction of instances using our formulation. My second contribution in this area is formulating the problem of policy calculation as a continuum of problems with varying levels of difficulty and studying the use of homotopy methods to solve this continuum.

**Solving self-interested EDI-CR using optimization techniques:** For an existing formulation of calculating an equilibrium as a bilinear program, I studied the effect of changing the amount of interaction among agents on the size of the formulation and the speed of solving it. I experimentally compared this to representing the same instances as extensive form games (EFGs) and solving them using the game theoretic package Gambit. The amount of interaction is varied by varying the amount of communication the agents can have. Starting from an existing formulation of calculating an equilibrium in general sequential games as a system of nonlinear equations, I developed a formulation for EDI-CR that addresses the

structured interactions among agents.

**Communication among cooperative agents**: I exploited the structure explicitly represented by EDI-CR to make offline reasoning about communication tractable. Starting with a no-communication version of the problem, I devised heuristics that strategically choose communication decision points to add to the problem. This results in a new decision problem far smaller than what would be obtained if full-fledged communication were available, while including enough communication that allows agents to coordinate their actions and get higher rewards. My main heuristics decide whether to add a communication point based on the impact of communication at that point on the other agent. For the case of bi-directional interactions, my heuristic calculates beliefs by formulating the problem as a Bayesian Network and relegating belief calculations to an inference engine. Experimental results show that in the case with uni-directional interactions, our most sophisticated heuristic can almost achieve the full benefit of communication at a fraction of the problem size (thus solution time) of full-fledged communication. In the bi-directional case, the full-fledged communication problem is too large to solve, so we do not know what the maximum reward is, but we see that this heuristic again achieves higher reward at an overall lower computational cost than a simpler heuristic. Experiments also show that in many cases, our heuristic can determine how many communication points that are actually needed.

In spite of several attempts to get around the complexity of offline reasoning, ours is the first work to focus on making it more tractable by restricting the problem size in a way that has little or no effect on solution quality, thereby making it possible to reason about long-term consequences of communication without incurring the prohibitive costs typically associated with doing so. Also, by controlling the amount of communication introduced, our approach allows a modeler to control the tradeoff between solution quality and problem

size.

**Communication among self-interested agents:** I studied the problem of multiple agents deciding whether to communicate information when doing so is necessary to accomplish a collective task, but incurs individual costs. I modeled this situation as a sequential game of incomplete information and developed an anytime hill-climbing algorithm that finds an approximate Nash equilibrium. Our algorithm has three novel features: it collapses the game tree as a pre-processing step, resulting in more tractable trees; it generates local measures that guide the search by indicating which parts of a strategy profile are least stable; and it proposes a global measure of the stability of a profile as a whole by calculating upper bounds on players' regrets when playing this profile. Experimental results show that the pre-processing step is very effective in reducing the tree size. They also show that our search has a better anytime performance than a state-of-the-art game-theoretic package.

## 7.2 Future Directions

In the following paragraphs, we discuss some of the possible future directions of our work.

### 7.2.1 Decomposition-based optimization

A large part of the work done in this thesis uses optimization methods and available packages to solve mathematical formulations of EDI-CR instances. The optimization techniques we used try to find policies of all agents *simultaneously*. However, given that EDI-CR caters to problems where interactions among agents are relatively few, the idea of *separately* optimizing policies of different agents then somehow piecing them together is very attractive. We referred to some existing works that employ this idea, but noted that current approaches do not solve the sub-problems of the different agents in a principled manner, and are therefore prone to getting stuck in local optima.

Decomposition methods have been used very successfully in mathematical optimization [16]. The general idea is that to overcome the super-linear growth in problem complexity, a problem is broken down into a set of subproblems that can either be solved sequentially or in parallel. We identify a set of *complicating* variables; ones which, if fixed, render the subproblems independent. There are several ways of handling interactions (in the form of complicating variables) among the subproblems. In primal decomposition, an iterative solution algorithm optimizes each of the subproblems then adjusts the values of the complicating variables until convergence. In dual decomposition, we give each subproblem its local version of the complicating variables and enforce equality of the local versions.

