Agent-Driven Representations, Algorithms, and Metrics for Automated Organizational Design

by

Jason Lee Sleight

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2015

For Miriam.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my adviser Ed Durfee. Throughout my graduate studies, Ed has had enormous influence not only on my research and professional development but on my life and personal growth more broadly. He consistently allowed me the freedom to pursue topics that I found engaging (though sometimes I think it was against his better judgment), and the lessons learned from these experiences have been central in my development as a researcher. Ed's advice and critical feedback have immeasurably improved the quality of my research in this dissertation, and moreover have shaped my broader approach to problem solving, critical analysis, and effectively communicating technical information.

My work has also benefited from interactions with my other committee members, and I would like to thank Amy Cohn, Victor Lesser, and Satinder Singh for their efforts and feedback throughout my graduate career. Discussions with them has helped me to analyze my research from other perspectives, and this dissertation is all the better for it. In particular, collaborations with Victor (and others at the University of Massachusetts mentioned below) during the early stages of my research were especially significant in influencing the direction of this dissertation.

While many graduate students are fortunate enough to publish in passionate research communities, I have been exceptionally fortunate in that my research bridges several distinct research communities. The alternative perspectives afforded to me by discussions both within and across these communities have been especially engaging. In particular, interactions with Dan Corkill, Frans Oliehoek, Matthijis Spaan, Prashant Doshi, Chris Amato, Shimon Whiteson, Julian Padget, Robin Cohen, Virginia Dignum, Catholijn Jonker, Chongjie Zhang, Jie Jiang, Yoonheui Kim, and Diederik Roijers have inspired me to deeply think about the relationships between my research, the organizational-research community's work, and the multiagent-sequential-reasoning community's work.

I have had the pleasure to work with numerous other graduate students and alumni, especially Jim, Stefan, Johnathon, Monica, Alexander, Chris, Lynn, Bryce, Ben, Elaine, Erik, Rob, Nan, Ananda, Shiwali, Veronica, Steve, Konstans, and Shibu;

thank you for making working at Michigan interactive and enjoyable. I would also like to say thank you to the CSE staff and administrators, including Dawn Freysinger, Dia Moulton, Karen Liska, Rita Rendell, Kimberly Mann, Kelly Cormier, Cindy Estell, and Cindy Watts for helping me to navigate the administrative aspects of my graduate career.

My parents, Ron and Jan, have played a vital role in making me into who I am today. I cannot thank you enough for providing me with opportunities to succeed and encouraging me to pursue my scientific interests from a young age. Thank you also to my siblings, aunts, uncles, cousins, and grandparents for continually pushing me to be the best and inspiring me to accept nothing less than perfection from myself.

Finally, I owe an inexpressible amount of gratitude to my wife Miriam for accomplishing this milestone together with me. It is due to her unwavering support and encouragement that I was able to complete this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABSTRACT

As cooperative multiagent systems (MASs) increase in interconnectivity, complexity, size, and longevity, coordinating the agents' reasoning and behaviors becomes increasingly difficult. One approach to address these issues is to use insights from human organizations to design structures within which the agents can more efficiently reason and interact. Generally speaking, an organization influences each agent such that, by following its respective influences, an agent can make globally-useful local decisions without having to explicitly reason about the complete joint coordination problem. For example, an organizational influence might constrain and/or inform which actions an agent performs. If these influences are well-constructed to be cohesive and correlated across the agents, then each agent is influenced into reasoning about and performing only the actions that are appropriate for its (organizationally-designated) portion of the joint coordination problem.

In this dissertation, I develop an agent-driven approach to organizations, wherein the foundation for representing and reasoning about an organization stems from the needs of the agents in the MAS. I create an organizational specification language to express the possible ways in which an organization could influence the agents' decision making processes, and leverage details from those decision processes to establish quantitative, principled metrics for organizational performance based on the expected impact that an organization will have on the agents' reasoning and behaviors.

Building upon my agent-driven organizational representations, I identify a strategy for automating the organizational design process (ODP), wherein my ODP computes a quantitative description of organizational patterns and then searches through those possible patterns to identify an (approximately) optimal set of organizational influences for the MAS. Evaluating my ODP reveals that it can create organizations that both influence the MAS into effective patterns of joint policies and also streamline the agents' decision making in a coordinate manner. Finally, I use my agent-driven approach to identify characteristics of effective abstractions over organizational influences and a heuristic strategy for converging on a good abstraction.

CHAPTER 1

Introduction

Multiagent systems (MASs) are a promising approach for addressing many complex real world problems such as: disaster response, health care management, energy distribution, transportation management, and distributed sensing, among many others. However, as such systems become increasingly large, interconnected, resource constrained, and long lived, coordinating the agents' individual actions to collectively achieve desirable global outcomes becomes increasingly difficult. Larger MASs expand the space of possible joint actions, increasing the number of collective decisions the agents can consider making. More interconnected MASs limit the effectiveness of decoupling techniques (Witwicki, 2010; Zhang et al., 2010) that exploit structure in the ways that agents can interact to make joint decision making more efficient. Tighter resource constraints magnify the importance of enacting coordinated joint policies. Long lasting systems further compound these issues by making the agents account for the long-term effects of their joint actions' trajectory.

One approach for combating these issues in real-world, human systems (e.g., corporations, governments, etc.) is the use of *organizations*, which, broadly speaking, attempt to instill long term coordination patterns so as to decompose the system's joint decision problem into more manageable components. The hope is that, if these patterns are chosen well, people can make local decisions that contribute to effective joint actions without having to explicitly consider the entire joint decision problem. As a result, a well-designed organization allows humans to efficiently operate even when scaled to many people collectively working towards a long-term objective with limited resources. Complementarily and unsurprisingly, a poorly-designed organization can be disastrous for the system, and result in mis-coordination (e.g., poor utilization of shared resources or failure to realize collectively achievable objectives) and/or excessive or insufficient coordination to make collective decisions.

In an effort to realize organizational benefits for computational agents, MAS research has investigated how organizational concepts and strategies can be modeled and utilized, and shown that organizations can increase the expected performance of large-scale, cooperative MASs (So & Durfee, 1998; Fox et al., 1998). However, as I describe more fully in Section 2.2, existing organizational research fails to provide computational representations and algorithms that can explicitly identify appropriate organizational patterns from first principles without external expert information. I argue that such computational techniques for organizational design, which are the focus of this dissertation, are vital if organizations are to be reliably deployed as a beneficial technique for multiagent coordination, since a poorly-designed organization can have adverse consequences and designing an organization by hand can be complex, time-consuming, and error-prone.

1.1 Problem Statement

While organizations for MASs have been extensively studied, there is no consensus as to what constitutes a well-defined organizational design problem. However, there are several common properties in the MAS organizational design research literature that when taken together constitute an intuitive, informal conception of what an organization is. I begin here by characterizing these core properties, and outline how they relate to an intuitive understanding of organizations. Then, I describe how these properties can be more formally grounded to provide the well-defined organizational design problem for MASs that I solve in this dissertation.

Property 1.1: ***Cooperative Multiagent System.*** Fundamentally, an organization consists of multiple agents collectively working to achieve a known, shared objective, and carries a notion of at least some cooperation among the agents. If we further assume the agents are constructed for the purpose of operating in the organization, then the agents are also fully-cooperative.[1] That is, there are **not** self-interested notions such as personal gain (e.g., preferring actions that result in more individual glory) or fairness (e.g., preferring actions that require less individual effort). I make no assumptions about agent heterogeneity, and the agents may or may not have varied capabilities.

[1]It is noteworthy that organizations have also been studied in semi-cooperative settings where agents make decisions based on a balance between organizational and personal objectives (Brooks & Durfee, 2003; Stone et al., 2010). However, in this dissertation I limit my discussion to fully-cooperative settings.

Property 1.2: ***Interdependent Agents.*** Intuitively, a primary objective of any MAS coordination mechanism, including organizations, is to facilitate good interactions among the agents. This implicitly assumes a problem domain containing inter-agent dependencies (e.g., synchronized/sequential actions and/or contention for shared resources) that the agents must collectively resolve. In MAS research, such interdependencies have been characterized as agent coupling (Witwicki & Durfee, 2011), and several methods have been developed for efficient reasoning in loosely-coupled MASs (Varakantham et al., 2009; Witwicki, 2010; Velagapudi et al., 2011; Oliehoek et al., 2013). The relationship between organizational techniques and agent coupling is complex. For loosely-coupled systems, an organization can more safely facilitate agent interactions by imposing coordination patterns since agents can already operate without extensive coordination, intuitively allowing for high performing organizations in such systems. On the other hand, tightly-coupled systems present more prevalent and/or significant opportunities for facilitating agent interactions, and thus are intuitively more interesting systems for employing organizational techniques. In this dissertation, I will not make any strong assumptions about loose versus tight coupling, but will tend to evaluate my work in relatively tightly-coupled systems since they present a more challenging domain for organizational design.

Property 1.3: ***Knowledge Limitations.*** Agents in an organization are often assumed to have some degree of individual expertise about the problem domain, but may lack knowledge of how their individual actions should (best) contribute to the rest of the system. Complementarily, a centralized perspective of the global problem domain exists (for the organizational design process) that models how the agents can interact with each other; however, this perspective may be imprecise and/or incomplete with respect to the detailed problem specifics contained in the agents' expertise. A basic motivation for an organizational approach is to provide the agents with a more globally aware, organizational context within which to exercise their local expertise.

Property 1.4: ***Coordination Overhead.*** Agents are typically assumed to have some capacity for coordinating their decisions with other agents; however, doing so incurs overhead and thus must be performed judiciously. For example, agents might be capable of communicating to share information about the domain, create joint plans, or commit to agreed-upon responsibilities, however such message exchanges consume valuable resources (e.g., power, time, or bandwidth) that could be spent on other actions. An intuitive advantage of an organization is to decrease such overhead by reducing the amount of explicit coordination that the agents require.

Property 1.5: ***Temporally Extended.*** Organizations are typically assumed to be long-lived (i.e., persist for several similar or related problem instances that the agents face in sequence), and moreover to change slowly (i.e., the same, or nearly the same, organization is used for each of the instances). This serves to provide stability and continuity for the MAS (e.g., agents can be replaced without having to completely redesign how the other agents in the MAS should reason and behave), as well as to reduce the effective costs for designing an organization since the costs can be amortized. In this dissertation, I manifest this property as an episodic problem domain (i.e., the environment is either naturally episodic or can be punctuated to artificially create episodes), where actions executed in one episode do not impact choice of actions or their effects in any other episode. While distinct, I assume episodes are similar enough to each other that sufficient long-term coordination patterns exist to warrant an organizational approach.

MASs exhibiting Properties 1.1– 1.5 can be encoded in a variety of modeling paradigms. To ground my discussions in this dissertation, I focus on the decentralized Markov decision process (Dec-MDP) (Bernstein et al., 2002), which I describe in more detail in Section 2.1.4. Briefly, the Dec-MDP is a formal mathematical framework for describing distributed, sequential decision problems in partially observable, stochastic domains.

Collectively, Properties 1.1– 1.5 provide a high-level motivation for employing an organization in a MAS. Namely, an organization for a MAS provides information to each agent about the long-term patterns of organizationally-designated policies it should typically be responsible for reasoning about and/or executing (i.e., incorporates non-local considerations into the agents' local decision problems), which enables agents to make globally-useful local decisions without having to explicitly consider the entire joint coordination problem.

This characterization leaves many questions unanswered. For example, how does an organizational designer decide which responsibilities each agent should have? To what extent should the organization impose policies on the agents versus leveraging their individual expertise? To what extent should an organization specify broad patterns that provide organizational context for agents' local decision making versus provide a collection of specific, narrow patterns for ensuring nuanced, coordinated interactions at critical junctures? These questions, among others to be addressed later, can be categorized as aspects of the organizational design problem, and fall under the overarching question of *how are (good) organizations created*?

Prior organizational design approaches, which I discuss more in Section 2.2, have centered either on employing a human to encode an organization using his/her expert knowledge to identify an organization for the problem domain, or on allowing implicitly-organized policies to emerge via experience gained by the agents repeatedly interacting with the environment and each other. Each of these approaches has limitations, for example human driven organizational design may be overly complex, error prone, and/or time consuming. Emergence of implicit organizations may require extensive interactions, have poor transient performance until the MAS's policy converges (if ever), and/or the resulting policy may be difficult for system administrators to comprehend or justify. In this dissertation, I bridge these alternative approaches, and develop an *agent-driven* approach for automated design of explicit organizations.

The main problem that I address in this dissertation is how to represent and design effective organizations for MASs via an automated computational process. A solution to this problem is an organizational design process (ODP) that satisfies the following criteria:

Principled, mathematical foundation. Reliance on external expert information to design organizations is ultimately unsustainable as MASs become increasingly large, long-lived, interconnected, distant from human experience, and populated by agents that are far from human-like, resulting in nuanced agent policies and interaction patterns that are difficult for humans to predict and organizationally encode. As such, a user of an ODP should not be expected to provide expert knowledge of appropriate organizational patterns for a MAS, but rather to provide knowledge about the MAS's characteristics that an ODP can analyze as part of creating an organization. In my dissertation, this means that an ODP should create organizations from the patterns it can identify from its Dec-MDP model of the domain, and not rely on (human-provided) domain-specific knowledge external to this model. In circumstances where a parameterized ODP could produce alternative organizations with different, non-Pareto dominating performance measures (e.g., imparted metareasoning regime, organizational abstraction level, etc.), a theoretical understanding of how parameter choices affect the resulting organization should also be provided to guide usage of the algorithm.

Flexibility to varied domain characteristics. While an ODP undoubtedly needs to recognize and exploit domain-specific information contained in the Dec-MDP model of the MAS, an ODP should **not** be reliant on the existence of specific, specialized domain characteristics (e.g., transition or reward independence, particular inter-agent influence topologies like loose-coupling or a directed acyclic influence graph, etc.). If a MAS is expressible as a Dec-MDP, then an ODP should be capable of designing an organization for that system.

Yield high-quality organizations. An ODP should create organizations that do not harm the performance of the MAS in expectation, and should typically improve the expected performance of the system.

Scalability. As implied by Properties 1.1– 1.5, organizations are assumed to be scalable in a multitude of dimensions including number of agents, agent coupling, knowledge availability/accuracy, overhead, and lifetime. Moreover, the benefits of organizations are believed to become increasingly important as these dimensions scale up (Corkill & Lander, 1998). As such, an ODP should be able to scale up to create organizations for MASs with interconnected agents that must interact for long lifetimes, where coordination incurs substantial overhead, and organizations must be created from imperfect knowledge of the domain and MAS.

1.2 Illustrating Example

To provide a more concrete example of the problems I am addressing, in this section I describe a simplified firefighting domain that I use for illustration and evaluation throughout this dissertation. Consider a grid world containing fires of various intensities/importances, and fire-fighting agents that move throughout the grid to extinguish those fires. The objective of the agents is to extinguish the fires as quickly as possible, and let us assume that only a single agent is necessary to fight a fire (e.g., an agent represents a fire brigade), implying the optimal joint policy is for agents to spread out and fight fires in parallel. Additionally, suppose that grid-cell delays cause movement throughout the grid to be non-deterministic, for example due to traffic or debris that stochastically impedes an agent's attempts to move into a grid cell. Finally, the world is episodic, where the agents begin each episode at a fixed location (e.g., their respective firehouses), then move throughout the grid and extinguish a set of fires before returning to their initial locations. In each episode, the agents are facing the same high-level problem (i.e., fight a set of fires), but the specifics of each episode may differ, for example fires may be in different locations with different intensities and the cell delays may be different. Figure 1.1 illustrates several example problem episodes.

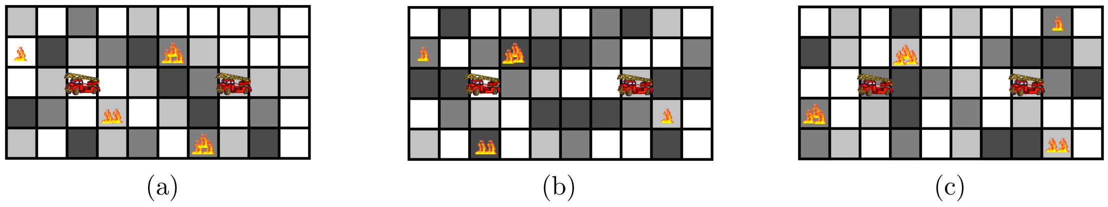


Figure 1.1: Example initial states for three different problem episodes. Darker cell shading indicates higher cell delays.

If each agent had full awareness of the global state and the other agents' capabilities, then the agents could each determine the optimal joint policy (though this could be prohibitively expensive to compute in some cases), and consequently also know the respective local actions that they should perform as part of that joint policy. However, a more practical assumption is that the agents are not inherently fully aware of the global state or each others' capabilities, but rather can only obtain such non-local information if they exchange messages. Intuitively, communicating the agents' capabilities for the full range of possible global states and/or maintaining global state awareness as the state stochastically changes over time could require an extensive amount of information being exchanged between the agents, which consumes valuable resources that could otherwise be spent fighting fires.

To see the advantages that an organization can provide for such a MAS, consider the simple organization in Figure 1.2a that designates partitioned regions of responsibility for each agent to fight fires within. If each agent focuses on fighting the fires in its organizationally-designated region, then the agents no longer need to know any non-local information or to (further) coordinate their local policies with other agents. It is non-trivial, however, to decide if this region-partitioning organization is appropriate for the problem domain. For example, notice that this region-partitioning organization will tend to work well if the fires are uniformly distributed among the partitions, but since the organization will persist across all episodes,[2] could be overly restrictive in other episodes, for example the episode in Figure 1.1b where by chance the fires are skewed to one partition.

Stepping back, how did an ODP come up with the region-partitioning organization

[2] Alternatively, a MAS could have different organizations for different episodes, alongside rules for multiplexing these alternative organizations, and indeed this strategy makes sense in many cases. (I revisit this notion in Section 6.2.1). Conceptually, however, such a multiplexing strategy is semantically identical to an overarching organization with several sub-components, each with interaction patterns tailored for sub-classes of environmental conditions. For simplicity and clarity in my discussions, I will treat multiplexed organizations as a single, overarching organization unless otherwise noted.

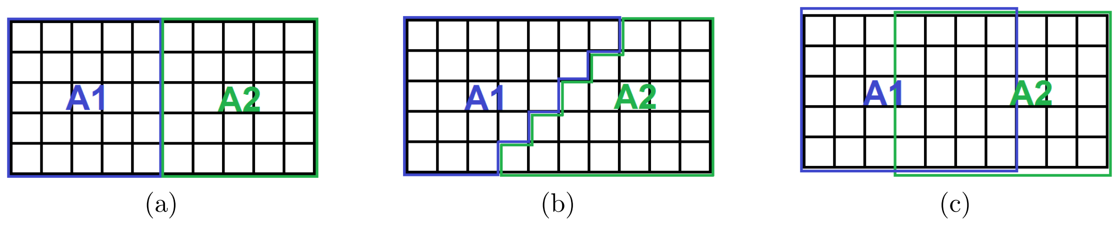


Figure 1.2: Example organizations that designate regions of responsibility for each agent to focus on fighting fires within.

in the first place? Why not partition the grid differently (e.g., Figure 1.2b), or have overlapping regions of responsibility to decrease the likelihood of mismatched episodes (e.g., Figure 1.2c), or time varying regions (confusing to represent on a spatial graphic, but see Section 5.4 for examples of these kinds of organizations), or fuzzy boundaries to designate regions of responsibility (again confusing to represent graphically, but see Section 3.5.3 for an example)? Indeed, each of these alternative examples could be the best organization for some (subset of) problem episodes.

The objective of the organizational design problem laid out in Section 1.1, and the focus of this dissertation, is to identify the (approximately) optimal organization for a given MAS from the first principles about the expected domain (as captured in the ODP's Dec-MDP model).

1.3 Solution Approach

Prior approaches to explicitly solving the organizational design problem have taken a problem-driven stance (see Section 2.2.2 for more details). That is, these approaches begin by designing an organization (externally to any specific MAS) to solve a problem, and then populate the organization with agents to embody that organization. In contrast, my approach is agent-driven, where I begin by assuming a specific MAS is already in place and then create an organization to optimize and streamline that MAS. As we will see throughout this dissertation, the fundamental advantage of my agent-driven approach over a problem-driven one is that decisions about the organizational design can be grounded in the expected impact that those decisions will have on the MAS. This mathematical basis provides opportunities for a deeper, theoretical comprehension of how organizations relate to MASs, establishes a basis for evaluating the effectiveness of an organization, and anchors computational algorithms for autonomous organizational design.

My agent-driven approach has far-reaching consequences, and affects issues ranging

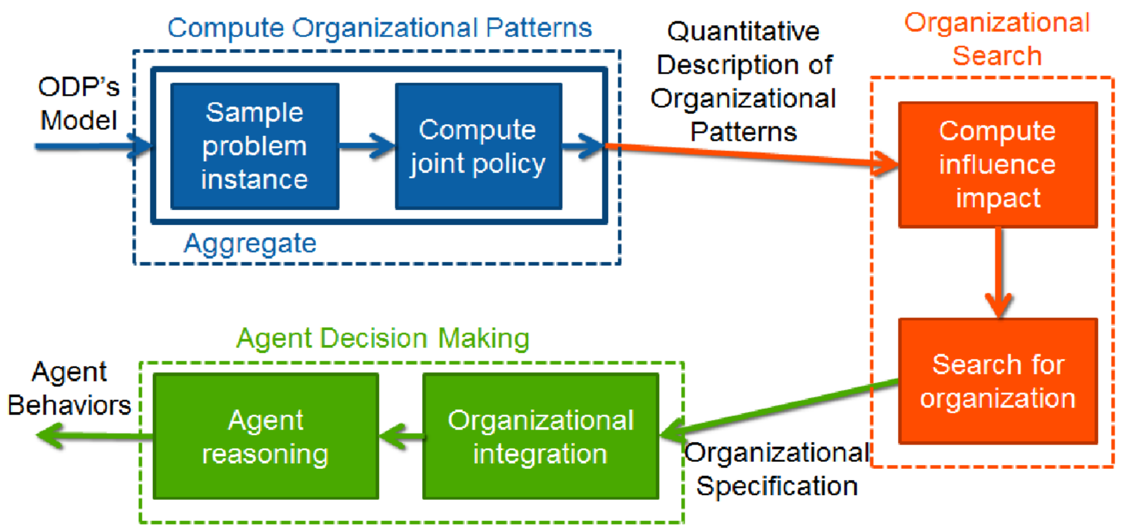


Figure 1.3: Overview of how my ODP operates and interacts with the MAS.

from the language for specifying an organization to agents in a MAS, to heuristics for guiding organizational design (e.g., either for a human designer or for setting parameters of an automated ODP), to how an autonomous ODP can create an appropriate organization, among others to be discussed later. In this dissertation I focus on three primary areas as depicted in Figure 1.3 (see Section 6.2 for a discussion and some preliminary results of how my approach could affect other points of interest in organizational and MAS research).

Agent Decision Making. Specifying an organization to a MAS such that the agents can each unambiguously understand what is organizationally expected of them is a complex problem (see Section 2.2.2 for an overview of prior work in this area). Moreover, the agents must be capable of integrating the organizational directives into their native, local reasoning processes. Leveraging my agent-driven approach, however, solves these issues at a fundamental level. Rather than construct a language by identifying important and/or useful organizational constructs and then building up middleware that permits agents to understand and integrate their organization into their local reasoning (as is typical in prior work), I instead begin by committing to a particular agent reasoning framework (Dec-MDP agents in this dissertation) and then derive the organizational constructs that are natively and unambiguously expressible in that framework. That is, my organizational specification language is directly derived from the agents' internal reasoning framework, which allows agents to naturally understand and integrate the organizational directives into their local reasoning.

Organizational Search. As alluded in Section 1.1, an ODP's objective is to create an organization by identifying appropriate patterns of agent actions and interactions and then encoding those patterns within the target organizational specification language. My ODP achieves this by searching through a space of *organizational influences*, which are the building blocks that comprise an organizational specification. Broadly speaking (formal definitions follow in Chapter 3), an organizational influence is a modification to an agent's local reasoning process For example in Dec-MDP agents, an organizational influence could prohibit an agent from considering an action in some state(s), augment an agent's reward function to incentivize organizationally-desired policies, etc. Thus, my ODP's objective is to identify a set of organizational influences to specify that will guide the agents into appropriate patterns of agent actions and interactions.

Leveraging my agent-driven approach, an ODP can measure the impact that a (set of) influences will have on the MAS, and use these measurements to inform a search over the organizational influence space. Intuitively, the optimal organization is then defined as the set of organizational influences with the optimal expected impact to the MAS; unsurprisingly however, it is computationally intractable to exhaustively search through the combinations of organizational influences. As such, an important part of my work in this dissertation is the identification of more efficient techniques for identifying an approximately optimal set of organizational influences.

Compute Organizational Patterns. My organizational search process just described relies extensively on statistical estimates of the impact that organizational influences are expected to have on the MAS. More broadly, for a computational ODP to make intelligent decisions about the long-term patterns it should encode as an organization for a MAS, it must have a sense as to how well each of the agents' various policies are expected to perform. In principle, an ODP could acquire this information from any of several sources, such as from a system administrator (e.g., an expert human that has general ideas of how the MAS should act) or from the agents themselves (e.g., if the MAS has been acting in the environment for some time already); however, leveraging my agent-driven approach for organizational design, in this dissertation I develop techniques by which an ODP can self-determine a quantitative description of organizational patterns.

Given that the ODP has a Dec-MDP model of the MAS, computing a quantitative description of organizational patterns can be done by completely solving the Dec-MDP for the MAS's optimal joint policy, which is a computationally intensive task, especially for MASs exhibiting Properties 1.1– 1.5. To mitigate the computational intractability of directly computing the MAS's complete, optimal joint policy, I adopt a Monte Carlo sampling approach to empirically estimate organizational patterns. Specifically, the ODP samples problem episodes from its Dec-MDP model, and for each computes the optimal joint policy for that episode's reachable state space, before finally aggregating across the samples to form an estimate of optimal organizational patterns. In this way, the ODP can compute a statistically-stable, quantitative description of organizational patterns, which the ODP can then use to inform its search of the organizational influence space.

1.4 Contributions

Fundamentally, the overarching question of my dissertation is *how is an organization created for a MAS*? In answering this question, I develop both novel representational strategies for describing organizations in MASs and solution techniques for deciding on an appropriate organization for a MAS. In what follows, I highlight the most significant contributions of my work, describe how each is evaluated, and discuss the implications of my results to further extensions by researchers and/or adoption by practitioners.

Agent-driven Organizational Specification Languages. From my agent-driven approach, a principled organizational specification language can be derived from the agents' native reasoning representations and processes. Fundamentally, the advantage of constructing a specification language this way is that the agents can unambiguously, directly integrate their organization into their local decision processes. Additionally, since the agents (presumably) have a finite vocabulary of native representational constructs (e.g., states, initial states, actions, rewards, transitions, and time horizon constructs for MDP-based agents), an agent-driven approach allows for formal analysis of a specification language's necessity and completeness, and also establishes a well-defined organizational design space.

I evaluate this methodology by demonstrating its application to Dec-MDP agents (Chapter 3). Stepping through my methodology, I first use the Dec-MDP decision framework to enumerate the constructs of my organizational specification language, and then use the mathematics of the Dec-MDP framework to formally prove the necessity and completeness of my language. While the semantics associated with each construct of my specification language are precisely defined by the underlying theory of the Dec-MDP framework, I additionally perform an empirical evaluation for each language construct to demonstrate the practical applicability of my agent-driven organizational specification language.

My methodology could be used by other researchers studying organizations to construct agent-grounded specification languages for other agent reasoning frameworks. Additionally, by demonstrating my methodology on Dec-MDP agents, I provide a language for practitioners of Dec-MDPs to incorporate organizations into their systems as well as a launching point for further study of organizations in Dec-MDPs.

Principled, Quantitative Metrics for Organizational Performance. A significant advantage of my agent-driven approach is that it enables quantitative measurements of organizational performance stemming from how the MAS is expected to actually perform when using the organization. As I describe in Section 2.2.2, prior, problem-driven approaches to organizational design do not have a systematic way to measure organizational performance, which makes selecting an appropriate or optimal organization rather *ad hoc*. From my agent-driven perspective, however, organizational performance is directly defined by the MAS's performance when using the organization, thus providing a mathematically-sound foundation for reasoning about alternative organizations. Consequently, I identify that an organization not only impacts the quality of the MAS's joint policy, but also the amount of computation required for the agents to make decisions; that is, an organization can trade off between the agents' policy quality and their computational costs, a property often included as part of metareasoning decisions (see Section 4.3.1 for an overview of prior research on metareasoning). Leveraging my agent-driven approach, I quantify the effects of such metareasoning decisions to provide parameterized metrics of organizational performance (Section 4.3).

I evaluate the advantages of my agent-driven, quantitative performance metrics by developing an ODP based on them, and evaluating this ODP's effectiveness over a space of parameters as previously described in the problem statement (Section 1.1). My ODP evaluation shows that my organizational performance metrics are predictive of the actual performance of the MAS using the organization, and that the ODP is able to leverage these quantitative metrics to design effective organizations for the MAS.

The performance metrics I identify, and my agent-driven approach for identifying them more broadly, could be used by other researchers studying organizations to provide a principled, quantitative foundation for further study of organizational design techniques, thereby allowing for informed, rational decision-making about organizational designs. Possible directions for future research include further study of the ODP beyond what I describe in the scope of this dissertation (see Section 6.2), such as organizational adaptations (e.g., in response to shifting or unexpected environmental conditions), or hierarchical nesting of organizations.

Heuristics for Guiding Organizational Selection. Creating an organization is a challenging task, regardless of whether the ODP is performed by a computational algorithm or an expert human. To both broaden my results' accessibility to problem-driven organizational approaches, as well as guide usage of automated ODPs, I formulate heuristic guidelines for selecting an appropriate organization for a MAS. Specifically, I identify that an organization should focus on patterns of the agents' joint interactions rather than how an agent should execute its local components of those joint actions (Section 4.1), and that task-delineated abstractions are a good mechanism to include in an ODP's reasoning about which organizational influences to specify (Section 5.5).

Leveraging my agent-driven approach, I analyze these heuristics to identify their theoretical foundations. In addition, I demonstrate the heuristics' effectiveness empirically over a space of environmental parameters, and find that in expectation they yield organizations do not harm the MAS's performance, but that they perform significantly better under certain environmental parameterizations, e.g., when agents have meaningful local expertise (Property 1.3).

Beyond informing usage of my ODP algorithm, these heuristics could be used to inform usage of existing, problem-driven organizational approaches, for example, to delimit and/or focus a human designer's attention to the aspects of an organization that are most important to explicitly reason about. Additionally, MAS practitioners can use these heuristics to identify effective ways for organizing a MAS as a less-demanding alternative to a complete, computationally-intensive ODP.

Techniques for Automated Organizational Design. Finally, to provide a more comprehensive solution for the organizational design problem than heuristics, I develop representations and algorithms for an automated ODP (Chapters 4 and 5). In doing so, I develop novel strategies for efficiently estimating an organizational influence's incremental impact, and embed these statistical estimates in an incremental search algorithm of the organizational influence space. I also develop a novel framework for analyzing how the abstraction level of the organizational influence space affects the

ODP's search algorithm and the resulting performance of an organization created by my ODP.

I evaluate my ODP using the solution criteria previously set forth in Section 1.1. I empirically evaluate both the ODP's effectiveness as well as the performance of the organizations the ODP creates, and when meaningful, additionally provide theoretical analysis of my ODP's characteristics (e.g., computational complexity). Briefly, I find that my ODP is able to efficiently identify an approximately optimal organization for a MAS, where such an organization exploits structure in the domain to achieve performance significantly better than the baseline MAS as well as better than the best hand-tailored organizations that I create using my organizational selection heuristics.

As is, my ODP can be used to create organizations for MASs represented as Dec-MDPs, which Dec-MDP practitioners could use to organize their systems. Moreover, my results here serve as a springboard for further study of organizational reasoning techniques, providing not only a starting point for further research (e.g., see Section 6.2) but also a performance benchmark against which to compare alternative organizational design approaches.

CHAPTER 2

Background

The research I present in this dissertation straddles and builds upon ideas from multiple areas of prior work that have traditionally been pursued by disparate research communities. It is therefore unsurprising that a comprehensive discussion of all background work relevant for my research would fill many volumes of books and is well beyond the scope of my dissertation. In this chapter, I overview the most pertinent areas of prior work so that the reader can better understand how my contributions and approach are situated with respect to the existing research literature. When more detailed understanding is necessary for comprehending my representational and algorithmic contributions, I delve more deeply into the necessary technical material.

The rest of this chapter is structured into two major components, operational decision making (Section 2.1) and organizational decision making (Section 2.2). Broadly speaking, operational decision making focuses on the representations and algorithms that agents use to reason about their local and/or joint decisions within a single problem episode. That is, given a specific decision problem, how does an agent plan its actions and coordinate with other agents so that the MAS can operate (approximately) optimally for the currently encountered episode. Complementarily, organizational decision making focuses on the representations and algorithms that the MAS and/or ODP use to reason about the long-term coordination patterns exhibited across problem episodes. That is, given a distribution of expected decision problems, how should the MAS be organized so as to streamline the agents' operational reasoning and promote effective interactions across the space of episodes the agents are likely to encounter.

2.1 Operational Decision Making

As mentioned in Section 1.3, I ground my agent-driven approach to organizational design by first committing to a particular agent reasoning framework. In this dissertation, I have elected to adopt the Dec-MDP modeling paradigm, which falls under the broader class of decision-theoretic models. Decision-theoretic models are a widely studied and commonly adopted paradigm within the research community due to their principled mathematical foundation, and their inclusion of expressively-powerful modeling constucts like uncertainty, utility, and sequential reasoning. In this section, I provide a description of the decision-theoretic modeling techniques and concepts that are necessary to understand the organizational representations and algorithms that I develop throughout this dissertation.

I begin with the single agent decision-theoretic model my agents utilize, namely the Markov decision process (MDP) in Section 2.1.1, and then discuss a related model that incorporates partial observability, the partially observable Markov decision process (POMDP) in Section 2.1.2. While the basic MDP and POMDP models provide the formal basis for my agents' local reasoning, my methods also make heavy use of factoring, a concept that extends the basic models by decomposing the representation of model components into conditionally independent factors, which provides specification compactness and computational efficiency (Section 2.1.3). I then turn to the cooperative multiagent case, the decentralized (partially observable) Markov decision process (Dec-(PO)MDP) in Section 2.1.4, which provides a multiagent foundation for decision-theoretic models. To mitigate computational intractability, my algorithms incorporate hierarchical abstractions into the decision-theoretic models (Section 2.1.5), which allows agents to reason about tasks to accomplish instead of primitive actions to perform.

Formal definitions will be provided in the sections that follow, but generally speaking, the idea behind decision-theoretic models is that there exists a problem *environment* that is currently in some *state*. The agent obtains information about the current state by making an *observation*, and must make a decision about which *action* it should execute. The agent decides which action to execute based upon a function that associates a *reward* with each state, where the agent's objective is to maximize its expected total reward.[1] After the agent executes an action, the environment *transitions* to a (potentially) new state according to its (stochastic) dynamics and the agent's action. This process (i.e., the agent making an observation, then executing an action, and then the environment transitioning) repeats for a predetermined, finite number of iterations.

[1] In this dissertation, I only consider finite horizon models, and thus the expected total reward is the agent's optimization objective. More generally speaking, decision-theoretic models can be infinite horizon, in which case there would be a discount factor instead of a time horizon, and the agent's objective would be to optimize the expected discounted reward.

2.1.1 Markov Decision Process

The Markov decision process (MDP) (Bellman, 1957) is a mathematical framework for sequential decision making for a single agent that incorporates rewards as well as transition uncertainty. More formally, a MDP is defined as follows.

Definition 2.1. *A* ***finite horizon MDP*** *is given by the tuple* $\mathcal{M} = \langle S, \alpha, A, P, R, T \rangle$*, where:*

- S *is the finite set of possible states in the environment. For notational clarity, I will sometimes use superscripts to denote the decision point at which a state is encountered, for example,* s^t *is state* s *at the* t*-th decision point.*

- $\alpha : S \mapsto [0, 1]$ *is the initial state distribution, where* $\alpha(s^0)$ *specifies the probability that the environment will initially begin in state* $s^0 \in S$*.*

- A *is the finite set of possible actions. Each state may have a different (sub)set of available actions, and* $A_{s^t} \subseteq A$ *represents the actions available to the agent in state* $s^t \in S$*. I will be notationally explicit in this initial definition, but neglect this notational nuance in subsequent discussions to reduce notational clutter (though the set of actions available in a state will still be a function of that state).*

- $P : S \times A_S \times S \mapsto [0, 1]$ *is the transition function, where* $P(s^{t+1}|s^t, a)$ *denotes the probability of the environment transitioning to state* $s^{t+1} \in S$ *after the agent executes action* $a \in A_{s^t}$ *in state* $s^t \in S$*.*

- $R : S \times A_S \times S \mapsto \mathbb{R}$ *is the reward function, where* $R(s^t, a, s^{t+1})$ *denotes the immediate reward the agent associates with executing action* $a \in A_{s^t}$ *in state* $s^t \in S$ *and the environment transitioning to state* $s^{t+1} \in S$*.*

- $T \in \mathbb{N}$ *is the finite time horizon and specifies the number of decision points, after which the problem terminates.*

An important property for computational tractability in an MDP is the **Markov property**, which asserts that the transition dynamics, $P(s^{t+1}|s^t, a)$, and reward, $R(s^t, a, s^{t+1})$ are conditionally independent of any other past states, actions, transitions, or rewards, given the current state.

Consider a single agent version of the firefighting domain from Section 1.2 (I present a multiagent version in Section 2.1.4), where a firefighting agent and fires to be fought exist in a grid world with C cells (Figure 2.1). The (factored) environment state representation captures: the system time, $t \in \mathbb{N}$; the location of the agent, $\ell \in C$;

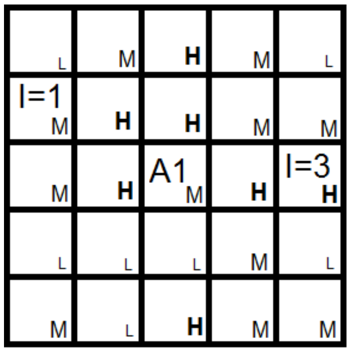


Figure 2.1: Example state of a 5×5 firefighting grid world. A1 designates the cell corresponding to the location of the agent, and $I = x$ indicates that there is a fire in that cell with intensity x. Letters designate a (H)igh, (M)edium, or (L)ow delay in that cell.

the fire intensity, $I_c \in \mathbb{N}$ for each cell c; and a delay, $\delta_c \in [0, 1]$ for each cell c, which stochastically prevents movement into that cell with probability δ_c. Figure 2.1 shows an example environment state, with the location of the agent, along with the intensity of fire in the two cells with $I_c > 0$. For illustration, suppose (H)igh, (M)edium, and (L)ow delay in Figure 2.1 correspond to δ_c equal to $0.8, 0.5,$ and 0.0 respectively. For simplicity (this will be relaxed in the multiagent case), suppose the agent can precisely observe exactly which state the environment is in. The agent has six actions: a NOOP action that makes no change to the environment state except to increment the time by one (the other five actions also increment the time by one); four possible movement actions, (N)orth, (S)outh, (E)ast, and (W)est, that each stochastically move the agent one cell in the specified direction (into cell c_dest) with probability $1 - \delta_{c\_dest}$, and equates to a NOOP with probability δ_{c_dest} (or if there is no cell in that direction); and a fight-fire (FF) action that decrements by one the intensity of the agent's current cell (to a minimum of zero) and otherwise behaves like a NOOP. The agent executes actions for a predetermined number of steps, T. Suppose the reward associated with a state is $-\sum_c I_c$.

The agent's objective in a MDP is to maximize its expected total reward, which is recursively defined by the Bellman equation.

$$Q^*(s^t, a) \equiv \sum_{s^{t+1} \in S} P(s^{t+1}|s^t, a) \left[R(s^t, a, s^{t+1}) + \max_{a' \in A} Q^*(s^{t+1}, a') \right] \tag{2.1}$$

$Q^*(s^t, a)$ designates the expected total reward of executing action $a \in A$ in state $s^t \in S$ and then behaving optimally (i.e., executing actions that maximize expected total reward) from that point onward (i.e., the next $T - t - 1$ decision points).

An agent's plan for how it will act in each state is described as a **policy**, $\pi : S \times A \mapsto [0, 1]$, where $\pi(s^t, a)$ specifies the probability that the agent will execute action $a \in A$ in state $s^t \in S$. Reformulating Equation 2.1 in terms of π yields $Q^\pi(s^t, a)$, the expected total reward of executing action $a \in A$ in state $s^t \in S$ and then following policy π from that point onward:

$$Q^\pi(s^t, a) \equiv \sum_{s^{t+1} \in S} P(s^{t+1}|s^t, a) \left[R(s^t, a, s^{t+1}) + \sum_{a' \in A} \pi(s^{t+1}, a') Q^\pi(s^{t+1}, a') \right] \tag{2.2}$$

With this formulation, the agent's objective is restated as finding an **optimal policy** π^* in its policy space Π that maximizes the expected total reward.

$$\pi^* \equiv \arg\max_{\pi \in \Pi} \sum_{s^0 \in S} \alpha(s^0) \sum_{a \in A} \pi(s^0, a) Q^\pi(s^0, a) \tag{2.3}$$

The optimal policy is also referred to as the *solution* to the MDP, and similarly, *solving a MDP* refers to the process of computing π^*.

Several alternative algorithms exist for solving MDPs, the best of which have P-complete computational complexity in the number of states, actions, and representational size (Papadimitriou & Tsitsiklis, 1987). Throughout this proposal, I will refer to the computational cost of solving for π as $C(\pi)$. Notable algorithms for solving a MDP include value and policy iteration (Russell & Norvig, 2009) which iteratively apply the Bellman equation to gradually converge on π^*, as well as linear programming, which I utilize throughout this dissertation. In particular, I utilize a variation of the linear program as formulated by Kallenberg (1983), which frames the policy creation process as a linear optimization problem:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{s \in S} \sum_{a \in A} x(s, a) \sum_{s' \in S} P(s'|s, a) R(s, a, s') \\ & \left| \begin{array}{l} \forall s' \in S, \sum_{a' \in A} x(s', a') - \sum_{s \in S} \sum_{a \in A} x(s, a) P(s'|s, a) = \alpha(s') \\ \forall s \in S, \forall a \in A, x(s, a) \geq 0 \end{array} \right. \end{aligned} \tag{2.4}$$

where the vector of variables $\mathbf{x}$ is referred to as the *occupation measures*, and $x(s, a)$ denotes the total expected number of times that action $a \in A$ is performed in state $s \in S$. Further, if the state space is non-recurrent, that is, upon transitioning from any state $s \in S$ it is impossible to ever transition back to that state (e.g., in the firefighting domain, system time is included within the state representation, and since time increments with every action, the state space is non-recurrent), then $x(s,a)$ is equal to the probability of reaching state $s \in S$ and executing action $a \in A$. Upon solving this linear program, the optimal policy can be directly computed from the optimal occupancy measures, $\mathbf{x}^*$:

$$\pi^*(s,a) = \frac{x^*(s,a)}{\sum\limits_{a' \in A} x^*(s,a')} \tag{2.5}$$

Note that occupancy measures are effectively an alternative representation for a policy, and one can easily convert between policies and occupancy measures using Equation 2.5.

2.1.2 Partially Observable Markov Decision Process

In the MDP framework, it is assumed that the environment is *fully observable* to the agent, that is, the agent makes an observation that informs it of exactly which state the environment is in. However, in many domains, the environment is only *partially observable* to the agent, that is, the agent makes an observation that might only partially inform it of which state the environment is in. Thus the agent might have uncertainty about the actual environment state, and the agent must reason about its beliefs of the environment's actual state. This type of uncertainty is formally represented within the partially observable Markov decision process (POMDP), which is defined as follows.