The advantages of applying decomposition to EDI-CR are obvious. If each subproblem corresponds to an agent, we will be solving a series of MDPs rather than a single EDI-CR. So in essence, decomposing a monolithic decision process will not only be done in the representation (which EDI-CR already does), but also in the computation. Another important advantages is that all the approaches we discuss in this thesis involve centralized planning. Decomposition-based approaches allow each agent to work on its subproblem, with some supervision to direct future subproblem and integrate their solutions.

### 7.2.2 Homotopy-inspired optimization

The idea behind homotopy methods is to move along a continuum between solving an easy variant of the given problem and solving the given problem itself. These methods find a solution to the easy variant then calculate the change that needs to be made to the previous solution to make it solve a new problem that is a little closer to the original problem. As we reported in Section 3.4, homotopy methods did not work out-of-the-box for solving EDI-CR instances. However, we believe that there is a lot of potential to the *idea* of gradually transitioning from an easy to a hard problem, especially for EDI-CR where these problems have physical significance (the easy problem is a set of MDPs ignoring all

interactions and the hard problem considers all interactions). The open question is what form this gradual transition from easy to hard should take. The formulation we gave gradually increases the emphasis on interactions, but there are other possibilities, one of which is gradually increasing the number of interactions taken into consideration while planning. Note that although this is homotopy-*inspired*, it is not allowed under homotopy because of discontinuities it introduces (at one point a given interaction is not considered, at the next it is). Another possibility is alternate between adding a new interaction and smoothly increasing the weight on this interaction. The first step can use heuristics to pick interactions in the order of importance and the second step can use the usual curve-tracing methods from homotopy. We suspect that the heuristics we developed for assessing potential communication points in terms of their impact on the receiver (Section 4.5) can be useful in assessing the impact of a given interaction.

### 7.2.3 Optimization for self-interested EDI-CR

Our work presented in Chapter 6 raises many interesting questions regarding the use of optimization techniques to find equilibria in self-interested EDI-CR. First, it would be interesting to see how the idea of binning that we used for solving cooperative problems can be applied to formulations of problems involving self-interested agents. Chapter 3 showed the very good results we obtained from binning in the cooperative case, and we feel that similar advantages can be obtained in the selfish case.

Second, our discussion of formulations of finding equilibria as a system of non-linear equations is just the beginning of a line of research that can potentially yield a formulation that is both open to having more than two agents and amenable to efficient calculation using the special structure in EDI-CR. Again, the indifference of an agent to many of the details of another agent's actions can make much of the calculations in a key step of the continuation method (computing the Jacobian) redundant.

### 7.2.4 Measuring problem difficulty

In Section 4.4, we presented experimental results suggesting that it may be possible to automatically determine the number of communication points needed to achieve optimal reward. After adding a sufficient number of communication points, a good scoring heuristic will give a low score to remaining points, indicating that adding these points will not have a strong impact on reward.

One direction of future research is to investigate the use of the number of points added by a heuristic as a measure of how much coordination is needed in a given instance. If it is true that problems requiring tighter coordination are more difficult to solve, the number of added points can serve as an indicator of the degree of difficulty of a probelm.

### 7.2.5 State representation

The work in this thesis employs a state representation that stores the actions taken by an agent (in the order they were taken) and their outcomes. Where there is communication, it keeps track of sent and received messages and their timestamps as well. A received message's content and time provides information about where the other agent is and what it will do in the future. Likewise, the time at which an affected action resulted in a given outcome provides information about the probability that the affecting action was done by another agent at a given point in the past.

This verbose state representation is important in a tightly coupled DEC-MDP where an agent cares about all the future actions of another agent. However, in the case of EDI-CR, an agent only cares about the critical actions in another agent's future. And because of structured interactions, two different states are more likely to induce the same belief over the other agent's future critical actions (in which case the optimal action is the same at these states) than in a general DEC-MDP.

This observation suggests that we can obtain more a compact state representation if we discard the exact action and message history and instead keep track of an agent's belief

over the other agent's execution of future critical actions. In this alternative representation, the size of the state space will change depending on the number of interactions, unlike our current representation. More interactions result in a wider range of beliefs about the other agents.

Expressing policies as mappings from beliefs, rather than entire observation histories, to actions has been considered in the literature. In the single-agent setting, many POMDP algorithms reason over belief states. In the multi-agent setting, this approach is less common. The work of Zettlemoyer et. al [98] presents a way of calculating infinitely-nested beliefs (beliefs over what the other agents belief...etc.) about the global state at a given time. Their main concern, however, is efficient belief update given a new action and observation. They do not address planning; they assume policies for both agents are given up front.

Oliehoek et al. [68] use policies that map observation histories to actions, but they *cluster* observation histories that induce the same belief over the other agent's histories and over states, either in a lossy or lossless manner. This results in a smaller number of distinct (with respect to induced beliefs) observation histories and thus a smaller policy space. And because they do not directly deal with beliefs, they do not worry about belief updates.

**Incomplete representations:** So far, we have only been discussing representations that retain all information necessary for making optimal decisions; when we suggested forgetting certain information, it was because it doesn't not affect the optimal decision. A different approach would be to deliberately omit necessary pieces of information, resulting in sub-optimal solutions in return for a smaller state space. The question then would be which pieces of information have the least impact on solution quality. Intuitively, if, for agent $j$, a certain dependency does not affect agent $i$ that much (e.g., does not dramatically change its transition function), then $j$ need not remember exactly when the affecting action was started and both agents can just ignore this interaction. We would like to investigate whether some of the analysis approaches used in our communication heuristics can discover

such weak interactions. The result of this investigation would be another way to control the tradeoff between problem size and solution quality in addition to the one provided by limiting communication possibilities.

...

# BIBLIOGRAPHY


[1] http://www.robocuprescue.org.

[2] IBM ILOG Cplex Academic Initiative.

[3] Knitro, Ziena Optimization, Inc. http://www.ziena.com/knitro.htm.

[4] Ahuja, Kapil, Watson, Layne T., and Billups, Stephen C. Probability-one homotopy maps for mixed complementarity problems. *Journal Of Computational Optimization and Applications 41* (December 2008), 363–375.

[5] Allen, Martin. *Agent Interactions In Decentralized Environments*. PhD thesis, University of Massachusetts, Amherst, 2008.

[6] Allen, Martin, Petrik, Marek, and Zilberstein, Shlomo. Interaction structure and dimensionality reduction in decentralized mdps. In *Proceedings of the Twenty-Third National Conference on Artificial Intelligence* (Illinois, July 2008).

[7] Altman, Eitan, Avrachenkov, Konstantin, Marquez, Richard, and Miller, Gregory. Zero-sum constrained stochastic games with independent state processes. *Mathematical Methods of Operations Research 62*, 3 (2005), 375–386.

[8] Aras, Raghav, and Dutech, Alain. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research 37* (2010), 329–396.

[9] Aras, Raghav, Dutech, Alain, and Charpillet, François. Quadratic Programming for Multi-Target Tracking. In *MSDM Workshop, AAMAS 2009* (Budapest, Hungary, 2009), pp. 4–10.

[10] Becker, Raphen. *Exploiting Structure In Decentralized Markov Decision Processes*. PhD thesis, University of Massachusetts, Amherst, 2006.

[11] Becker, Raphen, Carlin, Alan, Lesser, Victor, and Zilberstein, Shlomo. Analyzing Myopic Approaches for Multi-Agent Communication. *Computational Intelligence 25*, 1 (2009), 31–50.

[12] Becker, Raphen, Lesser, Victor, and Zilberstein, Shlomo. Decentralized markov decision processes with event-driven interactions. In *Proceedings of The Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2004)* (USA, 2004), pp. 302–309.