Definition 2.2. *A* ***finite horizon POMDP*** *is defined by the tuple* $\mathcal{M} = \langle S, \alpha, A, P, R, \Omega, O, T \rangle$*, where:*

- *S is the state space, α is the initial state distribution, A is the action space, P is the transition function, R is the reward function, and T is the time horizon, exactly as they were defined in the fully observable MDP (Definition 2.1).*
- *Ω is the finite set of observations the agent could receive. As with actions, each state may have a subset of possible observations the agent could receive, $\Omega_s \subseteq \Omega$. I will be notationally explicit in this initial definition, but neglect it in subsequent discussions to reduce notational clutter.*
- *$O : S \times A_S \times S \times \Omega_S \mapsto [0,1]$ is the observation function, where $O(o|s^t, a, s^{t+1})$ specifies the probability that the agent makes observation $o \in \Omega_{s^{t+1}}$ after executing action $a \in A_{s^t}$ in state $s^t \in S$, and the environment arriving in state $s^{t+1} \in S$.*

Note that the MDP is a special case of the POMDP where the agent's observation uniquely determines which state the environment is in, $\forall o \in \Omega, \exists s \in S, Pr(s|o) = 1$.

Although the state space for a POMDP still possesses the Markov property, the agent is unable to fully exploit this property since it does not necessarily know the actual state of the environment. Rather, the agent knows only observations that inform it of which state(s) the environment could potentially be in, implying the *history* vectors of past observations and actions are relevant for determining which action the agent should execute next. Consequently, a policy for a POMDP is defined as $\pi : \vec{\Omega} \times \vec{A} \times A \mapsto [0, 1]$, where $\vec{\Omega}$ and $\vec{A}$ are the history vector spaces of past observations and actions respectively, and $\pi(\vec{o}, \vec{a}, a)$ specifies the probability the agent will execute action $a \in A$ when its observation and action histories are $\vec{o} \in \vec{\Omega}$ and $\vec{a} \in \vec{A}$ respectively.

The above policy definition is problematic from a computational perspective because the policy space increases exponentially as the history of observations and actions grows. To combat this issue, prior research has identified that the agent's probability distribution over states is a sufficient statistic to capture its knowledge of which state(s) the environment might be in (Smallwood & Sondik, 1973). This distribution over states is referred to as the agent's *belief state*, and is comprised of a vector $\mathbf{b}$ containing the probability of each $s \in S$ being the environment's current state. Let $b(s) = Pr(s|\vec{a}, \vec{o})$ refer to $\mathbf{b}$'s estimate of the probability that state $s \in S$ is the current environment state, and let $\mathbf{b}$ be initialized to α to reflect the initial uncertainty about the environment state. Smallwood & Sondik (1973) also identified a means for an agent to update its belief state using the following update rule:

$$b^{t+1}(s^{t+1}) = \frac{O(o|s^t, a, s^{t+1}) \sum_{s^t} P(s^{t+1}|s^t, a) b^t(s^t)}{\text{normalizing factor}} \tag{2.6}$$

It is important to note that constructing belief states this way makes them display the Markov property, and moreover Equation 2.6 defines a transition function over the belief state space. This means any POMDP to be reduced to an equivalent, belief-state MDP, where: the state space is the belief-state space; the initial state distribution is the singleton $\mathbf{b}$ initialized to the POMDP's α; the transition function is Equation 2.6; the reward function is the expected reward given the belief distribution; and the action space and time horizon are identical to the respective POMDP components. The belief-state MDP can then be analyzed and solved using any of the techniques from Section 2.1.1.

2.1.3 Factored Markov Decision Process

In the formulation of MDPs given in Section 2.1.1, states are represented as atomic objects, and then the other MDP model components are defined in terms of these atoms. The basic MDP is thus simple and easy to describe, but has the notable limitation that, as the size and/or complexity of the model increases, representing the MDP becomes computationally expensive (or even infeasible). In many domains, however, the environment consists of several state **factors**, which together constitute the state of the environment, as opposed to opaque, atomic states. Moreover, these factors often exhibit structure such as conditional independence between factors that provides opportunities to more compactly represent the model. For example, the firefighting domain previously described was naturally defined in terms of a factored state representation (e.g., system time, the location of the agent, the intensity of fire in each cell, etc.). Notice that some of these factors are conditionally independent; for example, the intensity in a cell is conditionally independent of the cell delays given the agent's location.

Factored MDPs (Boutilier et al., 2000; Guestrin et al., 2003) were developed to explicitly exploit factored structure, and are defined in what follows. While I present only the definition for a factored MDP, an analogous factored POMDP definition also exists by additionally factoring the observation set, Ω, and observation function, O.

Definition 2.3. *A **factored MDP** is given by the tuple $\mathcal{M} = \langle S, \alpha, A, P, R, T \rangle$ as in an unfactored MDP (Definition 2.1), but now the components are factored to exploit the structure of the domain:*

- $S = F_1 \times F_2 \times \cdots \times F_{m_S}$ *where F_j is the finite domain of state factor j and there are m_S factors.*

- $\alpha = \alpha_1 \times \alpha_2 \times \cdots \times \alpha_{m_\alpha}$ *where $\alpha_j : (\otimes_k F_k) \mapsto [0, 1]$ is the jth factor of the initial state distribution (out of m_α factors), and $\{F_k\}$ is partitioned across the α_j's.*

- *A is the finite set of possible actions. As before, each state may have a different (sub)set of available actions, and $A_{s^t} \subseteq A$ represents the actions available to the agent in state $s^t \in S$. I will be notationally explicit in this initial definition, but neglect it in subsequent discussions to reduce notational clutter.*

- $P = P_1 \times P_2 \times \cdots \times P_{m_P}$ *where $P_j : (\otimes_k F_k) \times A_S \times (\otimes_{k'} F_{k'}) \mapsto [0, 1]$ is the jth factor of the transition function (out of m_P factors), and $\{F_{k'}\}$ is partitioned across the target of the P_j's. $\{F_k\}$ need not be partitioned across the source of* *the P_j's (i.e., a state factor may contribute to the transition dynamics of several state factors).*

- *$R = \sum_{i=1}^{m_R} R_i$ where $R_j : (\otimes_k F_k) \times A_S \times (\otimes_{k'} F_{k'}) \mapsto \mathbb{R}$ is the jth factor of the reward function (out of m_R factors). Neither $\{F_k\}$ nor $\{F_{k'}\}$ need be partitioned across their respective elements of the R_j's (i.e., state factors may contribute to multiple reward factors).*

- *$T \in \mathbb{N}$ is the finite time horizon and specifies the number of decision points, after which the problem terminates.*

Note that every unfactored MDP can be represented as a factored MDP where each model component has a single factor. Additionally, every factored MDP can be represented as an unfactored MDP, for example by enumerating all possible combinations of factors; however, doing so loses information about which factors are conditionally independent. Thus the factored MDP representation is strictly more expressive than the unfactored representation since it explicitly captures conditional independencies among factors.

Factored MDP's are often depicted graphically as two-stage dynamic Bayesian networks (2DBNs) (Guestrin et al., 2003), which allow for intuitive visualization of the dependencies among the factors. Figure 2.2 presents an example 2DBN for the single agent firefighting domain. The state factors are time (t), the agent's current position (ℓ), the intensity of fires in each of the C cells (I_c for cell c), and the delay conditions in each cell (δ_c for cell c), thus in this example, $m_S = 2C + 2$. The agent has six actions ($m_A = 6$), NOOP, N, S, E, W, FF. The fire intensity factors determine the reward factors (there are C reward factors, $m_R = C$), each of which is the negative intensity of the fire at that location. There are four transition factors ($m_P = 4$), the first (P_1) increments the time every step. The second (P_2) decrements a cell's intensity if the agent performs the fight fire action in that cell. The third (P_3) changes the agent's position depending on the agent's current position, action, and delay conditions. Finally, the fourth (P_4) represents that delay conditions in each cell do not change. The agent has four initial state distribution factors ($m_\alpha = 4$), which are analogous to the transition factors.

2.1.4 Decentralized (Partially Observable) Markov Decision Process

The first formal model for cooperative multiagent sequential decision making is that of Boutilier (1996) who created the multiagent Markov decision process (MMDP).

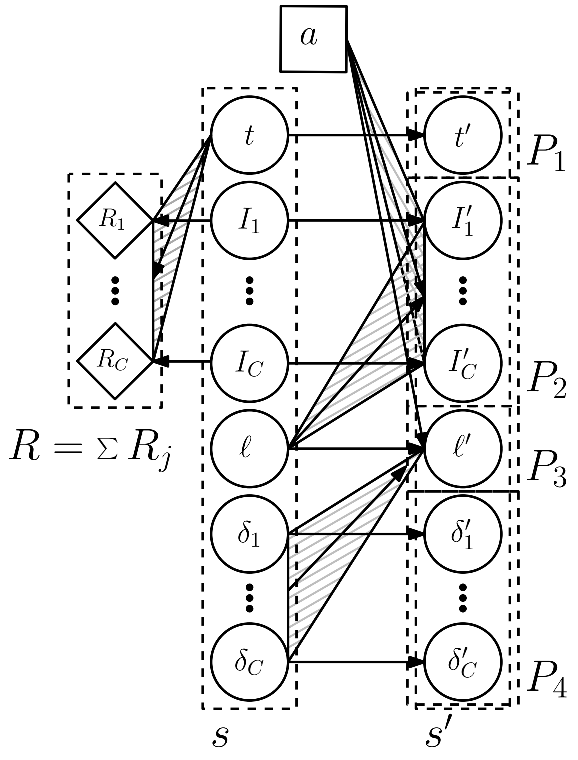


Figure 2.2: An example factoring for the single agent firefighting domain represented as a two-stage dynamic Bayesian network (2DBN).

While the MMDP was groundbreaking in its explicit focus on cooperative multiagent settings, it makes the often unreasonable assumptions that every agent can fully observe the entire state of the environment as well the actions executed by every agent. For these reasons, the MMDP was extended to the *decentralized partially observable Markov decision process (Dec-POMDP)* by Bernstein et al. (2002), which they defined more precisely in what follows. Note that the following definition is presented in an unfactored form, but factored versions of the Dec-POMDP have been constructed via factoring analogous to the methods discussed in Section 2.1.3.

Definition 2.4. *A* ***finite horizon Dec-POMDP*** *is a tuple* $\mathcal{M} = \langle \mathcal{N}, S, \alpha, A, P, R, \Omega, O, T \rangle$*, where:*

- $\mathcal{N}$ *is the finite set of* n *fully cooperative agents.*
- S *is the finite set of global states.*
- $\alpha : S \mapsto [0, 1]$ *is the initial state distribution.*
- $A : A_1 \times A_2 \times \cdots \times A_n$ *is the joint action space, where* A_i *is the finite action set for agent* i*. As with previous models, each state may have a subset of available*

actions for each agent, $A_{i_{s^t}} \subseteq A_i$, and thus each state effectively may have a subset of available joint actions $A_{s^t} \subseteq A$. I will be notationally explicit in this initial definition, but neglect it in subsequent discussions to reduce notational clutter.

- $P : S \times A_S \times S \mapsto [0, 1]$ *is the joint transition function.*
- $R : S \times A_S \times S \mapsto \mathbb{R}$ *is the joint reward function.*
- $\Omega : \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$ *is the joint observation space, where Ω_i is the finite observation set for agent i. As with actions, each state may have a subset of possible observations for each agent, $\Omega_{i_{s^t}} \subseteq \Omega_i$, and thus each state effectively may have a subset of available joint observations $\Omega_{s^t} \subseteq \Omega$. I will be notationally explicit in this initial definition, but neglect it in subsequent discussions to reduce notational clutter.*
- $O : S \times A_S \times S \times \Omega_S \mapsto [0, 1]$ *is the joint observation function.*
- $T \in \mathbb{N}$ *is the finite time horizon.*

Bernstein et al. (2002) also define the related model of the *Dec-MDP*, which is a special case of the Dec-POMDP.

Definition 2.5. *A **finite horizon Dec-MDP** is a Dec-POMDP in which the model is jointly observable. That is, there exists a mapping $J : \Omega \mapsto S$ such that if $O(o|s^t, a, s^{t+1}) \neq 0$, then $J(o) = s^{t+1}$.*

They then go on to prove that both the MDP and MMDP are subclasses of the Dec-MDP, which is in turn a subclass of the Dec-POMDP, and finally prove that both the Dec-MDP and Dec-POMDP have NEXP-complete complexity for two or more agents. Due to this high complexity, both the Dec-MDP and Dec-POMDP are widely considered computationally intractable.

Analogously to MDPs, the agents' objective in a Dec-POMDP is to maximize their expected total joint reward, which can be accomplished through the creation of an optimal joint policy, π^*. Several algorithms have been proposed for solving Dec-POMDPs and their subclasses using both centralized approaches (Boutilier, 1996; Guestrin et al., 2001; Hansen et al., 2004; Szer et al., 2005; Bernstein et al., 2009), as well as through distributed methods (Nair et al., 2003; Becker et al., 2004a; Varakantham et al., 2009; Witwicki, 2010; Velagapudi et al., 2011). In centralized approaches, the entire Dec-POMDP model is known by a single computational entity (or equivalently the entire model is known by each of the agents), who searches through the joint policy space to find the optimal joint policy, and then informs the agents of their constituent parts within the joint solution. To execute this policy, the agents must communicate with each other (e.g., exchanging their local observations after each action) to maintain a joint belief about the environment state. Such approaches are ensured to yield a globally-optimal, joint policy, but make often unreasonable assumptions about the centralized availability of the complete model and communication capabilities required to execute the resulting policy. Distributed approaches seek to mitigate the weaknesses of centralized ones by exploiting structure within the Dec-POMDP. In distributed approaches each agent i creates its own local policy π_i, and the joint policy is defined as $\pi = \langle \pi_1, \pi_2, \cdots, \pi_n \rangle$. This type of approach may or may not yield a globally-optimal, joint policy depending on the structural properties of the problem (see Section 2.1.4.1) and how the agents coordinate their local policies (if at all), but makes fewer assumptions about the centralized availability of the joint model and runtime communication capabilities.

Consider now a multiagent version of the firefighting domain. The (factored) environment state representation captures: the system time, $t \in \mathbb{N}$; the location of each agent, $\ell_i \in C$ for each agent i; the fire intensity, $I_c \in \mathbb{N}$ for each cell c; and a delay, $\delta_c \in [0, 1]$ for each cell c, which stochastically prevents movement into that cell with probability δ_c. Figure 2.3 shows an example environment state, with the location of each of two agents, along with the intensity of fire in the two cells with $I_c > 0$. Suppose (H)igh, (M)edium, and (L)ow delay in Figure 2.3 correspond to δ equal to $0.8, 0.5,$ and 0.0 respectively. For illustration, suppose that each agent can precisely observe the system time, the fire intensity in each cell, the delay in each cell, and its position. That is, an agent can *not* observe the position of the other agent. Each agent has the same six actions, NOOP, N, S, E, W, and FF, which behave similarly to the single agent case. Here, however, suppose movement actions are independent (agents can occupy the same location), but FF actions are not: the intensity of a cell only decreases by 1 even if multiple agents simultaneously fight it. The agents simultaneously execute actions for a predetermined number of steps, T. Suppose the reward function is identical to the single agent case, and the joint reward associated with a state is $-\sum_c I_c$.

2.1.4.1 Special Cases of the Dec-POMDP

While the general form Dec-POMDP is generally considered computationally intractable, substantial progress has been made by constraining the problem to

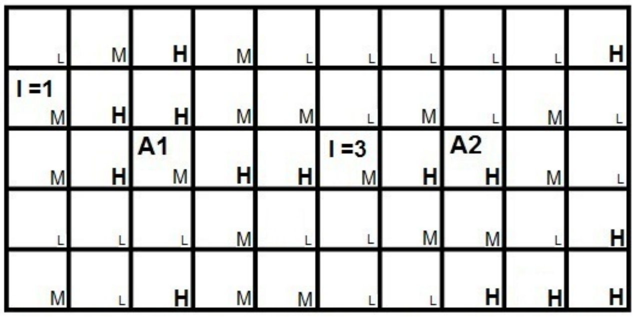


Figure 2.3: Example state of a 10×5 firefighting grid world. Ai designates the cell corresponding to the location of agent i, and $I = x$ indicates that there is a fire in that cell with intensity x. Letters designate a (H)igh, (M)edium, or (L)ow delay in that cell.

subclasses of the Dec-POMDP which contain certain types of structural properties. These subclasses are typically based upon factored versions of the Dec-POMDP formalism (Goldman & Zilberstein, 2004; Becker et al., 2004a,b; Varakantham et al., 2009; Witwicki, 2010; Velagapudi et al., 2011). Several definitions and properties are useful for categorizing these subclasses, which I overview now by adapting definitions taken from the above citations.

Definition 2.6. *Given a factored state representation, agent i's* ***local state*** *is defined as the (sub)set of global state factors that agent i's local observations are informative about (either partially or completely).* $S_i = \otimes_{k_i} F_{k_i}$, *refers to agent i's local state.*

Definition 2.7. *A state factor,* F_k *is* ***affectable*** *by agent i if* $\exists a_i \in A_i$ *such that* a_i *affects the transition dynamics for* F_k*. Note that a state factor that is affectable by agent i need not be in* S_i*, and a state factor in* S_i *also need not be affectable by agent i.*

Property 2.1. *We say that a Dec-POMDP is* ***locally fully observable*** *if each agent's local observations uniquely determine its local state,* $\forall i \forall o_i \exists s_i, Pr(s_i|o_i) = 1$.

Note that in Definitions 2.6 and 2.7, and Property 2.1, global state factors need not be partitioned across the agents, and it is possible for multiple agents to observe and/or affect the same state factor(s). Moreover, in interesting problems, it will almost always be the case that some state factors are observable and/or affectable by multiple agents, since otherwise the agents cannot interact and the MAS is just n independent, isolated agents.

Property 2.2. *We say that a Dec-POMDP has* ***transition independence*** *(Becker et al., 2004b) if the transitions over each agent's local state factors are independent of* *non-locally modeled state factors and the actions of other agents,* $\forall i, P(s_i^{t+1}|s^t, a) = P(s_i^{t+1}|s_i^t, a_i)$.

Property 2.3. *We say that a Dec-POMDP has* ***observation independence*** *(Becker et al., 2004a) if the joint observation function is decomposable into local observation functions. That is,* $O(o|s^t, a, s^{t+1}) = \prod_i O_i(o_i|s_i^t, a_i, s_i^{t+1})$*, where* $O_i(o_i|s_i^t, a_i, s_i^{t+1})$ *designates agent i's local observation function.*

Property 2.4. *We say that a Dec-POMDP has* ***reward independence*** *(Becker et al., 2004b) if the joint reward function is decomposable into a function of local reward functions. That is,* $R(s^t, a, s^{t+1}) = f\left(R_1(s_1^t, a_1, s_1^{t+1}), R_1(s_2^t, a_2, s_2^{t+1}), \cdots, R_n(s_n^t, a_n, s_n^{t+1})\right)$*, where* $R_i(s_i^t, a_i, s_i^{t+1})$ *is agent i's local reward function and* $f(\cdots)$ *is monotonic (typically the summation function).*

These properties can be combined together (perhaps with variants to reduce the expressive limitations of the property) to formulate a Dec-POMDP subclass in terms of **local models** (Varakantham et al., 2009; Witwicki, 2010; Velagapudi et al., 2011). That is, rather than define the problem in terms of a joint model as in Definition 2.4, we can instead define it in terms of a set of local models. The problem is then modeled by $\mathcal{M} = \langle \mathcal{N}, \{\mathcal{M}_i\} \rangle$, where $\mathcal{N}$ is a set of n fully cooperative agents, and $\mathcal{M}_i = \langle S_i, \alpha_i, A_i, P_i, R_i, T_i \rangle$ is the local model for a MDP agent i, or $\mathcal{M}_i = \langle S_i, \alpha_i, A_i, P_i, R_i, \Omega_i, O_i, T_i \rangle$ is the local model for a POMDP agent i (depending on the specific structure present in the domain).

While expressing most of these local components is intuitive, expressing the transition function in terms of local models is more challenging. That is, if a local state factor is affectable by other agents, then the transition function for that factor is dependent on the actions of other agents (i.e., the factor contains *non-local effects* (Witwicki, 2010)). Prior research has identified methodologies to address this issue by calculating a summary of the expected non-local effects and incorporating them into P_i (Varakantham et al., 2009; Witwicki, 2010; Velagapudi et al., 2011). Given the set of expected non-local effects on each agent, the agents' local models are conditionally independent, which allows the agents to reason in parallel without further coordination, and the joint policy is the aggregation of the local policies. Several strategies exist to compute expected non-local effects such as: assume (potentially incorrectly) that there are no non-local effects; have the agents coordinate to calculate the optimal set of non-local effects (Witwicki, 2010); or have the agents coordinate to approximate the likely non-local effects (Varakantham et al., 2009; Velagapudi et al., 2011).

For illustration of representing a problem in terms of local models, consider the multiagent version of the firefighting domain. Each agent i's local state consists of: t, ℓ_i, I_c for each c, and δ_c for each c. That is, an agent does not observe the positions of the other agents, and thus its local state does not include the other agents' positions. Each agent has the same six local actions as before: NOOP, N, S, E, W, and FF. Since movement actions are independent, P_i can precisely represent the transitions for ℓ_i. Similarly, t, and δ_cs are unaffectable by other agents, and P_i can represent them precisely as well. I_cs, however, are affectable by other agents, and thus I_c transitions contain non-local effects, which must be summarized (e.g., via one of the previously described methods) within P_i in addition to agent i's effects on I_c transitions. As before, each agent executes actions for a predetermined number of steps, $\forall i, T_i = T$, and since each agent can observe the fire intensity in each cell, let the local reward of each agent be identical to the previously described joint reward.

2.1.4.2 Approximate Techniques

Another, orthogonal approach for coping with the computational intractability of general case Dec-POMDPs is to utilize techniques that yield approximately optimal solutions. Intuitively, such approximation techniques have been shown to reduce computational costs of solving the Dec-POMDP, but do not ensure that the resulting policy is globally optimal, and may not make any solution quality guarantees at all. For example, in the JESP algorithm (Nair et al., 2003) the agents calculate a joint policy by iteratively revising their local policies to be the optimal best response to the other agents' policies on the previous iteration. The agents' policies are ensured to converge to a (Nash) equilibrium that must also be a local optimum in the joint policy space, but may not be a (Pareto-efficient) global optimum. Another approximately optimal technique is TREMOR (Varakantham et al., 2009) and the subsequent extension D-TREMOR (Velagapudi et al., 2011). In these methods, agents again compute best response policies, but use pre-identified coordination locales as a means to shape each agent's local model to estimate the non-local effects, resulting in approximate best responses. In this way, TREMOR and D-TREMOR do not provide any bounds on the joint policy's quality, but have been shown to scale better than many other Dec-POMDP subclass approaches.

One other potential caveat that can arise when the agents plan using approximate techniques is that their local policies may not encompass every possible situation that could arise. Namely, if an agent only creates a policy for states in its reachable state space, it is possible that when the agent is executing its policy, it could encounter a state for which it has not determined the action it should execute. This can occur, for example, if the expected non-local effects reflected in the agent's local model are inaccurate. In such situations, we say the agent has *fallen off policy*, and requires additional reasoning to determine an action to execute. This problem is the study of plan repair research that attempts to reuse (Krogt, 2005; Fox et al., 2006) and/or warp (Musliner et al., 2007) existing sub-policies as well as incremental planning techniques (Hansen & Zilberstein, 2001a; Koenig & Likhachev, 2002; Wu & Durfee, 2007). The idea in such approaches is to reuse the agent's existing policy as a basis for creating a new policy, that is to "fix" the current policy rather than replan from scratch. In this way, if the necessary changes to the agent's policy are relatively small (i.e., most of the current policy is still valid), the computational costs for the agent to determine a new policy can be much less than completely replanning.

2.1.5 Hierarchical Abstractions

The just-described decision-theoretic models each assumed that the actions specified within an agent's model are *primitive actions*, that is, each $a \in A$ is a single, atomic action that an agent can execute in exactly one time step. However, in many approaches (notably in problem-driven organizational approaches like those I describe in Section 2.2.2), it is common to identify a hierarchical task structure that defines abstract **tasks** in terms of primitive actions and/or other sub-tasks. By utilizing a hierarchical task structure, an agent can reason about its high-level behavior policy without complicating the process with the typically larger space of low-level primitive actions. For ease of explanation, I will assume a single agent, non-factored domain in the descriptions that follow; however, hierarchical abstractions have also been developed for both factored and multiagent models (Barto & Mahadevan, 2003; Stone et al., 2005; Ghavamzadeh et al., 2006; Goldman & Zilberstein, 2008; Amato et al., 2014) using concepts similar to those in Sections 2.1.3 and 2.1.4.

Researchers have identified several methods for incorporating hierarchical decompositions into decision-theoretic paradigms (Parr & Russell, 1998; Sutton et al., 1999; Dietterich, 2000). At the core of these approaches is the formalism of the *semi-Markov decision process (SMDP)*, which generalizes upon the MDP by allowing the time between one decision point and the next to be a random variable (Howard, 1971). The modeling differences between the SMDP and MDP are centered on the addition of τ, which is the (positive) time until the next decision point after executing action $a \in A$ in state $s \in S$. The transition and reward functions are then extended to incorporate τ, and yield the following definition.

Definition 2.8. *A* ***discrete-time SMDP*** *model is defined as* $\mathcal{M} = \langle S, \alpha, A, P, R, T\rangle$*, where:*

- S *is the finite state space,* α *is the initial state distribution,* A *is the action space, and* T *is the time horizon as in an MDP (Definition 2.1).*
- $P : S \times A_S \times S \mapsto [0, 1]$ *is the transition function, where* $P(s^{t+\tau}|s^t, a)$ *specifies the probability of the environment transitioning to state* $s^{t+\tau} \in S$ *after* $\tau \in \mathbb{N}$ *time steps when the agent executes action* $a \in A_{s^t}$ *in state* $s^t \in S$*.*
- $R : S \times A_S \times S \mapsto \mathbb{R}$ *is the reward function, where* $R(s^t, a, s^{t+\tau})$ *specifies the immediate reward associated with executing action* $a \in A_{s^t}$ *in state* $s^t \in S$ *yielding state* $s^{t+\tau} \in S$ *after* $\tau \in \mathbb{N}$ *time steps.*

The Bellman equation is reformulated for SMDPs as follows:

$$Q^*(s^t, a) = \sum_{\tau} \sum_{s^{t+\tau} \in S} P(s^{t+\tau}|s^t, a) \left[R(s^t, a, s^{t+\tau}) + \max_{a' \in A} Q^*(s^{t+\tau}, a') \right] \tag{2.7}$$

and solved using analogous methods to those in Section 2.1.1.

Using this underlying formalism, Sutton et al. (1999) define *options* as a means to hierarchically decompose a task structure and represent the decomposition within SMDPs.

Definition 2.9. *An* ***option*** *is defined as* $o = \langle \mathcal{S}, \mu, \beta, \rangle$*, where:*

- $\mathcal{S} \subseteq S$ *is set of states in which the option may be initiated.*
- $\mu : S \times A \mapsto [0, 1]$ *is a policy for executing the option, where* $\mu(s, a)$ *specifies the probability that the agent will decide to execute action* $a \in A$ *in state* $s \in S$*.*
- $\beta : S \mapsto [0, 1]$ *is the termination condition, where* $\beta(s)$ *specifies the probability that the option will terminate upon reaching state* $s \in S$*.*

Throughout this document, I will use the dot operator to refer to an option's components. That is, $o.\mathcal{S}$, $o.\mu$, and $o.\beta$ refer to option o's initiation set, policy, and termination condition respectively.

An agent can decide to begin executing an option o from any $s \in o.\mathcal{S}$. The agent then executes actions according to $o.\mu$, the environment transitions to subsequent states according to the SMDP transition dynamics P, and the agent associates reward according to R as usual. In each state $s \in S$ the agent encounters, o stochastically terminates with probability $o.\beta(s)$, and after o terminates, the agent can select another option o' (assuming $s \in o'.\mathcal{S}$). For example, in the firefighting domain, an agent could have an option, $o_{move(1,2)}$ to move to the cell at coordinates $(1, 2)$, in addition to its primitive actions of N, S, E, W, NOOP, FF. The above option might have the following properties: $o_{move(1,2)}.\mathcal{S}$ contains all states with $I_{(1,2)} > 0$ (i.e., there is a fire in cell $(1, 2)$); $o_{move(1,2)}.\mu$ specifies the movement actions to move the agent to cell $(1, 2)$; and $o_{move(1,2)}.\beta(s) = 1$ if $\ell = (1, 2)$ in s, otherwise $o_{move(1,2)}.\beta(s) = 0$.

As just illustrated, one way to view an option is as a temporally extended macro-action. In this way, an option encodes a higher-level task, and the option's policy determines how that task is decomposed into subtasks. Along this line, Sutton et al. (1999) defined options to support recursive nesting and to allow one option to be defined in terms of other options, which are themselves composed of options, and so on, until only primitive actions remain at the lowest level. An agent's "action" space in the SMDP is thus actually its option space, $\mathcal{O}$, in addition to its primitive action space $A \equiv \mathcal{O} \cup A_{primitive}$.

For an agent to utilize an option, o, while solving the SMDP for its policy, models of the option's expected reward, $R(s^t, o, s^{t+\tau})$, and transitions $P(s^{t+\tau}|s^t, o)$ are required. These values are directly calculated using the SMDP's reward/transition components and o's policy and termination condition, using an algorithm similar to Algorithm 3.1 that unrolls the state space for τ iterations using P and $o.\mu$ to determine the next states at each iteration, and calculating the probability of being in each of the reachable states after exactly τ iterations. The expected reward of o is given by the reward accrued along the reachable trajectory while following $o.\mu$ for τ time steps.

Options have been shown to improve system performance (Sutton et al., 1999; Barto & Mahadevan, 2003; Stone et al., 2005) assuming that options are correctly identified and encoded. These studies found that a good heuristic for creating the option space, $\mathcal{O}$, is to associate an option with a subgoal, such that the option terminates when the subgoal is achieved, and its policy directs the agent to accomplish the subgoal. Subgoals can either be identified using expert knowledge and then hand coded, or alternatively discovered by automated methods (Iba, 1989; Stolle & Precup, 2002; McGovern, 2002). These automated methods identify good subgoals as those states/actions that the agent typically encounters/executes on successful trajectories but not on unsuccessful ones.

In this dissertation, I use the options framework (e.g., in Section 4.2.2) as a mechanism to both expedite computing optimal joint policies, and also to focus an ODP's computational efforts on the MAS's joint interactions rather than agents' primitive actions.

2.2 Organizational Decision Making

Despite a large volume of prior research, the organizational research community has been largely unsuccessful in creating a precise, consensus definition of what an organization or its purpose is (Butts & Carley, 2007). As alluded in Chapter 1, however, recent approaches roughly fall into two main categories:

Problem-driven. In this perspective (discussed more in Section 2.2.2), the starting point of an organization (and the subsequent MAS) is distribution over expected decision problems. The purpose of forming an organization, in this context, is to represent a strategy for decomposing and solving the problem as a MAS, where a top-down, knowledge-based algorithm first creates an organizational design. This design is subsequently populated with appropriate agents to enact the organization. In prior work, it is assumed that an organization exists as an explicit, first-class object independently of any agents that might enact it, and moreover, though the organization itself is assumed to change very little over time, the agents enacting the organization may change comparatively frequently. Traditionally, the problem-driven research community has emphasized the development of languages for expressing an organization, and given less attention to the study of how to create the organization. In these works, an expert human usually serves as the ODP, who uses their knowledge to identify and represent an appropriate organization for the domain.

Experience-driven. In this perspective (discussed more in Section 2.2.3), the starting point of a MAS (and subsequent organization) is a group of cooperating agents. These agents are already working together (or at least trying to), and the purpose of forming an organization, in this context, is to reason over and codify expectations about appropriate actions and interaction patterns in order to improve and streamline cooperation. In prior work, experience-driven approaches have an emergent, self-organization flavor, and do not explicitly represent the organization as a first-class object. Rather, the MAS's organization is only implicitly observable via the agents' policies. In contrast to problem-driven approaches, organizations in experience-driven approaches are fluid, and adapt in response to the environment the MAS encounters; however, the agents in the MAS are fixed and not easily replaceable because of their local expertise (Property 1.3) that has been finely-tuned in response to the agent's experiences. Traditionally, the emphasis of experience-driven approaches is on the process of creating coordinated polices (that are implicitly organized), and gives less attention to the end product (i.e., how the organization is represented). The policy adaptation algorithm serves as the ODP, and adapts agents' policies over time in response to problem episodes the MAS encounters.

Unsurprisingly, each of these general approach categories has advantages and disadvantages. In an effort to achieve the benefits of both approach categories, several mixed approaches have been proposed (my agent-driven approach is also a mixedapproach). Like problem-driven approaches, mixed approaches represent the organization as an explicit, first-class object, but like with experience-driven approaches, the specific influences that the organization exerts emerge dynamically in response to the environment. In this way, mixed approaches can leverage the advantages of each approach. The problem-driven aspect provides explicit context for the organization to adapt within, which helps to speed up convergence, increase the likelihood of convergence, and/or steer the adaptations into more globally desired organizations (as opposed to locally optimal). Meanwhile, the experience-driven aspect provides a basis for dynamic adaptations, which tailors the organization to the actual environment the agents encounter. I discuss such mixed approaches more in Section 2.2.4.

The remainder of this section is structured as follows. I begin in Section 2.2.1 with a discussion of the early MAS research that developed the foundational ideas of organizing a MAS. Then I provide more detailed discussions of problem-driven, experience-driven, and mixed approaches (Sections 2.2.2, 2.2.3, and 2.2.4, respectively) and their relationships to the organizational design problem I focus on in this dissertation and my approach towards solving it. Then, in Section 2.2.5, I discuss several operational reasoning techniques that do not directly consider organizations, but nevertheless yield valuable insights when viewed from an organizational perspective. Finally, I briefly discuss the relationship between human organizations and the techniques I develop for organizations for computational MASs in Section 2.2.6.

2.2.1 Early MAS Research

The relationship between organizations and MASs has been studied since the 1980s; however, early approaches are difficult to categorize into problem-driven or experience-driven. Rather, early approaches are more easily viewed as precursors that developed the foundations upon which problem-driven and experience-driven approaches were subsequently investigated (e.g., established the intuitions reflected in Properties 1.1– 1.5). Unsurprisingly, in some cases, early approaches are even proto-versions of later research. In this section, I briefly overview some of the most significant early MAS research as related to organizations, and discuss how it relates my agent-driven approach.

One of the earliest MAS research lines was done at the University of Massachusetts (Corkill, 1979; Corkill & Lesser, 1983; Durfee et al., 1987). In this work, organizational reasoning is viewed as a meta-level of operational reasoning. An important result of this research is the notion that organizational reasoning should provide explicit, high-level guidelines that steer the agents into coordinated interaction patterns. While an agent is planning, it then uses its local expertise to temper its organizational guidelines. As I will show in Section 4.1, my agent-driven approach reinforces this result both in my empirical experiments as well as my theoretical analysis. Additionally, later problem-driven approaches (and my agent-driven approach) built upon the emphasis of an explicit organizational representation found in this work.

Fox also viewed organizations as a mechanism for providing high-level guidelines, but focused on how to balance between uncertainty (e.g., in task distribution, resource utilization, etc.) and the decentralization of a MAS (Fox, 1981). Fox & Smith (1984) also looked at methods for decomposing and representing tasks and their associated resource requirements as part of organizing a MAS.

Another line of MAS research viewed organizational reasoning as part of a continuum alongside operational reasoning. For example, Durfee & Montgomery (1991) developed a framework showing how plans, schedules (i.e., a specific, applied plan), and organizations (i.e., the abstract, long-term MAS objectives) can be unified within a single reasoning framework. Subsequent research built on this perspective, investigating how agents can effectively coordinate over this hierarchical behavior space (Durfee, 1993; Castelfranchi, 1995). These techniques can be viewed as proto-influences (see Section 2.2.5 for a discussion of influences) in that they provide a framework for agents to efficiently coordinate their interactions at an abstract level.

Ishida et al. (1992) studied how organizations—in particular work-allocation and load-balancing—can be adapted in response to the environment. Subsequent research into experience-driven and mixed approaches builds on the intuitions of this research to develop techniques for adapting organizational structures (see Sections 2.2.3 and 2.2.4 respectively).

Finally, Fox & Gruninger (1998) investigated how to model enterprises, and discussed ontologies for representing concepts such as activities, resources, goals, etc.

Subsequent research into organizational modeling languages (OMLs) (see Section 2.2.2) builds on the intuitions developed in this proto-OML research.

2.2.2 Problem-driven Approaches

As previously mentioned, problem-driven approaches focus on utilizing expert knowledge of a problem domain to identify and encode an organizational strategy for decomposing and solving the distribution of expected decision problems as a MAS. Commonly, this knowledge is specified via an *organizational modeling language* (OML), which provides syntax for expressing organizational knowledge that can later be enacted as a MAS. The research community has proposed a large number of OMLs including: Gaia (Wooldridge et al., 2000), SODA (Omicini, 2001), ISLANDER (Esteva et al., 2001), OperA (Dignum, 2004), Tropos (Bresciani et al., 2004), OMNI (Vázquez-Salceda et al., 2005), MOISE$^+$ (Hübner et al., 2007), ODML (Horling & Lesser, 2008), and ORG4MAS (Hübner et al., 2010) among others. While the specifics of these OMLs vary, there are several features that the community has identified as important for encoding organizational knowledge. In what follows, I overview the most commonly included features to provide a greater intuition behind the types and forms of knowledge that are encoded within a OML, but for more details please consult the papers cited above. Throughout the following discussion, I make use of a scientific conference domain that the research community frequently uses to illustrate knowledge-based approaches (e.g., Vázquez-Salceda et al. (2005)).

Task Structure. Problem-driven approaches typically assume an organization creates a MAS that performs a complex task involving many interconnected subtasks that themselves might recursively contain interconnected sub-subtasks, and so on. For example, the TAEMS framework (Lesser et al., 2004) is a common method for representing such a hierarchical task structure. Therefore, part of the expert knowledge encoded in an OML is how the global task is decomposed into its respective subtasks, and the relationships between those constituent subtasks (i.e., must the agents perform all subtasks to complete the parent task or only a subset of them; are there sequentiality/simultaneity constraints; etc.). For example, in the conference domain the global task is to hold a successful conference, and the subtasks are to collect a set of high-quality papers, secure a venue, etc. Each of the above subtasks must be completed to achieve the global task, although there is no strict ordering requirement between them. Further, each subtask itself consists of multiple sub-subtasks, for example the collect-papers subtask might include sub-subtasks to issue a call for papers, gather submissions, review those submissions, make decisions to accept/reject the submissions, etc. In this case, the sub-subtasks must be performed sequentially, and each of them must be completed to accomplish the parent collect-papers task.

Environment Model. Related to the task structure is an environment model, which captures information about the expected environmental parameters such as: available resources and constraints over resources; expected resource requirements and costs for completing tasks; expected agent capabilities; expected communication availability, channels, bandwidth, and latency; etc. In the conference domain, environment parameters include: a set of deadlines for the tasks; financial limitations for securing the venue; expectations that agents can freely communicate with high bandwidth and low latency; etc.

Roles. In each of the above OMLs, a primary mechanism for encoding expert knowledge is through *organizational roles*, which are a mechanism for summarizing the organization's expectations about the agents' actions and interactions. More precisely, an organizational structure can contain multiple roles that are then adopted by (some of) the available agents (this mapping can be many-to-one, one-to-one, many-to-many, or one-to-many depending on the system). For example, in the conference domain, some roles might be AUTHOR, REVIEWER, SENIOR-REVIEWER, ATTENDEE, etc. Specific definitions of roles vary across OMLs, but typically a role is associated with:

- Constraints about which agents can/should enact the role. These constraints can be in terms of capabilities the agent must/should have (e.g., an agent enacting the REVIEWER role should have expertise in the conference's field) as well as in terms of relationships between roles (e.g., an agent enacting the AUTHOR role can not also enact the REVIEWER role for that paper).

- Expected relationships that an adopting agent should have with other roles, along with associated communication protocols. Continuing the scientific conference example, the REVIEWER role is expected to interact with an associated SENIOR-REVIEWER role to make a decision about the paper's acceptance to the conference. This interaction could be communicated in the form of numerical scores of the paper's quality and associated text discussing thoughts on the paper.

Norms. Norms are a formal specification of required, permitted, obliged, and/or forbidden actions expressed using deontic logic. Within OMLs, a norm can be viewed as a singleton piece of organizational guidance that is independent of any one agent, but rather associated with some organizational role(s). Additionally, a norm is often associated with a means of enforcement and penalty for non-conformance to the specified actions by any agent enacting the associated role. For example, the REVIEWER role might have an associated norm requiring each REVIEWER agent to review one paper by a pre-determined deadline, and if an agent enacting the REVIEWER role fails to meet this norm it will not be allowed to adopt the REVIEWER role in the future.

Unsurprisingly, there are a large number of structures that one could use to organize a MAS (e.g., hierarchy, coalition, etc.). However, as surveyed by Horling & Lesser (2004), several popular classes of structures account for the majority of organizations studied in MAS research.

Given an OML specification of an organization, the next step to is to populate it with agents to enact the various roles, which can be performed using a mechanism such as contract net (Davis & Smith, 1983; Sandholm, 1993), STEAM (Tambe, 1997), service oriented computing (Papazoglou, 2003; Bichier & Lin, 2006), or RETSINA (Sycara et al., 2003), among others. The specifics of these mechanism vary, but broadly they serve to identify, recruit, and enlist agents with the necessary expertise (as specified by the norms of a role) to adopt the roles in the organizational specification.

Compared to my agent-driven approach described in Section 1.3, problem-driven approaches have many similar components (e.g., norms are analogous to organizational influences, both approaches have an environment model, etc.); indeed my agent-driven approach intentionally builds on the problem-driven idea that an organizational representation should exist as an explicit first-class object. Despite these similarities, however, there are significant differences in how these ideas are manifested between my approach and problem-driven ones. Most importantly, in problem-driven approaches, the task structure and environment model features described above constitute the *output* of a human ODP's expert knowledge, and serve to delineate the necessary task decompositions, agent capabilities, communication channels, etc. required for the organized MAS to solve the expected distribution of decision problems. In stark contrast, the task structure and environment model features constitute the *input* to my agent-driven, computational ODP techniques, which my ODP analyzes to identify organizational patterns from first principles.

As a consequence, a fundamental disadvantage of problem-driven approaches is an absence of solid, theoretical foundations. As illustrated by the large number of OMLs cited above, the research community has incrementally identified additional and/or alternative organizational modeling techniques that provide more expressive OMLs and/or more intuitive OMLs for human organizational designers to encode organizational expertise. However, since the OMLs lack a formal mathematical foundation, it is impossible to determine if prior OMLs are complete, or if future research will yield yet more evolutions. Moreover, since the OMLs are intentionally designed to be independent of any particular agent architecture, the agents in the eventual MAS can not be assumed to necessarily understand how to map the organizational specification to their local reasoning processes. As a result, middleware has been developed to allow agents to understand and integrate the organization's influences into their local reasoning (Pynadath & Tambe, 2003; Esteva et al., 2004; Hübner et al., 2005, 2007). Since my agent-driven approach bases the organization's representation on the agents' reasoning framework, agents can natively incorporate their organizational influences into their local reasoning, and such middleware systems are unnecessary in my research.

Complementarily, the fundamental advantage of problem-driven approaches compared to my agent-driven one is that they do not commit to a particular group of agents, which makes the methods particularly useful for *open systems*. In an open system, the agents enacting an organization are allowed (and moreover assumed) to join and leave the organization as they please, but the organization structure as encoded in the OML remains relatively unchanged regardless of which particular agents are currently enacting it. For example in the conference domain, the organizational structure responsible for setting up and leading the conference is practically unchanging from year to year (although could change, for example, to add a new track to the conference); however, the specific agents enacting the organization typically change each year, with existing agents adopting different roles over time. In contrast, my agent-driven approach commits to specific group of agents, which provides a mathematical basis for organizational reasoning, but also makes it more cumbersome for agents to come and go as in open systems (see Section 6.2.7 for brief thoughts on extending my agent-driven approach to open systems).

2.2.3 Experience-driven Approaches

As opposed to the problem-driven approaches described in Section 2.2.2, experience-driven approaches begin by assuming a fixed set of agents that gradually make local adaptations to their policies as they interact with the environment and each other. Over time, these adaptations (hopefully) converge the agents' joint policy to *de facto* organized patterns of joint interactions. However, in stark contrast to the explicit, first-class organizational representations in problem-driven and my agent-driven approach, experience-driven approaches do not explicitly represent the organization, rather the *de facto* organization is only implicitly observable in the MAS's joint behaviors. As such, the focus of experience-driven approaches is on the *process* of determining appropriate policies, with little consideration given towards representing the final organizational product.