[13] Becker, Raphen, Lesser, Victor, and Zilberstein, Shlomo. Analyzing Myopic Approaches for Multi-Agent Communication. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 05)* (Compiegne, France, September 2005), IEEE Computer Society, pp. 550–557.

[14] Becker, Raphen, Zilberstein, Shlomo, Lesser, Victor, and Goldman, Claudia V. Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research 22* (2004), 423–455.

[15] Bernstein, Daniel, R. Givan, R., Immerman, Neil, and Zilberstein, Shlomo. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research 27*, 4 (2002), 819–840.

[16] Bertsekas, D. P. *Nonlinear Programming*, second ed. Athena Scientific, 1999.

[17] Beynier, Aurélie, and Mouaddib, Abdel-Illah. A polynomial algorithm for decentralized markov decision processes with temporal constraints. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2005), ACM, pp. 963–969.

[18] Beynier, Aurélie, and Mouaddib, Abdel-Illah. Communicative opportunity cost decentralized mdps for improving agent coordination. In *Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)* (2008).

[19] Bhat, Navin A. R., and Leyton-Brown, Kevin. Computing nash equilibria of action-graph games. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence* (July 2004), University of Toronto.

[20] Blakeley, Josh A, Larson, Per Ake, and Tompa, Blank Wm. Efficiently updating materialized views. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data* (Washington, D.C., USA, 1986).

[21] Blum, Ben, Shelton, Christian R., and Koller, Daphne. A continuation method for nash equilibria in structured games. *Journal of Artificial Intelligence Research* (2006).

[22] Borkovsky, Ron N., Doraszelski, Ulrich, and Kryukov, Yaroslav. A user's guide to solving dynamic stochastic games using the homotopy method. *Operations Research 58* (2010), 1116–1132.

[23] Boutilier, Craig. Planning, learning and coordination in multiagent decision processes. In *TARK '96: Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge* (San Francisco, CA, USA, 1996), Morgan Kaufmann Publishers Inc., pp. 195–210.

[24] Cafieri, Sonia, Lee, Jon, and Liberti1, Leo. Comparison of convex relaxations of quadrilinear terms. *Ma, C., Yu, L., Zhang, D., Zhou, Z. (eds.) Global Optimization: Theory, Methods and Applications I. Lecture Notes in Decision Sciences 12(B)* (2009), 999–1005.

[25] Candan, K., Agrawal, D., W. Li, O. Po, and Hsiung, W. View invalidation for dynamic content caching in multitiered architectures. In *Proceedings of the 28$^{th}$ Very Large Data Bases Conference* (Hongkong, China, August 2002).

[26] Carlin, Alan, and Zilberstein, Shlomo. Value-based observation compression for dec-pomdps. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 501–508.

[27] Chalkiadakis, G., Robu, V., Kota, R., Rogers, A., and Jennings, N. Cooperatives of distributed energy resources for efficient virtual power plants. In *Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2011)* (Taipei, Taiwan, 2011).

[28] Choi, Chun Yi, and Luo, Qiong. Template-based runtime invalidation for database-generated web contents. In *Proceedings of Advanced Web Technologies and Applications, 6$^{th}$ Asia-Pacific Web Conference, APWeb 2004* (Hangzhou, China, 2004).

[29] Chow, Shui-Nee, Mallet-Paret, John, and Yorke, James A. Finding zeroes of maps: Homotopy methods that are constructive with probability one. *Mathematics of Computation 32*, 143 (1978), 887–899.

[30] Conitzer, Vincent, and Sandholm, Tuomas. New complexity results about nash equilibria. *Games and Economic Behavior 63*, 2 (July 2008), 621–641.

[31] Cozman, Fabio. The javabayes system. *The ISBA Bulletin 7* (2001), 16–21.

[32] Dudik, Miroslav, and Gordon, Geoffrey. A sampling-based approach to computing equilibria in succinct extensive-form games. In *Proc. 25th Conference on Uncertainty in Artificial Intelligence* (Montreal, Canada, 2009).

[33] Floudas, Christodoulos A., and Pardalos, Panos M. *Encyclopedia of Optimization, Volume 1*. Springer, 2001.

[34] Gilboa, Itzhak, and Zemel, Eitan. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior 1* (1989), 80–93.

[35] Gilpin, Andrew, and Sandholm, Tuomas. Finding equilibria in large sequential games of imperfect information. In *ACM Conference On Electronic Commerce* (2006), pp. 160–169.

[36] Gl, Faruk, Pearce, David, and Stacchetti, Ennio. A bound on the proportion of pure strategy equilibria in generic games. *Mathematics of Operations Research 18*, 3 (1993), pp. 548–552.

[37] Goldman, Claudia, and Zilberstein, Shlomo. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS)* (New York, NY, USA, 2003), ACM, pp. 137–144.

[38] Goldman, Claudia, and Zilberstein, Shlomo. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research 2* (2004), 143–174.

[39] Govindan, Srihari, and Wilson, Robert. A global newton method to compute nash equilibria. *Journal of Economic Theory 110*, 1 (May 2003), 65–86.

[40] Gupta, Ashish, Mumick, Inderpal Singh, and Subrahmanian, V. S. Maintaining views incrementally. In *Proceedings of ACM SIGMOD Conference on Management of Data* (Washington D.C., USA, 1993).

[41] Hansen, Eric A., Bernstein, Daniel S., and Zilberstein, Shlomo. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)* (2004), pp. 709–715.

[42] Harsanyi, John C. Games with incomplete information played by bayesian players, part i. the basic model. *Management Science* (November 1967), 159–182.

[43] Harsanyi, John C., and Selton, Reinhard. *A General Theory of Equilibrium Selection in Games*. MIT Press, 1988.

[44] Herings, P. Jean-Jacques, and van den Elzen, Antoon. Computation of the nash equilibrium selected by the tracing procedure in n-person games. *Games and Economic Behavior 38*, 1 (2002), 89 – 117.

[45] Howard, R. A., and Matheson, J. E. Influence diagrams. *Readings on the Principles and Applications of Decision Analysis* (1984), 721–762.

[46] Jiang, Albert, Leyton-Brown, Kevin, and Pfeffer, Avi. Temporal action-graph games: A new representation for dynamic games. In *Proc. 25th Conference on Uncertainty in Artificial Intelligence* (Montreal, Canada, 2009), pp. 268–276.

[47] Kearns, Michael, Littman, M., and Singh, S. An efficient exact algorithm for singly connected graphical games. In *Proceedings of Advances in Neural Information Processing Systems* (British Columbia, Canada, 2001).

[48] Kearns, Michael, Littman, M., and Singh, S. Graphical models for game theory. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence* (CA, USA, 2001).

[49] Kim, Yoonheui, Nair, Ranjit, Varakantham, Pradeep, Tambe, Milind, and Yokoo, Makoto. Exploiting locality of interaction in networked distributed pomdps. In *AAAI Spring Symposium on Distributed Planning and Scheduling* (2006).

[50] Koller, Daphne, Megiddo, Nimrod, and von Stengel, B. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior 14* (1996).

[51] Koller, Daphne, and Milch, B. Multi-agent influence diagrams for representing and solving games. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (2001), pp. 1027–1034.

[52] La Mura, Pierfrancesco. Game networks. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, 2000), pp. 335–343.

[53] Lemke, Carlton, and Howson, Joseph. Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics 12* (1964), 413–423.

[54] Manjhi, Amit, Gibbons, Phillip B., Ailamaki, Anastassia, Garrod, Charles, Maggs, Bruce M., Mowry, Todd C., Olston, Christopher, Tomasic, Anthony, and Yu, Haifeng. Invalidation clues for database scalability services. In *Proceedings of the 23rd International Conference on Data Engineering* (Istanbul, Turkey, April 2007).

[55] Maranas, C., and Floudas, C. Finding all solutions to nonlinearly constrained systems of equations. *Journal of Global Optimization 7* (1995), 143–182.