The research community has proposed several classes of experience-driven methods that I summarize below; for more details, please consult the respective papers cited below. In this section, I reuse the firefighting domain previously presented in Section 1.2 to illustrate concepts.

Swarm Intelligence. (Bonabeau et al., 1999; Dorigo et al., 2006; Gauci et al., 2014). Fundamentally, the idea of swarm intelligence is to utilize a multitude (i.e., often hundreds or thousands) of simple agents that each learn and follow primitive rules for deciding their actions rather than collectively creating policies accounting for future effects of sequential decision trajectories. These agents then either leave signals in the environment (e.g., pheromones that fade over episodes) to influence the other agents' actions (in subsequent episodes), and/or learn primitive decision rules to directly translate environmental stimuli into actions. For example, in the firefighting domain, each agent could leave a pheromone in each cell that it visits that fades over time, which could signal to the other agents that the cell is already covered (and thus the other agents should focus their efforts elsewhere). The agents' cooperation patterns (and thus organizational influences) are implicitly represented within these environment-encapsulated signals and/or decision rules, and only observable indirectly via the agents' actions.

Multiagent Reinforcement Learning (MARL). (Hu & Wellman, 1998; Bowling & Veloso, 2002; Stone et al., 2005; Busoniu et al., 2008). In MARL, individual agents learn local policies via reinforcement both within and across problem episodes, where these local policies are conditional on the local policies of the other agents. There are several strategies that the agents could employ to coordinate their local policies (see Busoniu et al. (2008) for a more detailed survey), such as: explicitly communicating with each other to select a joint action; implicitly cooperating by estimating the policies of the other agents (e.g., estimated joint action learning); or incorporating additional, organization-like constraints such that an agent can make globally-useful local decisions (see Section 2.2.5 for more details of these techniques). In the firefighting domain, for example, the agents could over time learn policies where each agent tends to fight fires in a respective region of the grid.

A fundamental advantage of experience-driven approaches is the principled, mathematical foundation that the algorithms use as a basis for adapting the agents' policies. That is, adaptations are made when they increase the expected performance of the MAS, resulting in a (locally optimal) joint policy. My agent-driven approach builds on this concept by extending these mathematical foundations to explicit organizations, where an organization is selected if it has (approximately) optimal benefit to the MAS. Moreover, experience-driven approaches do not require extensive domain knowledge in order to yield an organization, thus making them particularly useful when there is high *a priori* uncertainty about the environment or agent capabilities. In such cases, the agents themselves can still converge towards an appropriate, implicitly-organized policy through repeated adaptations, whereas a problem-driven approach might lack sufficient initial information to create an equally effective organization. Experience-driven approaches, however, can sometimes require extensive iterations of adaptations before the agents' policies converge (if ever) due to the large space of possible joint actions and/or limited local awareness of each agent. Further, while convergence has been theoretically proven in some cases (Bowling & Veloso, 2002), the resulting joint policy is ensured to be only a Nash equilibrium, and thus could be suboptimal from the global, fully-cooperative perspective. My agent-driven approach lies between problem-driven and experience-driven approaches in this regard, requiring a model of the domain from which to derive organizational patterns, but not for an expert to pre-determine organizational patterns. This is an intentional choice, however, since model based planning allows an ODP to account for rare, critical problem episodes without needing to actually experience them.

As mentioned, a drawback of experience-driven approaches is the distinct lack of explicit organizational representation, though function approximation techniques (Sutton et al., 2000; Whiteson & Stone, 2006; Busoniu et al., 2010) can provide a higher-level perspective of the agents' policies with aspects more akin to an organizational representation. This deficit can make understanding or justifying the agents' final policy difficult for system administrators, and also challenging for agents to join, leave, or change roles within the MAS (e.g., as part of an open system).

2.2.4 Mixed Approaches

The research community has proposed several mixed approaches to organizational reasoning in an attempt to achieve the advantages of both problem-driven and experience-driven approaches. These approaches explicitly represent the organization, which is typically initialized via top-down knowledge (like in problem-driven approaches). Then, as the agents solve problem episodes, they adapt the organization in response to the environment (like in experience-driven approaches). Thus, in many senses, mixed approaches can be viewed as employing a problem-driven approach for designing the components of the organization, and then utilizing a experience-driven approach to tune and configure the details of how their organization is implemented. By explicitly representing the organization, the agents have additional context from which to make adaptations to their policies, which can increase the likelihood of convergence, increase the quality of the converged upon joint policy, and/or decrease the number of iterations until convergence (Zhang, 2011). Meanwhile, the adaptive aspect allows the agents to overcome imperfections in the organization (e.g., brought about by unanticipated problem episodes), and tailors the organization to the actual execution environment.

The research community has proposed several classes of mixed approaches, which I overview below; for more details, please consult the respective papers cited below.

Structural Configuration and Adaptation. (Sims et al., 2003; Gaston & desJardins, 2005; Butts & Carley, 2007; Hoogendoorn, 2007; Horling & Lesser, 2008; Sims et al., 2008; Kota et al., 2009). The primary idea in these techniques is to configure and/or adapt the organization (especially the structure between roles) to be suited for the environment that the MAS experiences. Like problem-driven approaches, the organization is an explicit, first-class object that exists independently of any agents that might enact it. The significant difference from problem-driven approaches, is that after a (human) ODP initially designs the overarching organizational components (e.g., task structure, roles, etc.), the organization is configured via a computational optimization algorithm. The configuration algorithm (and conceptually-identical, subsequent adaptation algorithm) is related to experience-driven concepts, since the organization is optimized in response to expected and/or experienced problem episodes that the MAS encounters.

Organizationally Adept Agents (OAAs). (Horling et al., 2001; Corkill et al., 2011, 2012). The underlying principle of OAAs is that agents should be empowered to make informed decisions about their organization in response to the environment they are encountering. The difference between OAAs and the other mixed approaches lies in the distinction between adaptations and adeptness, where OAAs not only adapt their organization, but explicitly make adaptations because they are aware of the broader context and relationship between their organization and environment. That is, OAAs are made aware of the underlying expectations and assumptions that the ODP has about their organization, and use this second-order information as a basis for adapting their treatment of the organization. In this way, the OAAs can adapt to an organization that is aligned with the ODP's intent (which presumably is appropriate even if its current configuration was not), which serves as a heuristic to reduce and/or focus the space of possible adaptations. In the firefighting domain, for example, the organization could designate regions of responsibility for each agent, and the underlying expectation for this organization is that the fires are uniformly distributed in these regions. If the OAAs observe that this expectation is not being met in the execution environment, they could adapt the regions within the organization, such that the fires are uniformly distributed among these revised regions.

Coalition and Team Formation. (Brooks & Durfee, 2003; Stone et al., 2010). The premise of these techniques is for the (often self-interested) agents to dynamically form cooperative groups, which can be thought of as (typically shorter-term) organizations. While details vary, the group formation process is usually initiated by the agents, who identify that cooperating with other agents would be mutually beneficial, for example, cooperating with the group could provide expertise and/or resources that would otherwise be unavailable to an individual. Like in experience-driven approaches, the cooperative groups emerge via repeated interactions with other agents; however, as in problem-driven approaches, the organizational structure and influences are explicitly represented. This explicit representation provides context for the agents' decisions, which allows them to more efficiently compute effective, coordinated policies. Unlike problem-driven approaches, however, the organization exists only within the context of the agents enacting it, and if agents leave the organization, then it ceases to exist (of course, agents could (re-)form a new group as necessary).

Since my agent-driven approach is also a mixed approach designed to achieve the benefits of both problem-driven and experience-driven approaches, it shares several significant attributes with the related works described above. Namely, mixed approaches represent an organization as an explicit, first-class object, and utilize computational algorithms to create, configure, and/or adapt the organization to optimize expected MAS performance.

The distinctions between my agent-driven approach and the techniques described above lie in where the techniques fall in the spectrum between problem-driven and experience-driven approaches. Structural configuration/adaptation and OAA techniques are closely related to problem-driven approaches, but additionally incorporate experience-driven concepts to permit the organization to be fitted to the actual environment. My agent-driven approach takes the experience-driven ideas a step further, where rather than just configuring/adapting the organization in response to the environment, I use expectations about the MAS's agents (e.g., capabilities, expected joint interactions, etc.) as the foundation for *creating* the organization. The coalition/team formation techniques are yet another step closer to experience-driven approaches; as opposed to a globally-informed ODP creating the organization (as in my agent-driven approach and problem-driven approaches), self-interested agents locally decide to form cooperative groups for mutual benefit.

2.2.5 Operational Techniques from an Organizational Perspective

Beyond organizational approaches, there are a number of operational reasoning techniques that are conceptually related to organizational reasoning, despite traditionally being studied in an operational context. In this section, I discuss the benefits and limitations of these approaches when viewed from an organizational perspective.

Reward Shaping. (Ng et al., 1999; Wolpert & Tumer, 2001; Agogino & Tumer, 2005; Babes et al., 2008; Zhang, 2011). The central idea of reward shaping is to alter the agents' local reward functions so as to bias each agent into globally desirable local actions, for example via a potential function, $\phi(s)$, such that

$$R_{i\_shaped}(s_i^t, a_i, s_i^{t+1}) = R_i(s_i^t, a_i, s_i^{t+1}) + \phi(s_i^{t+1}) - \phi(s_i^t)$$

where $\phi(s_i^t)$ estimates the expected joint value of being in s_i^t. By doing so, reward shaping can lead an agent to establish conditions that have no inherent local reward, but that enable other agents to then perform actions that result in high joint reward. Traditionally, reward shaping has been applied as part of the agents' operational decision making, with specialized reward shaping values (ϕ) developed for a single problem episode. I build on these ideas to apply reward shaping for organizational reasoning in Section 3.5.3, the primary difference being that the shaped reward function must apply across problem episodes, and thus the method for shaping rewards must account for the variability of appropriate agent actions across the episode space.

Coordination Locales. (Varakantham et al., 2009; Velagapudi et al., 2011). The central idea in this approach is that if agents have a relatively small number of explicitly-specified joint interaction possibilities, then they can plan independently of each other most of the time, and in the explicitly identified coordination locales, the agents' local reward and transition models are shaped to bias the agents into a good joint interaction. For example, in disaster rescue domains where agents can detrimentally collide in narrow corridors, two agents can plan independently unless they anticipate both (potentially) being in a single corridor at the same time. In these states, the agents' local transitions are shaped to reflect the joint transition (e.g., the agents would be disabled due to colliding), and their local rewards are shaped so that the agents avoid that state. Unsurprisingly, prior research has focused on leveraging coordination locales for operational decision making within a single problem episode, and indeed, pre-specifying all possible coordination locales for every possible problem episode could be a daunting task. Stepping back, however, the idea of shaping the ways in which the agents interact is similar to what an organization is trying to accomplish, albeit at a single episode scope rather than across episodes. Tractably extending coordination locale techniques to an organizational perspective implies constructing more abstract coordination locales that, for example, represent interaction patterns across episodes.

Influence Abstraction. (Witwicki, 2010; Witwicki et al., 2012; Oliehoek et al., 2012). The central idea of this line of research is that the non-local effects on an agent's local model can be summarized via *influences*. Broadly speaking, an influence shapes an agent's transition model to reflect non-local effects of the other agents, and given a statistically-sufficient set of influences, the agents' decision problems are conditionally independent. Put another way, an influence summarizes a joint interaction, and the approach provides an abstraction layer upon which the MAS can directly coordinate the agents' interactions for a problem episode without conflating the coordination with details about how the agents' will execute their respective portions of the coordinated global policy. My research extensively leverages the ideas of influence abstraction, but generalizes on them in two primary ways. Firstly, I adapt the idea of an influence from an operational perspective to an organizational one, where rather than an influence representing the expected non-local effects in a single problem episode, it represents the expected non-local effects over a space of episodes. Secondly, I identify that in an organizational perspective, it can sometimes be beneficial to summarize the non-local effects on other modeling components besides transitions (e.g., states, actions, etc.), and as such, the definition of organizational influence I develop in Section 3.1 encompasses the full set of an agent's modeling components.

Constrained Multiagent Reinforcement Learning. (Abdallah & Lesser, 2007; Zhang, 2011; Lau et al., 2012). As a technique for improving the convergence properties of multiagent reinforcement learning algorithms, research has identified that additional constraints can be added to an agent's local action space (both hard and soft constraints have been considered), which serve to guide the agents' action selections into more globally desired policies. These constraints are created via a hierarchical supervisor, who reasons using an abstracted view of the environment created from the agents' observations as a summary of the execution environment they have experienced. The supervisor is repeatedly invoked as the agents experience the environment, and determines the appropriate action constraints to enforce through a process similar to reinforcement learning (and in some of the approaches cited above is exactly reinforcement learning). Though these techniques have been applied for operational reasoning, from an organizational perspective, such action constraints are analogous to organizational influences, and the supervisor is analogous to an ODP. A difference however, is that, even viewed from an organizational reasoning perspective, they represent a sequence of organizations that is learned rather than a single organization created upfront via model-based planning. This is significant if the performance of the transient organizations is relatively poor, the learning period is extensive, and/or rare episodes occur where performance is critical.

Hierarchical Learning/Planning. (Barto & Mahadevan, 2003; Stone et al., 2005; Ghavamzadeh et al., 2006; Goldman & Zilberstein, 2008). Recall from Section 2.1.5, that utilizing a hierarchical task structure, an agent can reason about its high-level behavior policy without complicating its decision process with the typically larger space of low-level primitive actions. In decision-theoretic models, hierarchical methods are typically utilized as part of an agent's operational reasoning, although hierarchical methods have been used for knowledge-based models as an aspect of organizational reasoning as seen in Sections 2.2.2 and 2.2.4. The abstract reasoning of hierarchical methods is essentially computing the best ways for the agents to interact in a given problem episode. Aggregating the results of such reasoning across episodes provides an organizational perspective of the agents' interactions, which an ODP could use as a basis for constructing an organizational design. Indeed, the ODP techniques I develop in Section 4.2.2 build on exactly this notion, and use hierarchical options to construct statistics of the expected impact to the MAS associated with an organizational influence.

2.2.6 Human Organizations

Organizational structuring and design for human organizations has received considerable attention from a managerial and operations research perspective (Galbraith, 1973; Wood & Bandura, 1989; Rivkin & Siggelkow, 2003; Bernstein, 2012). While concepts developed for human organizations can provide useful insights in to how one might approach organizations for computational agents, it is important to note that human organizations are targeted towards a problem with inherently different requirements. That is, human organizations must address issues such as motivation, satisfaction, ego, personality conflicts, etc. that are not present with computational agents. Thus, many human-organizational techniques (e.g., inserting a manager between two individuals who dislike each other but must collaborate) are unnecessary for computational agents.

Similarly, computational agents present several challenges and opportunities for organizational techniques that are not present with humans. For example, the ways in which an organization can affect a human might be limited as compared to the ways in which it can (potentially) affect a computational agent. Human organization might be limited to providing (dis-)incentives for performing some action or reaching some state; however, for computational agents, in addition to (dis-)incentives, an organization might directly affect the ways in which an agent can observe the world, or the actions an agent can even consider executing. This additional expressive power creates opportunities for more rigid, fail-safe organizations (i.e., with provable guarantees on the agents' policies), but also increases search space size for creating an appropriate organizational design.

CHAPTER 3

Organizational Design Problem

Intuitively, the first step in my investigation of agent-driven organizational design is to formulate a well-defined organizational design problem. Viewing organizational design as search, I formulate an organizational design problem via two primary steps. First, I develop an agent-driven organizational specification language, which serves to define and delimit the organizational design (i.e., search) space. Second, I develop quantitative, agent-driven metrics of organizational performance, which serve to define the objective function over the organizational design space.

Like in the problem-driven and mixed organizational reasoning approaches discussed in Section 2.2, an important aspect of my agent-driven approach is representing a MAS's organization as an explicit, first-class object. The significant departure in my approach, however, is that I do not adopt an existing organizational specification language from the research literature or construct a new top-down knowledge-driven language, as is typical in prior organizational design research (e.g., see Section 2.2.2). Rather, I follow my agent-driven approach to construct a specification language that is mathematically grounded in the agents' reasoning framework. In Section 3.1, I describe my agent-driven specification language for Dec-MDP-based agents, and show how the constructs in my language are natively integrated into the agents' local reasoning processes. Then, I formally analyze the theoretical properties of my specification language in Section 3.2.

Leveraging the agent-driven foundation of my specification language, I identify quantitative metrics of organizational performance in Section 3.3, and use them to formulate a well-defined organizational design problem. One consequence of my agent-driven treatment of organizational specification languages is a theoretical understanding that *all* MASs have some form of (perhaps implicit) organization, and that the inherent default, baseline organization of a MAS—as well as the inherent demands on the MAS (i.e., problem episodes are neither trivial nor insurmountable)—can have substantial impact on the effectiveness of additional organizational design. In Section 3.4, I provide a more well-defined baseline against which to compare my organizational design techniques, derived from the premise that a MAS's default organization should be faithfully representative of the execution environment but should not contain further information to distinguish effective coordination patterns beyond what is necessary for faithful problem representation.

To demonstrate the potential for my organizational techniques to improve MAS performance as well as to empirically validate the effects that organizational influences in each of my language constructs has on MAS performance, in Section 3.5, I present empirical evaluations of organizations that I hand-construct and encode using my specification language. Finally, I briefly discuss the generality of my approach to other agent reasoning frameworks in Section 3.6.2 before concluding with a summary of the chapter's contributions and results in Section 3.7.

3.1 Specification Language Formalism

To illustrate my agent-driven approach for specifying an organization (I describe the generality of my approach more in Section 3.6), I commit to a factored Dec-MDP agent reasoning framework (Definition 2.5) where the agents have a locally-fully observable local state representation (Property 2.1). Consequently, this means the problem domain is modeled by $\mathcal{M} = \langle \mathcal{N}, \{\mathcal{M}_i\} \rangle$, where $\mathcal{N}$ is a set of n fully cooperative agents, and $\mathcal{M}_i = \langle S_i, \alpha_i, A_i, P_i, R_i, T_i \rangle$ is the local MDP model for agent i. (MDP semantics are given in Definition 2.1.) The objective of this section is to leverage the mathematical formalism of this reasoning framework to derive a principled organizational specification language where each of the constructs has well-defined effects on the agent(s) local reasoning processes.

3.1.1 Organizational Specification Language

At the center of my specification language is the concept of an *organizational influence*, so named because the idea is a generalization of inter-agent transition influences (Section 2.2.5) from operational to organizational reasoning. Intuitively, an organizational influence is an atomic piece of information that biases an agent's local reasoning process. Formally, I define an organizational influence as follows.[1]

[1] From this point forward, "influence" refers to an organizational influence (Definition 3.1) unless otherwise specifically noted.

Definition 3.1. *An **organizational influence** for agent i, $\Delta_i : (\otimes_j F_j) \times A_i \times (\otimes_k F_k) \times \mathbb{R} \mapsto (\otimes_j F_j) \times A_i \times (\otimes_k F_k) \times \mathbb{R}$ is a modification to $\mathcal{M}_i$ at $(s_i^t \in S_i) \times (a_i \in A_i) \times (s_i^{t+1} \in S_i)$.*

Note that for expressive generality, Definition 3.1 defines organizational influences in terms of modifications to a tuple of state factors, action, successor state factors, and a real number, but as we will see, this expressive power is not always necessary.

An organization is then defined in terms of organizational influences for each agent.

Definition 3.2. *An **organizational design**, $\Theta \equiv \langle \theta_1, \cdots, \theta_n \rangle$, where $\theta_i \equiv \{\Delta_i\}$ is the set of organizational influences for agent i.*

Analyzing the components of the agents' reasoning framework allows enumeration of the possible ways that an influence could modify $\mathcal{M}_i$. I step through these now, along with brief, high-level examples of why an organization might want to modify $\mathcal{M}_i$ in each of the modeling constructs. Together, modifications to each of these modeling components constitutes the specification language I use to express organizational influences to agents that reason with the Dec-MDP framework I have adopted.

- State factors: $(\otimes_j F_j) \subseteq S$. For example, this type of modification could inform agent i that, given its part in the organization, a locally-observable state factor is unimportant for deciding its local policy, or that a new, previously-unobservable state factor is critical for deciding its policy (in which case additional Δ_is are also necessary for the agent to correctly model this new factor).

- Action space: $a_i \in A_i$. For example, this type of modification could prevent agent i from considering and executing an organizationally undesirable action, or that previously-neglected actions are important to consider and/or execute. Such Δ_is are similar to constrained action choices in MARL (Section 2.2.5) but summarize expectations over the expected distribution of problem episodes rather than within a single episode.

- Transition function factors: $(\otimes_j P_j) \subseteq P_i$. For example, this type of modification could inform agent i of organizationally-determined non-local effects that other agents are expected to have on its local state space. Such Δ_is are similar to operational transition influences (Section 2.2.5) but summarize expectations over the expected distribution of problem episodes rather than within a single episode.

- Reward function factors: $(\otimes_j R_j) \subseteq R_i$. For example, this type of modification could persuade agent i into executing actions and/or achieving states that might look poor from a local perspective, but actually contribute positively to collective performance (or *vice versa* for dissuading actions/states). Such Δ_is are similar to reward shaping (Section 2.2.5) but summarize expectations over the expected distribution of problem episodes rather than within a single episode.

- Initial state distribution factors: $(\otimes_j \alpha_j) \subseteq \alpha_i$. For example, this type of modification could inform agent i about the initial distribution over new, previously-unobservable state factors.

- Time horizon: T_i. For example, this type of modification could inform agent i that, given its part in the organization, the agent can expect its local actions to conclude sooner, and thus it should plan for fewer decision points (or *vice versa* for a longer planning horizon).

3.1.2 Incorporating Organizational Influences into Local Reasoning

Broadly, there are three classes of modifications that can be done to a model component: adding an entirely new factor, removing an existing factor, or overwriting an existing factor. When incorporating an organizational specification into its local model, each agent i overlays θ_i onto $\mathcal{M}_i$ to create its augmented model, $\mathcal{M}_i^{|\theta_i}$, by modifying its $\mathcal{M}_i$ according to each $\Delta_i \in \theta_i$. That is, each agent adds new factors to its model, removes existing factors from its model, and overwrites existing factors in its model as directed by its respective organizational influences. This overlaying process thus resembles how, for example, coordination locales model domain dynamics by overriding an agent's local transition/reward models (Varakantham et al., 2009; Velagapudi et al., 2011), and social model shaping augments those local models to coerce coordination (Babes et al., 2008). Each agent then computes its optimal local policy, $\pi_i^{*|\theta_i}$, from $\mathcal{M}_i^{|\theta_i}$ in exactly the same way as it did before.

To illustrate this overlaying process, consider a simple organization for the fire-fighting domain that designates a region of responsibility for each agent to consider fighting fires within. Suppose the organization specifies these responsibility regions by removing fire intensity factors (I_cs) from the agents' state representation for fires outside of the agent's region, adding a new reward factor for being located within its region, and overwriting the I_c-transition factor to account for I_cs in its region decreasing over time (due to other agents' efforts). Figure 3.1a illustrates agent i's $\mathcal{M}_i$ as a two-stage dynamic Bayesian network (2DBN), and Figure 3.1b illustrates the

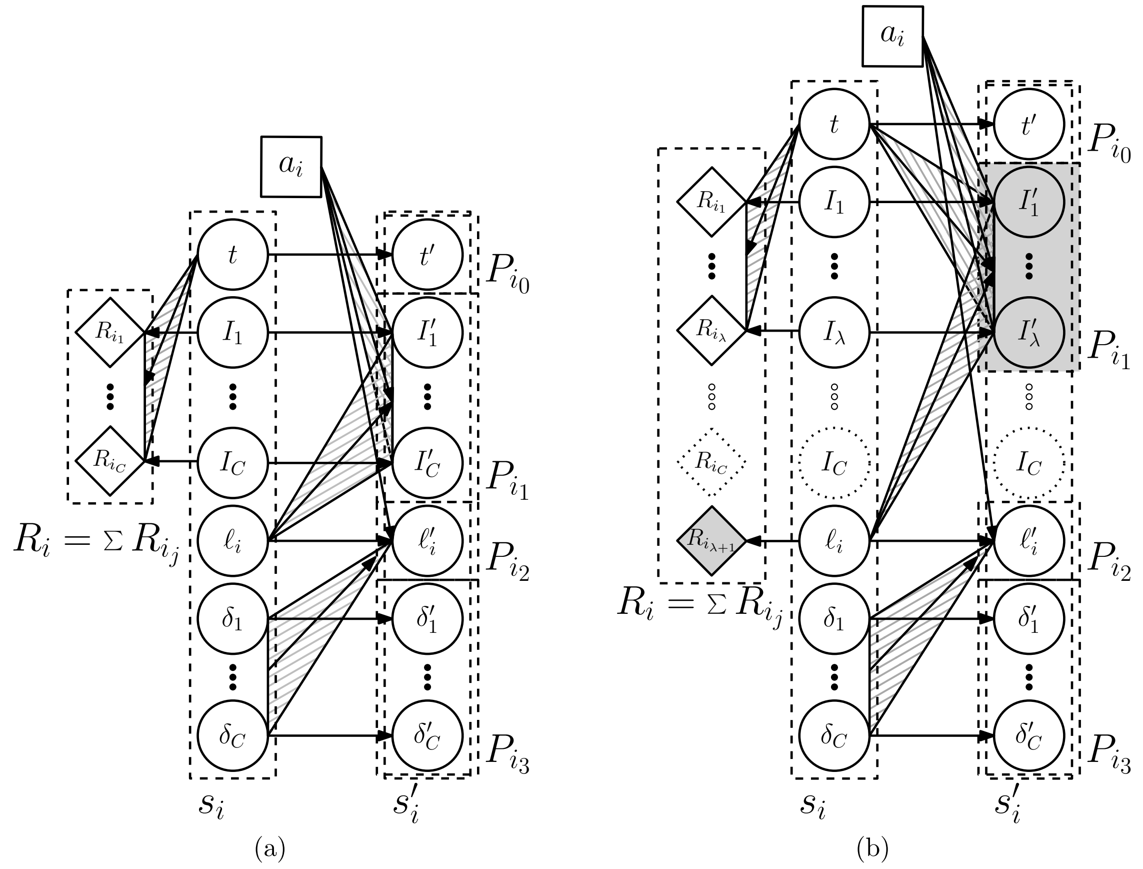


Figure 3.1: (a): An example $\mathcal{M}_i$ for the firefighting domain represented as a 2DBN. (b): An example $\mathcal{M}_i^{|\theta_i}$ created by organizationally modifying the $\mathcal{M}_i$ from (a). Shaded regions indicate factors that were organizationally overwritten or added, while dotted regions indicate factors that were organizationally removed.

$\mathcal{M}_i^{|\theta_i}$ obtained after incorporating the example organization, where agent i is assigned responsibility for cells 1 through λ.

To prevent several under-defined scenarios that could occur when overlaying θ_i onto $\mathcal{M}_i$ to create $\mathcal{M}_i^{|\theta_i}$, I make the following restrictions on valid Θ specifications:

- The influences must be internally consistent. That is, a well-formed Θ can not contain two contradictory Δ_is. For example if one Δ_i modifies a transition factor in $\mathcal{M}_i$ to make $P_i(f_k^{t+1}|s_i^t, a_i) = 0.5$ for state factor F_k, then another Δ_i cannot modify that factor to $P_i(f_k^{t+1}|s_i^t, a_i) = 0.2$.

- A valid Θ must be well-formed with respect to $\mathcal{M}_i$. That is, an influence in Θ cannot modify something that does not exist in $\mathcal{M}_i$, except by adding new modeling factors. For example, Δ_i cannot remove a state factor that does not exist in $\mathcal{M}_i$.

- A Θ must leave each agent with a well-defined MDP. Specifically, $\mathcal{M}_i^{|\theta_i}$ must contain: a finite time horizon; at least one state factor; at least one action in every state; at least one reward factor; transition factors that cover $\mathcal{M}_i^{|\theta_i}$'s state factors; and initial state distribution factors that cover $\mathcal{M}_i^{|\theta_i}$'s state factors.

Note that these additional restrictions on valid Θs could potentially be relaxed if the agent and ODP iteratively communicated with each other to resolve any such inconsistencies; however, for simplicity, in this dissertation I will require these restrictions to enforce that Θ is consistent with what the agents can internally model.

3.2 Language Properties

A primary advantage of my agent-driven approach for creating an organizational specification language is that the mathematical formalism of the agents' underlying reasoning framework is inherited by the specification language, which provides the necessary axioms to theoretically analyze my specification language and formally prove several important characteristics.

3.2.1 Size of Organizational Design Space

Naïvely examining my definition of organizational influence (Definition 3.1) and organization (Definition 3.2) reveals that a distinct organization exists for every legal combination of modifications that could be made to the agents' local models. By stepping through each of the model components, one can conclude that there are an uncountably infinite number of possible modifications than an organization specification could possibly represent. More precisely, if one assumes a finite number of modeling factors, the set of possible modifications has cardinality $\beth_1$. This can be seen by considering, for example, that modifications to the reward constructs include a real number (i.e., an influence could modify a reward to be any real value), thus the set of reward modifications alone has cardinality $\beth_1$. Taking the cross product between (a finite number of) factors with cardinality no higher than $\beth_1$ yields the final cardinality result of $\beth_1$.

A more pragmatic perspective for viewing an organizational influence, however, is that Δ_i represents a constraint and/or re-prioritization of agent i's local policy space (brought about via a modification to $\mathcal{M}_i$). The difference in perspective allows us to see that multiple influences could have the same net effect on the agent's reasoning and behaviors; for example, a Δ_i that modifies $R_i(s_i^t, a_i, s_i^{t+1}) = 4.3$ may have the same effect on agent i's reasoning and behaviors as an alternative Δ_i' that modifies $R_i(s_i^t, a_i, s_i^{t+1}) = 4.3 + \epsilon$, where ϵ is some small constant. From this perspective, the size of the pragmatically-distinct organizational design space is given by the number of different joint policy spaces an organization could induce a MAS to consider, and equates to the number of total orderings of every subspace of the joint policy space. Formally, this quantity is given by $\sum_{i=1}^{|\boldsymbol{\pi}|} \frac{|\boldsymbol{\pi}|!}{(|\boldsymbol{\pi}|-i)!} = O(|\boldsymbol{\pi}|!)$, where $|\boldsymbol{\pi}|$ is the cardinality of the joint policy space. The above quantity is finite for the Dec-MDP framework I have adopted; however, unsurprisingly, direct enumeration of the organizational design space is computationally intractable. Consequently, the automated ODP techniques I develop in later chapters only search the design space for an approximately optimal organization.

3.2.2 Language Completeness and Necessity

Intuitively, my organizational specification language is complete since it can modify any of the agents' reasoning components. This idea serves as the premise for a more formal proof.

Theorem 3.1. *My organizational specification language (Section 3.1) is complete for the Dec-MDP reasoning framework I have adopted.*

Proof. **By Contradiction.** Assume my specification language is not complete, then by definition there must exist an influence to the agents' decision making processes that is not expressible with my specification language. However, by construction, my specification language covers the full range of information expressible to an agent in the Dec-MDP framework I have adopted. That is, beyond state factors, initial state distribution factors, actions, transition factors, reward factors, and time horizon, there does not exist another mechanism that such an agent uses to make decisions. Thus, there cannot exist an influence to the agents' decision making processes that is not expressible with my specification language, which is a contradiction. □

Leveraging the mathematical foundations of the Dec-MDP framework I have adopted, it is also possible to formally prove the necessity of each construct in my organizational specification language. Broadly speaking, these proofs each construct an example organizational influence that an ODP might want to express for various reasons, and then formally prove by exhaustion that a particular construct in the specification language is the only way to express that influence. In many of these examples, a critical aspect is that an agent has local expertise that is not known to the ODP (i.e., Property 1.3). I discuss a particularly evident example of when agents possess local expertise and how this information uncertainty can be compactly represented in an ODP (i.e., abstract organizational influences) more thoroughly in Chapter 5. However, in order to present my formal proofs of language necessity as clearly as possible, I have elected to utilize the associated terminology that I develop in that chapter for discussing such issues. As such, to avoid unnecessarily confusing readers with terminology that has yet to be presented, the formal proofs for language necessity can be found in Appendix A.

It is worth noting that under special, more-restrictive circumstances, some of the specification language constructs may be unnecessary (though the language will always be complete). For example, suppose an ODP knows a specific policy π (e.g., the optimal policy π^*) that it wants the agents to always execute for every episode—e.g., an assembly line in a factory. In this case, the organizational specification could prevent the agents from considering any action counter to π by modifying their action spaces, and all the other specification language constructs are thus unnecessary. However, as formally proven in Appendix A, all of the language constructs are necessary within the general space of possible organizational design problems.

3.3 Measuring Organizational Performance

Building on the organizational design space defined by my agent-driven specification language in Section 3.1, in this section I identify principled, quantitative metrics of organizational performance to serve as an objective function over that design space. Together, my specification language and performance metrics constitute a well-defined, agent-driven organizational design problem (Definition 3.6 below).

Following my agent-driven approach, the performance of an organization reflects the performance of the MAS while solving problem episodes. For the Dec-MDP model I have adopted, the metrics of interest are the expected joint reward and expected computational costs of the MAS, which are quantitatively defined as follows:

Definition 3.3. *The **operational reward** under Θ, $\mathbb{R}_{Op}(\Theta)$, is given by the expected joint reward of the agents' joint policy w.r.t. their organization, $\pi^{|\Theta}$:*

$$\mathbb{R}_{Op}(\Theta) \equiv \sum_{s^0 \in S} \alpha(s^0) \sum_{a \in A} \pi^{|\Theta}(s^0, a) Q^{\pi^{|\Theta}}(s^0, a)$$

Definition 3.4. *Assuming agents reason in parallel, the **operational reasoning cost** under Θ, $\mathbb{C}_{Op}(\Theta)$, is given by the expected operational reasoning cost for an agent* *to calculate its individual $\pi_i^{*|\theta_i}$, notated as $C(\pi_i^{*|\theta_i})$:*

$$\mathbb{C}_{Op}(\Theta) \equiv E_i \left[C(\pi_i^{*|\theta_i}) \right]$$

Throughout this dissertation, I compute $C(\pi_i^{*|\theta_i})$ as the agents' actual CPU time required to compute $\pi_i^{*|\theta_i}$.

Definition 3.5. *The **operational performance** of Θ, $\mathbb{P}_{Op}(\Theta)$, selects from the Pareto front of the operational reward and operational reasoning cost:*

$$\mathbb{P}_{Op}(\Theta) \equiv f\left(\mathbb{R}_{Op}(\Theta), \mathbb{C}_{Op}(\Theta)\right)$$

Naturally, the specific form of f is defined by the problem domain; however, throughout this dissertation, I will assume that f is monotonic in each dimension such that higher $\mathbb{R}_{Op}$ and lower $\mathbb{C}_{Op}$ is preferable.

Given these performance metrics, it is straightforward to formulate a well-defined organizational design problem.

Definition 3.6. *The **optimal organization**, Θ^* is the Θ with maximal operational performance.*

$$\Theta^* \equiv \arg\max_{\Theta} \mathbb{P}_{Op}(\Theta)$$

As just shown in Section 3.2.1, enumerating the organizational design space is computationally intractable, and thus an important aspect of my research in future chapters is the identification of more efficient techniques for (approximately) solving this organizational design problem. For now, however, Definition 3.6 provides a well-defined organizational design problem, and consequently also provides a method for comparing alternative organizations.

3.4 Baseline Organization

As described in Section 3.1.2, an organizational design modifies the agents' local models by adding new factors, removing existing factors, and/or overwriting existing factors. This approach raises an important question: Where does an agent's (original) local model come from? Clearly, the opportunities for an organization to impact a MAS depends on how (dis-)organized the agents are when following their initial local models. Consequently, one could demonstrate arbitrarily good performance improvements from an organizational design by initializing the agents that will adopt the design with arbitrarily bad local models to begin with.

Examining the following biconditional relationship $\mathcal{M}_i^{|\theta_i} = \mathcal{M}_i \leftrightarrow \theta_i = \emptyset$, we can see that the combination of initial local models of agents essentially *do* comprise an organization. This observation is essentially an embodiment of the experience-driven aspects of my problem-drivenapproach, where a MAS's organization is implicitly observable in the agents' joint actions. Alternatively, viewed from a problem-driven perspective, when assembling a MAS, agents might be selected based on the inherent alignment between their local models and the (organizational) biases of whomever is assembling the system. The actions agents are capable of, the states they can represent, their predispositions about what states are rewarding, etc. can all factor into decisions about which agents are included in the system.

Evaluating my agent-driven approach thus depends on defining a baseline organization. To develop as even-handed a baseline as possible, I advocate initializing local decision models by performing an uninformed mapping of the joint Dec-MDP models into localized versions. In this way, the local models are perforce aligned with the global model, but they are not crafted to differentiate and/or coordinate the policies of the agents. Essentially, the philosophy is to endow each agent with a local model that directly makes the individual agent responsible for solving the global problem, to the extent its awareness and capabilities allow.

Specifically, my methodology for initializing agents' local models to provide an experimental baseline is as follows:[2]

1. The subset of state features directly observable to the agent defines its local state representation.

2. The action space of an agent is simply its component of the joint action space.

3. The local reward function is the same as the global reward function, except that any components involving features outside of the agent's local state representation are dropped, since the agent does not have values for those features.

4. The local transition model corresponds to the joint transition entries where the other agents do not affect the agent.

5. The initial local state distribution maps the global distribution into the local state space.

6. The local finite time horizon is identical to the global value.

[2] Throughout this dissertation, I use the terms "local organization", "baseline organization", and "local baseline" interchangeably to refer to the baseline MAS constructed using my methodology.

In the firefighting domain, the baseline organization is the local model as I have previously described in Section 2.1.4.1.

While this method for creating a baseline model is still dependent on somewhat arbitrary decisions (e.g., which features are included in an agent's local state), the idea is that aspects that influence how an agent formulates a policy (what is rewarding, what might happen in the world, etc.) are aligned with the "true" global model but contain as little information as possible about what an agent might expect others to do in the world. An ODP's primary objective is to provide such information.

Despite adopting this uninformed-but-aligned baseline, other factors can also influence the difference that an organizational design can make. A simple example is that the initial configuration of state can greatly affect whether the baseline organization is effective. In the firefighting domain, if we suppose that the fires pop up across the space with uniform probability, then where should the agents initially be located? If they are uniformly distributed in the environment, then their local models (where they tend to prefer fighting nearby fires) inherently lead to a good allocation of fires to agents, and results in a relatively high-performing local baseline. If they instead all start in the same location, on the other hand, then the local models inherently lead to agents moving around *en masse* and yields no parallelism benefits (but note that if multiple firefighters on the same fire had a super-additive effect, instead of a sub-additive effect, then initially spreading out could be disadvantageous, while moving around in a pack might be beneficial). Even randomly placing firefighters is not conclusive, because distributing fires and agents in the same uniformly random way introduces its own bias. To mitigate these concerns in my later empirical evaluations, I present results from the two extreme initial conditions: the agents beginning uniformly distributed; and the agents beginning clustered in the center of the grid world, which represents the best and worst case in expectation for the baseline organization respectively.

Beyond the effects that the baseline organization can have on organizational performance, the inherent demands on the MAS can also have substantial impact on improvement from organizational design (Corkill et al., 2015). That is, the nature of the domain impacts the degree to which an organization matters. For example, in the firefighting domain, if there were only one fire per problem episode, then an organization cannot improve joint reward because the agents can achieve the optimal joint policy without coordination (although an organization could reduce the agents' computational costs). Similarly, if every grid cell contained a fire, then organization also does not improve joint reward because severe over-utilization of the agents' capabilities precludes meaningful joint interactions (though again, an organization could reduce the agents' computational costs). In these cases, an organization could be a significant improvement for some $\mathbb{P}_{Op}$ Pareto topologies (where low computational costs are important), but will not improve the quality of the agents' joint policy. However, organizations can improve the MAS's joint reward when the agents' capabilities are in the middle, transient portion of domain "difficulty". We will explicitly observe these effects more in Chapter 4 (e.g., Figure 4.1 illustrates a manifestation of this observation in the firefighting domain with two fires per episode).

3.5 Empirical Demonstration

While the semantics and impact of influences to each of the agents' modeling components is well-defined by the agents' reasoning framework, it is nonetheless illustrative to demonstrate that my specification language can be practically used to specify an organization for a MAS. In this section, I describe two empirical evaluations performed using organizations that I hand-constructed and tuned for use in the firefighting domain. I begin next, by describing the overarching process that I use for empirically evaluating organizations in a MAS throughout this dissertation. In Section 3.5.2, I use my specification language to demonstrate how organizations can be used to improve the performance of a MAS. Then, in Section 3.5.3, I investigate the effects that encoding organizational influences in each of the model components has on MAS performance.

3.5.1 Empirical Evaluation Process

In this section, I describe several aspects of my empirical evaluation pertaining to simulation details. Unless otherwise explicitly noted, the details I describe in this section hold for *all* of the empirical evaluations I perform in this dissertation.

Episodic performance. To test the degree to which an organization provides long-term benefit to a MAS, I run a fixed organization over a space of randomly-generated problem episodes, where by the luck of the draw, some episodes might be well suited to one organization over another. I focus on expected performance over episodes not only to smooth out the randomness of the episodes but moreover to identify an organization's effectiveness over the long term, since an organization should be relatively stable over an extended period (i.e., Property 1.5). To reduce the statistical variance of my results, each organization that I evaluate encounters exactly the same problem episodes (for a given experiment) in exactly the same order (though since the organization is fixed, episode order does not impact performance).

Agents' Operational Reasoning. In each episode, each agent uses its $\mathcal{M}_i^{|\theta_i}$ (as computed via the methods in Section 3.1.2) to locally compute its $\pi_i^{*|\theta_i}$. To create its local policy, an agent uses CPLEX (IBM, 2012) to calculate the optimal local policy for its reachable state space (see below for an overview of reachability) using the linear program as formulated by Kallenberg (1983) (Equation 2.4). The joint policy is then simply defined as $\pi^{|\Theta} = \langle \pi_1^{*|\theta_1}, \pi_2^{*|\theta_2}, ..., \pi_n^{*|\theta_n} \rangle$, which is not ensured to be optimal with respect to Θ (i.e., $\pi^{|\Theta}$ is distinct from $\pi^{*|\Theta}$), since the agents do not perform any explicit operational coordination. (See the "Communication" paragraph below for a discussion of runtime coordination in my experiments.)

In the firefighting domain, it is the case that an agent i's *reachable* state space from a given initial state, which I denote as $\tilde{S}_i(S_i^0)$, for a problem episode, is several orders of magnitude smaller than the complete state space. A state is reachable, $\tilde{s}_i^t \in \tilde{S}_i(S_i^0) \subseteq S_i$, if and only if there exists a policy, π_i (note that this policy need not be optimal), such that by beginning in some $s_i^0 \in S_i^0$ and following π_i, $\tilde{s}_i^t$ will be reached with non-zero probability. The reachable state space can be directly enumerated using Algorithm 3.1, which performs breadth-first search by iteratively "unrolling" the reachable state space from S_i^0 forward via the transition function (Wu & Durfee, 2007).

Reachability is important because, if $|\tilde{S}_i(S_i^0)| \ll |S_i|$ as it is here, then creating a policy for only the reachable states can result in substantial computational savings as compared to creating a policy for the entire state space. As such, in my empirical evaluations the agents only create policies for their respective reachable state spaces (w.r.t. $\mathcal{M}_i^{|\theta_i}$). However, since an agent i is unaware of the actions of the other agents at policy creation time, and the effects those actions have on agent i's local state space are also unknown, it is common for an agent to "fall off policy" during execution (i.e., reach a state that was thought to be unreachable). In these cases, I assume that agent i extends its current policy to include the reachable state space from the new state forward, and that this planning is instantaneous with respect to events in the world. For the purposes of my experiments this assumption favors less informed organizations (that fall off policy more frequently) more than informed ones. As my empirical results in the next sections show, the organizations I create in my experiments are more informed than the local baseline, and thus they typically fall off policy less frequently. Consequently, the benefits of an organization relative to the local baseline will tend to

Algorithm 3.1 Reachable state space enumeration via "unrolling"

```
Input: The agent $i$'s MDP model $\mathcal{M}_i$, set of possible starting states $S_i^0$
Output: The set of states reachable from $S_i^0$, $\tilde{S}_i(S_i^0)$
 1: $\tilde{S}_i(S_i^0) \leftarrow \emptyset$
 2: List $l \leftarrow S_i^0$
 3: while $l$ is not empty do
 4:     $s_i^t \leftarrow l$.pop
 5:     if $s_i^t \notin \tilde{S}_i(S_i^0)$ then
 6:         $\tilde{S}_i(S_i^0) \leftarrow \tilde{S}_i(S_i^0) \cup s_i^t$
 7:         $l$.add($\forall s_i^{t+1} \in S_i$ s.t., $\exists a_i \in A_i$ where $P(s_i^{t+1}|s_i^t, a_i) > 0$)
 8:     end if
 9: end while
10: return $\tilde{S}_i(S_i^0)$
```

be, if anything, *understated*.