[56] McKelvey, Richard D., and McLennan, Andrew M. Computation of equilibria in finite games. *Handbook of Computational Economics* (1996), 87–142.

[57] McKelvey, Richard D., McLennan, Andrew M., and Turocy, Theodore. Gambit: Software tools for game theory, 2007.

[58] McKelvey, Richard D., and Palfrey, T. Quantal response equilibria for extensive form games. *Games and Economic Behavior 10* (1995), 6–38.

[59] McKelvey, Richard D., and Palfrey, T. Quantal response equilibria for extensive form games. *Experimental Economics 1* (1998), 9–41.

[60] Mostafa, Hala, and Lesser, Victor. Offline planning for communication by exploiting structured interactions in decentralized MDPs. In *2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (Italy, 2009), IEEE Computer Society, pp. 193–200.

[61] Mostafa, Hala, and Lesser, Victor. Exploiting structure to efficiently solve loosely coupled stochastic games. In *AAMAS 2010 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains* (Toronto, Canada, 2010), pp. 46–53.

[62] Mostafa, Hala, and Lesser, Victor. Compact Mathematical Programs For DEC-MDPs With Structured Agent Interactions. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)* (Barcelona, Spain, July 2011).

[63] Mostafa, Hala, Lesser, Victor, and Miklau, Gerome. Self-interested database managers playing the view maintenance game. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-Agent Systems* (Estoril, Portugal, May 2008), IFMAAS, pp. 871–878.

[64] Myerson, Roger B. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

[65] Nair, R., Tambe, M., Yokoo, M., Pynadath, D., and Marsella, S. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)* (2003), pp. 705–711.

[66] Nair, Ranjit, Tambe, Milind, Roth, Maayan, and Yokoo, Makoto. Communication for improving policy computation in distributed pomdps. In *Proceedings of The Third International Joint Conference on Autonomous Agents and Multiagent Systems* (2004), ACM Press, pp. 1098–1105.

[67] Nair, Ranjit, Varakantham, Pradeep, Tambe, Milind, and Yokoo, Makoto. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)* (2005).

[68] Oliehoek, Frans A., Whiteson, Shimon, and Spaan, Matthijs T. J. Lossless clustering of histories in decentralized POMDPs.

[69] Petrik, Marek, and Zilberstein, Shlomo. Anytime coordination using separable bilinear programs. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence* (2007), pp. 750–755.

[70] Pynadath, David V., and Tambe, Milind. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research 16* (2002), 2002.

[71] Roth, Maayan, Simmons, Reid, and Veloso, Maria Manuela. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems* (2005).

[72] Saad, Youcef, and Schultz, Martin H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing 7*, 3 (1986), 856–869.

[73] Seuken, Sven, and Zilberstein, Shlomo. Improved memory-bounded dynamic programming for decentralized pomdps. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)* (July 2007).

[74] Seuken, Sven, and Zilberstein, Shlomo. Memory-bounded dynamic programming for dec-pomdps. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)* (January 2007).

[75] Seuken, Sven, and Zilberstein, Shlomo. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems 17*, 2 (2008), 190–250.

[76] Shakshuki, Elhadi, Koo, Hsiang-Hwa, Benoit, Darcy, and Silver, Daniel. A distributed multi-agent meeting scheduler. *Journal of Computer and System Sciences 74*, 2 (2008), 279–296.

[77] Shen, Jiaying, Becker, Raphen, and Lesser, Victor. Agent interaction in distributed mdps and its implications on complexity. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2006)* (Japan, 2006), ACM, pp. 529–536.

[78] Shen, Jiaying, Lesser, Victor, and Carver, Norman. Minimizing communication cost in a distributed bayesian network using a decentralized mdp. In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003)* (Melbourne, AUS, July 2003), vol. AAMAS03, ACM Press, pp. 678–685.

[79] Sims, Mark, Mostafa, Hala, Horling, Bryan, Zhang, Haizheng, Lesser, Victor, and Corkill, Dan. Lateral and Hierarchical Partial Centralization for Distributed Coordination and Scheduling of Complex Hierarchical Task Networks.