Communication. A commonly studied aspect of many MASs is communication; for example, in Dec-MDPs, agents could exchange messages to share non-local information or coordinate joint actions. As described in Section 2.2.2, inter-agent communication has also been studied within the context of organizations, where an organization might streamline the agents' communication by influencing when the agents initiate communication and what information is exchanged. In the firefighting domain, it is intuitive that runtime communication to coordinate the agents' local policies (i.e., to compute $\pi^{*|\Theta}$ rather than $\pi^{|\Theta}$) could increase $\mathbb{R}_{Op}$ (at the expense of increased $\mathbb{C}_{Op}$), and gainfully augment an organizational approach. For example, an organization could inform the agents that each agent should send a message to inform the others of which fire it is currently pursuing, which would ensure the agents do not redundantly pursue the same fire.

In my empirical experiments, however, I prevent *all* of the agents' communication, instead forcing each agent to make decisions from its local perspective, and moreover forcing all non-local information to come from the organization. Intuitively, this pushes the problem domain to an extreme parameterization, and presents a compelling evaluation space that stresses the ODP to identify an organization that does not rely on runtime coordination to facilitate the agents' interactions. Additionally, preventing communication eliminates a confounding factor in my results since the agents are unable to use runtime coordination to overcome a poorly designed organization. For example, in the firefighting domain, agents could achieve optimal coordination with a relatively simple communication protocol (i.e., exchange information about which fire each agent should pursue), and consequently only poorly designed organizations (i.e., that preclude the optimal joint policy) would not achieve the optimal joint reward.

In Dec-MDP frameworks, communication is often modeled as an action, albeit communicative actions are explicitly, distinctly represented (Pynadath & Tambe, 2002; Goldman & Zilberstein, 2003). One natural approach for incorporating communication into my representations and algorithms would be to simply represent the agents' various communication capabilities as actions in the agents' action spaces. While this simplistic approach would not theoretically require any modifications to my organizational representations and algorithms, it is arguably an oversimplification in many cases. That is, communicative actions have qualitatively different effects on the MAS's decision problem as compared to non-communicative actions. For example, communicative actions often have delayed-effects, a distinct class of resource constraints (e.g., bandwidth), etc.—indeed, this is a reason that communicative actions are distinctly modeled in the prior works cited above. Thus, while it seems intuitive that the techniques I develop in this dissertation should extend to communication, identifying the best, practical way to incorporate communication into my representations and algorithms is an interesting direction for future work.

One particular way that I have identified where influencing the agents' communication could be especially significant is if the organization adds a new state factor to the agents' state representation. For example, a new state factor could act as a blackboard (Corkill, 1991) for facilitating cooperation among the agents. In this example, influences to the agents' communication protocols could inform the agents of when and how they should modify the blackboard.

Agent Interactions in the Firefighting Domain. In the firefighting domain, it is intuitive that the agents could potentially interact in a variety of ways. For example, simultaneously fighting the same fire could have: a redundant effect where multiple agents have diminished returns or punitive effects (i.e., sub-linear scaling); an additive effect where each agent contributes the same regardless of how many other agents simultaneously fight a fire (i.e., linear scaling); or a bonus effect where multiple agents as a team are better than a group of individuals (i.e., super-linear scaling). While the linear scaling case is less interesting since joint interactions are essentially unimportant, both the sub- and super-linear cases have joint interactions that can force the MAS to cooperate in order to achieve high performance. Throughout this dissertation, I somewhat arbitrarily decided between the sub- and super-linear cases, and elect to experiment with the sub-linear case where multiple agents have the same effect as a single agent when fighting a fire (i.e., simultaneously fighting a fire in a cell is completely redundant). However, the super-linear case would have been equally valid, and the important aspect here is that the domain has joint interactions that force the MAS to cooperate, where failure to effectively cooperate diminishes performance.

Another intuitive way for the agents to interact is in their movement actions, where simultaneously occupying the same grid cell could be: prevented (e.g., agents bounce off of each other); punished (e.g., collisions cause damage to each other); positively rewarded (e.g., allows agents to share supplies or information); ignored (i.e., no effect for occupying the same cell); etc. In this dissertation, I elected to not associate any special effects with simultaneously occupying the same grid cell (i.e., the ignored case above). This choice essentially reduces the number of joint interactions in the domain, which is arguably a drawback; however, it also delimits the space of joint interactions to the firefighting actions, which affords me greater control of the evaluation domain.

3.5.2 Specification Language Demonstration

To provide a more concrete example of how my specification language can be used to encode an organization, I step through specifications for several hand-designed organizations, and then use them to illustrate how an organization can increase $\mathbb{P}_{Op}$ compared to the baseline MAS. I designed the example organizations in this section for a relatively straightforward two agent version of the firefighting domain, which allows comparison of $\mathbb{P}_{Op}$ against not only the local baseline but also the optimal joint policy, which can be computed for these small problems.

Specifically, for this set of experiments, the domain is a simple 10×5 grid world with 2 cooperative agents and 2 fires, as illustrated in Figure 3.2. The distribution of fires' locations in the initial state is uniformly random (without replacement) over the entire grid, and the initial fires' intensities are i.i.d. uniformly random over $\{1, 2, 3\}$. The agents always begin each episode in the same locations (those in Figure 3.2). Note that in this set of experiments there are no cell delays—movement actions are deterministic. To speed up the tests without pruning any viable solutions, the finite time horizon, T, is the maximal time either agent would require to put out both fires alone (varies per episode). Each agent's local reward (as well as the joint reward) in a state prior to reaching T is $-\sum_c I_c$. When T is reached, the problem episode ends, and each agent's local reward is $-10\sum_c I_c$, encouraging the agents to put all the fires out before the deadline.

For each organization that I develop in this section, the underlying structure is that each agent is assigned a *primary area of responsibility (PAR)*. The first organization,

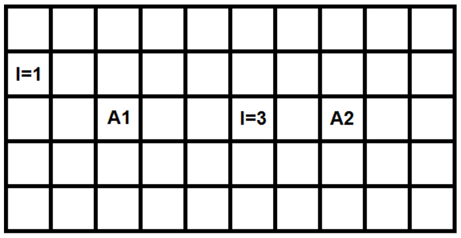


Figure 3.2: Example initial state of a 10×5 firefighting grid world domain. Ai is the position of agent i, and $I = x$ indicates that there is a fire in that cell with intensity x.

called **fullOverlapOrg**, assigns both agents to be responsible for all cells in the entire grid. However, unlike the baseline organization where agents have no model of each other, fullOverlapOrg influences the agents' transition factors to account for the expected non-local effects of the other agent. Specifically, the organizational specification encodes the heuristic assumption that an agent will always move (within its PAR) to fight the fire closest to its current location. So, the organizational influences modify the other agent's transition function to anticipate that some fires (typically on the other side of the grid) will have decreasing intensities even without fighting them itself, helping it refrain from rushing to distant high-intensity fires that will be addressed by the other agent.

A second organization, called **partitionOrg**, partitions the grid cells, assigning responsibility for fires in the western 5×5 subgrid to A1, and the eastern subgrid to A2. This assignment is encoded by preventing the agents from considering a local action that would move them out of their respective PARs (i.e., modifying the agents' action spaces so those actions are no longer available in the appropriate states).

The third organization is called **smallOverlapOrg**, in which the 4 middle columns of the grid are in both agents' PAR. Like in partitionOrg, agents' action spaces are modified so that an agent doesn't consider moving out of its PAR. Additionally, like fullOverlapOrg, the smallOverlapOrg modifies each agent's transition factors to reflect that fire intensity state factors in its PAR (i.e., those in the overlapping PAR space) have a chance of going out without it fighting them.

I generated 1,500 episodes with random initial states (as described above) and the agents solved each using the 3 organizational designs (partitionOrg, smallOverlapOrg, and fullOverlapOrg), as well as the uninformed baseline organization. I also used CPLEX with the joint problem specification to generate the optimal joint policy for each episode to compute the optimal attainable $\mathbb{R}_{Op}$, if the agents could afford the computational costs to generate it.

These experiments are summarized in Table 3.1, and highlight several important

	$\mathbb{R}_{Op}$	$\mathbb{C}_{Op}$	Replans
Baseline	-15.97	86	1.32
ParitionOrg	-16.15	12	0.26
SmallOverlapOrg	-14.74	27	0.16
FullOverlapOrg	-14.70	70	0.14
Joint Policy	-14.37	24558	0.00

Table 3.1: Results for experiments in Section 3.5.2 for $\mathbb{R}_{Op}$, $\mathbb{C}_{Op}$, and expected number of times the replanning mechanism was invoked per agent per episode.

points. Firstly, even these simple organizations can improve $\mathbb{R}_{Op}$ considerably compared to the baseline, but overly restrictive organizations (e.g., partitionOrg) can degrade $\mathbb{R}_{Op}$ because the same agent too often must fight both fires. As one would expect, more restrictive organizations increasingly simplify agents' local decision problems, resulting in lower $\mathbb{C}_{Op}$. Moreover, note that all of the organizations decrease $\mathbb{C}_{Op}$ compared to the baseline, because in the baseline both agents solve larger problems (plan to put out all the fires by themselves) than when they are informed (e.g., through the transition function) that they will have help.

Note that the fullOverlapOrg provides greater global awareness to the MAS than the other organizations, which allows that organization to achieve superior $\mathbb{R}_{Op}$ at the expense of increased $\mathbb{C}_{Op}$. Unsurprisingly, Table 3.1 illustrates that $\mathbb{R}_{Op}$ is inversely correlated with $\mathbb{C}_{Op}$. To provide a sense of when a system administrator should prefer each of the organizations, I encoded a Pareto front of $\mathbb{P}_{Op} = \mathbb{R}_{Op} - \frac{1}{b}\mathbb{C}_{Op}$, which is shown in Figure 3.3. Figure 3.3 confirms intuitions that when computation is expensive (low b) paritionOrg is best due to its highly simplified decision process. Then as computation becomes cheaper (b increases), the more flexible organizations become superior, and finally when computation is very cheap, computing the optimal joint policy becomes best. However, note that the local baseline is never optimal.

3.5.3 Isolated Impact of Specification Constructs

To provide an empirical perspective of the impact of specifying an organization along different influence specification dimensions, I developed another set of hand-designed organizations for the firefighting domain. Unlike Section 3.5.2 where I evaluated different organizational structures (i.e., different methods for allocating PARs), in this set of experiments, I encode a single overarching organizational structure in a variety of ways using the different specification language constructs individually.

For this set of experiments, the domain has 10 cooperative agents and 10 fires

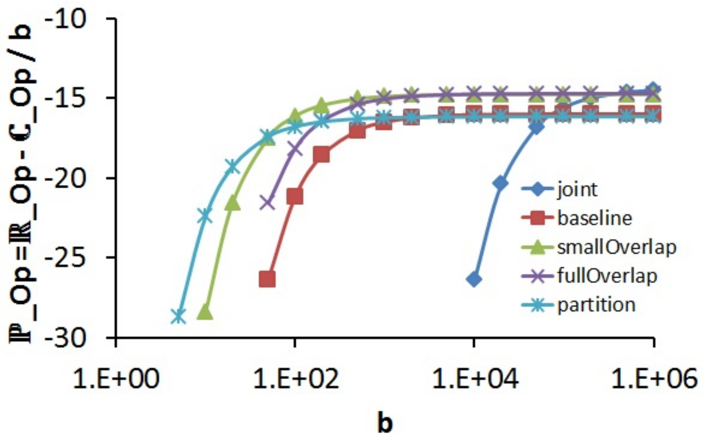


Figure 3.3: $\mathbb{P}_{Op}$ curves for each organization.

on a 25×10 grid. The rest of the experimental domain parameters for this set of experiments are identical to those in Section 3.5.2.

The overarching structure for the organizations I develop in this section inherits from the smallOverlapOrg previously described (Section 3.5.2), which leverages the notion of organizationally-directed PARs to narrow the local policy space that each agent respectively considers while still providing the agents with some flexibility to load balance by encoding overlapping regions of responsibility. Specifically, the 25×10 grid is divided into 10 distinct 5×5 subgrids, one for each agent, to act as the agent's PAR (Figure 3.4). In each direction (where cells exist), the subgrid is expanded by 3 cells to introduce overlap; conceptually, this is an agent's secondary area of responsibility (SAR). I encoded five organizations that implement this fundamental structure in different ways:

- **actionOrg** modifies an agent's action space to prevent it from moving out of its combined PAR and SAR.
- **stateOrg** modifies an agent's state factors so that it cannot model the fire intensity state factors for cells outside of its combined PAR and SAR.
- **rewardOrg** modifies the agent's reward factors to penalize the agent with increasing severity for leaving its PAR (specifically, Manhattan distance from the PAR squared).
- **transitionOrg** modifies the agent's transition factors to model how fires in its PAR and SAR might go out due to another agent's actions using the same heuristics as in the fullOverlapOrg from Section 3.5.2. Additionally, like stateOrg,

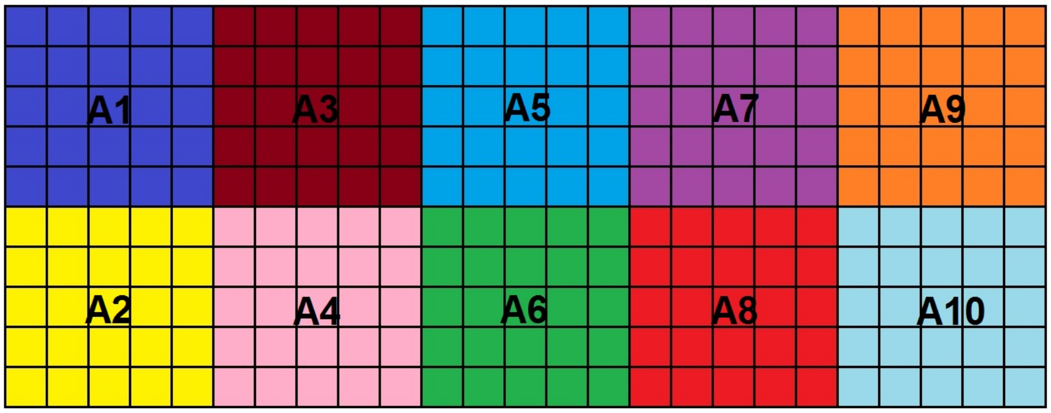


Figure 3.4: Illustration of PARs for 10 agent organizations.

transitionOrg modifies the agent's state factors to prevent it from representing fire intensity state factors for cells outside its combined PAR and SAR, which is necessary to curb the substantial increase to an agent's reachable state space resulting from the richer transition model.

- **fullOrg** uses all of the dimensional levers just described.

Recalling from Section 3.4, the effectiveness of the local baseline can depend on the problem domain, especially on the available opportunities to improve coordination by organizationally differentiating agents' policies to yield effective interactions. To provide a general idea of these performance bounds, in these experiments, I consider two extreme cases of initial locations for the agents (corresponding to the "best" and "worst" case for the baseline respectively): where they are evenly spread throughout the grid; and where they are clustered at the center of the grid. Note that these cases do not represent formal bounds on performance (indeed the formally-provable bounds are so loose as to be rendered practically meaningless), and it is possible to construct domains where the local baseline would perform (arbitrarily) better/worse than found in these cases. The point of presenting results from these two cases is merely to demonstrate the effects that the problem domain can have on baseline effectiveness, and to evaluate organizational effectiveness in domains indicative of good/poor baseline performance.

I generated 100 random episodes using the initial state distribution previously described, for each of the spread and clustered variations of agents' initial locations. For each episode, I evaluated each of the five organizations above, as well as the baseline organization—computing optimal joint policies to compare against for these problems was computationally intractable. For each episode, I recorded each organization's $\mathbb{R}_{Op}$,

$\mathbb{C}_{Op}$, and number of replanning invocations just as was done in Section 3.5.2.

Tables 3.2 and 3.3 present the results for these experiments, and illustrate many of the intuitions for how each influence construct impacts an agent's decision-making process. As others have discovered, reward influences (aka reward shaping) can be a powerful tool for increasing $\mathbb{R}_{Op}$; however, they do not generally reduce $\mathbb{C}_{Op}$ since an agent must still consider every policy even though some turn out to have less reward. Influencing the agents' transition functions can also yield a large increase to $\mathbb{R}_{Op}$; however, doing so substantially increases $\mathbb{C}_{Op}$ since the richer transition model increases the size of the reachable state space. Notice that organizations with modified transition functions invoke the replanning mechanism much less, indicating that if recovering from falling off policy incurs non-negligible penalties (i.e., results in decreased $\mathbb{R}_{Op}$ or increased $\mathbb{C}_{Op}$), then transition influences could be the implementation strategy of choice. Note that the Baseline having no replanning invocations on Table 3.3 is because the agents have exactly identical local policies, thus move around *en masse* and each think they are the one responsible for fighting the fires. I also find that influencing the agents' action or state spaces can greatly simplify the agents' decision problems (and thus reduces $\mathbb{C}_{Op}$) and can also increase $\mathbb{R}_{Op}$. Finally, with fullOrg, I identify that despite attempting to encode the same overarching organizational structure, the organizational influences in the components are not completely redundant, as it is largely possible to obtain the additive benefits found in each of the other organizations. The drop in $\mathbb{R}_{Op}$ of fullOrg as compared to transitionOrg is due to the reward influences urging agents to quickly go to their respective PARs (and stay there) rather than stop and fight fires along the way and discouraging them from moving into their respective SARs to fight fires. However, also note that $\mathbb{C}_{Op}$ is drastically reduced, indicating that the Pareto tradeoff would be beneficial unless $\mathbb{C}_{Op}$ is an insignificant contribution to $\mathbb{P}_{Op}$. (Optimal organizations for different points along the Pareto front could be determined from a figure analogous to Figure 3.3.)

Examining the results between the two different initial agent configurations highlights the significant impact that the problem domain can have on an organization's relative effectiveness as compared to the baseline MAS. When the agents initially begin spread throughout the grid, the local baseline performs relatively well (since agents can generally just focus on fighting the fires nearest to their current location without coordinating those actions with other agents), and there are fewer, limited opportunities for an organization to improve the agents' coordination. However, when the agents begin clustered in the center of the grid, it is important for an organization to differentiate the agents' policies so that they will efficiently disperse throughout the

	$\mathbb{R}_{Op}$	$\mathbb{C}_{Op}$	Replans
Baseline	-107.40	1646	7.89
RewardOrg	-91.45	1817	7.49
TransitionOrg	-85.14	14606	0.86
ActionOrg	-94.14	551	7.70
StateOrg	-94.14	1237	2.60
FullOrg	-87.51	5476	0.88

Table 3.2: Results for experiments in Section 3.5.3 when agents initially began spread throughout the grid for $\mathbb{R}_{Op}$, $\mathbb{C}_{Op}$, and number of times the replanning mechanism was invoked per agent per episode.

	$\mathbb{R}_{Op}$	$\mathbb{C}_{Op}$	Replans
Baseline	-436.7	10912	0.00
RewardOrg	-242.0	11051	9.38
TransitionOrg	-222.5	10859	0.55
ActionOrg	-264.5	621	8.56
StateOrg	-254.4	1588	1.50
FullOrg	-250.4	2652	1.02

Table 3.3: Results for experiments in Section 3.5.3 when agents initially began clustered in the center of the grid for $\mathbb{R}_{Op}$, $\mathbb{C}_{Op}$, and number of times the replanning mechanism was invoked per agent per episode.

grid (to avoid redundantly pursuing the same fire), and consequently the organizations all perform substantially better than the baseline.

The reader may have noted that these experiments did not evaluate the impact of influencing the initial state distribution or the time horizon. One could envision organizations that modify T_i to give agents specific roles for planning horizons, where some agents focus on the near-term and others on the long-term, but this kind of organizational decomposition is obviously unsuited for the firefighting domain since the focus is on (relatively) short time horizons. If α_i summarizes the exogenously-determined initial state, then an organization can only map this into the agent's modified state space, as was implicitly done for the organizational variations above. However, as seen in the relative performance between the spread and clustered domains, if the organization can impose initial states on agents (e.g., spreading them out in anticipation of expected fire configurations), then this provides an additional lever for improving collective performance.

3.6 Generality of Approach

In order to provide more precise discussions throughout this chapter, I have made various restrictions on the space of problems that I considered. In this section, I briefly describe the generality of my approach beyond the specific problems that I previously described. Note that these discussions are by no means comprehensive, but rather are intended to shed insight on my approach's generality.

3.6.1 Generality to Other Problem Domains

Throughout this chapter, and indeed throughout this dissertation, my empirical evaluations are limited to the firefighting domain, which could be viewed as a weakness of my evaluations. It is important to recognize, however, that the problem domain specifics are only significant insofar as they present interesting challenges for organizational design. In this regard, the firefighting domain is a nuanced testbed for evaluating my organizational design techniques. By altering the domain parameters (as I will do throughout this dissertation), the firefighting domain can be controlled to pose progressively difficult organizational design challenges. Moreover, by consistently using different parameterizations of the same problem domain, I can systematically investigate the isolated effects of various organizational techniques, and incrementally provide the reader with an intuitive understanding of how my techniques are affecting the MAS.

Of course, generality to other problem domains is nevertheless important. Theoretically speaking, my specific techniques rely only on the existence of a well-formed Dec-MDP model, and nowhere did my representations rely on specific domain information. That said, one possible challenge to generalizing my discussions in this chapter would be in hand identifying appropriate organizational influences for other domains. In the firefighting domain, I relied heavily on my intuitions and recognition of spatial patterns to hand design organizations, which could be difficult in other domains. A possible solution to this concern would be to use automated organizational design techniques like the ones I develop in Chapters 4 and 5.

3.6.2 Generality to Other Reasoning Frameworks

To delimit the scope of my discussions in this dissertation, I have elected to adopt a Dec-MDP framework to model the agents' reasoning processes, which subsequently provides the theoretical foundations for the specific organizational specification language I developed (and for the ODP techniques I create in Chapters 4 and 5.). Fundamentally, however, the specific representations and algorithms I develop for this Dec-MDP framework are an illustration of my broader agent-driven approach, which could be applied to other agent reasoning frameworks. That is, an agent-driven organizational specification language can hypothetically be created for any well-defined reasoning framework by enumerating the framework's dimensions. To give a better sense of this generality, in the remainder of this section, I briefly describe what the agent-driven specification language would consist of for two other popular agent reasoning frameworks. More rigorous examination (e.g., formal analysis like in Section 3.2, and/or empirical demonstrations like in Section 3.5) for these other reasoning frameworks is beyond the scope of this dissertation.

Factored Dec-POMDP. Closely related to the Dec-MDP model I use throughout this dissertation, the Dec-POMDP model additionally models partial-observability (whereas the Dec-MDP model I adopted is locally-fully observable). Recalling Definition 2.4, the Dec-POMDP model is expressed as states, an initial state distribution, actions, a transition function, a reward function, and a finite horizon just like the Dec-MDP model I utilized, but additionally includes an observation set and an observation function. Intuitively, then, the agent-driven organizational specification language would include influences that can modify any of those dimensions. An organization might modify an agent's observation set to reflect that certain observations are no longer possible due to the non-local effects of the other agents' actions. Similarly, an organization might modify factors of an agent's observation function to reflect the non-local effects of the other agents' actions.

Belief-Desire-Intention (BDI). BDI frameworks for agent decision making have been widely studied by the research community (Rao & Georgeff, 1991; Wooldridge, 2000; Bordini et al., 2007; Dastani, 2008) but, unfortunately, there is no consensus, formal definition of the components in a BDI-agent's reasoning framework. Broadly though, BDI frameworks consist of belief, desire, and intention dimensions, each of which an organization could modify via organizational influences.

- Beliefs represent the information that the agent believes to be true, and are analogous to (belief) states in (PO)MDP models. An organization could modify an agent's belief space to, for example, inform the agent that given its part in the organization, certain aspects of the environment are unimportant for it to model, much like influences to state factors in (PO)MDP based frameworks.

- Desires represent the agent's motivations, often in terms of goals for the agent to achieve, and are roughly analogous to rewards in (PO)MDP models. An organization could modify an agent's desires to, for example, influence the goals that the agent tries to achieve to be better aligned with the MAS's collective needs, much like influences to reward factors in (PO)MDP based frameworks.

- Intentions represent what an agent has chosen to do, often in terms of (hierarchical) plans that the agent can execute, and are roughly analogous to a mix of actions and transitions in (PO)MDP models. An organization could modify an agent's intentions to, for example, influence the agent into specific plans (e.g., that contend for fewer global resources), much like influences to actions in (PO)MDP based models.

3.6.3 Generality to Hierarchical Organizational Structures

A common attribute of previously developed organizational modeling languages (e.g., see Section 2.2.2) is the ability to express a hierarchical organizational structure (Horling & Lesser, 2004). Hierarchical organizational structures can also be expressed in my agent-driven approach, albeit they do not need to be explicitly represented like in prior approaches. Whereas prior approaches are intended for open systems and thus must explicitly represent the organizational structure to allow agents to be recruited to enact the roles entailed in the structure, in my agent-driven perspective, an organization is designed for a particular MAS. Consequently, an agent only needs to be informed of how its local reasoning and behaviors fit into the organizationally-influenced MAS, and an explicit representation of the structural hierarchy is unnecessary.

Implicitly representing a *de facto* hierarchical structure in my agent-driven approach instead relies on the agents in the MAS possessing the requisite individual capabilities (e.g., to fuse information from subordinates, accept task-delegations from superiors, etc.). For example, an agent could be organizationally influenced into acting as a manager in a hierarchical organization if its local model included managerial capabilities. In the firefighting domain, managerial actions might include actions to obtain expected response times for subordinate agents to fight a fire at a target location, and actions to designate which fires its subordinate agents should pursue next.

Noteworthy, however, is that in my agent-driven approach, such managerial capabilities must be included in the organizational designer's model of the environment.

That is, some prior research (Horling & Lesser, 2008; Sims et al., 2008) studies how to optimize the organizational structure under the expectation that agents can be recruited to enact the full range of possible structures. My agent-driven approach, on the other hand, instead optimizes over the organizational structures possible within a given MAS, and will not attempt to consider any organizations that would require changing the population of agents in that MAS, such as injecting a new agent with managerial capabilities into the MAS. Of course, one could circumvent this limitation by encapsulating my agent-driven techniques with an outer loop to search through alternative, hypothetical MASs, albeit at the expense of greatly increased computational costs. Keeping the number of hypothetical MASs tractable could require good heuristics and/or local search techniques, such as greedily adding managers into the MAS until performance doesn't improve. However, further exploration of such ideas are beyond the scope of this dissertation

3.7 Conclusion

There are two primary contributions in this chapter. Firstly, I contribute an agent-driven approach to defining the organizational design space. The fundamental advantage of this approach is that the design space inherits the mathematical formalism of the agents' underlying reasoning framework. In Section 3.1, I defined the concept of an *organizational influence* as a modification to an agent's local decision problem, and formulated the definition of an organization in terms of such influences. After committing to a specific Dec-MDP based agent reasoning framework, I illustrated my agent-driven approach by enumerating the agents' modeling constructs to create a principled organizational specification language for this Dec-MDP framework. Specifically, for the Dec-MDP framework I adopted, the specification language consists of modifications to the agents' state factors, initial state distribution factors, action spaces, transition function factors, reward function factors, and finite time horizon. Then, in Section 3.2, I demonstrated an advantage of my agent-driven approach by leveraging the mathematical foundation of the agents' reasoning framework to formally analyze the theoretical properties of my organizational specification language.

As is, my specification language contributes to Dec-MDP based reasoning techniques since it provides a mathematically-sound approach for incorporating organizational reasoning techniques (e.g., like those I develop in later chapters) into the agents' operational reasoning processes. More broadly, my agent-driven definition of the organizational design space contributes to the body of organizational reasoning techniques, and provides a well-defined, systematic method for understanding how an organization can relate to a MAS given the agents' reasoning framework, whereas prior research (Section 2.2) only informally understood this relationship. Moreover, my agent-driven definition of the organizational design space establishes a mathematical foundation for the study and development of computational representations and algorithms for organizational design, in contrast to prior research that relies on implicit organizational representations (e.g., Section 2.2.3) or human expertise to identify an organization (e.g., Section 2.2.2).

The second primary contribution of this chapter is the development of a well-defined, principled organizational design problem (Section 3.3), which provides a mathematical basis for discussing the effectiveness of an organization and choosing from among alternative organizational designs. By examining the formal, agent-driven characterization of the organizational design space, in Section 3.4 I identified that all MASs have an inherent local organization which can play a significant role in determining the effectiveness of additional organizational design. I then defined a systematic methodology for constructing a baseline MAS that is faithfully aligned with the environment model, but does not differentiate and/or coordinate the agents' policies. Finally, in Section 3.5, I illustrated the practical applicability of my agent-driven approach by empirically demonstrating how hand-encoded organizations can improve the expected performance of the MAS relative to the local baseline.

My agent-driven definition of the organizational design problem contributes to the body of organizational reasoning techniques, and provides a formal, mathematical foundation for selecting between alternative organizations. Moreover, my problem formulation establishes the basis for computational ODP techniques that create an organization from first-principles (e.g., like I develop in later chapters), in contrast to prior research that relies on an expert human to identify appropriate organizational constructs (e.g., Section 2.2.2) that may subsequently be configured/adapted (e.g., Section 2.2.4). Finally, my methodology for constructing a baseline organization contributes a performance benchmark to the organizational research community that can be used to evaluate the effectiveness of an ODP technique to identify beneficial organizational influences.

CHAPTER 4

Selecting Organizational Influences

In Section 3.2 (and Appendix A), I formally proved that my organizational specification language is a general purpose programming language for the Dec-MDP framework I have adopted (i.e., since it is a necessary and complete language for modifying agents' local models), and that the size of the influence space is intractably large. Moreover, in Section 3.5 I illustrated that the choice of organizational influences and encoding mechanism can have significant impact on the performance of a MAS following an organization, even between seemingly similar organizations. For example, the organizations in Section 3.5.3 are all intended to embody the same overarching organizational patterns, yet have substantially different $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ performance characteristics when evaluated in the MAS.

Combining these observations, it is evident that selecting appropriate organizational influences to specify to a MAS can be challenging. Not only is the space of possible organizational specifications large and complex, but naïvely innocent choices (e.g., encoding an influence as a modification to a reward function factor versus to an action space) can have significant impact on the MAS's subsequent $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ performance when following the organization. This argues for techniques to focus the organizational reasoning of an ODP (either human or computational) on the most pertinent aspects of organizing the MAS as well as to aid an ODP in selecting and specifying appropriate organizational influences.

In this chapter, I leverage my agent-driven approach to identify the mathematical foundations of selecting appropriate organizational influences for a MAS. I begin in Section 4.1 by describing a heuristic approach for restricting the space of influences that an ODP should consider. Specifically, an ODP should focus its design efforts on influencing the agents' interdependencies rather than on how the agents complete their individual aspects of a joint interaction. Empirically evaluating this heuristic shows that it results in organizations that are robust to imperfections in the ODP's information (e.g., when agent's have local expertise), but still increase MAS performance relative to the local baseline.

Even with this design principle, however, designing an organization can be complex, time-consuming, and error-prone if done by hand, as teasing out the pertinent interdependencies can be difficult, which argues for automated ODP techniques. In Section 4.2, I describe a general-purpose, agent-driven strategy for automated organizational design, and formulate concrete representations and algorithms to implement this automated ODP approach. Briefly, the underlying insight behind my automated ODP is to apply my agent-driven approach to model and reason about the agents' joint policy abstractly to predict desirable patterns of joint action, and then influencing the agents into these coordination patterns. My empirical evaluations demonstrate that my ODP implementation is able to identify organizations that are both intuitively sensible and also find and exploit domain structure that my hand-generated organizations overlook.

Finally, my experiments illustrate that a single organizational design typically cannot dominate the entire $\mathbb{P}_{Op} = f(\mathbb{R}_{Op}, \mathbb{C}_{Op})$ Pareto front, which argues that an ODP should intentionally design an organization to optimize for the anticipated Pareto parameterization. In Section 4.3, I extend my ODP techniques to explicitly reason about the metareasoning regime (i.e., balance between $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$) that the designed organization will impart on the MAS. My evaluation of these extensions shows that my ODP implementation can efficiently identify an organization with (approximately) optimal $\mathbb{P}_{Op}$ given a provided Pareto parameterization.

4.1 Influence Selection Heuristic

In Chapter 3, one may have noticed that a syntactically correct Θ could completely overwrite an agent i's entire local model, $\mathcal{M}_i$, and if desired could completely micromanage the agent by dictating exactly which action should be executed in each state. While such micromanagement could be desirable if the ODP has perfect information about the agents' capabilities and the problem episodes the MAS will encounter, it is more common to assume that the agents possess a non-negligible amount of expertise that is not precisely known by the ODP (Property 1.3). Otherwise, the agents are really just effectors for a centralized (ODP) reasoner. If the agents have local expertise, then influences that completely micromanage an agent would override that expertise, and likely lead to an organization with diminished effectiveness. Natural questions, therefore, are whether there is a mathematically-grounded explanation for these organizational intuitions, and how these intuitions can be formally understood and codified for aiding an ODP. In this section, I answer these questions by proposing and testing one such principle to guide decisions about which portions of the agents' models an organizational design should influence.

As just mentioned, organizations should not dictate or micromanage because individual agents might (and generally do) possess their own expertise, and leaving them room to exercise their local expertise benefits the collective organizational objectives. Assuming that agents are locally skilled, then, what an ODP possesses that individual agents lack is global awareness of how the agents' local actions assemble into coordinated, collective interaction patterns, where if individual agents had such awareness they could make more informed decisions. Hence, an ODP should use its more global perspective to influence agents into acting like they would if they were more globally aware themselves, and otherwise should allow agents to exercise their local capabilities.

I codify this observation in the following organizational design principle:

Principle 4.1. *A well-designed organizational specification should influence only the factors of agents' models that are associated with agent interactions.*

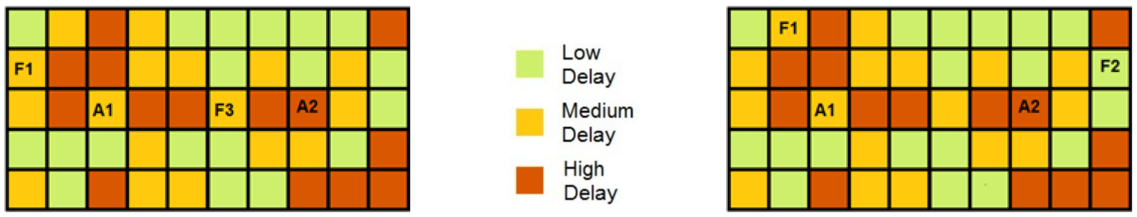
This principle is surprisingly applicable for creating organizational designs for Dec-MDP based agents, where factors associated with interactions are directly captured in joint reward/transition functions, and indeed where the specification of agents' decision models sometimes explicitly separates out the dependent factors (e.g., coordination locales (Varakantham et al., 2009; Velagapudi et al., 2011)) from the independent ones. This is neatly illustrated in my specification of the firefighting domain, where agent movements are independent (agent i's movement actions do not directly affect the states/rewards of another agent), but firefighting actions are not (agent i's states/rewards are affected by another agent fighting a fire).

To test Principle 4.1, I enumerated a space of factored organizations, each of which draws on the same, fully-specified organization (specifically the smallOverlapOrg from Section 3.5.1), but adopts a different subset of factors taken from that full specification. In particular, in its factored form, the transition component for the firefighting domain has one factor for the effects of agent movement actions, and another for firefighting actions. (There are also transition factors for the system time and cell delays, but these factors are unaffectable by the agents and thus omitted from my discussion here.) I draw on my organizational design principle to combine only the factors that correspond to ***reward/transition-dependencies (R/T-Ds)***. Revisiting the smallOverlapOrg's components, R/T-D factors include: the entire state and action components (since these reflect the organization's global awareness that agents can depend on each other to fight fires in their respective PARs); and the transition factor for I_cs (summarizing expectations of when fires will be extinguished by other agents in the overlapping PARs). On the other hand, the transition factor for agents' movement actions is a non-R/T-D factor, because agent movements are independent.

Principle 4.1 thus leads to the hypothesis that a specification including only these R/T-D factors will capitalize on the global perspective of the ODP without overstepping into micromanagement: it will perform as well or better than other combinations of factors, especially when the agents possess significant local expertise. To test this hypothesis, I constructed this **R/T-DOrg** organizational specification, as well as an organization that includes only non-R/T-D factors, the **non-R/T-DOrg**. For completeness, I also considered the full set of factors yielding the **unfactoredOrg** (identical to the unfactored smallOverlapOrg), and the empty set of factors corresponding to the local baseline (where agents are uninfluenced by any explicit organization).

My experiments in this section use the two agent version of the firefighting domain previously described in Section 3.5. However, to stress that agents can have local expertise that is imprecisely known to an ODP, the cell delays, δ_c, are allowed to be non-zero, which introduces stochastic movement actions to the problem domain. In these experiments, cells delays are i.i.d. uniformly sampled from $\delta_c \in \{0.0, 0.5, 0.8\}$ for each cell c, but do not change from their initial values within a single episode. That is, at the beginning of each episode, the cell delays are randomly selected (independently of each other as well as any delays from past episodes), but then they are static for the episode. Figure 4.1 shows example initial states for this domain.

I artificially controlled the relative amount of the agents' local expertise by degrading the ODP's view of the cell delays via applying a smoothing filter over the true environment's ($\mathcal{M}^*$) delays (Figure 4.2). As the ODP's view of the cell delays is increasingly blurred by the smoothing filter, the agents possess comparatively more local expertise (e.g., about the best paths to take through the grid to most efficiently reach the fires to fight). By controlling the ODP's information in this way, the ODP's model remains accurate in terms of cells' mean delays but loses precision (about the physical distribution of delays) as additional smoothing iterations are performed. For example, the 100-smoothed setting had a smoothing filter iteratively applied to the grid 100 times, leaving the ODP with what amounts to the average cell delay in each



(a) "Hard" firefighting episode.

(b) "Easy" firefighting episode.

Figure 4.1: Illustration of "easy" and "hard" firefighting episodes. An episode is "hard" if the local baseline MAS miscoordinates.

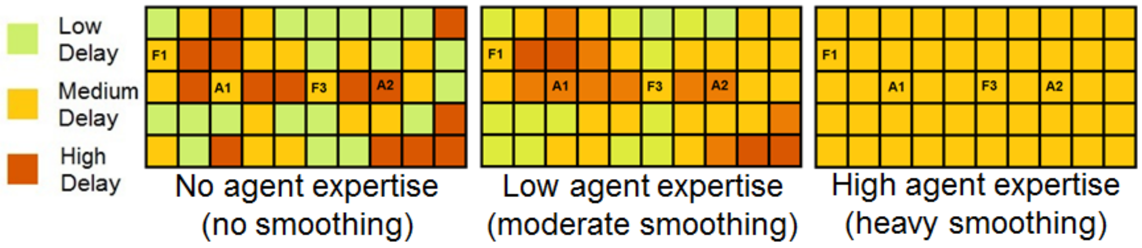

Low Delay
Medium Delay
High Delay
F1
A1
F3
A2
No agent expertise (no smoothing)
Low agent expertise (moderate smoothing)
High agent expertise (heavy smoothing)

Figure 4.2: Illustration of how I controlled agent expertise.

cell. Comparatively, in the 0-smoothed setting the ODP's view precisely matches the real delays in $\mathcal{M}^*$, etc. An agent's knowledge of the cell delays (contained in $\mathcal{M}_i$) is perfect in these experiments, and thus exactly equals $\mathcal{M}^*$ in this regard.

I generated 3000 episodes for the two agent firefighting domain described above, and simulated each episode 5000 times to smooth out the effects of stochastic movement actions. The procedure described in Section 3.5.1 was used to incorporate organizational specifications into the agents' local models and to compute their local policies with respect to their organization. To evaluate the hypothesis that agents in an organization should be allowed to retain and contribute their local expertise, I evaluated each of the four organizations across a spectrum of settings where I varied the relative quality of agent expertise compared to the ODP's. Unfortunately, I was unable to fully contrast the organizations against computing the optimal joint policy, π^*, as I was computationally unable to calculate π^* for all 3000 episodes. However, in a small subsample of the episodes that I could complete ($\sim$100), I observed that computing π^* costs approximately two orders of magnitude more $\mathbb{C}_{Op}$ and achieves approximately three percent more $\mathbb{R}_{Op}$ as compared to the local baseline.

Tables 4.1, 4.2, and 4.3 present results from these experiments and highlight several important points. Firstly, observe in Table 4.1 that, in expectation over all problem episodes, my organizations only increase $\mathbb{R}_{Op}$ by 1.45% in this domain. While this initially seems disappointing, on reflection it is unsurprising: for most episodes, an

	$\mathbb{R}_{Op}$ **Improvement (%)**			$\mathbb{C}_{Op}$ **Reduction (%)**		
# Smoothing Iterations	**0**	**10**	**100**	**0**	**10**	**100**
R/T-DOrg	1.45	1.40	1.40	34.08	33.02	33.12
unfactoredOrg	1.45	-6.24	-7.30	34.08	33.00	33.13
non-R/T-DOrg	0.01	-7.41	-8.29	0.00	0.00	0.00

Table 4.1: Expected percent $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ improvement of organizations compared to the local baseline for all 3000 problem episodes.

	$\mathbb{R}_{Op}$ **Improvement (%)**			$\mathbb{C}_{Op}$ **Reduction (%)**		
# Smoothing Iterations	**0**	**10**	**100**	**0**	**10**	**100**
R/T-DOrg	6.05	5.92	5.98	34.81	34.04	33.77
unfactoredOrg	6.10	-9.61	-10.70	34.84	30.83	30.90
non-R/T-DOrg	0.00	-12.26	-13.02	0.00	0.00	0.00

Table 4.2: Expected percent $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ improvement of organizations compared to the local baseline for the 750 problem episodes with largest expected impact (top 25%) to $\mathbb{R}_{Op}$. The same subset of episodes is used to calculate the $\mathbb{C}_{Op}$ reduction. The subset of episodes used varies by organization as well as by number of smoothing iterations.

	$\mathbb{R}_{Op}$ **Improvement (%)**			$\mathbb{C}_{Op}$ **Reduction (%)**		
# Smoothing Iterations	**0**	**10**	**100**	**0**	**10**	**100**
R/T-DOrg	23.27	23.55	23.44	27.88	27.79	27.76
unfactoredOrg	23.21	-19.48	-19.76	28.03	28.63	29.36
non-R/T-DOrg	0.00	-24.88	-24.47	0.00	0.00	0.00

Table 4.3: Expected percent $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ improvement of organizations compared to the local baseline for the 150 problem episodes with largest expected impact (top 5%) to $\mathbb{R}_{Op}$. The same subset of episodes is used to calculate the $\mathbb{C}_{Op}$ reduction. The subset of episodes used varies by organization as well as by number of smoothing iterations.

agent's lack of global awareness is inconsequential, because the initial placement of agents and fires is such that, for most cases, agents' local decisions lead them into complementary actions (e.g., Figure 4.1b). However, in episodes where a high-intensity fire is located between the agents' initial positions (e.g., Figure 4.1a), agents acting locally often miscoordinate, providing opportunities for an organization to improve $\mathbb{R}_{Op}$. This observation is a manifestation of how the domain inherently impacts the potential effectiveness of an organizational design, as also observed by Corkill et al. (2015)

Moreover, this observation suggests that the average performance hides a heavy-tailed distribution. If we sort the episode results by the magnitude of the percent $\mathbb{R}_{Op}$ difference versus the local baseline, $\frac{|baseline-org|}{baseline}$, and filter the sorted results to only include the episodes where an organization has the largest impact to $\mathbb{R}_{Op}$ (either positively or negatively), then we observe in Tables 4.2 and 4.3 that when an organization does impact performance, it has a noticeable impact. For example, R/T-DOrg has a 23.27% expected improvement to $\mathbb{R}_{Op}$ in 5% of the episodes (Table 4.3). Note that the decrease in the $\mathbb{C}_{Op}$ improvement of the unfactoredOrg and R/T-DOrg as episodes are filtered (i.e., moving from Table 4.1 to Table 4.2 to Table 4.3) is also caused by this domain property. These R/T-D inclusive organizations (i.e., unfactoredOrg and R/T-DOrg) coordinate correctly when a high-intensity fire is located between the agents due to I_c transition shaping (thus increasing $\mathbb{R}_{Op}$ relative to the baseline); however, transition shaping also increases the size of the agents' state spaces (thus decreasing their $\mathbb{C}_{Op}$ advantages).

Secondly, as shown in all three tables, the organizations that influence R/T-D factors outperform those without R/T-D based influences, both in terms of $\mathbb{R}_{Op}$ as well as reducing $\mathbb{C}_{Op}$. Moreover, the non-R/T-D based influences (i.e., in unfactoredOrg and non-R-/T-DOrg) can severely degrade MAS performance as agents possess relatively more local expertise, as demonstrated by the distinct $\mathbb{R}_{Op}$ drops for these organizations as the ODP's information becomes increasingly imprecise via smoothing its view of cell delays. This illustrates the costs of heavy-handed micromanagement that undervalues agents' expertise, and supports my claim that organizations that omit non-R/T-D based influences allow agents to exercise their expertise to avoid these performance deficiencies.

While this concludes my explicit evaluation of Principle 4.1, as we will see in the remainder of this chapter, Principle 4.1 serves as an underlying premise of my computational ODP techniques. As such, the subsequent experiments I perform to evaluate my ODP techniques can also be viewed as further evidence in support of

Principle 4.1 This complete body of experiments, along with the theoretical derivatioPrinciple 4.1 earlier in this section, suggest that Principle 4.1's effectiveness is not limited to the specific experiments I performed in this section, but rather it is a more robust, generally-validated heuristic for selecting effective organizational influences.

4.2 Automated Organizational Design

Thus far, the organizations I have presented in this dissertation have been hand created and tuned for use in the firefighting domain. Generally speaking, designing an organization for a MAS by hand can be challenging. For example, the organizations I hand-created in the previous sections required intricate knowledge of how the agents should best interact as well as several iterations of careful tuning to get the organizational influences just right. Automated techniques for organizational design can address these challenges in two primary ways:

1. A computational ODP could serve as an alternative to hand-designing an organization. Rather than relying on a human ODP to comprehend the MAS's nuanced interaction patterns and my specification language's influence encoding strategies, an automated ODP can determine interaction patterns itself and search through the space of influence encodings to find the best one.

2. Computational techniques for organizational design necessitate a more formal, explicit understanding of how to create an organization for a MAS. As such, developing an automated ODP provides mathematically-grounded insights that could guide a human ODP in creating organizations (e.g., as part of a problem-driven approach).

In this section, I describe the core representations and algorithms that my automated ODP techniques utilize to create an organization for a MAS.

4.2.1 ODP Overview

As described in Section 4.1, an ODP should use its global perspective to identify patterns of interactions that would arise when agents cooperate effectively, and then codify these patterns into influences that agents internalize. For example, the R/T-DOrg stops agents from even thinking about fighting fires that another agent is clearly better positioned to fight, and focuses them on fighting nearby fires. Stepping back, the organizational design principle I presented in Section 4.1 suggests the foundations

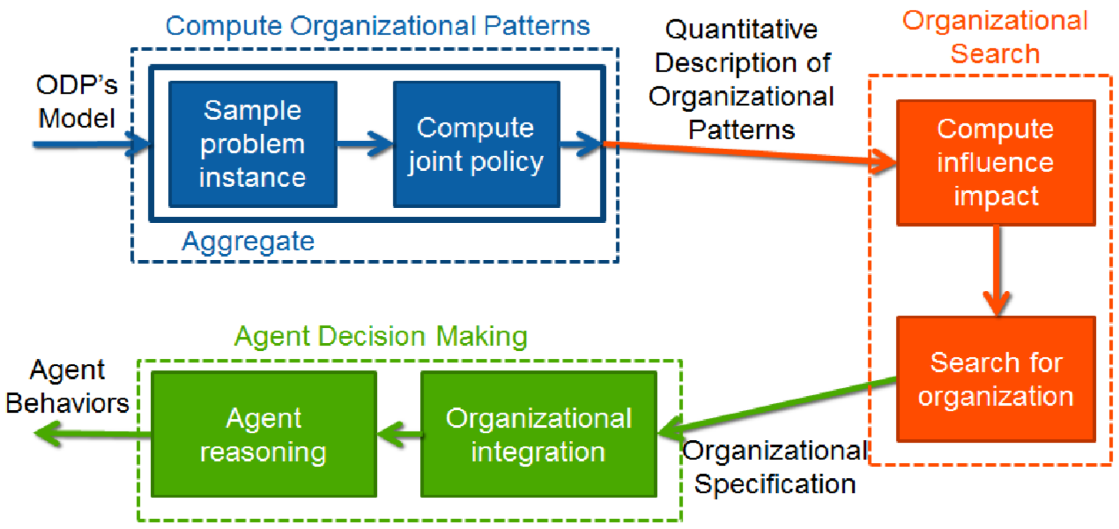


Figure 4.3: Overview of how my automated ODP and how it relates to the MAS.

of a process for automating organizational design: use the ODP's global perspective to identify the R/T-Ds; deduce from these a space of joint actions to seek/avoid; and then use these patterns to select influences to agents' local models that steer agents to/from local decisions that lead to the good/bad interactions.

Unfortunately, identifying R/T-Ds can sometimes be difficult in models where joint interactions are not explicitly provided as part of the model specification. For this reason, the automated ODP I have devised exploits domain knowledge if it is provided, but still functions (although with increased computational costs) without explicit R/T-D specifications. Figure 4.3 illustrates a high-level overview of my automated ODP and how it relates to the MAS. The ODP begins with a Dec-MDP model of the global domain, and I assume that this model is accurate with respect to the global interactions, but may be imprecise with respect to the details of how agents will complete their respective components of a joint interaction (i.e., Property 1.3). Then, as shown in Figure 4.3, there are three main stages to my ODP:

1. Compute organizational patterns. The ODP uses its global perspective of the domain to compute a quantitative description of organizational patterns. This stage of the ODP will be described in detail in Section 4.2.2.

2. Organizational search. Using the quantitative description of organizational patterns from the previous stage, the ODP searches through the space of organizational influences to identify the subset that comprise an organization to specify to the MAS. This stage of the ODP will be described in detail in Section 4.2.3.

3. Agent decision making. The agents then incorporate the organization into their local reasoning processes and solve their local decision problems as previously described in Section 3.1.2.

It is important to note that my ODP is intentionally decomposable into these distinct stages. Not only does the stage decomposition help to conceptualize my representations and algorithms, but moreover, it allows a user to employ a subset of the stages. For example, suppose a system administrator has expert knowledge consisting of a quantitative description of organizational patterns. This administrator could use the organizational-search stage to compute an organization from those statistics, bypassing the unnecessary compute-organizational-patterns stage.

4.2.2 Compute Organizational Patterns

The objective of this stage in my ODP approach is to compute a quantitative description of organizational patterns. At a high level, a quantitative description of organizational patterns corresponds to statistical information about the expected performance of the MAS's possible interaction patterns. A quantitative description of organizational patterns could be represented in a variety of forms; however, in terms of the organizational design problem I previously defined (Definition 3.6) it corresponds to the expected operational performance (i.e., expected reward and reasoning costs) associated with each of the agents' interactions. For example, an action a_i is expected to increase agent i's computational costs by some amount, and also contribute and/or enable some amount of expected joint reward to the MAS. (Note statistics about the expected joint reward correspond to Q-values and thus encapsulate both immediate and future rewards.)

At a high-level, my ODP computes a quantitative description of organizational patterns via a Monte Carlo estimation since enumerating the agents' optimal actions for all possible problem episodes the MAS could encounter is assumed to be computationally intractable. The ODP begins by randomly sampling a problem episode from its global Dec-MDP perspective of the domain (e.g., using the initial state distribution α). Then, the ODP computes the optimal joint policy for the sampled episode using a centralized perspective. Finally, the ODP repeats this process for numerous samples, and aggregates the joint policies across the sampled episodes to form its quantitative description of organizational patterns. In the remainder of this section, I step through this process in full detail.

Also, note that while my discussions in this section may suggest a sequential

Algorithm 4.1 Compute Organizational Patterns

Input: ODP's domain model, $\mathcal{M}$, and (optionally) a set of subgoals for task decomposition, G.
Output: A quantitative description of organizational patterns, Φ

```
1: $\Phi \leftarrow \emptyset$
2: while $\exists sample =$ NextSample() do
3:     $\pi \leftarrow$ ComputeJointPolicy($sample, \mathcal{M}, G$)
4:     $\Phi \leftarrow$ AggregateInformation($\Phi, \pi$)
5: end while
6: return $\Phi$
```

algorithm with three stages (i.e., sample episodes, solve the episodes for joint policies, and aggregate across samples), it is straightforward to interleave these stages to create a more anytime-esque algorithm. Indeed, my ODP implementation (Algorithm 4.1) interleaves these stages.

Episode Sampling. Using its Dec-MDP model of the global domain, it is straightforward for the ODP to sample the initial state distribution for a problem episode. Note, however an important subtlety that must be considered in the ODP's sampling process: if the ODP does not sample a set of episodes that covers the space of important agent interactions patterns, then the ODP's quantitative description of organizational patterns will not be fully informative of the interaction patterns the ODP should influence the agents into. For example, in the firefighting domain, in order for the ODP to recognize the value of an agent i fighting a fire in a cell c near i's initial location, the ODP must sample a problem episode where there is a fire in cell c. Information inaccuracies caused by poor episode sampling could negatively impact to the ODP's organizational search algorithm (Section 4.2.3) that uses these statistical estimates.

Consequently, the ODP should not simply use i.i.d. sampling of the initial state distribution to generate problem episodes. Rather, the ODP should sample problem episodes to provide maximal coverage of the patterns of the agents' interactions weighted by the importance of the interaction (e.g., interactions with inconsequential impact on MAS performance are less important for the ODP to recognize than ones with critical impact). Unfortunately, this optimal sampling strategy is theoretically infeasible since the ODP cannot know how the agents should interact for an episode without actually solving for the agents' optimal joint policy in that episode (at which point that data may as well be included in the ODP's statistics). As a proxy for this optimal strategy, for the firefighting domain my ODP generates samples in batches, where the samples in a batch are biased such that the ODP is ensured to encounter a fire in each of the grid cells. Specifically, in my experiments, the batch size is equal to the number of grid cells, where the first fire for the kth sample in the batch is placed into the kth cell, and then subsequent fires are selected uniformly randomly (without replacement) from the other cells. More generally, Bayesian sampling of the initial state distribution could be used to generate samples, although doing so implies that different initial states lead to different agent interactions.

Computing a Joint Policy. Theoretically, the ODP could use any Dec-MDP solution technique to solve a problem episode for the optimal joint policy (e.g., from among the methods described in Section 2.1.4). Practically, however, the computational intractability of computing a joint policy limits an ODP's ability to naïvely solve the Dec-MDP model for an episode, especially for domains with tightly interconnected agents (Property 1.2).

As such, my ODP uses the options framework (Section 2.1.5) from the hierarchical learning community to abstract its primitive-action Dec-MDP model into a task-level Dec-MDP model that focuses on tasks to accomplish rather than actions to take. Reasoning with task-level options serves two primary purposes. Firstly, reasoning with options reduces the ODP's computational burden since a Dec-MDP solution algorithm can exploit structure in the action sequences an agent should typically perform (i.e., rather than considering primitive actions individually, sequences of primitive actions can be considered as a whole). Secondly, task-level reasoning naturally emphasizes the most significant interactions among agents while remaining largely agnostic about how the agents will translate their options into detailed actions, which embodies Principle 4.1.

As is customary, my ODP creates one option for each task in the problem episode, where a task corresponds to achieving a particular subgoal. For simplicity, in the experiments that follow, I informed the ODP that good subgoals are states where $I_c \to 0$; however, subgoal detection could be automated using techniques from the hierarchical learning community (see Section 2.1.5). Of course, the ODP still requires an estimate of each option's properties (i.e., its expected reward, $R_i(s_i^t, o_i, s_i^{t+1})$, and transitions, $P_i(s_i^{t+1}|s_i^t, o_i)$), which my ODP computes by standard hierarchical reasoning techniques (Section 2.1.5).

The ODP then solves the task-level Dec-MDP as a centralized process using standard algorithms (see Section 2.1.4), which results in occupancy measures, $x(s^t, o)$, and Q-values, $Q^*(s^t, o)$, for state-option pairs in the optimal task-level joint policy.

Finally, the ODP inverts its option abstraction using the respective policy of each option (i.e., $o.\mu$), which projects the task-level joint policy downward to estimate the primitive-action joint policy, $\hat{\pi}$. For example, the occupancy measures in the primitive-action joint policy are the product of the task-level occupancy measures multiplied by the occupancy measures of the options' policies, $x(s^t, a) = \sum_o (x(s^t, o) \cdot o.\mu(s^t, a))$. Note that while the resulting $\hat{\pi}$ is not exactly identical to if the ODP had directly solved the primitive-action Dec-MDP for π^*, it does have the same optimal policy trajectory. That is, for any states that are reachable when following π^*, both $\hat{\pi}$ and π^* are ensured to be identical. As I discuss below, my ODP only aggregates information from along the optimal policy trajectory, so this distinction is ultimately inconsequential.

The particular options I created for my ODP are fairly domain-dependent—although automated techniques exist for option creation (Iba, 1989; Stolle & Precup, 2002; McGovern, 2002). Moreover, it is important to note that the abstraction to the task-level representation has no effect on the resulting occupancy measures and Q-values in the detailed problem representation. Rather, the ODP would compute exactly the same detailed policy if it did not use the options abstraction, albeit at a significantly higher computational cost. The options abstraction serves only to expedite the policy solving process. The issue of identifying appropriate options is addressed further in Section 4.2.4.

Aggregating Information Across Samples. To aggregate information across the sampled problem episodes, my ODP computes a weighted expectation. Specifically, information (e.g., Q-values or probabilities of non-local effects) is weighted by the occupancy measures (recall that in a non-recurrent state space, an occupancy measure $x(s^t, a)$ is equivalent to the probability of reaching state s^t and then performing action a). For example, the probability of a non-local effect $P_i(f_k^{t+1}|s^t, a)$ exerted on agent i is weighted by the occupancy measure $x(s^t, a)$.

A caveat in this method is that the linear program used to calculate occupancy measures (Equation 2.4) only returns informed values for states along the optimal policy trajectory. That is, state-action pairs that are not part of the optimal policy trajectory have zero occupancy by definition (i.e., there is no probability of reaching these states); however, $x(s^t, a) = 0$ for these types of states is not informative of whether the MAS should perform a, only that s^t should not be reached. Consequently, including these zero-valued occupancies in the expectation calculation dilutes the ODP's statistics without providing any actual information. This effect is especially significant in domains where the optimal policy trajectory is a small subset of the state space (like in the firefighting domain), since the informative occupancy measures will be scarce relative to the complete set of occupancies (essentially making it incorrectly appear as though all agent actions have no value). As such, my ODP disregards any occupancy measures for states that are not part of the optimal policy trajectory, i.e., states where $\sum_a x(s^t, a) = 0$. In contrast, state-action pairs along the optimal policy trajectory could be zero or non-zero. Obviously a non-zero occupancy is informative of the MAS's actions in the ODP's statistical estimates; however, $x(s^t, a) = 0$ for these types of states informs that a is *not* the optimal action, and thus should be included in the ODP's statistical estimates as well.

Illustrative Example. Figure 4.4 illustrates an example of how my ODP computes its quantitative description of organizational patterns via Algorithm 4.1. On each iteration (i.e., row of Figure 4.4), the ODP first samples a problem episode, translates its representation of the episode into the options-level representation, and solves for the optimal options-level joint policy, which corresponds to the first column of Figure 4.4. Then, the ODP inverts its options to compute the occupancy measures and Q-values (note that Q-values are not shown in Figure 4.4) in the detailed problem representation, which corresponds to the second column of Figure 4.4. Finally, the ODP aggregates the information from the current sample into its cumulative statistics, which corresponds to the third column of Figure 4.4.

Stepping down the rows, one can see that, as the ODP samples problem episodes, its statistics give it an increasingly accurate quantitative description of organizational patterns. Looping this process until the cumulative statistics stabilize results in the information shown at the bottom of the third column in Figure 4.4.

4.2.3 Selecting Organizational Influences

As illustrated in Figure 4.3, the second stage of my ODP is to use the quantitative description of organizational patterns as a basis for selecting organizational influences. Perhaps unsurprisingly, this a complex, multifaceted subject (essentially tantamount to the organizational reasoning problem), and even limiting scope to within my agent-driven approach there are numerous methods and orthogonal aspects of the algorithm that could be investigated (e.g., metareasoning issues like I do in Section 4.3 or abstract organizational influences like I do in Chapter 5). In this section, I describe a relatively simple version of the influence selection stage in my ODP. Not only does this simple version provide a starting point for subsequent discussions in this dissertation (and

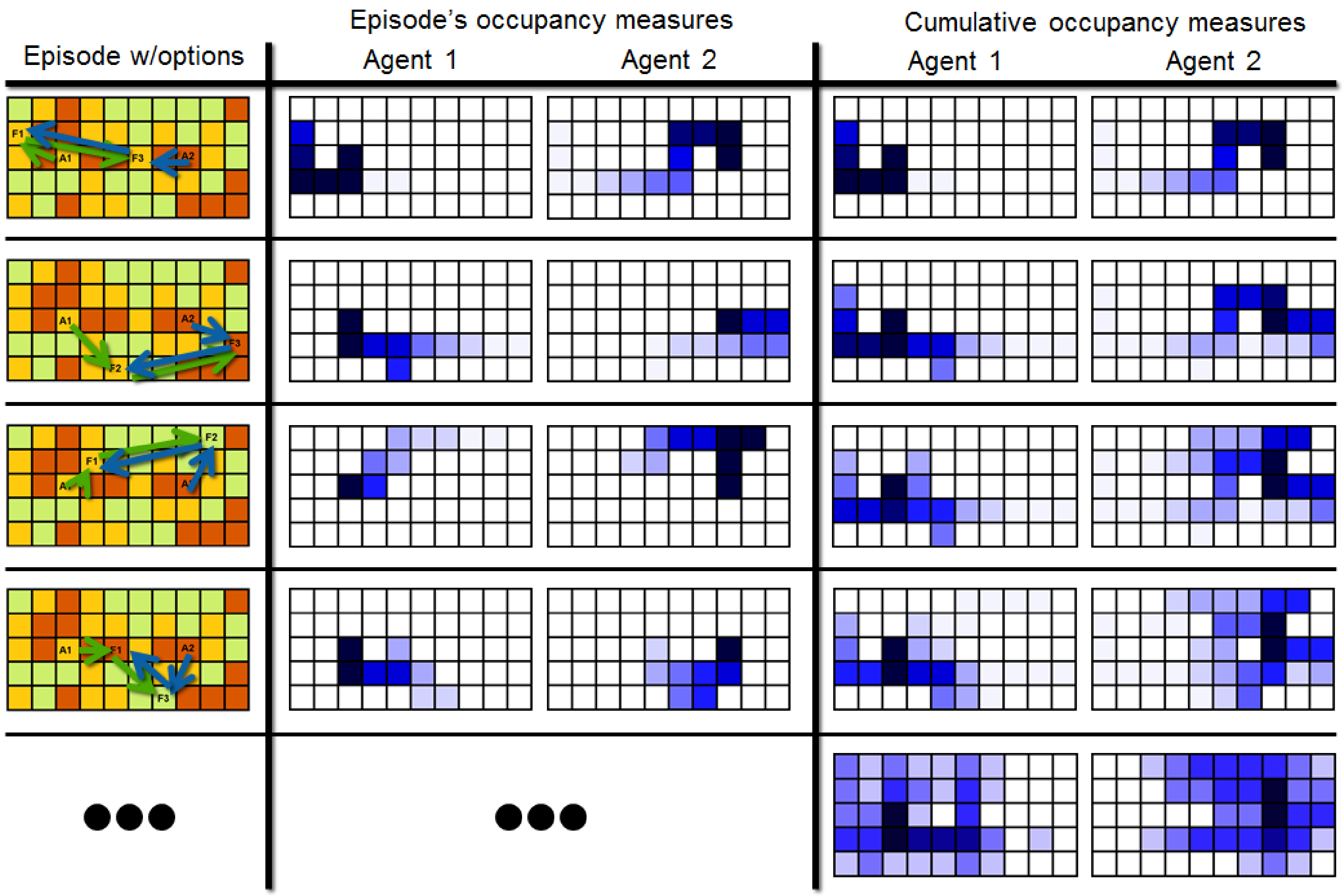


Figure 4.4: Walkthrough of my ODP's process for computing its quantitative description of organizational patterns. Darker shading in the occupancy measure columns indicates higher occupancy measure.

future research beyond the scope of my dissertation), but moreover, its deficiencies provide context for guiding my subsequent efforts. At a high level, my broader ODP approach selects organizational influences by first computing the expected impact that each candidate organizational influence will have on MAS performance, then searching through the influence space to identify an (approximately) optimal set of influences, and finally encoding these influences in my organizational specification language.

For now (this will be revisited in Section 4.3), my ODP selects influences as follows.

- **Actions:** For agent i, if the occupancy measure $x(s_i, a_i) = 0$ then remove a_i from the set of available actions in s_i for agent i. For example, in the firefighting domain, if the ODP's statistics from computing joint policies for sample episodes never have an agent move into certain cells (e.g., cells always serviced by closer agents), then actions that would move the agent into those cells are removed from consideration.

- **States:** If agent i's action choice under the joint policy is invariant with respect to state factor F_{i_k} given any values for the other state factors, then F_{i_k} can be removed from agent i's local state factors (since it contributes no information). For example, in the firefighting domain, the intensity of cells distant to an agent (always fought by someone else) do not impact the agent's action selection and thus are removed from the agent's state representation.

- **Transitions:** For an agent, modify the transition factors for each of its remaining state factors (after removing state factors as above) to include the probabilistic non-local effects of the other agents. For example, in the firefighting domain, an agent's transition factor for an overlapping cell's intensity would be altered to reflect the probability that some other agent executes the FF action in that cell at certain times.

The reader may note that influences to the agents' reward function factors are not included above (nor are initial state distribution and time horizon influences). I focus on action, state, and/or transition influences over reward influences since they provide hard constraints that can affect the agents' computational costs ($\mathbb{C}_{Op}$), e.g., as demonstrated in Section 3.5. However, reward influences could be useful as a fallback mechanism when hard constraints are not possible to formulate or otherwise undesirable. For example, if the ODP has an option-level model, but lacks knowledge of how options translate into local actions, states, and transitions, it could influence agents via local reward factors to induce coordinated policies. As another example, reward influences could be useful if the ODP has doubt in the agents reliably adopting organizational influences (e.g., in semi-cooperative settings beyond the scope of this dissertation), where an agent might be unwilling to constrain its action space, but could be influenced by providing it with additional incentives. Like in Section 3.5, influences to an agent's initial state distribution and time horizon are broadly inappropriate for the firefighting domain. I revisit the issue of other influence mechanisms again in Section 6.2.2.

4.2.4 ODP Limitations and Concerns

In this section, I describe some of the potential limitations and/or concerns of my ODP thus far, and discuss some preliminary thoughts of how they might be addressed (either later in this dissertation or in future work).

Heuristics for Identifying Hierarchical Options. The techniques I employ in my ODP leverage relatively conservative assumptions about the availability of external knowledge about the problem domain (specifically the identification of subgoals for the ODP to translate into options); however, there are ways of generalizing my ODP to incorporate additional knowledge, should it be available, or function without the option abstraction, should it be unavailable. For example, if an explicit specification of the R/T-Ds is provided, then the joint policy creation process could simply create a policy directly from the R/T-Ds. Even stronger, if (partial) information of good joint interactions is known (e.g., a partial-order plan), then the joint policy creation process could be sped up by constraining the Dec-MDP solution algorithm to reflect this knowledge. Alternatively, knowledge of the R/T-Ds could be reflected by directly providing the ODP with an options-level model (along with properties of the options to be used for inverting the abstraction) as opposed to the primitive-action Dec-MDP that the ODP translates into a task-level model.

In my experiments that follow, I informed the ODP that good subgoals are states where $I_c \rightarrow 0$, but did not systematically validate this abstraction choice (though intuitively, extinguishing a fire seems to be the natural task-decomposition for the firefighting domain). I did not systematically validate this abstraction because ultimately the options abstraction is only to expedite solving Dec-MDP problem episodes, and has no impact on the organization that the ODP creates. As such, optimal abstraction choice is not a core ODP problem, but rather a secondary topic for future research.

Centralized ODP Limitations. My ODP algorithms throughout this dissertation employ centralized techniques for computing a quantitative description of organizational patterns, which unsurprisingly is intractable as the problem domain scales (see Section 2.1.4). This computational intractability inevitably limits the scale of the problems that this aspect of my specific ODP implementation can solve (note, however, that the other aspects of my organizational representations and algorithms are not limited by this scaling problem). Intuitively speaking, there are several approaches one could use to mitigate these scaling concerns, including:

- Compute an approximately optimal joint policy, for example using techniques as described in Section 2.1.4.2.
- Use the options framework hierarchically, like I do in Section 5.7.
- Obtain a quantitative description of organizational patterns from a different source such as a system administrator or the agents themselves.

Approximate Dec-MDP Solution Techniques. If information about pertinent subgoals is unavailable for seeding an options abstraction (and optimally solving the primitive-action Dec-MDP is computationally infeasible), the ODP could instead rely on approximate solution techniques for computing an approximately optimal joint policy (e.g., see Section 2.1.4.2). Throughout this dissertation, however, I avoid using such approximation techniques to avoid confounding factors in my results. That is, solution techniques for solving Dec-MDPs are not the focus of this dissertation, but rather a means for my ODP to estimate a quantitative description of organizational patterns. If the joint policies found were only approximately optimal, then it would be difficult to discern whether an ineffective organization were the result of the approximate statistics, or a more fundamental deficiency of my organizational reasoning techniques. Of course, robustness to information inaccuracies is a desirable characteristic for an ODP, but I have elected to evaluate such robustness using more controlled methods (e.g., degrading the ODP's domain model like in Section 4.1) rather than the more unpredictable information inaccuracies possible with approximate joint policy solvers.

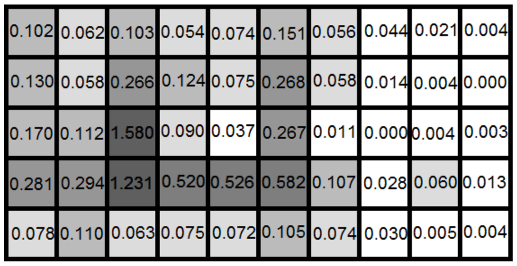
Rigidity of Influence Selection Criteria. It is important to recognize that the influence mechanisms in Section 4.2.3 use very strict criteria for when to remove a factor. Essentially, the ODP finds the maximal reduction to an agent's model that does not decrease the expected joint reward. In principle, however, further reductions could sacrifice some amount of reward in order to further influence the agents. For example in the firefighting domain, it could be the case that agent i rarely needs to know the intensity of a relatively distant cell, so the expected reward loss from not modeling that state factor is very small. Thus, removing that factor from consideration has negligible impact on the expected joint reward, but could yield significant computational savings during agent planning. Tradeoffs like these are the focus of metareasoning research, and I investigate this issue more deeply in Section 4.3.

4.2.5 Evaluation

To evaluate the initial ODP algorithm just described, I use the same parameterization of the firefighting domain as in Section 4.1. However, to evaluate this simple version of the ODP, I did not perform any smoothing of the ODP's model because I wanted to evaluate if the ODP could exploit structure that it identifies in the domain (this assumption will be relaxed in later experiments). In this domain, the space of possible initial global states is exceedingly large (∼22,000), and furthermore, the total reachable state space from the set of initial states contains millions of states. For these reasons, the ODP is computationally unable to exactly solve for the complete, optimal joint policy in every state, and instead relies on the sampling process described in Section 4.2.2. To test the impact the sample size has on the resulting organizations, I present results from two different parameter settings, 50 (0.23% of possible initial states) and 150 (0.68%). ***X*** **AutoOrg** refers to the organization designed using $X \in \{50, 150\}$ initial state samples. The effects of sampling size on the ODP will also be revisited in Chapter 5.

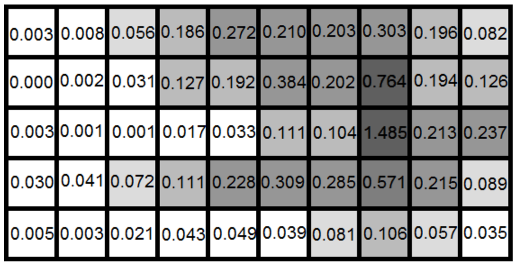
I begin my evaluation by confirming that the ODP's designs are intuitively sensible. Figures 4.5a and 4.5b show the cumulative occupancy measures by cell (shaded by magnitude), $\sum_{a_i} x_i(s_i, a_i)$, for each agent, created in response to the delays in Figure 4.5c, and represent a summary of the action shaping specified to each agent (i.e., cells with low cumulative occupancy measure typically have more tightly restricted actions). Darker shaded cells thus represent those that the ODP expects the agent will more likely visit. Observe in Figures 4.5a and 4.5b that the agents' influences are correlated, and each agent is more or less expected to be responsible for a particular region (with some overlap in between). Further, as seen by comparing Figure 4.5c to Figures 4.5a and 4.5b, the ODP recognized the domain structure, and tailored its influences to skew the agents' regions towards those cells they can efficiently reach.

I also tested the XAutoOrgs using the same empirical methodology and episodes as in Section 4.1 to ensure that they improve $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ in addition to being intuitively sensible. Table 4.4 presents the results of these experiments as well as repeats the



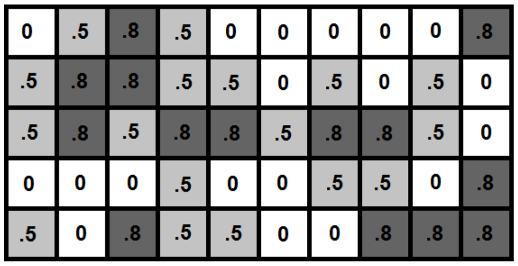
0.102	0.062	0.103	0.054	0.074	0.151	0.056	0.044	0.021	0.004
0.130	0.058	0.266	0.124	0.075	0.268	0.058	0.014	0.004	0.000
0.170	0.112	1.580	0.090	0.037	0.267	0.011	0.000	0.004	0.003
0.281	0.294	1.231	0.520	0.526	0.582	0.107	0.028	0.060	0.013
0.078	0.110	0.063	0.075	0.072	0.105	0.074	0.030	0.005	0.004

(a) Agent 1



0.003	0.008	0.056	0.186	0.272	0.210	0.203	0.303	0.196	0.082
0.000	0.002	0.031	0.127	0.192	0.384	0.202	0.764	0.194	0.126
0.003	0.001	0.001	0.017	0.033	0.111	0.104	1.485	0.213	0.237
0.030	0.041	0.072	0.111	0.228	0.309	0.285	0.571	0.215	0.089
0.005	0.003	0.021	0.043	0.049	0.039	0.081	0.106	0.057	0.035

(b) Agent 2



0	.5	.8	.5	0	0	0	0	0	.8
.5	.8	.8	.5	.5	0	.5	0	.5	0
.5	.8	.5	.8	.8	.5	.8	.8	.5	0
0	0	0	.5	0	0	.5	.5	0	.8
.5	0	.8	.5	.5	0	0	.8	.8	.8

(c) Cell Delays δ_c

Figure 4.5: Cumulative cell occupancy measures, $\sum_{a_i} x_i(s_i, a_i)$, for each agent that the ODP calculated in response to the cell delays in (c)

% Results Included	**100%**	**Top 25%**	**Top 5%**
R/T-DOrg	1.45%	6.05%	23.27%
50AutoOrg	2.06%	3.33%	4.11%
150AutoOrg	2.17%	5.25%	13.63%

(a) Expected $\mathbb{R}_{Op}$ improvement.

% Results Included	**100%**	**Top 25%**	**Top 5%**
R/T-DOrg	37.07%	34.81%	30.39%
50AutoOrg	38.84%	39.29%	49.15%
150AutoOrg	19.36%	20.75%	38.91%

(b) Expected $\mathbb{C}_{Op}$ reduction.

Table 4.4: Percent $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ improvement compared to the baseline MAS for experiments in Section 4.2.5. The same subsets of episodes are used to calculate the $\mathbb{C}_{Op}$ reduction. The subset of episodes used varies by organization as well as by number of smoothing iterations.

results from the best hand-designed organization from Section 4.1. As Table 4.4 illustrates, the XAutoOrgs compare well against the hand-designed R/T-DOrg, but make different tradeoffs as demonstrated by the top 25% and 5% columns. That is, R/T-DOrg has little to no $\mathbb{R}_{Op}$ impact in most episodes, but then has substantial $\mathbb{R}_{Op}$ gains in a few episodes, whereas the XAutoOrgs yield a slightly larger overall $\mathbb{R}_{Op}$ improvement, but accomplish this by having moderate $\mathbb{R}_{Op}$ gains in many episodes. This observation suggests that the XAutoOrgs are not over-specialized to particular situations the ODP might have encountered, but rather provide general influences, since their $\mathbb{R}_{Op}$ gains are more uniformly distributed over the problem space relative to R/T-DOrg. Also observe that, as the ODP gains a more complete input model, it uses this additional information to infer more specialized organizational patterns and thus exerts more specialized influences, as evidenced by the 150AutoOrg having statistically significant $\mathbb{R}_{Op}$ gains relative to 50AutoOrg in the 25% and 5% cases. Finally, observe that the XAutoOrgs' $\mathbb{C}_{Op}$ improvements actually increase as we filter out episodes—in stark contrast to the R/T-DOrg. Recalling from Section 4.1, the episodes where organizational influence are most meaningful are those where a high-intensity fire is between the agents' initial locations. While the R/T-DOrg approaches these cases with I_c transition shaping, the XAutoOrgs instead address them with state/action shaping (e.g., by delegating the northern cells to one agent and the southern to the other), which reduces $\mathbb{C}_{Op}$ rather than increasing it.

4.3 Metareasoning through Organizational Design

As one may have noticed throughout my empirical results thus far, $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ are essentially inversely correlated, where an ODP can create an organization that improves $\mathbb{R}_{Op}$ by inducing the MAS to reason about additional actions, which increases $\mathbb{C}_{Op}$ (and *vice versa*). Moreover, there is often not a single organization that Pareto dominates the others for all operational performance, $\mathbb{P}_{Op} = f(\mathbb{R}_{Op}, \mathbb{C}_{Op})$, parameterizations. For example, in Figure 3.3, computing the optimal joint policy is best when $\mathbb{C}_{Op}$ is unimportant compared to $\mathbb{R}_{Op}$, but as the $\mathbb{P}_{Op}$ parameterization increasingly values low $\mathbb{C}_{Op}$, the fullOverlapOrg becomes best, then the smallOverlapOrg, and finally the partitionOrg is best when $\mathbb{R}_{Op}$ is relatively unimportant compared to $\mathbb{C}_{Op}$. A natural question, therefore, is whether an ODP can design an organization that optimally selects from the $\mathbb{P}_{Op}$ Pareto front, given a target Pareto parameterization as input.

This question also relates to an observation from Section 4.2.4, where the mechanism for selecting organizational influences only specified influences that were not expected to decrease $\mathbb{R}_{Op}$. Even if an influence would only sacrifice a negligible amount $\mathbb{R}_{Op}$ to reduce $\mathbb{C}_{Op}$ by orders of magnitude, the mechanism would not include that influence in the organizational specification. If agents have unlimited/sufficient computational resources (i.e., $\mathbb{C}_{Op}$ is insignificant), then such a strategy is justified; however, commonly (especially in Dec-MDP models where policy spaces can quickly become impractical to solve) computational costs can exceed the time before decisions must be made. A possible technique in response to this observation is to allow the ODP to specify an influence that sacrifices some amount of $\mathbb{R}_{Op}$ (beyond zero) in proportion to the $\mathbb{C}_{Op}$ savings that the influence imparts.

Determining how much and/or which reasoning an agent should perform is the subject of metareasoning research, which I briefly overview next in Section 4.3.1. Taking an organizational stance to multiagent metareasoning, in Section 4.3.2, I describe how to extend my ODP to (approximately) optimally solve the organizational design problem (Definition 3.6) including metareasoning issues by framing it as incremental search of the organizational influence space. Then, in Section 4.3.2.3, I illustrate an implementation of this search process for influences to agents' action spaces. Finally, I evaluate my ODP in Section 4.3.3 and find that it generates organizations that effectively impart a desired metareasoning regime upon the MAS.

4.3.1 Background

Metareasoning is typically studied in the context of real-time systems and/or agents with bounded rationality, which each naturally emphasize the premise that agents must make Pareto tradeoffs between exerting effort on reasoning about actions to execute, and exerting effort on actually executing those actions. This premise is especially true with multiagent decision-theoretic frameworks (like the Dec-MDP framework I have adopted), where high computational complexity inevitably limits the scale of problems that can be solved. While metareasoning comes in many forms (see Cox & Raja (2011) for a thorough discussion of work in this field), the type that is most relevant for this dissertation addresses the followowing question: how does an agent decide whether the improvements to decisions from additional reasoning are expected to outweigh the costs of delaying enacting its decisions? When metareasoning issues are considered, an agent's objective shifts from computing π^* to instead striking the optimal balance between its computational costs and the quality of the best policy it can identify within those costs. Several approaches have been developed towards solving the metareasoning problem, including anytime algorithms (Hansen & Zilberstein, 2001b) and model shaping (Bratman et al., 2012).

Unsurprisingly, metareasoning becomes even more complicated in MASs, since the benefits of additional reasoning might depend on the reasoning and actions of other agents (Raja & Lesser, 2007). For example, if one agent assumes responsibility for (reasoning about) performing a task, then there might be no additional benefit for other agents to also reason about that task. Thus, research into multiagent metareasoning has typically been formulated as a metacoordination problem, where agents individually make metareasoning decisions but coordinate those decisions to strike a good collective balance between their expected joint performance and reasoning costs (Raja & Lesser, 2007; Alexander et al., 2007).

In the remainder of this chapter, I investigate an alternative approach to solve the multiagent metareasoning problem through organizational design, where a good multiagent organization should both guide agents into coordinated local policies, and also guide them into coordinated reasoning about their individual decision problems. Of course, this approach simplifies the multiagent metareasoning problem that the agents face by complicating the organizational design problem to find a design that not only leads to coordinated action in the world, but also coordinated utilization of the agents' distributed reasoning resources.

The idea that an organization can impact agents' reasoning and behaviors is well-established. For example, social laws (Shoham & Tennenholtz, 1995) affect the reasoning that agents perform as well as the actions they execute. In prior work, however, the impact that an organization has on the agents' reasoning has been an incidental side effect rather than something an ODP *explicitly* leveraged to *intentionally* impart a specific, desired metareasoning regime upon the agents, i.e., a specific $\mathbb{P}_{Op}$ parameterization. Additionally, typical metareasoning approaches try to dynamically assess the predicted benefit of additional reasoning, whereas the fundamental idea of my approach is to have an ODP utilize its global view of the problem domain to identify optimally-coordinated policy patterns, and then influence the agents to avoid *even thinking* about acting counter to those patterns. For example, using its global perspective, an ODP might identify that agents should typically fight fires near their initial locations. It might then codify this pattern by restricting an agent from reasoning about fighting fires in distant cells, which imposes a metareasoning regime that trades computational speedup (due to never considering fighting fires in those distant cells) for small expected reward loss (in the rare cases that it should fight those fires).

4.3.2 Extending the ODP

Extending the ODP to incorporate metareasoning issues means that it will have to solve my organizational design problem (Definition 3.6): $\Theta^* \equiv \arg\max_\Theta \mathbb{P}_{Op}(\Theta)$. Since direct enumeration of the organizational design space is infeasible (Section 3.2.1), I focus on incremental search, and on techniques for computing the incremental impact of an individual influence. In what follows, I formulate a simple greedy hill-climbing search; however, other incremental search algorithms (e.g., Monte Carlo, A*, etc.) could be used instead (see Section 4.3.4 for further discussion).

Naïvely, a greedy algorithm computes the $(j+1)$-th step of the hill climb by determining the influence, Δ_i, with maximal organizational performance improvement (I will return to the topic of determining Θ^0 later in Section 4.3.2.3):

$$\Theta^{j+1} = \Theta^j + \arg\max_{\Delta_i} \mathbb{P}_{Op}(\Theta^j + \Delta_i) \tag{4.1}$$

Notice however, that Equation 4.1 requires recomputing the performance contribution of Θ^j for each $\mathbb{P}_{Op}(\Theta^j + \Delta_i)$, which could waste substantial computational effort. If the calculation of $\mathbb{P}_{Op}(\Theta^j + \Delta_i)$ was instead decomposed into $\mathbb{P}_{Op}(\Theta^j)$ and the conditional, incremental impact of Δ_i w.r.t. Θ^j, then the ODP could avoid this redundant computation. I achieve this by estimating $\mathbb{P}_{Op}(\Theta^j + \Delta_i)$ as a linear approximation.

Assuming $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ are everywhere differentiable,[1] and somewhat abusing notation, I can write the equation in standard linear approximation form:

$$\mathbb{R}_{Op}(\Theta^j + \Delta_i) \approx \mathbb{R}_{Op}(\Theta^j) + \Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$$
$$\mathbb{C}_{Op}(\Theta^j + \Delta_i) \approx \mathbb{C}_{Op}(\Theta^j) + \Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$$

Substituting these linear approximations into the definition of $\mathbb{P}_{Op}$ (Definition 3.5) yields:

$$\mathbb{P}_{Op}(\Theta^j + \Delta_i) = f\left(\mathbb{R}_{Op}(\Theta^j) + \Delta.\frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j), \mathbb{C}_{Op}(\Theta^j) + \Delta.\frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)\right)$$

Now, taking the linear approximation for $\mathbb{P}_{Op}(\Theta^j + \Delta_i)$ (assuming it is also everywhere

[1] While the everywhere differentiable assumptions for $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ are theoretically required, in practice I have not found them necessary since my ODP does not explicitly compute the derivatives (see Sections 4.3.2.1 and 4.3.2.2).

differentiable[2]):

$$\begin{aligned}\mathbb{P}_{Op}(\Theta^j+\Delta_i) \approx & f\left(\mathbb{R}_{Op}(\Theta^j), \mathbb{C}_{Op}(\Theta^j)\right) + \Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)\frac{\delta f}{\delta \mathbb{R}_{Op}}(\Theta^j) \\ & + \Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)\frac{\delta f}{\delta \mathbb{C}_{Op}}(\Theta^j)\end{aligned}$$

Substituting into Equation 4.1:

$$\begin{aligned}\Theta^{j+1} = \Theta^j + \arg\max_{\Delta_i} \Bigg[& f\left(\mathbb{R}_{Op}(\Theta^j), \mathbb{C}_{Op}(\Theta^j)\right) + \Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)\frac{\delta f}{\delta \mathbb{R}_{Op}}(\Theta^j) \\ & + \Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)\frac{\delta f}{\delta \mathbb{C}_{Op}}(\Theta^j)\Bigg]\end{aligned}$$

Finally, the $f(\mathbb{R}_{Op}(\Theta^j), \mathbb{C}_{Op}(\Theta^j))$ term can be dropped since it is independent of Δ_i, yielding the incremental organizational design problem my ODP solves:

$$\Theta^{j+1} = \Theta^j + \arg\max_{\Delta_i} \left[\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)\frac{\delta f}{\delta \mathbb{R}_{Op}}(\Theta^j) + \Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)\frac{\delta f}{\delta \mathbb{C}_{Op}}(\Theta^j)\right] \quad (4.2)$$

Significantly, Equation 4.2 avoids redundantly computing how Θ^j impacts the operational performance, and instead only computes the conditional operational performance impact of Δ_i w.r.t. Θ^j.

In Sections 4.3.2.1 and 4.3.2.2, I describe a general methodology for efficiently computing $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ respectively. Then, in Section 4.3.2.3, I illustrate in detail how an ODP can implement this methodology for influences to the agents' action spaces.