[80] Spaan, Matthijs T. J., Gordon, Geoffrey J., and Vlassis, Nikos. Decentralized planning under uncertainty for teams of communicating agents. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)* (2006), pp. 249–256.

[81] Spaan, Matthijs T. J., Oliehoek, Frans A., and Vlassis, Nikos. Multiagent planning under uncertainty with stochastic communication delays.

[82] Spaan, Matthijs T.J., and Melo, Francisco S. Interaction-driven markov games for decentralized multiagent planning under uncertainty. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (Estoril, Portugal, 2008), pp. 525–532.

[83] Stengel, Bernhard Von. Efficient computation of behavior strategies. *Games and Economic Behavior 14* (1996), 220–246.

[84] Stengel, Bernhard Von, van den Elzen, H., and Talman, A. J. J. Tracing equilibria in extensive games by complementary pivoting. Discussion paper No. 9686, CentER for Economic Research, Tilburg University, 1996.

[85] Szer, Daniel, and Charpillet, Francois. Point-based dynamic programming for dec-pomdps. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (2006).

[86] Turocy, Theodore. A dynamic homotopy interpretation of the logistic quantal response equilibrium correspondence. *Games and Economic Behavior 51* (2006), 243–263.

[87] Varakantham, Pradeep, young Kwak, Jun, Taylor, Matthew, Marecki, Janusz, Scerri, Paul, and Tambe, Milind. Exploiting coordination locales in distributed pomdps via social model shaping. In *Proceedings of the International Conference on Automated Planning and Scheduling* (Thessaloniki, Greece, 2009).

[88] Velagapudi, Prasanna. *Distributed Planning for Large Teams*. PhD thesis, Carnegie Mellon University, 2010.

[89] Vickrey, David, and Koller, Daphne. Multi-agent algorithms for solving graphical games. In *Eighteenth national conference on Artificial intelligence* (Alberta, Canada, 2002), pp. 345–351.

[90] Watson, Layne T. Theory of globally convergent probability-one homotopies for non-linear programming. *SIAM Journal on Optimization 11* (March 2000), 761–780.

[91] Watson, Layne T., Sosonkina, Maria, Melville, Robert C., Morgan, Alexander P., and Walker, Homer F. Algorithm 777: Hompack90: a suite of fortran 90 codes

for globally convergent homotopy algorithms. *ACM Transactions on Mathematical Software (TOMS) 23* (December 1997), 514–549.

[92] Witwicki, Stefan, and Durfee, Edmund. Commitment-driven distributed joint policy search. In *Proceedings of the 6th international joint conference on Autonomous Agents and Multiagent Systems* (2007), pp. 480–487.

[93] Witwicki, Stefan, and Durfee, Edmund. Flexible approximation of structured interactions in decentralized markov decision processes. In *Proceedings of the 8th international joint conference on Autonomous Agents and Multiagent Systems* (2009), pp. 1251–1252.

[94] Witwicki, Stefan J., and Durfee, Edmund H. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS-2010)* (Toronto, Canada, May 2010), pp. 185–192.

[95] Wu, Jianhui, and Durfee, Edmund H. Mixed-integer linear programming for transition-independent decentralized MDPs. In *AAMAS 2006* (Hokkaido, Japan, 2006).

[96] Xuan, Ping, and Lesser, Victor. Incorporating Uncertainty in Agent Commitments. In *Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)* (January 1999), pp. 57–70.

[97] Xuan, Ping, and Lesser, Victor. Multi-agent policies: From centralized ones to decentralized ones. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi Agent Systems* (2002), ACM Press, pp. 1098–1105.

[98] Zettlemoyer, Luke S., Milch, Brian, and Kaelbling, Leslie Pack. Multi-agent filtering with infinitely nested beliefs. In *NIPS* (2008).

[99] Zhang, Chongjie, and Lesser, Victor. Multi-Agent Learning with Policy Prediction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence* (Atlanta, 2010), pp. 927–934.