4.3.2.1 Computing Incremental Reasoning Costs

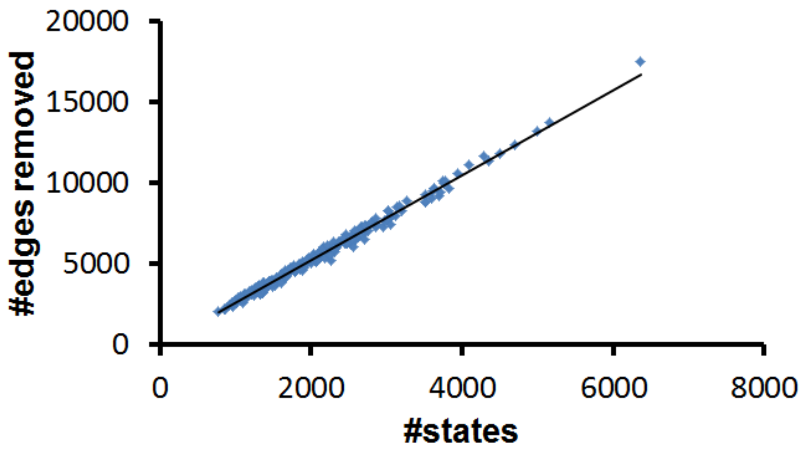
In this section, I describe how an ODP can compute $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$, the conditional impact to the agents' computational costs from adding Δ_i w.r.t. Θ^j. A Dec-MDP agent's computational costs are determined by two primary factors (Littman et al., 1995), the number of states in its decision problem and the number of edges in its state graph. Thus, computing incremental computational costs relies on determining the expected marginal costs of adding a new state/edge, and then calculating the expected change to the number of states and edges caused by adding Δ_i into Θ^j.

[2] The assumption of $\mathbb{P}_{Op}$'s differentiability, unlike for $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ is practically required since my ODP directly computes derivatives of $\mathbb{P}_{Op}$. As a result, my ODP can only consider metareasoning Pareto fronts that are everywhere differentiable.

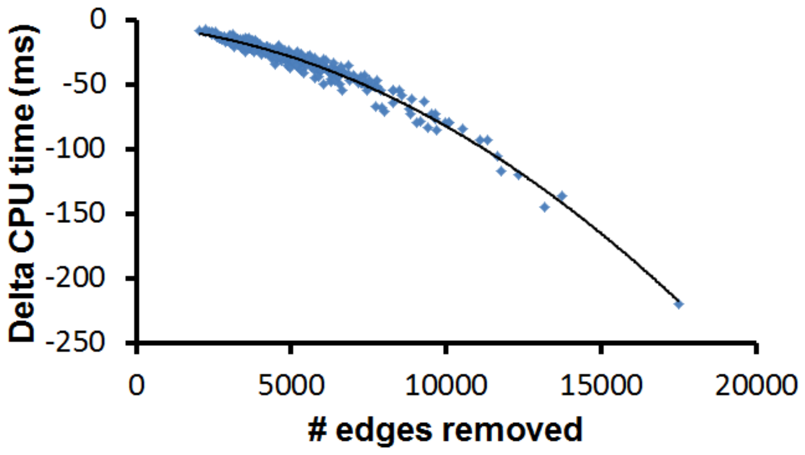
My methodology for empirically estimating the marginal cost of a state and/or edge is as follows. An agent first uses its local model, $\mathcal{M}_i$, to compute π_i^* for an episode in a set of randomly sampled episodes. Then, for each episode, I create a modified version of $\mathcal{M}_i$, labeled as $\mathcal{M}_i'$, that contains the minimal number of edges between states such that the reachable state space from the initial states is unchanged, and the optimal policy is unchanged. I include the latter condition so that the bias of the estimate matches desired organizational influences that streamline the agents' reasoning without precluding optimal policies. The agent then solves each problem episode again, but plans using the respective $\mathcal{M}_i'$ for that episode instead of $\mathcal{M}_i$. Taking the relative computational difference between these experiments provides an empirical estimate of an edge's marginal cost. To compute an estimate of a state's marginal cost, I looked across the episodes at the computational differences for the $\mathcal{M}_i'$ experiments (each episode typically has a different numbers of reachable states). Since the state graph in the $\mathcal{M}_i'$ for each episode is "minimally" connected, the agents' computational costs are almost completely derived from the number of states (at least insofar as possible). Consequently, this methodology "maximally" disentangles the cost of a state and the edges to connect it to the state space, and provides a good estimate of a state's marginal cost.

To demonstrate the use of my methodology, Figure 4.6 shows its application in the firefighting domain, using 300 randomly-generated episodes, along with best fit lines. Taking the derivative of Figure 4.6a, $\mathcal{M}_i'$ removes approximately 2.6 edges for every state. Taking the derivative of Figure 4.6b shows that an edge's marginal computational cost is approximately $1.2e_i + 2000$ (ns), where e_i is the current number of edges. Taking the derivative of the $\mathcal{M}_i'$ line in Figure 4.6c shows that a state's marginal computational cost is approximately $5.28s_i + 3000$ (ns), where s_i is the current number of states. The exact values I found here are clearly only applicable for my agents' specific policy creation implementation within the firefighting domain; however, the methodology generalizes to any problem domain expressed as a Dec-MDP, and to any Dec-MDP solution techniques. More broadly, the approach of enumerating the factors that contribute to an agents' planning costs, then computing the expected marginal cost for each factor, is extensible to any well-defined agent reasoning framework.

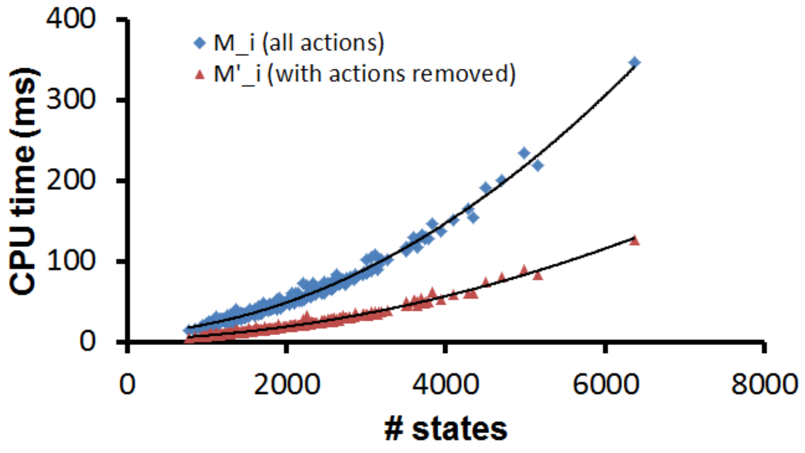
It is worth noting that measuring the marginal costs of states/edges using this methodology does have a bias. Depending on the reward and transition topologies and on the agent's reasoning algorithm, the agent might not consume equal computational costs on optimal versus non-optimal actions. Since I am only removing non-optimal actions, I am biasing the measurements towards the costs of removing non-optimal



(a) # of edges removed versus # of states.



(b) CPU savings vs. # of edges removed from the agent's reasoning problem



(c) CPU time for agent vs. # of states in the agent's reasoning problem

Figure 4.6: Marginal cost estimates for adding or removing a state/edge from an agent's local reasoning problem.

actions (as opposed to any action). However, since the ODP seeks influences that remove non-optimal actions rather than optimal ones, the measurement is biased in the same way that we would want an ODP to exert influences.

4.3.2.2 Computing Incremental Reward

The incremental reward, $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$, corresponds to the expected Q-value change from adding Δ_i into Θ^j. Examining Equation 2.2, a Q-value $Q^{\pi}(s^t, a)$ only changes if Δ_i alters π, the immediate reward $R(s^t, a, s^{t+1})$, or the transition probabilities $P(s^{t+1}|s^t, a)$. Since alterations to $R(s^t, a, s^{t+1})$ or $P(s^{t+1}|s^t, a)$ also induce changes to π (and otherwise Δ_i's impact to Q-values is trivial to compute), I focus on how Δ_i alters the agents' joint policy w.r.t. Θ^j. While the ODP could do this by calculating $\pi^{|\Theta^j}$ for each iteration of the incremental search, such an approach is computationally daunting given the complexity of computing optimal policies and the number of iterations the search might require.

Instead, the insight my ODP exploits is that it can use its global view to compute/estimate the optimal joint policy, π^* (i.e., via the techniques from Section 4.2.2), once, and then only consider candidate Θ^js that preserve π^* while steering agents away from taking, and even considering, actions outside of π^*. So long as Θ^j does not preclude π^*, then the calculation of Δ_i's impact to the agents' joint policy is independent of Θ^j, and the ODP does not need to compute $\pi^{|\Theta^j}$. In the event that Θ^j does preclude π^* (e.g., which could be optimal in a Pareto topology where computation is extremely prohibitive), then statistics computed from π^* represent an optimistic upper bound of the influences' incremental $\mathbb{R}_{Op}$ impacts (see Section 4.3.4). While this methodology (unavoidably) requires the ODP to determine what good actions are by calculating an optimal joint policy, the ODP only need do this costly calculation once (rather than once for each of the $O(|\mathbf{\Delta_i}|)$ search iterations), and then amortize those costs over all of the search iterations, which results in substantial computational savings.

4.3.2.3 Action Influences

In this section, I illustrate how an ODP can implement the general methodologies from Sections 4.3.2.1 and 4.3.2.2 for action influences. I revisit extensions to other influence mechanisms again in Section 6.2.2. I chose to implement action influences because they are more straightforward (as explained below) while also being a particularly commonplace organizational mechanism in previous research (Shoham & Tennenholtz, 1995; Pacheco & Carmo, 2003; Horling & Lesser, 2008). Note however, that prior work has not given explicit, quantitative consideration to how such influences affect the agents' metareasoning regime, which is the focus here.

As I will discuss further in Section 6.2.2, actions are also a computationally simpler influence mechanism for which to compute the incremental impact $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$. That is, as I will show below, the ODP can compute the incremental impact of an influence that prevents an agent i from considering a_i in s_i^t by examining π^* and $\mathcal{M}_i$ at s_i^t, since the effects of the influence are so locally scoped with respect to the agent's reasoning process. In contrast, other influence types (i.e., states and transitions) can have far-reaching effects, and affect the agent's decisions process at multiple states and at potentially different system times.

An action influence, Δ_i, that removes action a_i from consideration in state s_i^t, removes one edge for each possible successor state upon taking a_i in s_i^t, and removes any now-unreachable states. By enumerating the successor states (via the transition function), an ODP can calculate the expected change to the number of edges, $|E_i^{\Delta_i}|$, and states $|S_i^{\Delta_i}|$, caused by adding Δ_i to Θ^j. Combining those quantities with the previous marginal cost estimates in Section 4.3.2.1 yields:

$$\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j) = \left(5.28|S_i^{\Theta^j}| + 3000\right)|S_i^{\Delta_i}| + \left(1.2|E_i^{\Theta^j}| + 2000\right)|E_i^{\Delta_i}|$$

where $|S_i^{\Theta^j}|$ and $|E_i^{\Theta^j}|$ are the expected number of states and edges respectively for agent i given that it conforms to Θ^j. $|S_i^{\Theta^j}|$ and $|E_i^{\Theta^j}|$ are given from the previous search iteration, meaning this computation requires only $O(|S_i^{successor}|)$ time for enumerating the successor state space.

The expected Q-value change associated with an action influence Δ_i, that removes action a_i from consideration in state s_i^t, is equal to the expected difference between the Q-value of a_i and the next best action. Mathematically this yields,

$$\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j) = E_{s^t \mapsto s_i^t}\left[\left(\left(\max_{a=\langle\cdot,a_i,\cdot\rangle} Q^{\pi^*}(s^t,a)\right) - \left(\max_{a'\neq\langle\cdot,a_i,\cdot,\rangle} Q^{\pi^*}(s^t,a')\right)\right) x(s^t,a)\right]$$

This computation requires $O(|A||S|)$ time in the worst case, but the $|S|$ term represents the number of states that map into s_i^t. In expectation, this term is $\frac{|S|}{|S_i|}$, and assuming the agents have meaningful local observation capabilities, will typically be much less than the total number of global states. For example, in the firefighting experiments that follow in Section 4.3.3, 50 global states (from the typically hundreds of thousands) map to a single local state.

Algorithm 4.2 shows how an ODP can embed the above computations in a greedy hill-climbing search to create an organizational design that modifies the agents' local action spaces so as to impart a target metareasoning regime. The algorithm begins by initially preventing all actions from consideration in every state for every agent (lines 1–3). For the firefighting domain, I choose to begin with an organizational design (Θ^0) that prevents everything and then the ODP adds actions back in, as opposed to beginning by allowing consideration of everything and subtracting actions out, due to the topologies of $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ and the tradeoff function, $f(\cdot)$ between them. That is, in the firefighting domain, Δ_i's with low $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ tend to also have low $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$. This is because actions with low expected Q-value (over all sampled problems as calculated within $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$) tend to occur in states that are infrequently reached, in large part since $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ is based on the occupancy measures. For example, moving to a distant cell has low Q-value both because fighting fires there is typically the responsibility of another agent, and because few feasible trajectories bring the agent to that cell. As such, if the algorithm begins by allowing consideration of all actions, it begins at a point where the magnitude of the gradient is small and thus is prone to premature halting. However, by beginning by preventing consideration of everything, the algorithm begins at a point with a large gradient, and is likely to find a better solution. Intuitively speaking, this pattern seems like it would be common in many domains, since computation of $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ is based on occupancy measures, but if not, then the algorithm should ideally be initialized to a point of the organizational design space with a high performance gradient. Other alternatives to discourage the ODP from identifying poor local optima include standard hill-climbing techniques like Monte Carlo, random restarts, simulated annealing, etc.

After initialization, the algorithm greedily selects local actions to add back in (lines 4–12). An important subtlety here is that I restrict the space of possible Δ_is to those actions possible in each agent's currently reachable local state space. This is important because efficiently utilizing the methodologies from Sections 4.3.2.1 and 4.3.2.2 relies on the assumption that only the current increment, Δ_i, to Θ^j impacts the performance. For example, when calculating $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$, the only information the ODP must determine is $|S_i^{\Delta_i}|$ and $|E_i^{\Delta_i}|$. If portions of the unreachable state space could already be connected (i.e., due to adding back an action in some unreachable state in an earlier iteration), then $|S_i^{\Delta_i}|$ and $|E_i^{\Delta_i}|$ could depend on the history introducing of prior Δ_is, which would greatly increase the complexity of the computation. Moreover, influences to unreachable states by definition have zero expected impact, thus the search algorithm can safely ignore such influences. Finally (lines 13–21), Algorithm 4.2

uses the $noAStates_i$ sets to ensure that each agent will have a well-defined local problem by forcing every reachable state to have at least one available action.

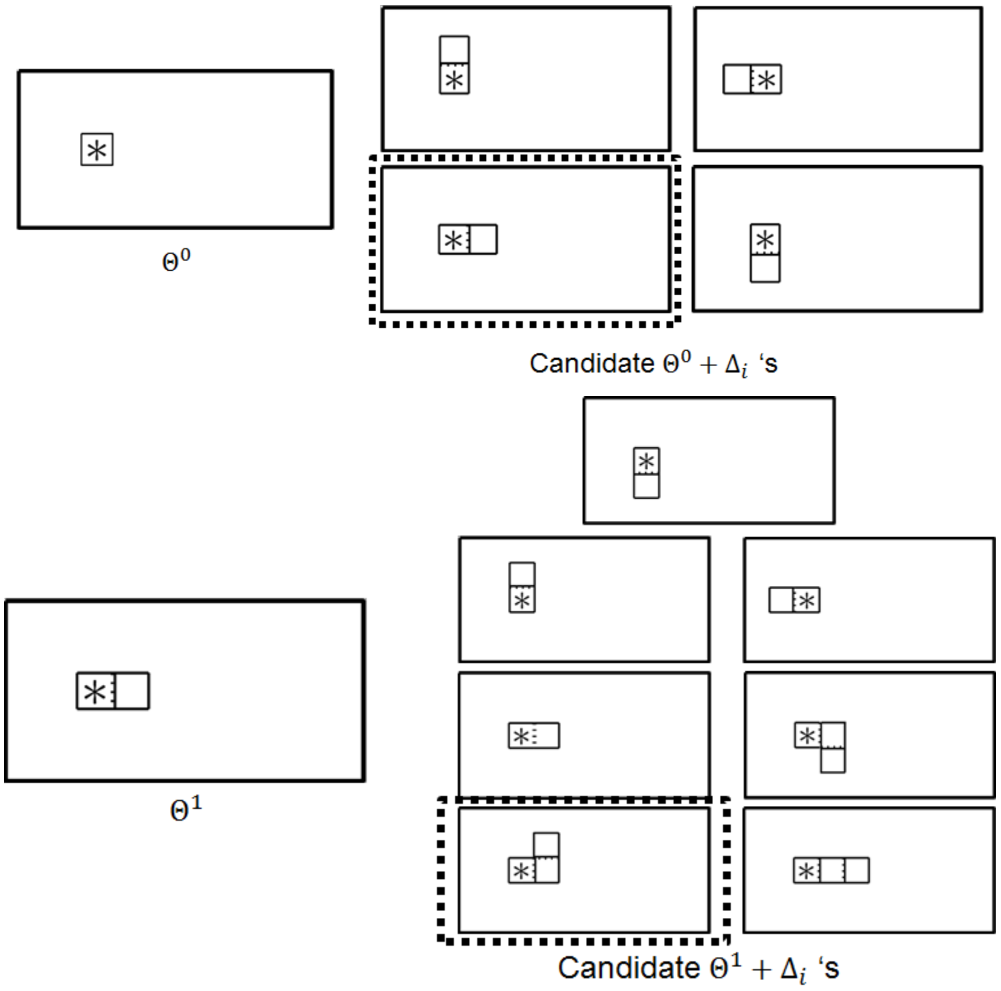
Figure 4.7 illustrates how Algorithm 4.2 searches through the influence space for agent 1's movement actions. (Influences to the fight fire and NOOP actions as well as all influences for agent 2 are omitted for ease of illustration.) As the hill climbing progresses, the ODP adds additional actions for the agent to consider, where actions with the highest Pareto improvement are added first.

The computational complexity of Algorithm 4.2 is $O(|\mathbf{\Delta_i}|^2 \cdot \mathbb{P}_{Op\ cost})$, where $\mathbb{P}_{Op\ cost}$ is the complexity of calculating $\mathbb{P}_{Op}(\Theta - \Delta_i)$. From the discussions earlier in this section, $\mathbb{P}_{Op\ cost}$ for an action influence Δ_i that prevents consideration of a_i in state s_i^t has complexity $O(|S_i^{successor}| + |A||S|)$ corresponding to the total cost to compute $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$. It is worth noting, however, that this result hides the computational cost for calculating the optimal joint policy necessary for computing $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$, which could in fact be a substantial fraction of the ODP's computation in practice (e.g., in my empirical evaluation in Section 4.3.3, computing joint policies for sampled problem episodes accounts for $\gg 99\%$ of the ODP's computational costs). Thus, more precisely, the total complexity of Algorithm 4.2 is $O\left(|\mathbf{\Delta_i}|^2 \left(|S_i^{successor}| + |A||S|\right) + |A_i|^{|S_i|^n}\right)$. Of course, the $|A_i|^{|S_i|^n}$ term is quite poor; however, intuitively *any* computational ODP would require knowing a quantitative description of organizational patterns (without which it could not decide which interactions the MAS should pursue), and thus for a Dec-MDP based MAS, would also include this term in its complexity analysis.

4.3.3 Evaluation

I begin my empirical evaluation by briefly describing some implementation parameters of the evaluation domain and ODP. These experiments use the firefighting domain, where in each episode there are: two agents; two fires, each with initial intensity independently and uniformly selected from $\{1, 2, 3\}$, and with a uniformly random, but distinct location; delay in each cell independently and uniformly chosen from $[0, 1]$; and a time horizon of 10. Agents incorporate their organizations into their local models and solve their decision problems using the methodology developed in Section 3.5.1.

In principle, an ODP could associate an action influence Δ_i with a specific local state (i.e., with a distinct influence for each agent-action-state combination). Section 4.1 as well as previous research (Dignum et al., 2005), however, has shown that providing abstract influences as opposed to detailed micromanagement can be beneficial when



Θ^0
Candidate $\Theta^0 + \Delta_i$'s
Θ^1
Candidate $\Theta^1 + \Delta_i$'s

Figure 4.7: Illustration of influence search (Algorithm 4.2) for agent 1's movement actions. The agent can move into a cell in a direction where it first passes a dotted line, but not a solid line. The agent begins in the cell indicated with $*$. Dashed boxes indicate the optimal influence to add into the organization for each search iteration.

Algorithm 4.2 Action Influence Creation

Input: ODP's domain model, and the Pareto optimality function f

Output: Set of organizational influences, Θ, that modify agents' action spaces

```
1:  $\Theta \leftarrow \forall i, \forall t, \forall s_i^t, A_{i,s_i^t}$        \\initially prohibit all actions in all states
2:  $\forall i, \tilde{S}_i \leftarrow \{s_i^0\}$        \\$\tilde{S}_i$ is agent $i$'s reachable state space
3:  $\forall i, noAStates_i \leftarrow \{s_i^0\}$        \\$noAStates_i : \forall s_i^t \in \tilde{S}_i$ s.t. $A_{i,s_i^t} = \emptyset$
4:  while $\exists(\Delta_i = a_{i,s_i^t}) \in \Theta$ s.t. $\mathbb{P}_{Op}(\Theta) < \mathbb{P}_{Op}(\Theta - \Delta_i)$ do
5:      $\Delta_i^* \leftarrow \underset{\Delta_i \in \{\forall i, \forall s_i^t \in \tilde{S}_i, a_{i,s_i^t} \in \theta_i\}}{\arg\max} - \left[\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta}(\Theta)\frac{\delta f}{\delta \mathbb{R}_{Op}}(\Theta) + \Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta}(\Theta)\frac{\delta f}{\delta \mathbb{C}_{Op}}(\Theta)\right]$
6:      $\theta_i \leftarrow \theta_i - \Delta_i^*$
7:      if $\Delta_i^*$'s $s_i^t \in noAStates_i$ then $noAStates_i \leftarrow noAStates_i - s_i^t$
8:      for possible successor states, $s_i^{t+1}$, from $\Delta_i^*$ that $\notin \tilde{S}_i$ do
9:          $\tilde{S}_i \leftarrow \tilde{S}_i \cup s_i^{t+1}$
10:         $noAStates_i \leftarrow noAStates_i \cup s_i^{t+1}$
11:     end for
12: end while
13: while $\exists i, noAStates_i \neq \emptyset$ do
14:     $\Delta_i^* \leftarrow \underset{\Delta_i \in \{\forall i, \forall s_i^t \in noAStates_i, a_{i,s_i^t} \in \theta_i\}}{\arg\max} - \left[\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta}(\Theta)\frac{\delta f}{\delta \mathbb{R}_{Op}}(\Theta) + \Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta}(\Theta)\frac{\delta f}{\delta \mathbb{C}_{Op}}(\Theta)\right]$
15:     $\Theta \leftarrow \Theta - \Delta_i^*$
16:     $noAStates_i \leftarrow noAStates_i - s_i^t$
17:     for possible successor states, $s_i^{t+1}$, from $\Delta_i^*$ that $\notin \tilde{S}_i$ do
18:         $\tilde{S}_i \leftarrow \tilde{S}_i \cup s_i^{t+1}$
19:         $noAStates_i \leftarrow noAStates_i \cup s_i^{t+1}$
20:     end for
21: end while
22: return $\Theta$
```

agents possess local expertise. For now (I revisit this issue in depth in Chapter 5), I incorporate this principle in two ways:

1. By presenting the ODP with a model where it only knows the mean cell delay as opposed to the specific delay of each cell for an episode. This also provides the agents with local expertise relative to the ODP (e.g., as in Section 4.1).

2. By having the ODP consider action influences for each agent i that remove an action, a_i, from an abstract (rather than specific) local state, $\hat{s}_i^t$, where the abstraction drops all state factors excluding agent i's location. The ODP computes the incremental operational reward and cost associated with an abstracted influence $\hat{\Delta}_i$ by taking the expectation over the set of influences, $\{\Delta_i\}$, that map into $\hat{\Delta}_i$ (i.e., whose states map into $\hat{s}_i^t$). This abstraction was chosen

to prevent the ODP from micromanaging the agents, and forces an influence to apply to a broader set of situations. Using an abstracted influence space additionally reduces the ODP's computational requirements, since the influence space is smaller, and fewer samples are needed to compute stable estimates for $\hat{\Delta}_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$. As we will see in Chapter 5, finer abstractions enable the ODP to find more refined organizational designs at the expense of greater ODP computation and/or overfitting (and *vice versa* for coarser abstractions).

The ODP sampled and solved training problems from its domain model until it had stable estimates for $\hat{\Delta}_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$, which took 300 sample episodes in these experiments. To test my claim that the algorithm correctly finds an organizational design that imparts a desired metareasoning regime upon the agents, I explored a space of environments with a range of metareasoning tradeoff demands, parameterized by $\mathbb{P}_{Op} = \mathbb{R}_{Op} - \mathbb{C}_{Op}/b$ for different values of the Pareto optimality parameter $b \in \mathbb{Z}^+$. I present results across b values such that at extremely costly reasoning (low b) the ODP designs an organization where the agents only consider executing a single action (FF in this case), and at extremely low reasoning costs (high b) designs an organization where every action the ODP expects an agent to ever want to execute is included. Note that the latter, max-bOrg, will still exclude local actions that would never be sensible (e.g., fighting fires in distant cells that are always another agent's responsibility).

Unexpectedly, I found that the ODP was able to encode surprisingly nuanced organizational designs despite being limited to a space of abstracted influences. For example, the ODP frequently imposes unidirectional movements (see Figure 4.8), where an agent is allowed to consider moving into a cell, but the action to move back and in effect "undo" the previous action is removed from consideration. This type of influence imparts a good metareasoning regime by forcing the agent to reason about complete, irreversible action trajectories rather than needlessly reasoning about reversing prior actions. These unidirectional movements also yield a coordinated joint policy by discouraging an agent from rushing to the other side of the grid (where the other agent is located) to fight a high-intensity fire since it would be unable to come back and fight an initially-closer fire. Instead, the agent will prioritize fighting fires close to its initial location before moving to the other side of the grid, by which time another agent could have fought those fires.

To quantitatively determine the expected joint reward and agent computational cost characteristics of each organizational design, I presented the MAS with a sequence of 1500 test problem episodes, randomly sampled from $\mathcal{M}^*$, and had the agents utilize each of the organizational designs (as well as a local baseline) in each of the episodes.

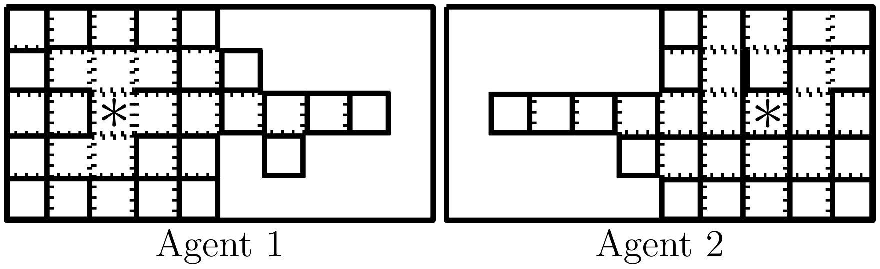
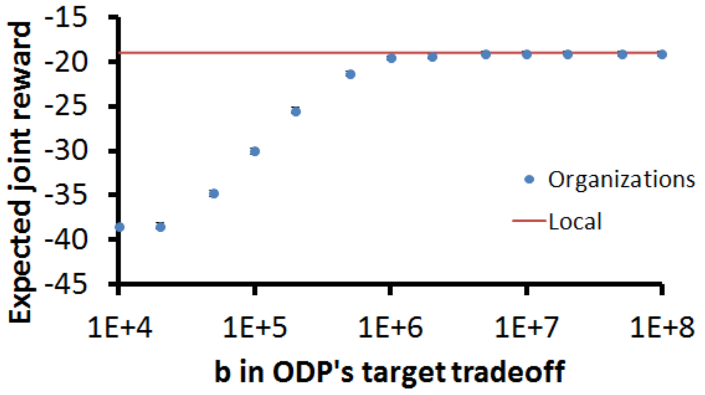


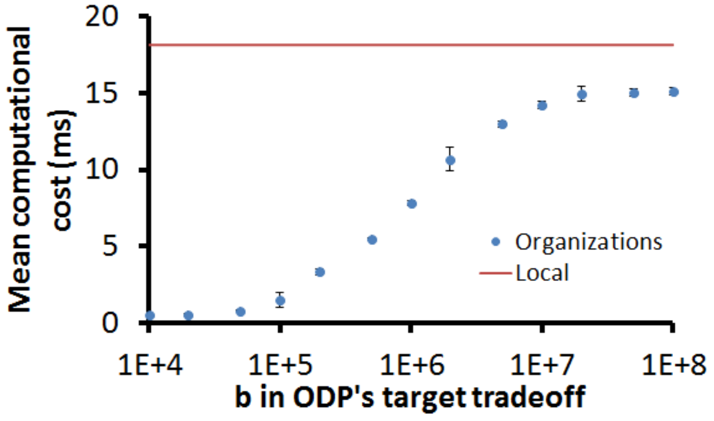
Figure 4.8: Movement action influences of the 1E6Org for each agent. An agent can move into a cell in a direction where it first passes a dotted line, but not a solid line. Agents begin in the cells indicated with $*$.

Figures 4.9a and 4.9b show the mean $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ respectively over the 1500 test episodes for each of the bOrgs and the local baseline. These graphs show that as the ODP faces different target metareasoning tradeoffs (i.e., values of b), the organizations it creates have monotonically increasing performance properties in both $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$. That is, as computation becomes cheaper (b increases), the ODP creates organizations that induce the agents to consider more actions (and thus utilize more computation), which yields increased expected joint reward. Also observe that these bOrgs, which are limited to influences that only remove actions from consideration, do not lead to agents finding better policies than they otherwise would have ($\mathbb{R}_{Op}$s of the bOrgs do not surpass the local baseline), but find these policies with significantly less computation (lower $\mathbb{C}_{Op}$). In Figure 4.9c, I use the expected $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ data to calculate the metareasoning regime imparted upon the agents by each of the organizations as a function of tradeoff parameterizations. This graph shows that, for any target tradeoff parameterization β, the best organizational design (i.e., maximizing the y-axis) is approximately the bOrg the ODP generates with $b = \beta$, which confirms that the ODP designs organizations that approximately optimize the Pareto front defined by the f function within $\mathbb{P}_{Op}$.

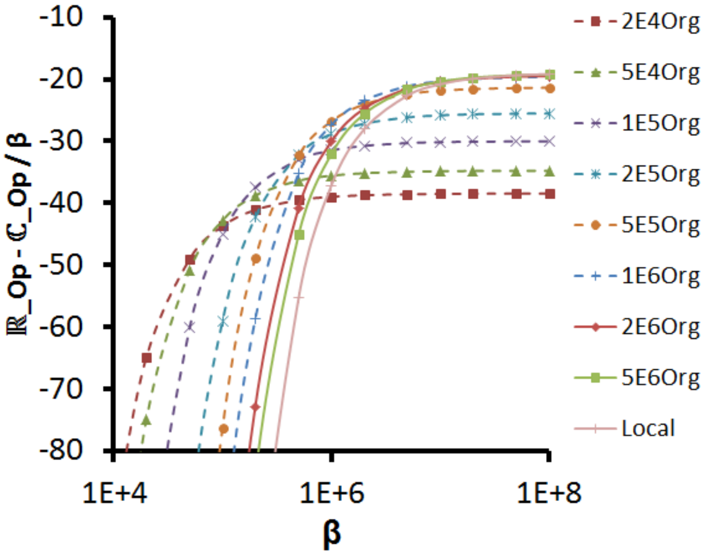
An additional dimension for evaluation is the ODP's computational savings gained by my incremental search techniques. Firstly, if the ODP enumerated the organizational design space, it would evaluate $O(2^{|\boldsymbol{\Delta}_\mathbf{i}|}) \approx 4.2\text{E}180$ candidate organizational designs in the firefighting domain as opposed to the $O(|\boldsymbol{\Delta}_\mathbf{i}|^2) = 360,000$ candidates it considers with my incremental search, which highlights the importance of selectively searching the design space. Secondly, despite being only a constant factor in the theoretical complexity, the ODP's computational costs in these experiments are overwhelmingly dominated by the costs to solve for optimal joint policies to determine $\hat{\Delta}_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$. While calculating optimal joint policies may seem daunting, the ODP would have



(a) $\mathbb{R}_{Op}(\Theta)$ for the bOrgs and local baseline



(b) $\mathbb{C}_{Op}(\Theta)$ for the bOrgs and local baseline



(c) Imparted metareasoning regime for the bOrgs and local baseline as a function of Pareto optimality parameter

Figure 4.9: Performance characteristics of the bOrgs created by my ODP.

to solve for optimal joint policies for *each* candidate design, $\Theta^j + \hat{\Delta}_i$, if it directly calculated the performance of each candidate instead of the (conditionally independent) incremental impact of $\hat{\Delta}_i$. Thus, the approximation techniques empirically result in numerous orders of magnitude of computational savings, and allow the ODP to efficiently compute locally optimal organizational designs.

4.3.4 Limitations and Concerns

The greedy hill-climbing algorithm my ODP utilizes (Algorithm 4.2) is but one choice for searching the organizational influence space. While the computational advantages of an *incremental* search algorithm seem crucial for regulating the ODP's computational complexity (Section 4.3.2), other incremental search algorithms, such as Monte Carlo, A*, etc., could supplant the greedy hill climb that my ODP utilized.

A particular weakness of the greedy hill-climbing approach that I have identified is that it can sometimes mis-apply influences that are along the front of the $\mathbb{P}_{Op}$ Pareto topology. Namely, the ODP's statistical estimates are constructed from optimally coordinated policies, and as such, the ODP calculates an influence's "optimal" incremental impact, i.e., the calculations for the $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ terms anticipate additional, complementary influences also being added into Θ. In the interior of the Pareto topology, this optimistic anticipation of complementary influences subsequently being added comes to fruition; however, at the Pareto front, the ODP may not add more influences (since they would decrease $\mathbb{P}_{Op}$). Thus, at the Pareto front, the ODP's estimates of $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ are overestimates. Given such overestimates, the ODP can add Δ_i into the organization when it actually shouldn't have. Obviously, a more intelligent search algorithm (e.g., that looks forward to consider the trajectory of influences added into Θ) could fix this inconsistency at the expense of additional computational costs for the ODP; however, it is important to note that the significance of these inconsistencies is small by definition (since they can only occur on the Pareto front). It is also worth noting that unlike other greedy hill-climbing applications where myopic reasoning can lead to halting too soon, my ODP implementation will only mis-apply influences by adding *too many* influences into Θ. That is, since $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ can never be an underestimate, my ODP will never undervalue an influence.

More broadly, other search algorithms could increase the expected $\mathbb{P}_{Op}$ of the Θ found by the ODP via increasing the likelihood of better local maxima (e.g., Monte Carlo search), or via a more comprehensive search of the organizational design space (e.g., A* search). An open question is whether such improvements are worth pursuing (given their higher computational costs), or if the greedy search is already identifying an Θ that is close to the global optimum, especially if the ODP were parameterized with an optimal abstraction (see Chapter 5).

4.4 Conclusion

In this chapter, I have made two primary contributions. Firstly, in Section 4.1, I contributed a heuristic principle for selecting effective organizational influences (Principle 4.1) that states that a well-designed organization should influence only the factors of agents' models that are associated with agent interactions. By doing so, an organization leaves the agents room to exercise their local expertise while still influencing how that local expertise contributes to the collective interaction patterns. I then identified that for the Dec-MDP framework I have adopted, these types of influences should stem from the agents' reward-/transition-dependencies. In my evaluation of Principle 4.1, I found that it yields robust organizations that outperform the local baseline MAS (especially in a subset of critical problem episodes) while avoiding micromanagement that disregards agents' local expertise.

My identification of Principle 4.1 contributes to the body of organizational reasoning techniques by providing a theoretically-derived, empirically-validated heuristic for delimiting a sub-class of effective organizational influences. Principle 4.1 contributes a design heuristic not only to problem-driven approaches (Section 2.2.2) where it guides a human to identify effective influences, but also to experience-driven approaches (Section 2.2.3) where it informs a MAS (and/or adaptation algorithm) of a sub-class of patterns that are organizationally significant. Moreover, Principle 4.1 contributes a strategy for focusing the efforts of mixed approaches to organizational reasoning (Section 2.2.4), including my agent-driven one, on pertinent influences for an ODP to specify, and suggests techniques for automated organizational design (e.g., like those I developed, see below).

The second contribution of this chapter is the development of computational representations and algorithms for automated organizational design built uponPrinciple 4.1's premise. In Section 4.2.1, I formulated a general-purpose, agent-driven methodology for constructing a computational ODP consisting of three stages, namely: compute organizational patterns; select organizational influences; and influence agent decision making. In the remainder of Section 4.2, I illustrated an implementation of these stages for the Dec-MDP framework I have adopted in this dissertation. I described how my ODP implementation can estimate the qualities of the agents' actions by: Monte Carlo sampling of the problem space; computing the optimal joint policy for each sampled episode; and then aggregating the information computed across the samples. My ODP implementation then selects organizational influences by identifying patterns in the statistical estimates of the quality of the agents' actions. Agents assimilate these organizational influences to their local models and solve their local reasoning processes as described in Chapter 3.1.2. My empirical evaluation of this ODP implementation showed that it creates organizations that are intuitively sensible and can exploit domain structure to yield MAS performance superior to that of my best hand-crafted organizations.

In Section 4.3, I discussed how an organization can be used to impart a metareasoning regime upon the MAS. I extended my ODP implementation to incrementally reason about the expected impact that candidate organizational influences would have on the agents' metareasoning, and developed techniques for efficiently estimating these statistics (for influences that modify an agent's action space). My evaluation demonstrated my ODP implementation's ability to identify intricate, nuanced organizations (despite being limited to action influences) that exploit domain structure while also following the Pareto topology between $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$.

My overarching strategy for automated organizational design contributes a mathematical, agent-driven approach for creating an explicit organization from first principles to the organizational reasoning community, which previously either: relied on human-expertise to provide an organization (e.g., Section 2.2.2); did not explicitly represent the organization (e.g., Section 2.2.3); or focused on configuring/adapting an organization rather than initially creating one (e.g., Section 2.2.4). Moreover, my overarching approach to automated organizational design contributes an agent-driven framework upon which the organizational reasoning community can further study specific aspects of organizational reasoning (e.g., as I do in Chapter 5 with abstract organizational influences). More narrowly, my specific ODP implementation contributes a proof of concept that my overarching ODP strategy is well-formed, and provides a functioning ODP implementation for the Dec-MDP community. Finally, extending my ODP to explicitly reason about and optimize the agents' metareasoning regime contributes the first systematic study of multiagent metareasoning through organizational design, which had previously been viewed as distinct issues and studied by distinct research communities. As such, my techniques in this regard contribute to both the metareasoning and organizational reasoning communities, providing a new solution approach for both metareasoning and organizational design as well as raising awareness of the cross-cutting nature of organizational design.

CHAPTER 5

Abstract Organizational Influences

As identified in Section 4.3.3 and prior organizational reason techniques (e.g., see Section 2.2.2), abstractions can play a central role in the specification of organizational influences. Broad, encompassing influences can be advantageous if an ODP wants to instill overarching organizational guidance for the agents' operational decisions. Alternatively, sets of narrow, detailed influences can be advantageous if an ODP wants to ensure specific coordination patterns specialized for particular environmental conditions. Recent OMLs and ODPs have recognized the significance of abstraction choice for creating effective organizational designs (Dignum et al., 2005; Horling & Lesser, 2008; Hübner et al., 2007; Sims et al., 2008; Sleight & Durfee, 2013, 2014), and typically include mechanisms for an organizational designer to formulate abstract influences. However, the field still lacks a formal understanding of how abstraction choices impact organizational design processes and outcomes.

In this chapter, I systematically construct a formal, agent-driven theory of abstract organizational influences. I begin in Section 5.1 by discussing the motivating factors for abstract organizational influences. Then, I formally define abstract organizational influences and present a mathematical framework for analyzing the dimensions of an influence abstraction mechanism in Section 5.2. In Section 5.3, I describe how to extend my ODP techniques developed in the previous chapters to incorporate abstract influences. Using my framework, I empirically analyze the effects of abstract influences on both the ODP and the performance of organizations designed with abstract influences, and identify characteristics of effective abstractions (Sections 5.4, 5.6, and 5.7). Given these results, in Section 5.5 I converge on task-delineated abstractions as a general-purpose heuristic for selecting an influence abstraction mechanism, and confirm this heuristic's effectiveness empirically.

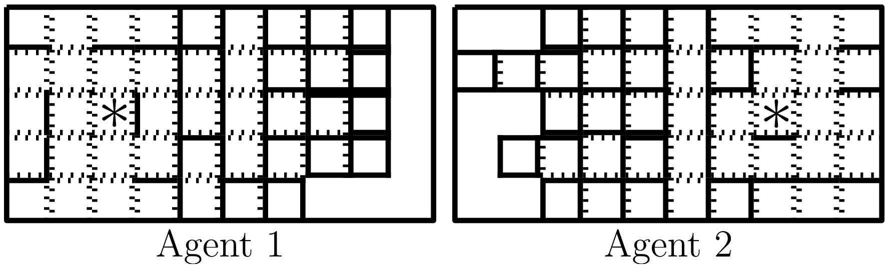


Figure 5.1: Movement action influences in the four-fire domain for each agent using a position abstraction. An agent can move into a cell in a direction where it first passes a dotted line, but not a solid line. Agents begin in the cells indicated with $*$.

5.1 Motivations for Abstract Influences

In Section 4.3.3, the ODP utilized an abstraction that relied on only an agent's position to identify patterns in that agent's reasoning and behaviors. The resulting organizations exploited specific problem properties—especially that a problem instance often had exactly one fire for each agent to fight—and specified influences (e.g., Figure 4.8) that essentially captured that, once an agent starts moving toward a fire, it shouldn't think about reversing its movements, and that it is not useful to (think about) moving to places more easily reached by the other agent. The ODP discovered that, because each agent typically fought only one fire, an organization that disallows reverse movements and movements deep in to the other agent's region of responsibility performs well.

More broadly, however, this class of patterns could be overly restrictive. For example, consider even the seemingly innocent extension of the (two-agent) firefighting domain to have four fires per episode instead of two. Since each agent might fight multiple fires in the extended version of the domain, an encompassing abstraction that at all times prohibits reverse movements could stop the agents from reaching a second fire, and as a result have poor operational performance. Alternatively, the ODP could allow the agents to consider more actions (e.g., Figure 5.1), but at the expense of the agents incurring additional computational costs. Either option results in a decreased net operational performance; however, a narrower abstraction (e.g., that partitions time) could provide the necessary organizational expressivity for the ODP to differentiate these coordination patterns in a more nuanced specification. For example, this could correspond to removing and/or redirecting the influences to prohibit reverse movements after some time has elapsed.

Stepping back, I identify two primary motivations for abstract organizational influences in an agent-driven approach:

1. By generalizing where influences apply beyond just the seen instances, an appropriate abstraction can improve organizational performance. In the firefighting domain, an example is where generalizing instances of purposeful movement toward a fire to prohibit reverse movements *everywhere* can be fruitful.

2. By abstracting over a wider space of instances, an appropriate abstraction can find influence patterns with greater confidence (i.e., it avoids overfitting). For example, in the firefighting domain, this could correspond to seeing enough instances to confidently constrain the agents to local partitions of the grid world.

Of course, these benefits can be lost if abstraction is taken too far. Overextending abstraction can misapply influences, and can conflate patterns or properties of influences that can harm operational performance and/or confuse an ODP's search algorithm. Alternatively, too little abstraction can too sparsely distribute the ODP's limited information, yielding poor statistical estimates that make the ODP's search algorithm sensitive to sampling artifacts, and organizational performance reflective of the agents' arbitrary priors.

Beyond these fundamental motivations, abstract influences could have several other benefits depending on the application. One intuitive example is that abstract organizational influences can reduce organizational specification size. This can be important when an agent queries its θ_i to find the influences associated with the part of its $\mathcal{M}_i$ currently being considered (i.e., as described in Section 3.1.2). In my computational agents, however, I employ hashing to provide $O(1)$ query complexity for organizational lookup, which makes specification size a non-issue. Nonetheless, for other application domains, specification size could be an important attribute, for example, with human agents or agents who must sequentially observe their local state factors and are thus unable to easily hash their entire $s_i^t \times a_i \times s_i^{t+1}$ space into an organizational specification.

The ODP's computational costs are another possible motivating factor for abstraction, where broader influences imply a smaller organizational design space. From Section 4.3.2.3, Algorithm 4.2's complexity is $O\left(|\mathbf{\Delta_i}|^2\left(|S_i^{successor}| + |A||S|\right) + |A_i|^{|S_i|^n}\right)$, which does depend on the size of the organizational design space (i.e., $|\mathbf{\Delta_i}|$). However, it is important to note that the search's computational costs are an insignificant portion (i.e., $\ll 0.1\%$) of the ODP's total computational costs in my experimental implementation; rather, the bulk of the computation is sampling problem instances and computing optimal joint policies to estimate the influences' incremental impacts, which is not affected by abstraction choice. Nonetheless, if a different (e.g., optimal, exhaustive) search algorithm were used instead of a greedy hill climb, then the effects of abstraction to reduce the size of the organizational design space could be an important consequence.

Yet another possible motivation for organizational abstraction is to provide flexibility for the agents to make local decisions within, while still providing organizational guidance for that reasoning. However, counter to possible intuitions, abstract organizational influences are orthogonal to the flexibility an agent retains in an organization. For example, highly flexible organizations can be specified as an aggregation of many fine-grained influences (e.g., in s_i^t consider a_i^1, and in s_i^t also consider a_i^2, and in s_i^t also consider a_i^3, etc.). Thus, flexibility stems from both number of influences and their abstraction. Of course, abstraction choice could impact the ODP's decision of which/how-many influences to specify, but such flexibility differences arise from ODP decisions rather than as necessary consequences of the abstraction choice.

5.2 Dimensions of Abstract Influences

To construct a framework for analyzing abstraction in organizational designs, in this section I provide precise, formal definitions for abstract influences and identify dimensions of abstraction pertinent to organizational design and outcomes.

Broadly speaking, an abstract organizational influence clusters together detailed influences and forces the ODP and agents into a monolithic treatment of the clustered influences. Figure 5.2 illustrates this concept. On the left are examples of optimal actions for A1 to take in its initial position, but sampled for different system times and fire configurations. On the right is the abstract influence that, based on the patterns seen in the samples, indicates that at this position (at any system time or fire configuration) A1 should just consider any of the West, East, or North movement actions. (As mentioned in Figure 5.2, this example is for illustrative purposes only, and not from empirical data.)

Using this intuition, I formally define an abstract influence as follows.

Definition 5.1. *An influence abstraction is a function $G : \Delta \mapsto \hat{\Delta}$, where $\hat{\Delta}_i \in \hat{\Delta}$ is an* ***abstract organizational influence****.*

Building from the clustering perspective, there are three primary dimensions for characterizing abstract influences.

Definition 5.2. *The* ***inclusivity*** *of an abstract influence, $\hat{\Delta}_i$, corresponds to how encompassing the influence is. Formally, the inclusivity of $\hat{\Delta}_i$ is the expected fraction* *of agent i's local model, $s_i^t \times a_i \times s_i^{t+1} \in S_i \times A_i \times S_i$, that $\hat{\Delta}_i$ modifies.*[1]

Definition 5.3. *The **uniformity** of an abstract influence, $\hat{\Delta}_i$, corresponds to how well its composing influences agree on the local model's modification. Formally, the uniformity of $\hat{\Delta}_i$ is the expected fraction of its composing influences that modify agent i's local model in the same way.*

Definition 5.4. *The **variance** of an abstract influence, $\hat{\Delta}_i$, is the expected variance of its composing influences' incremental $(\mathbb{R}_{Op}, \mathbb{C}_{Op})$ impact. This differs from uniformity in that $\hat{\Delta}_i$'s composing influences could all modify $\mathcal{M}_i$ in the same way (thus have uniformity of 1), but have varied estimates for the incremental $(\mathbb{R}_{Op}, \mathbb{C}_{Op})$ impact of the modification (thus have high variance).*

Figure 5.2 illustrates how the abstraction's values along each of these dimensions are calculated using an abstraction that drops all state factors except the agent's current position. I define the uniformity for an organizational design as the expected uniformity over all of the organization's influences (and *mutatis mutandis* for inclusivity and variance).

5.3 Incorporating Abstract Influences

Incorporating abstract organizational influences into my previous ODP algorithm (Algorithm 4.2) requires extending the calculation of an influence's incremental impact to an abstract influence's incremental impact. I do this by taking the expectation over $\hat{\Delta}_i$'s constituent influences.

$$\hat{\Delta}_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j) = E_{\Delta_i \mapsto \hat{\Delta}_i}[\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)]$$
$$\hat{\Delta}_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j) = E_{\Delta_i \mapsto \hat{\Delta}_i}[\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)]$$

Using Figure 5.2 as an example, the ODP's estimate for $\hat{\Delta}_i$'s incremental $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ impact is the expected impact of its constituent influences. The ODP search algorithm uses these values exactly as it would with un-abstracted influences.

Agents incorporate abstract organizational influences into their local reasoning in exactly the same way they incorporate un-abstracted influences (i.e., the process

[1]Beyond abstracting the domain of influences, one could also envision abstracting the range (i.e., the effect of the modification). However, such an approach often decreases uniformity, which is typically undesirable (see Section 5.4). Still, abstracting the range could be an interesting direction for future work (e.g., in the firefighting domain abstracting all movement actions to a single influence).

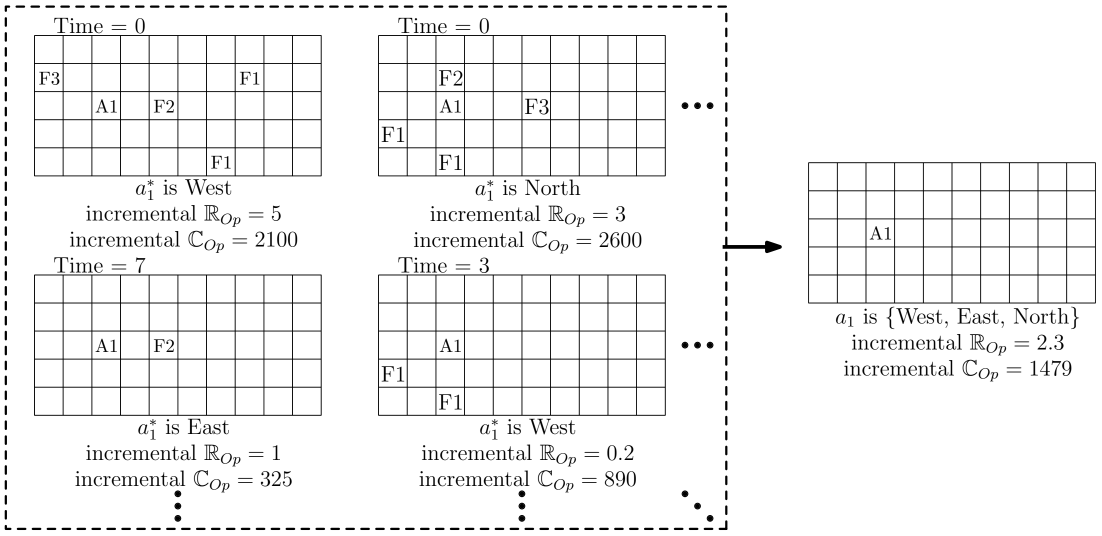

Time = 0
F3
F1
A1
F2
F1
a_1^* is West
incremental $\mathbb{R}_{Op} = 5$
incremental $\mathbb{C}_{Op} = 2100$
Time = 0
F2
A1
F3
F1
F1
a_1^* is North
incremental $\mathbb{R}_{Op} = 3$
incremental $\mathbb{C}_{Op} = 2600$
Time = 7
A1
F2
a_1^* is East
incremental $\mathbb{R}_{Op} = 1$
incremental $\mathbb{C}_{Op} = 325$
Time = 3
A1
F1
F1
a_1^* is West
incremental $\mathbb{R}_{Op} = 0.2$
incremental $\mathbb{C}_{Op} = 890$
A1
a_1 is {West, East, North}
incremental $\mathbb{R}_{Op} = 2.3$
incremental $\mathbb{C}_{Op} = 1479$

Figure 5.2: Illustration of the Pos abstraction's dimensions (see Table 5.1) for agent 1's initial position. The influence's inclusivity is $\frac{1}{50}$, its uniformity is $\frac{1}{3}$, and its $(\mathbb{R}_{Op}, \mathbb{C}_{Op})$ variance is $(4.63, 1.11 \times 10^6)$. Quantities shown are for the four example Δ_1s (rather than $\hat{\Delta}_1$'s entire domain), and are for illustrative purposes only and not from empirical data.

described in Section 3.1.2). As each agent i is solving its local decision problem, it queries θ_i to find the $\hat{\Delta}_i$ that applies to the $s_i^t \times a_i \times s_i^{t+1}$ currently being considered within its $\mathcal{M}_i$, and then modifies its $\mathcal{M}_i$ in accordance with that $\hat{\Delta}_i$. For simplicity in this work, I limit my consideration to abstractions where a Δ_i maps to a single $\hat{\Delta}_i$. In other words, G must be many-to-one, which implies the agents will only receive logically consistent Θs that do not entail incompatible modifications for any $s_i^t \times a_i \times s_i^{t+1}$. Investigating many-to-many abstractions could be an interesting line for future work (see Section 6.2.5), since it provides flexibility for hierarchical organizational influences like those found in modern organizational modeling languages (Dignum et al., 2005; Hübner et al., 2007) and/or conflicting influences brought about by simultaneous membership in multiple organizations.

For example, as agent 1 is solving for its organizationally optimal policy $\pi_1^{*|\theta_1}$, in any state whose location is its initial position, θ_1 specifies the $\hat{\Delta}_1$ shown in Figure 5.2. Using $\hat{\Delta}_1$, agent 1 will modify its model to only permit consideration of the West, East, and North movement actions in its currently considered state.

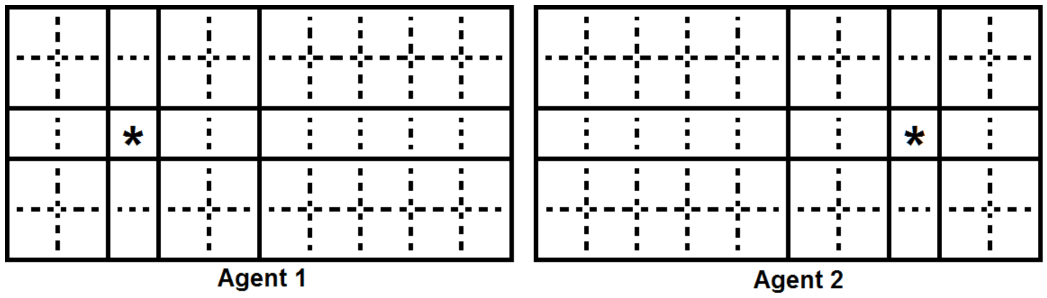


Figure 5.3: Position clusters for the PCluster based abstractions (dashed lines represent cell boundaries for reference). $*$ indicates the agent's initial location.

5.4 Influence Abstraction Effects

The research community has developed an extensive library of abstraction techniques such as: state abstraction (Li et al., 2006) and finite controllers (Bernstein et al., 2005; Poupart & Boutilier, 2003) for decision-theoretic problems; influence (Witwicki, 2010) and coordination locale (Varakantham et al., 2009) abstractions for efficient coordination; hierarchical planning (Amato et al., 2014; Sutton et al., 1999) and task networks (Sardina et al., 2006) for sequential reasoning; and the various abstract modeling constructs for an organizational modeling language (Dignum et al., 2005; Hübner et al., 2007), among many others. Two overarching commonalities within these techniques, however, are to approach abstraction as: 1) overlooking unimportant or irrelevant information and/or 2) clustering similar information together. Using these as a basis for designing abstractions over various points along abstraction dimensions, I crafted several families of abstractions for the firefighting domain that are summarized in Table 5.1. Broadly speaking, organizations created from these abstractions map a set of state factors (for specific mappings see Table 5.1) to the $\hat{\Delta}_i$ for that state, where $\hat{\Delta}_i$ informs agent i of which actions it should consider for that state. Together, these $\hat{\Delta}_i$s essentially construct regions of responsibility for each agent, which can vary over time if system time contributes to the abstraction mechanism.

These experiments utilized the firefighting domain, where there are: 2 agents; 4 fires uniformly distributed throughout the grid (without replacement) and with initial intensity i.i.d. uniformly sampled from $\{1, 2, 3\}$; cell delays i.i.d. uniformly sampled from $[0, 1]$; and a time horizon of 12. Additionally, to provide finite maximum durations before an agent could impact another, I introduce a cap on the maximum number of consecutive failed movement attempts before success is ensured (two in these experiments). I maintain the Markov property by adding a new state factor to maintain the number of consecutive failed moves.

Abstraction Name	**Expected Organizational** **Inclusivity**	**Variance** $(10^{-3}\mathbb{R}_{Op}, 10^{3}\mathbb{C}_{Op})$	**Description of Θs Constructed from Abstraction**
None	1	(2.50, 1297)	Every state maps to the same $\hat{\Delta}_i$.
Time	$\frac{1}{12}$	(35.2, 17.26)	System time in a state maps to $\hat{\Delta}_i$.
Pos	$\frac{1}{50}$	(2.08, 0.99)	Agent's position in a state maps to $\hat{\Delta}_i$.
TCluster	$\frac{1}{4}$	(34.5, 204)	Like **Time**, but system time is clustered into intervals. For example, states with times in $[1, 3]$ map to the same $\hat{\Delta}_i$.
PCluster	$\frac{1}{12}$	(3.00, 15.8)	Like **Pos**, but positions are clustered into neighborhoods as illustrated in Figure 5.3.
Time + Pos	$\frac{1}{600}$	(18.1, 0.02)	System time and agent's position in a state, together, map to $\hat{\Delta}_i$.
TCluster + Pos	$\frac{1}{200}$	(22.0, 0.22)	Clustered system time and agent's position in a state, together, map to $\hat{\Delta}_i$.
Time + PCluster	$\frac{1}{144}$	(28.1, 0.27)	System time and clustered agent positions, together, map to $\hat{\Delta}_i$.
TCluster + PCluster	$\frac{1}{48}$	(37.3, 2.84)	Clustered system time and clustered agent positions, together, map to $\hat{\Delta}_i$.

Table 5.1: Descriptions of the organizational abstractions I evaluate in Section 5.4. Uniformity for each abstraction relies on the specific Pareto characterization, but typically decreases as agent computation becomes less costly.

To observe the impact of abstract influences with respect to $\mathbb{P}_{Op}$'s Pareto topology (i.e., as agents have more or less available computational resources), I encoded the same Pareto topology as in Section 4.3.3, $\mathbb{P}_{Op}(\Theta) = \mathbb{R}_{Op}(\Theta) - \frac{1}{b}\mathbb{C}_{Op}(\Theta)$ for parameter $b \in \mathbb{Z}^+$, and for each abstraction had the ODP create organizations for different values of b. High b values represent when the agents have abundant computational resources relative to the pace of the environment, and *vice versa* for low values.

Additionally, to observe the impact of abstract influences with respect the amount of information the ODP possesses, I had the ODP create organizations from three different available information profiles. I controlled the amount of information available to the ODP by artificially manipulating the problem samples from which it constructed estimates of the $\hat{\Delta}_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ and $\hat{\Delta}_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ terms used in Algorithm 4.2. At one extreme, the ODP exactly sampled the evaluation problem set, which encodes that the ODP has perfect information. Then, as the ODP based its estimates off of fewer sample problems (i.e., the training set is a diminishing subset of the test problems), the ODP possessed increasingly imperfect domain information.

As in Section 4.3.3, I evaluated each organization on 300 problem instances to empirically compute $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ for each Θ, from which I calculated $\mathbb{P}_{Op}$ for various Pareto optimality parameterizations. Recall the motivating example from Section 5.1 where the influences to prohibit reverse movements were problematic in the four-fire domain. Figure 5.4 illustrates influences that the ODP specified using the TCluster+Pos abstraction. Notice how as time progresses, the actions that each agent can consider changes. Early on, each agent is forced into one-way paths, which streamlines their computation without typically precluding optimal actions. Then as time progresses, each agent is allowed to consider most movements in the locations near its initial location (in case it was able to quickly fight a fire and is now pursuing its second fire), but still is forced into one-way paths further out (in case it is still pursuing its first fire). Continuing to the next time cluster, the agent is afforded most movement actions on its side of the grid (in case there are still fires in this region), but is forced into one-way paths on the side of the grid opposite its initial location (in case there are no more fires on its initial side of the grid and it helping the other agent). Finally, at late times, the agents' movements are heavily restricted because they usually have fires extinguished by this time.

Figure 5.5 shows the $\mathbb{P}_{Op}$ curves for each organization. Figures 5.6, 5.7, and 5.8 show the separate $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for the organizations constructed from perfect information, 2/3 information, and 1/3 information respectively. To allow the reader to more easily visualize how the $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves change in response to the amount

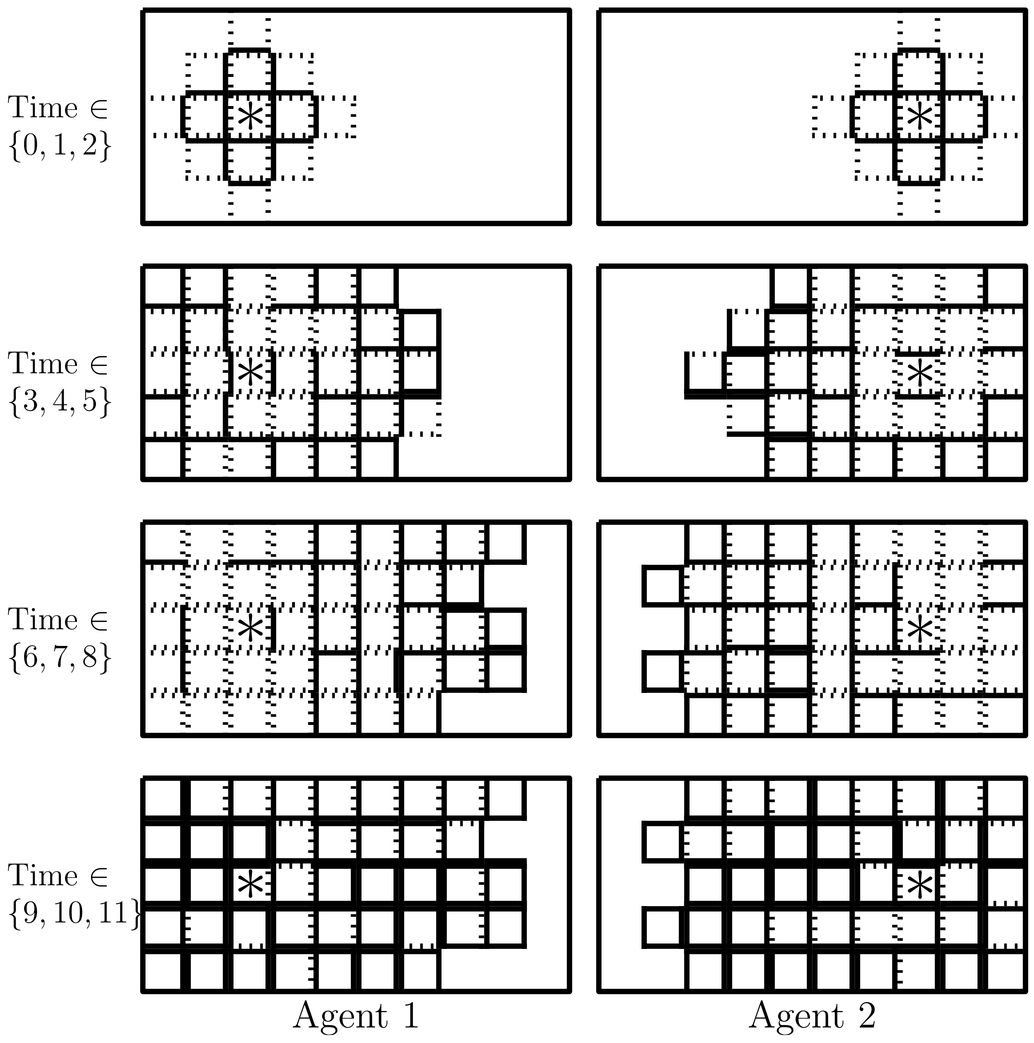
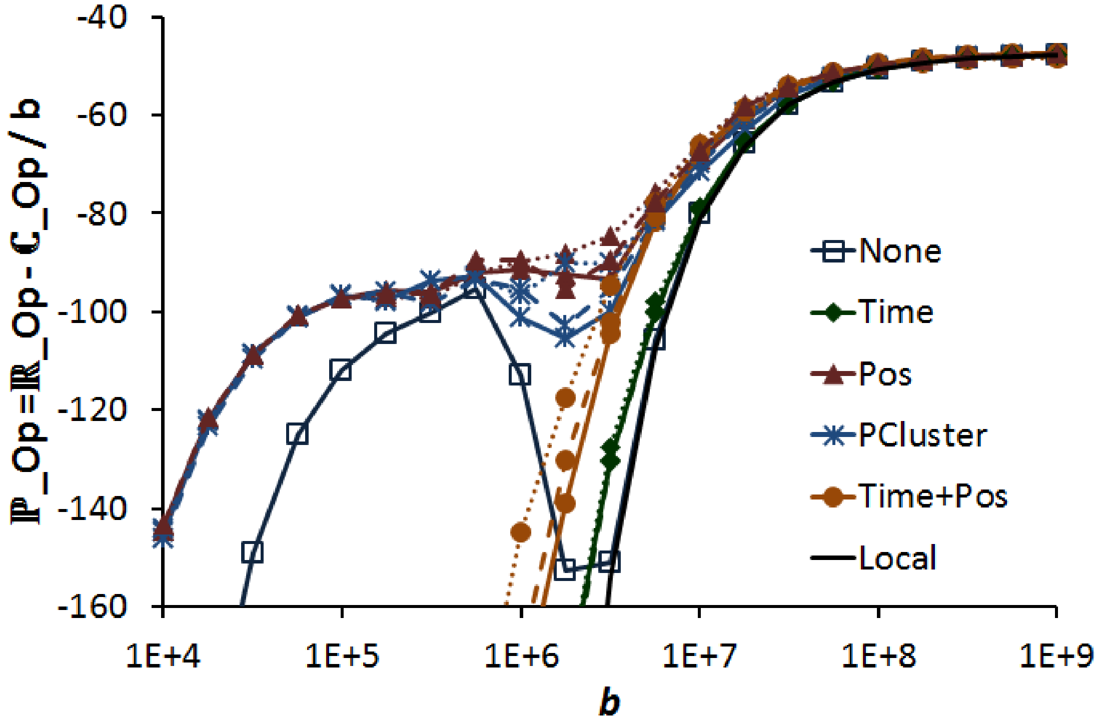


Figure 5.4: Movement action influences in the four-fire domain for each agent using the TCluster+Pos abstraction. An agent can move into a cell in a direction where it first passes a dotted line, but not a solid line. Agents begin in the cells indicated with $*$.

of information the ODP possessed, Figure 5.9 shows the $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for select organizations that are particularly susceptible to information scope effects in Figure 5.5. In the next sections, I systematically analyze these results to develop a framework for characterizing how abstraction dimensions impact: the operational performance of an organization (Section 5.4.1), the sensitivity of the ODP's search algorithm (Section 5.4.2), and the effects of information availability on the ODP (Section 5.4.3).



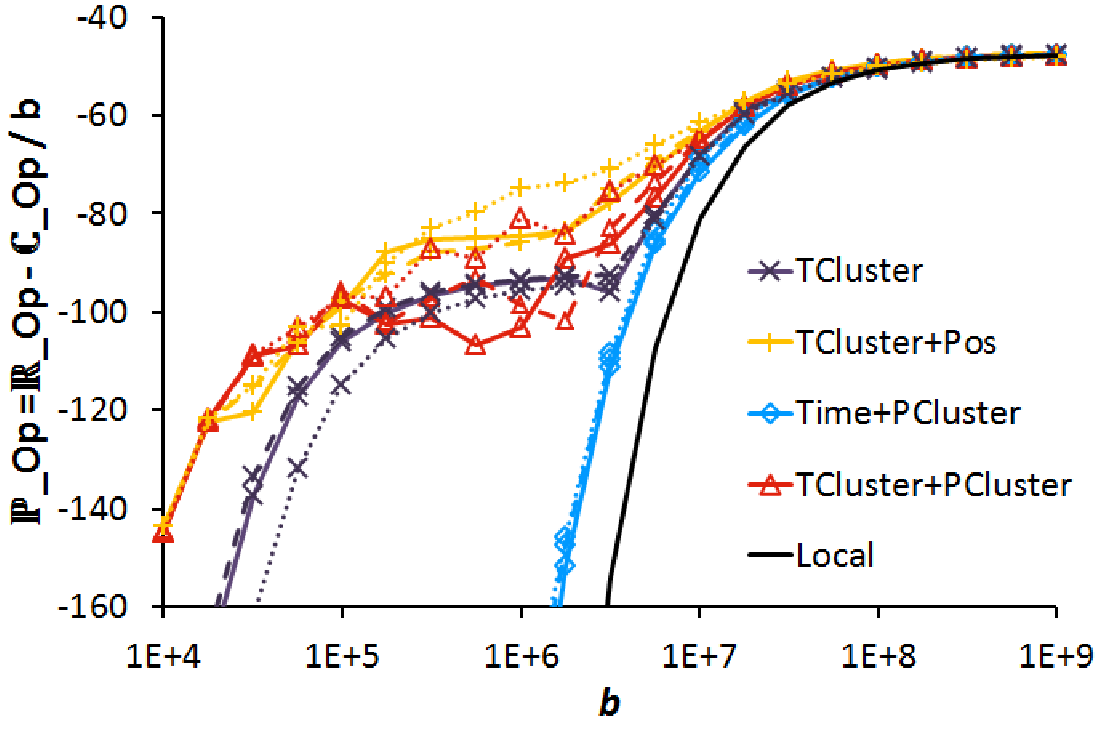


Figure 5.5: $\mathbb{P}_{Op}$ curves for each abstraction for different amounts of ODP information. Solid, dashed, and dotted curves correspond to organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.

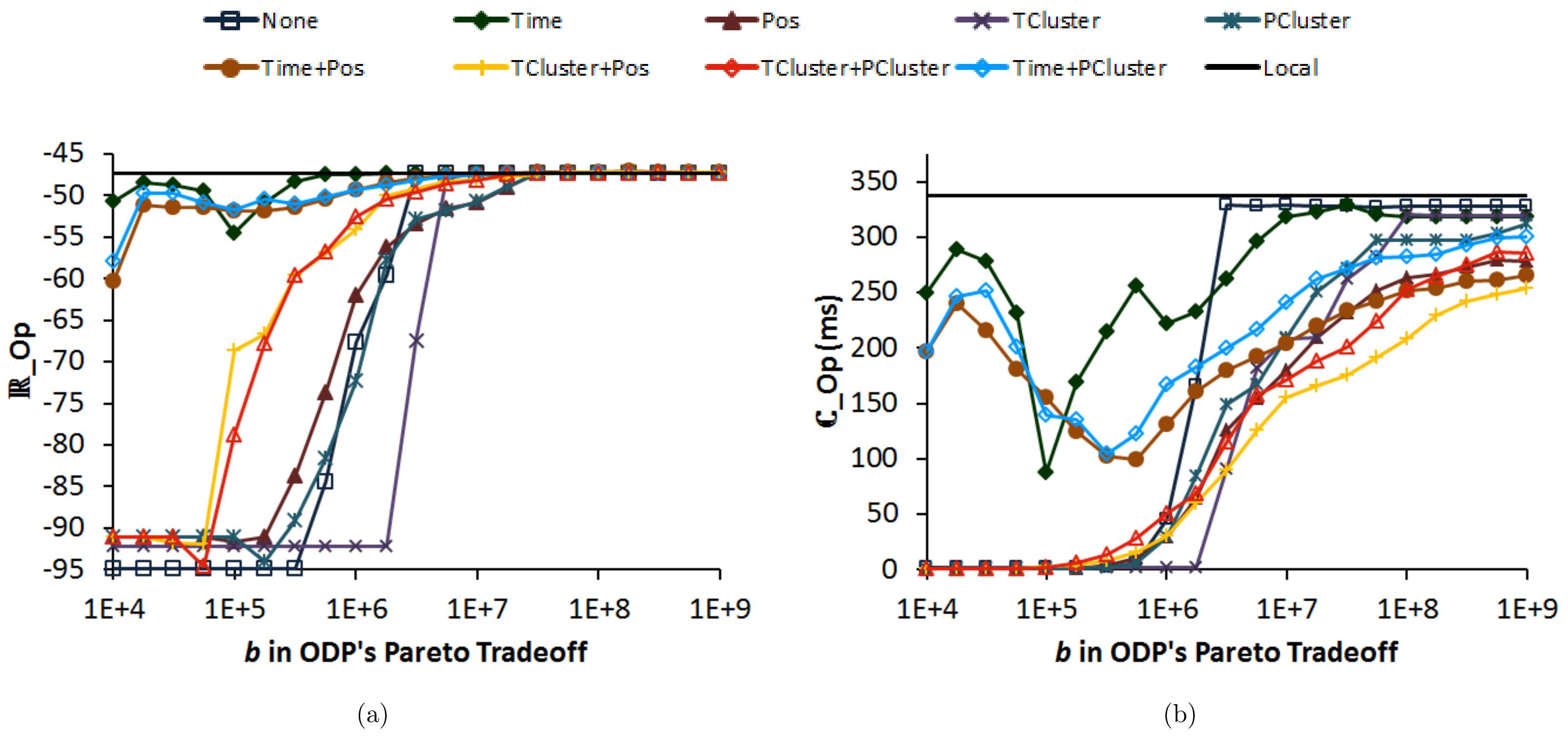
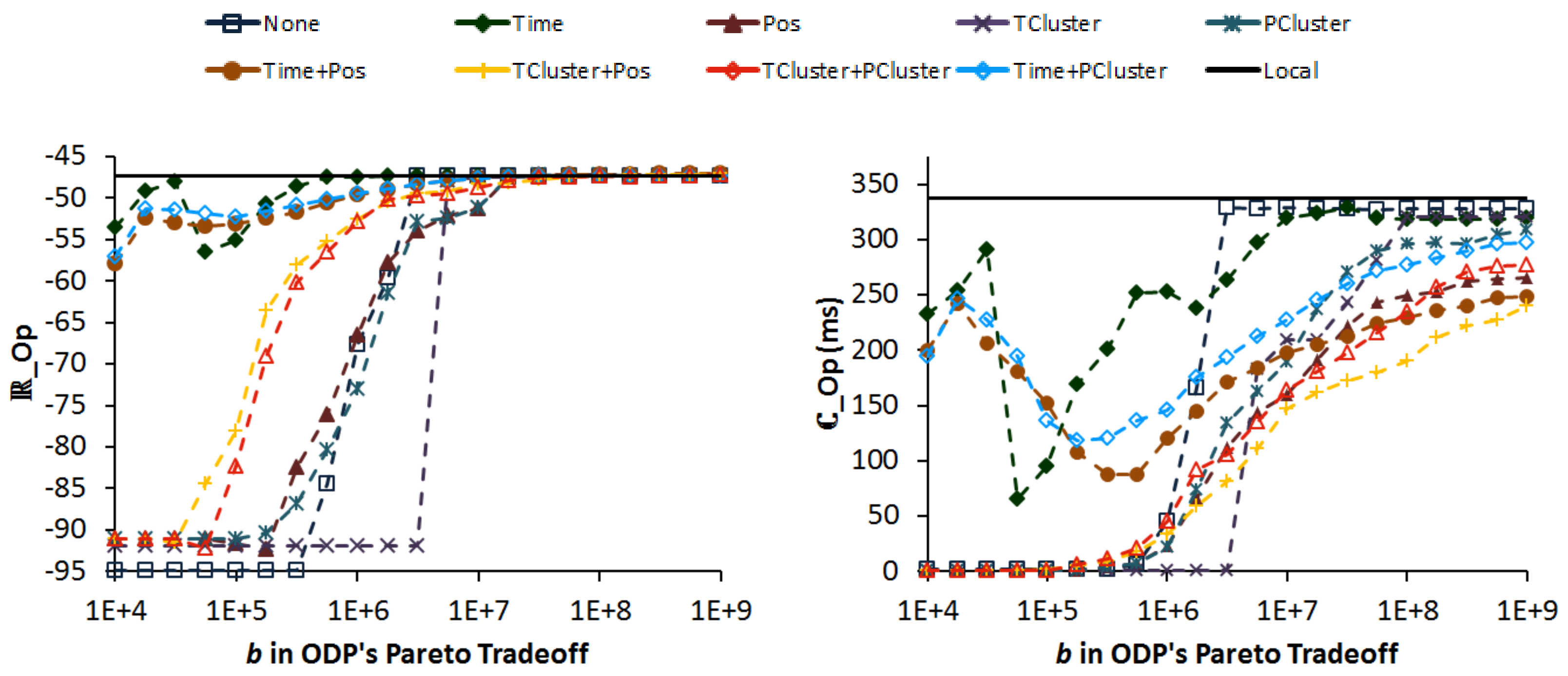


Figure 5.6: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from perfect information.



(a) (b)

Figure 5.7: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from 2/3 information.

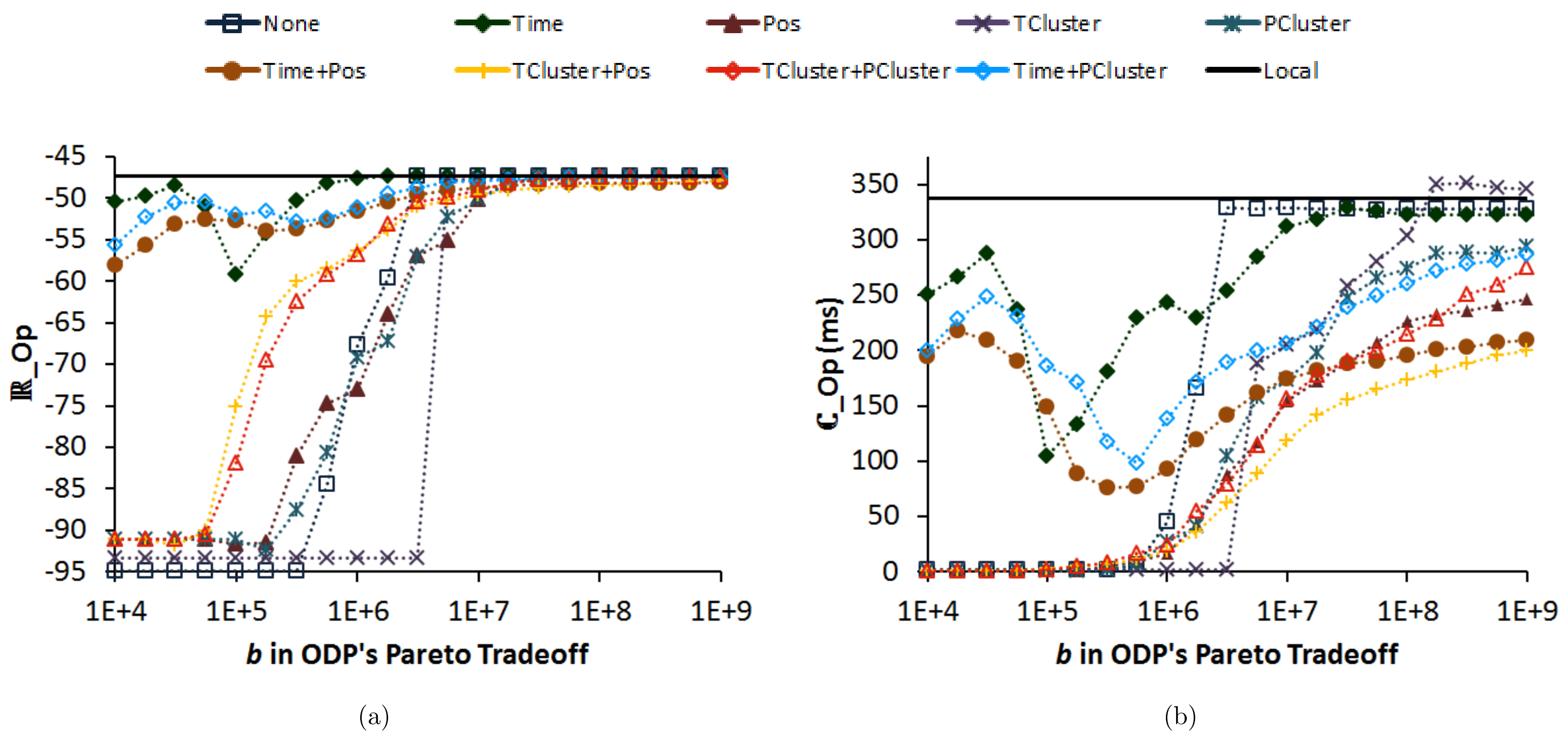
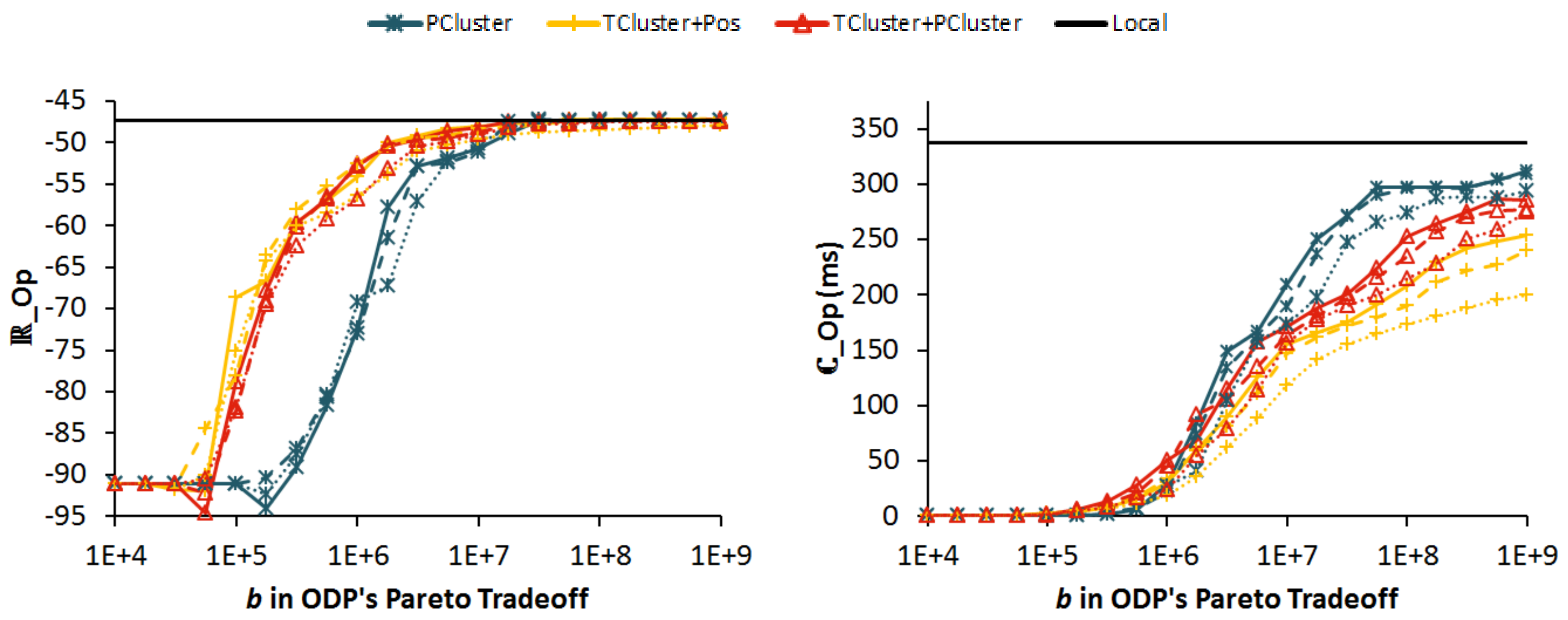


Figure 5.8: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from 1/3 information.



(a) (b)

Figure 5.9: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for select organizations to permit direct visualization of how information scope affects the ODP. Solid, dashed, and dotted curves correspond to organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.

5.4.1 Operational Performance

The $\mathbb{P}_{Op}$ curves in Figure 5.5 reveal two differentiating characteristics for operational performance across abstractions: a curve's smoothness (which will be discussed in Section 5.4.2) and a curve's raw quality (i.e., its vertical placement on the graph). Analysis reveals that an organization's uniformity is strongly correlated with performance quality. One exemplary case of this is the None abstraction, where the organizations at low b (i.e., when computation is expensive) restrict all but a single action from consideration (i.e., the same action must be taken in every state). These Θs by definition have maximal uniformity and also obtain relatively high $\mathbb{P}_{Op}$ as compared to other abstractions with similar inclusivity (e.g., Time). Then, as b increases, the None organizations permit consideration of additional actions, eventually decreasing uniformity to its minimal value, and these minimal-uniformity Θs obtain the worst $\mathbb{P}_{Op}$.

Considering this observation more deeply, decreased uniformity arises when alternative actions could be optimal for specific instances entailed in $\hat{\Delta}_i$'s domain, for example like in Figure 5.2. Rather than restrict the agent from considering some subset of Pareto-valuable actions that the ODP knows the agent might need in a specific problem instance, the ODP instead permits consideration of all actions that could be Pareto-valuable, and relies on the agent's local intelligence to appropriately select from among this set. As a result, the ODP under-constrains the agents' reasoning, which increases $\mathbb{C}_{Op}$ and thus decreases $\mathbb{P}_{Op}$. However, for high-uniformity influences, the ODP can aggressively restrict the agents' actions to a limited set of Pareto-valuable actions.

$\mathbb{P}_{Op}$'s reliance on inclusivity is interesting in that excess inclusivity can yield poor $\mathbb{P}_{Op}$ (e.g., the None and Time abstractions), and too little inclusivity can also result in poor $\mathbb{P}_{Op}$ (e.g., the Time+Pos abstraction). However, abstractions with moderate inclusivity are associated with the maximal $\mathbb{P}_{Op}$ curves. It is straightforward to show that excess inclusivity increases susceptibility to the effects of decreased uniformity; that is, additional states map into the same $\hat{\Delta}_i$ but may not have the same optimal actions. Too little inclusivity is detrimental for the opposite reason; that is, since a low-inclusivity $\hat{\Delta}_i$ applies to such a narrow space, there is insufficient diversity in the ODP's statistical estimates to generalize to unseen problem instances. Consequently, the agents can encounter states that do not map to any of the specified influences, in which case the agents default to their local, baseline reasoning and behaviors.

Examining the $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves in Figures 5.6, 5.7, and 5.8 illustrates that, broadly speaking, as the agents' computation becomes relatively cheaper (b increases), the ODP induces the agents to consider more actions, which in turn provides operational flexibility for the agents to achieve higher $\mathbb{R}_{Op}$. The Time and Time+Pos curves deviate from this pattern due to low uniformity across all of the Pareto conditions; that is, the specific action an agent should take is poorly correlated with system time, meaning that time-based abstractions cluster together different actions. This biases the ODP into permitting the agents to consider additional actions even when Pareto conditions discourage excessive agent reasoning.

5.4.2 ODP's Search Sensitivity

A striking feature of Figure 5.5 is the large dip in $\mathbb{P}_{Op}$ for some of the abstractions as computation becomes less costly (e.g., in the None abstraction), when we would normally expect smooth, monotonically increasing $\mathbb{P}_{Op}$ curves. In each of these cases, the pre-dip organization would actually be a preferable Θ to the one created by the ODP for these Pareto conditions, which implies that the ODP's greedy search algorithm performed poorly in these instances.

Notice that these dipping cases occur most significantly in the abstractions with high variance. Recalling Definition 5.4, high variance corresponds to $\hat{\Delta}_i$s composed of Δ_is with significantly different expected values for $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$, which in turn makes the incremental impact of $\hat{\Delta}_i$ have high statistical variance. Utilizing such imprecise estimates for a $\hat{\Delta}_i$'s incremental impact naturally makes the greedy search algorithm sensitive to initial conditions and small data errors introduced from the ODP's sampling process, which results in the unexpected $\mathbb{P}_{Op}$ performance dips. In other words, the ODP believes from its data that certain influences will improve $\mathbb{P}_{Op}$, when in reality, the ODP is overestimating the reward and/or underestimating the computational costs of adding the influences to the design, and experimentation in the evaluation domain ultimately reveals that these influences are actually detrimental to $\mathbb{P}_{Op}$.

5.4.3 ODP's Information Scope

Examining Figure 5.5, we unexpectedly observe that organizations created from less information tend to achieve higher $\mathbb{P}_{Op}$, whereas intuition would dictate that additional information should tend to improve the quality of an organizational design. Further analysis reveals that an abstraction's inclusivity is a primary determining factor for analyzing an ODP with respect to its available information. As the ODP receives additional information, it is being exposed to increasingly-unusual problem instances, analogously to how the cumulative probability of drawing a value three standard deviations from the mean of a Gaussian increases as you draw more samples. For abstractions with high inclusivity, the optimal actions from these unusual problem instances inevitably get clustered together with the more common optimal actions, essentially creating multimodal distributions for the $\hat{\Delta}_i$s' statistical estimates. Since my ODP computes a $\hat{\Delta}_i$'s incremental impact as the mean of its constituent Δ_is' incremental impacts, multimodal distributions fundamentally violate the assumptions of the ODP's statistical representation, and result in decreased $\mathbb{P}_{Op}$ from the resulting organizations. This argues that an ODP should employ more sophisticated statistical models for representing estimates of high inclusivity influences. An interesting direction to consider for future work would be an automated approach for correcting the ODP's statistical representation in response to influence inclusivity (see Section 6.2.4 for some initial thoughts in this direction).

Figure 5.5 also demonstrates that some abstractions are more robust to the effects of information availability than others. For example, the None abstraction is completely immune to these effects and the TCluster abstraction is also exceptionally resilient in this regard. While these abstractions have high inclusivity and thus should be susceptible to information availability effects, they also have extremely low uniformity. Thus, incorporating specialized Δ_is into a $\hat{\Delta}_i$ cannot induce the ODP to permit additional actions, because those actions are already permitted due to the common cases.

5.5 Task-Delineated Abstractions

To summarize the key findings from my analysis in the previous section, the best abstractions are ones that:

- **Have moderate inclusivity**. This provides enough leeway for an ODP to specify nuanced influences where appropriate but is broad enough to permit influences to generalize to the larger problem space.

- **Have high uniformity**. This allows the ODP to more aggressively restrict the agents' local models to a smaller set of actions for consideration, which streamlines computational effort thereby improving performance.

- **Have low variance**. Low variance reduces sensitivity in the ODP's search algorithm, resulting in smoother performance curves that better match the Pareto topology.

These observations lead to a high-level strategy of adopting abstractions that segment each problem instance into maximally-sized components that agree on the same action with (nearly) the same expected incremental computational and reward impacts. Although such a strategy is not computationally practical as it would involve searching through the space of clusterings, it does suggest a heuristic proxy, which is to group together states/actions that are collectively pursuing the same outcome. I refer to this heuristic clustering strategy as a **task-delineated abstraction**.

For example, in the fire-fighting domain, a task-delineated abstraction would imply segmenting a problem episode into tasks associated with putting out a specific fire. See Table 5.2 for the task-delineated abstractions I provided to the ODP for the firefighting domain, where the number of active fires serves as an indicator variable for which task an agent should currently be pursuing. It is important to note that while the FCount+Pos and FCountLocal+Pos abstractions have identical inclusivity and very similar variance, the FCount+Pos abstraction has much higher uniformity, and thus my framework predicts is the more desirable abstraction. The FCountLocal+Pos abstraction's uniformity is lower because the ODP's information is unevenly distributed across the abstraction space. For example, there are relatively few states where there are four fires on one half of the grid, causing the ODP's information to be concentrated in the portion of the abstraction space corresponding to 0-2 active fires in the agent's half of the grid, and subsequently decreasing uniformity. In contrast, the FCount+Pos abstraction more evenly distributes the ODP's information, resulting in higher uniformity.

Broadly, the task-delineated heuristic has theoretical foundations suggesting that it leads to abstract influences with moderate inclusivity, high uniformity, and low variance. Inclusivity is moderate because the abstraction allows the ODP to restrict the task-level behaviors of the agents while still allowing information to generalize within the scope of a single task (i.e., provides leeway for agents to use their local expertise to most effectively complete their organizationally designated tasks, Section 4.1). Uniformity is high and variance is low if appropriate tasks are identified that cluster similar actions with similar incremental impacts.

Figure 5.10 shows the $\mathbb{P}_{Op}$ curves of these abstractions using the same evaluation methodology as in Section 5.4, along with the local baseline and the best abstraction from Section 5.4 (TCluster+Pos). $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for these abstractions are given in Figure 5.11. Observe that the heuristically-recommended task-delineated abstraction (FCount+Pos) achieves essentially the same $\mathbb{P}_{Op}$ quality as the TCluster+Pos abstraction, which is unsurprising given that the clustered system times essentially

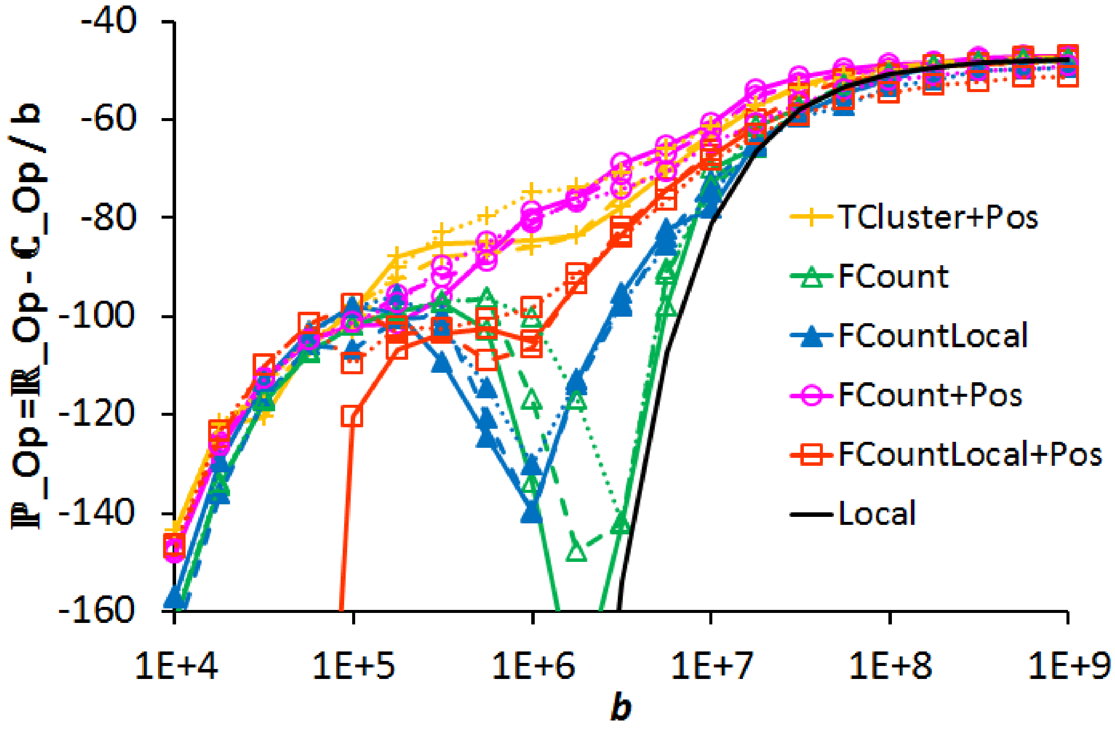
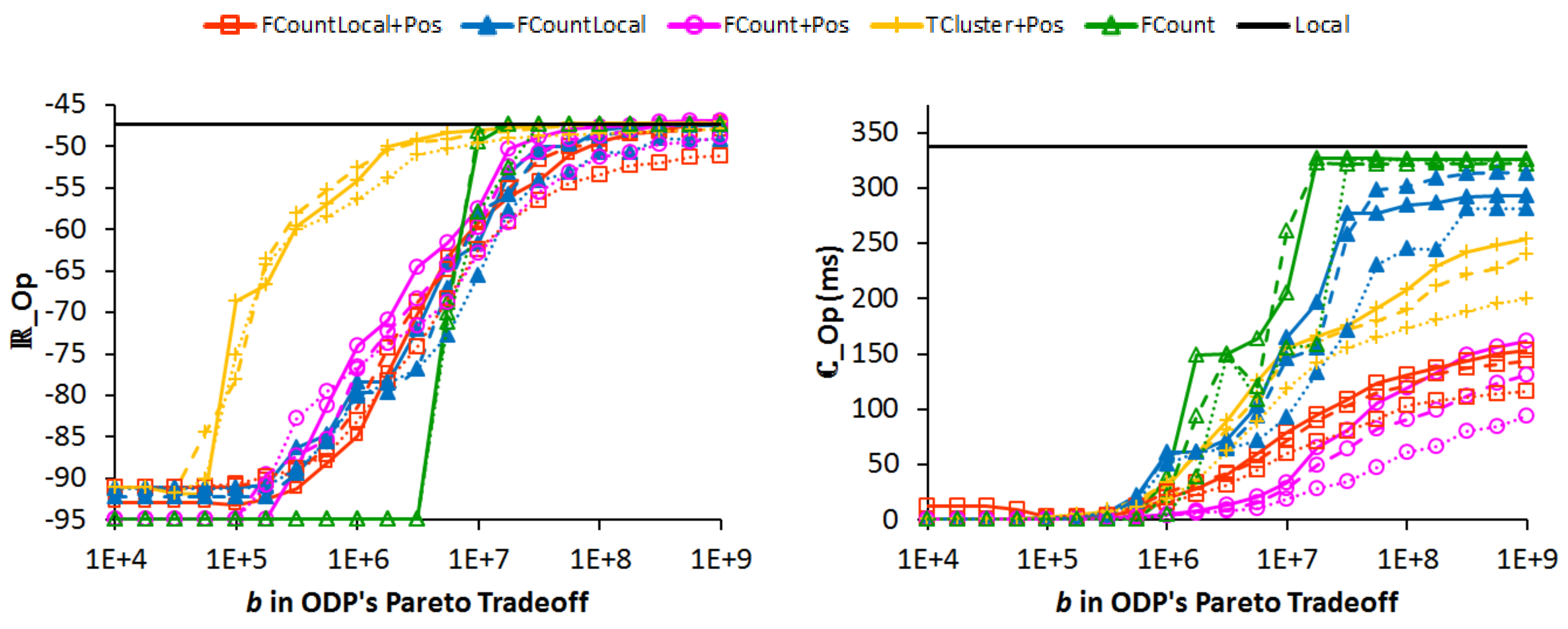


Figure 5.10: $\mathbb{P}_{Op}$ curves for task-delineated abstractions alongside bounding abstractions (TCluster+Pos and local baseline). Solid, dashed, and dotted curves correspond to organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.

Abstraction	**Expected Organizational**		
Name	**Inclusivity**	**Variance** $(10^{-3}\mathbb{R}_{Op}, 10^{3}\mathbb{C}_{Op})$	**Description of Θs Constructed from Abstraction**
FCount	$\frac{1}{4}$	(3.48, 120)	Number of active fires in a state maps to $\hat{\Delta}_i$.
FCountLocal	$\frac{1}{4}$	(2.26, 274)	Number of active fires on agent's half of grid in a state maps to $\hat{\Delta}_i$.
FCount + Pos	$\frac{1}{200}$	(3.46, 0.11)	Number of active fires and agent's position in a state, together, map to $\hat{\Delta}_i$.
FCountLocal + Pos	$\frac{1}{200}$	(2.76, 0.18)	Number of active fires on agent's half of grid and agent's position in a state, together, map to $\hat{\Delta}_i$.

Table 5.2: Descriptions of task-delineated abstractions. Uniformity for each abstraction relies on the specific Pareto characterization, but typically decreases as agent computation becomes less costly. For completeness, I included the FCount and FCountLocal abstractions despite my framework predicting that they have poor performance characteristics. I did not construct PCluster variants because PCluster and Pos were qualitatively identical in the experiments in Section 5.4.



(a) (b)

Figure 5.11: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for task-delineated abstractions alongside bounding abstractions (TCluster+Pos and local baseline). Solid, dashed, and dotted curves correspond to organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.

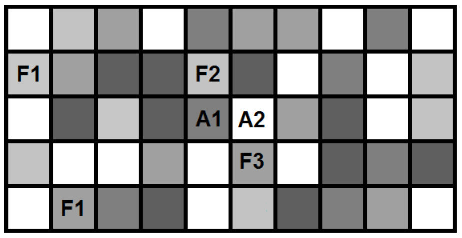


Figure 5.12: An example initial state with four fires and agents located initially adjacent to each other. Darker cell shading indicates higher cell delays. Ai indicates agent i's location, and Fx indicates a fire with intensity x.

proxy for task-delineation. Notice, however, that because FCount+Pos has lower variance, it is less sensitive to information availability effects (i.e., its three $\mathbb{P}_{Op}$ curves for information quantities are nearly identical), and also exhibits fewer ODP search sensitivities (i.e., the $\mathbb{P}_{Op}$ curves are smoother and monotonically increase over the Pareto topology). As my framework predicts, the other abstractions' high variance, low uniformity, and/or high inclusivity make them suboptimal, which provides additional evidence for the framework's effectiveness for analyzing influence abstraction mechanisms.

Finally, it is worth recognizing that, in retrospect, my previous position-based abstraction (Section 4.3.3) is essentially task-delineated for the version of the firefighting domain with only two fires, and as my framework predicts, performed well. That is, since each agent was expected to fight a single fire, there is a single task for each agent.

5.6 Evaluation With Initially Adjacent Agents

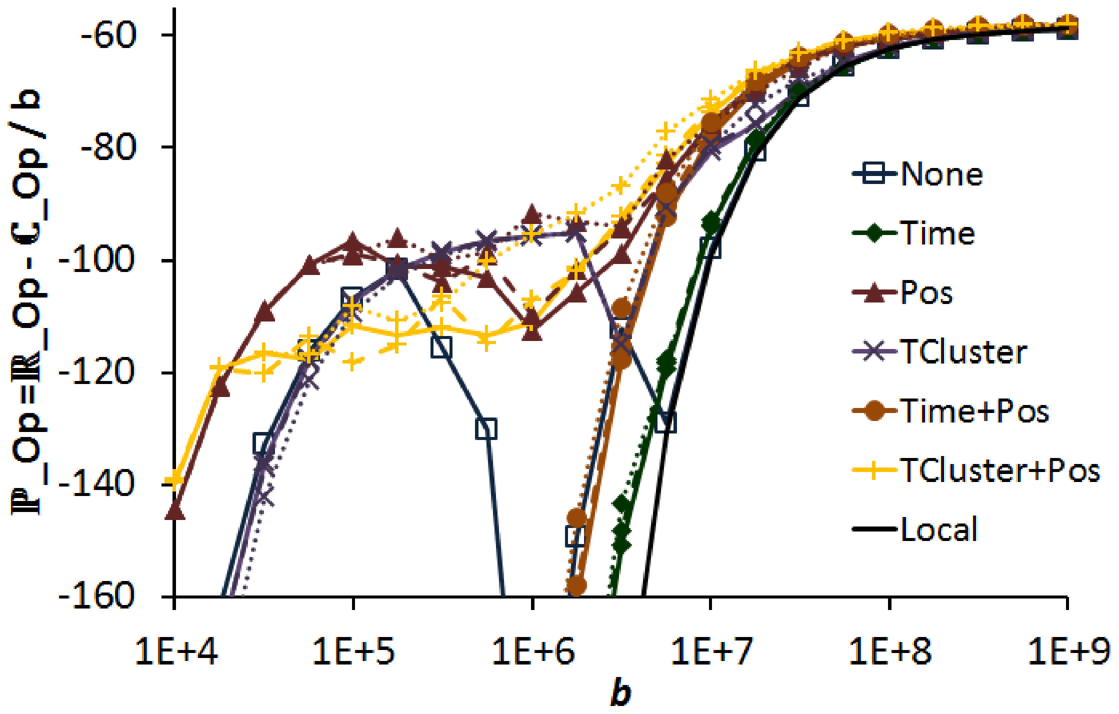
To provide additional empirical evidence of the effectiveness of my abstract influence techniques, in this section, I present results from a set of experiments where the agents begin adjacent to each other in the center of the grid (those in Figure 5.12). As described in Section 3.4 and demonstrated in Section 3.5.3, the initial configuration of state can have significant impact on the effectiveness of the local baseline, since a problem episode may have more or less opportunities for an organization to beneficially influence the MAS. As such, the experiments in this section present a qualitatively-distinct environment from the previous experiments in this chapter.

The experiments in this section follow the same parameterization and methodology as in Section 5.4, except for the agents' initial locations. Additionally, I did not construct PCluster variants because PCluster and Pos were qualitatively identical in the experiments in Section 5.4. Figure 5.13 shows the $\mathbb{P}_{Op}$ curves for the organizations in these experiments. Figures 5.14, 5.15, and 5.16 show the separate $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for the organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.

Firstly, notice that, because there are greater opportunities for an organization to improve the agents' coordination (i.e., by differentiating agents' task-level behaviors), many of the organizations are able to achieve higher $\mathbb{R}_{Op}$ than the baseline MAS in these experiments (only when computation is inexpensive enough), while still having lower $\mathbb{C}_{Op}$. This result reiterates the observations about baseline performance from Sections 3.4 and 3.5.3, and highlights the importance of evaluating an ODP technique in both domains where the baseline is effective (e.g., there are few opportunities for additional organization to improve coordination) and also where the baseline is ineffective (e.g., additional organization is required to differentiate agents' task-level behaviors).

Secondly, my framework for analyzing the effectiveness of an influence abstraction mechanism (Section 5.4) correctly predicts organizational performance. High-uniformity, moderate-inclusivity, low-variance abstraction mechanisms result in high-performing organizations that smoothly and robustly follow the Pareto topology for all of the ODP-information profiles.

Finally, observe that my task-delineated abstraction (FCount+Pos) remains a sound choice, and achieves the best $\mathbb{P}_{Op}$ for the majority of the Pareto space (and is nearly as good as the best otherwise), while smoothly and robustly following the Pareto topology. A notable exception to FCount+Pos's dominance of the other abstractions is that the FCountLocal+Pos is a superior choice when the agents' computational costs are inexpensive (because it has higher uniformity for these Pareto conditions). For the FCount+Pos abstraction, there are a more actions that are only Pareto-valuable when computation is exceptionally inexpensive (e.g., as indicated by the distinct inflection point for the FCount+Pos curve in Figure 5.14b), which inevitably get distributed across the abstraction space, resulting in decreased uniformity. However, the FCountLocal+Pos abstraction does not exhibit this property (e.g., has a smoother derivative in Figure 5.14b), meaning that the set of Pareto-valuable actions is smaller, and already included in the organization (resulting in higher uniformity). This occurs because the state space is unevenly distributed across the FCountLocal+Pos abstraction (i.e., there are few states with four fires on one side of the grid). While this unevenness was detrimental in Section 5.5 when the agents began spread out throughout the grid (since the fires agent i should fight could be in any direction from



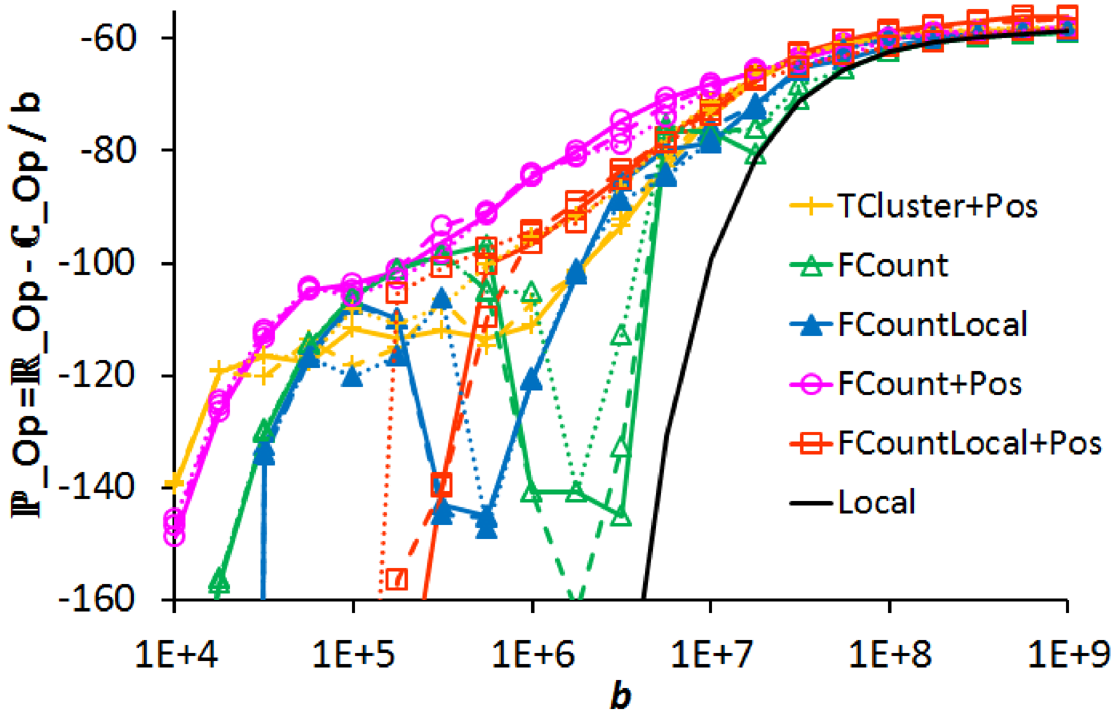
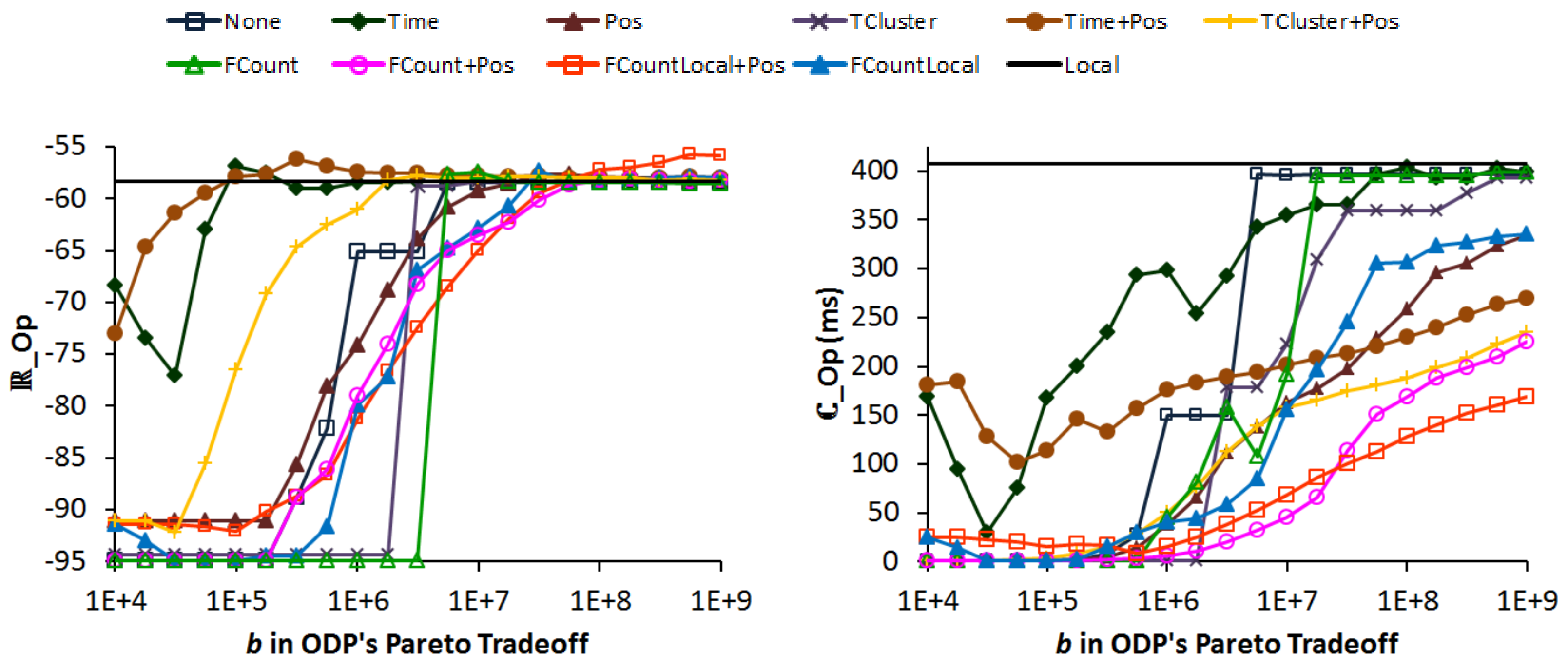
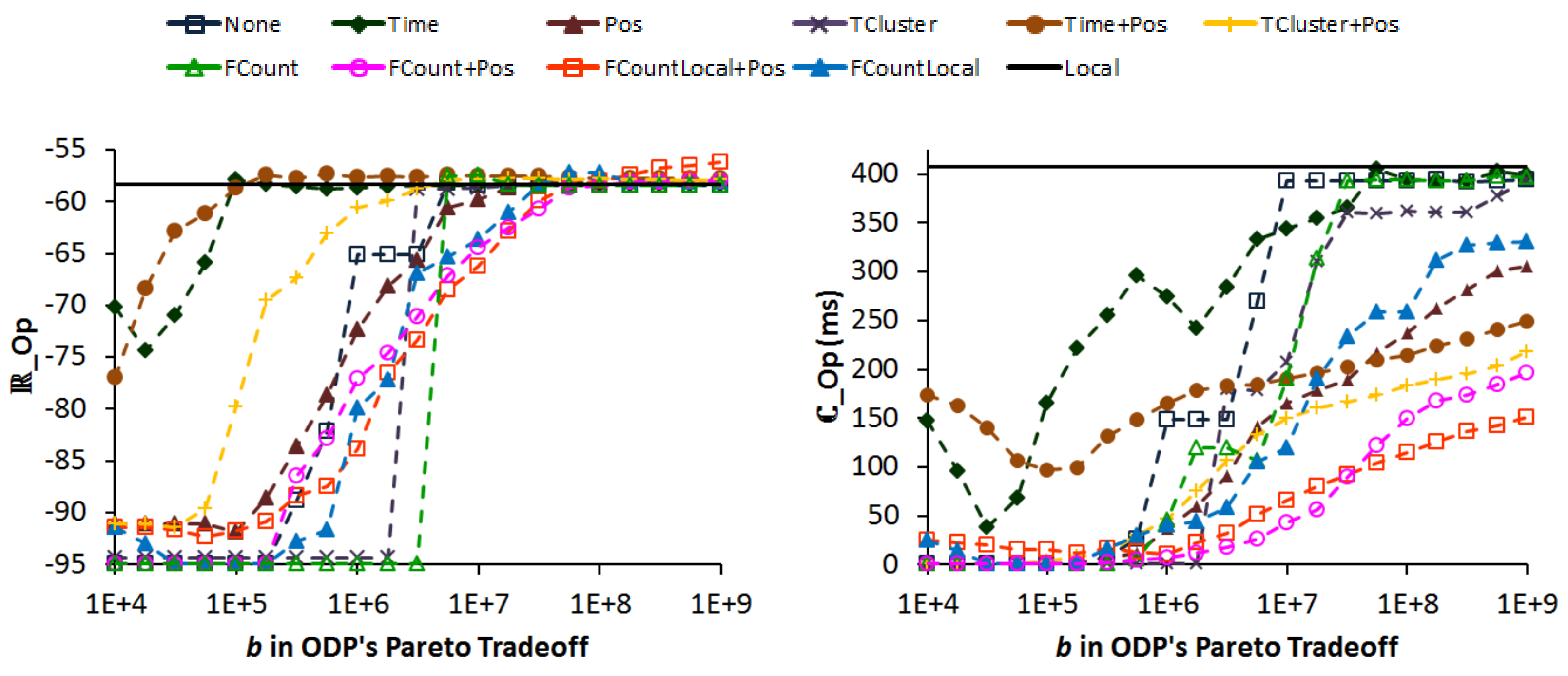


Figure 5.13: $\mathbb{P}_{Op}$ curves for each abstraction for different amounts of ODP information. Solid, dashed, and dotted curves correspond to organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.



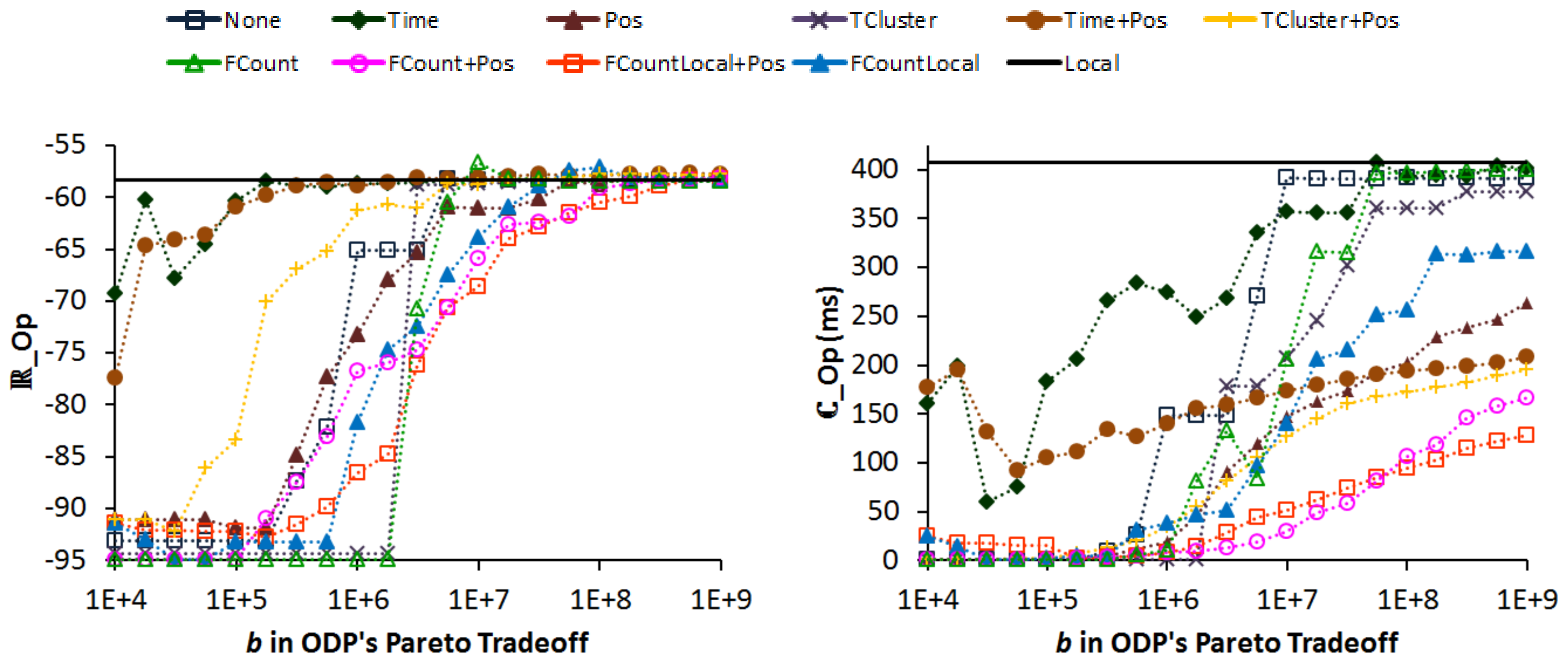
(a) (b)

Figure 5.14: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from perfect information.



(a) (b)

Figure 5.15: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from 2/3 information.



(a) (b)

Figure 5.16: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from 1/3 information.

the agent), when agents begin in the center of the grid, the fires agent i should fight are typically in a single direction.

5.7 Evaluation with Additional Agents

My experiments in this section utilize a version of the firefighting domain with: a 10×6 grid; 4 agents who begin at the locations in Figure 5.17; 8 fires, uniformly distributed throughout the grid (without replacement), and initial intensity i.i.d. uniformly randomly selected from $\{1, 2, 3\}$; no cell delays (i.e., movement actions are deterministic); and a time horizon of 8.

To compute optimal joint policies for sampled problem episodes as part of computing a quantitative description of organizational patterns (Section 4.2.2), my ODP performed an additional layer of task abstraction using the options framework. Specifically, to permit the ODP to compute optimal joint policies, it constructed options that each correspond to a pair of agents (agents 1 and 2 were grouped together, as were agents 3 and 4). The ODP constructed three options in a state for each pair of agents, where each option is associated with a pair of fires to fight, and I embedded a heuristic to inform the ODP of which pairs of fires should be translated into options. The heuristic selects three fires corresponding to the: (a) closest fire to the first agent i, (b) closest fire to the second agent j, and (c) fire with closest mean distance to the pair of agents. The three options the ODP constructs for a pair of agents i and j are then simply $\langle$Ai $\rightarrow$(a), Aj $\rightarrow$(b)$\rangle$, $\langle$Ai $\rightarrow$(c), Aj $\rightarrow$(b)$\rangle$, and $\langle$Ai $\rightarrow$(a), Aj $\rightarrow$(c)$\rangle$. To break ties between equally-distant fires, the ODP selected the fire with the largest, min-distance to another agent, which has the effect of prioritizing fires that are far from the other agents. While this heuristic is not theoretically ensured to retain the optimal joint policy, I hand-validated that it does in fact allow the ODP to compute the optimal joint policy for ten problem episodes (out of the ten that I checked), suggesting that the heuristic is at least sensible despite lacking theoretical guarantees.

For these experiments, I used the same methodology as in Section 5.4, and constructed a space of organizations over the same space of Pareto optimality parameters ($\mathbb{P}_{Op}(\Theta) = \mathbb{R}_{Op}(\Theta) - \frac{1}{b}\mathbb{C}_{Op}(\Theta)$), for each abstraction (excluding the PCluster abstractions which were qualitatively identical to the Pos abstractions in prior experiments) for the three available information profiles (300, 200, and 100 episode samples respectively). Importantly, however, note that the space of possible agent interactions has increased exponentially compared to Section 5.4, since there are additional agents and additional fires to be fought. Thus, despite the ODP sampling the same number

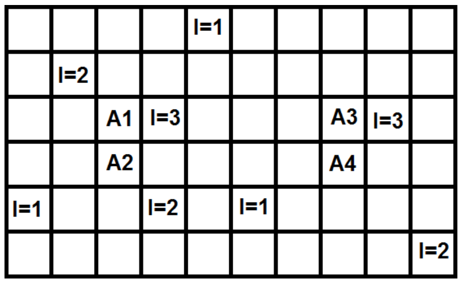


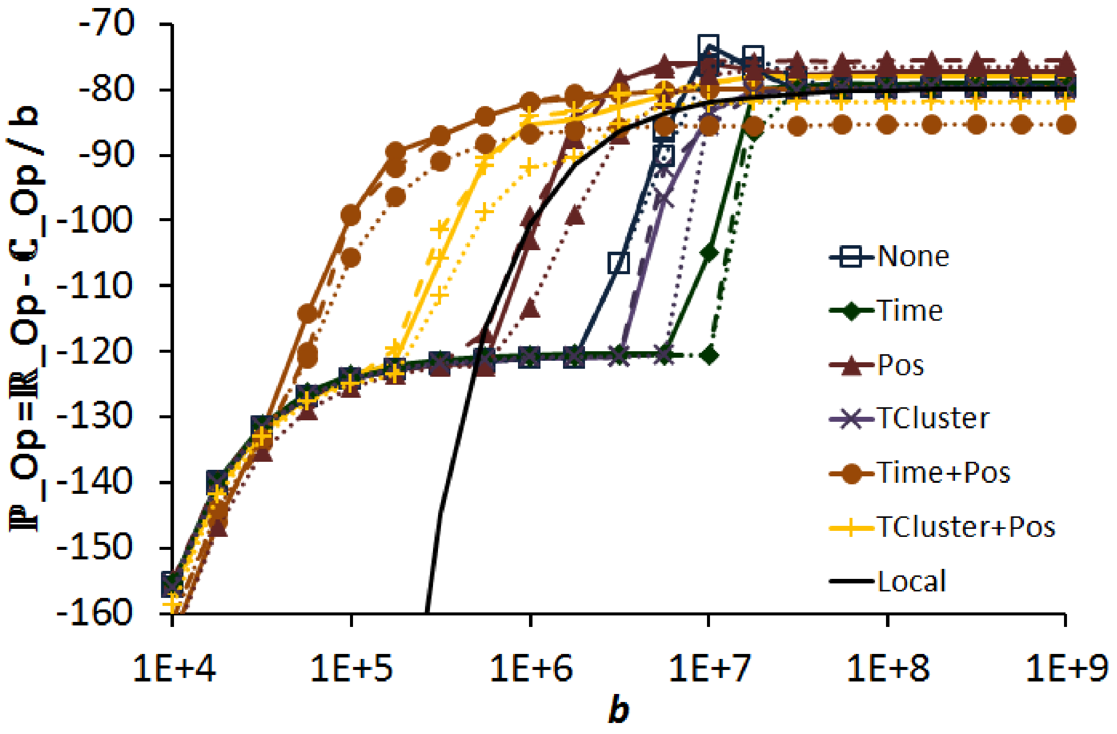
Figure 5.17: Example initial state in the four agent version of firefighting domain. Ai is the position of agent i, and $I = x$ indicates that there is a fire in that cell with intensity x.

of episodes as in those experiments, its knowledge of the MAS's interaction space is significantly diminished (from $1.6 \times 10^{-3}\%$ to $1.8 \times 10^{-9}\%$ of the initial fire configurations), to the point that its statistical estimates are no longer stable.

Figure 5.18 shows the $\mathbb{P}_{Op}$ curves for the organizations in these experiments. Figures 5.19, 5.20, and 5.21 show the separate $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for the organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.

Firstly, notice that in contrast to the experiments in Sections 5.4, 5.5, and 5.6 where additional information from more sample problems tended to degrade $\mathbb{P}_{Op}$, in the experiments here additional information almost universally results in organizations with increased $\mathbb{P}_{Op}$. This reflects the notion that despite sampling the same number of episodes, the ODP effectively has much less information in these experiments, and as it obtains additional information from more samples, the performance of the organizations can increase substantially. Interestingly, the task-delineated abstractions are exceptionally sensitive to this information deficit, which is caused by the information from the sampled episodes being unevenly distributed across the abstraction space. For example, there are relatively few states in the optimal joint policy trajectory where there are 6-7 active fires (since agents fight fires in parallel), yet those states are ones where influencing the agents into appropriate coordination is critical (so that redundant efforts of agents pursuing the same fire can be preempted). Consequently, the ODP has poor statistical estimates for how the MAS should act in these critical states, and mis-applies organizational influences, resulting in poor performance compared to Sections 5.4, 5.5, and 5.6 (though the FCount+Pos abstraction is still the third best abstraction).

The Time+Pos abstraction also exhibits interesting performance properties in these experiments. In Sections 5.4, 5.5, and 5.6, the Time+Pos abstraction performed



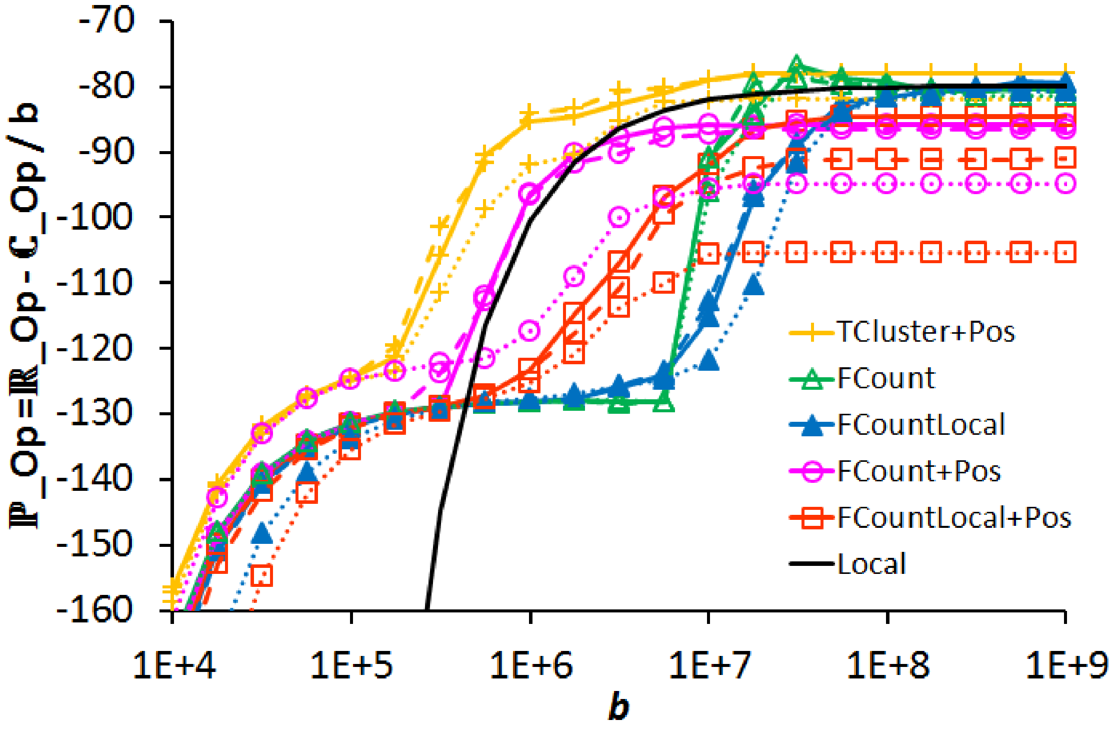
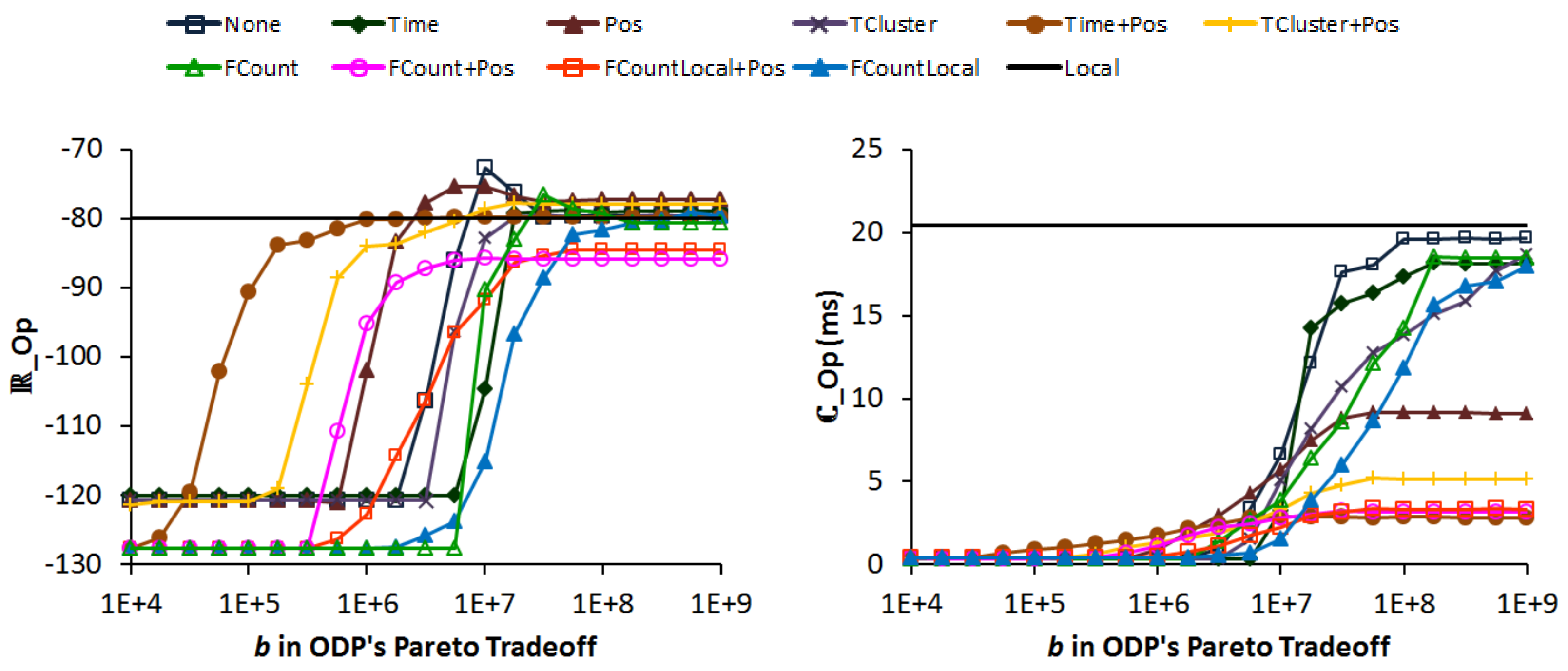
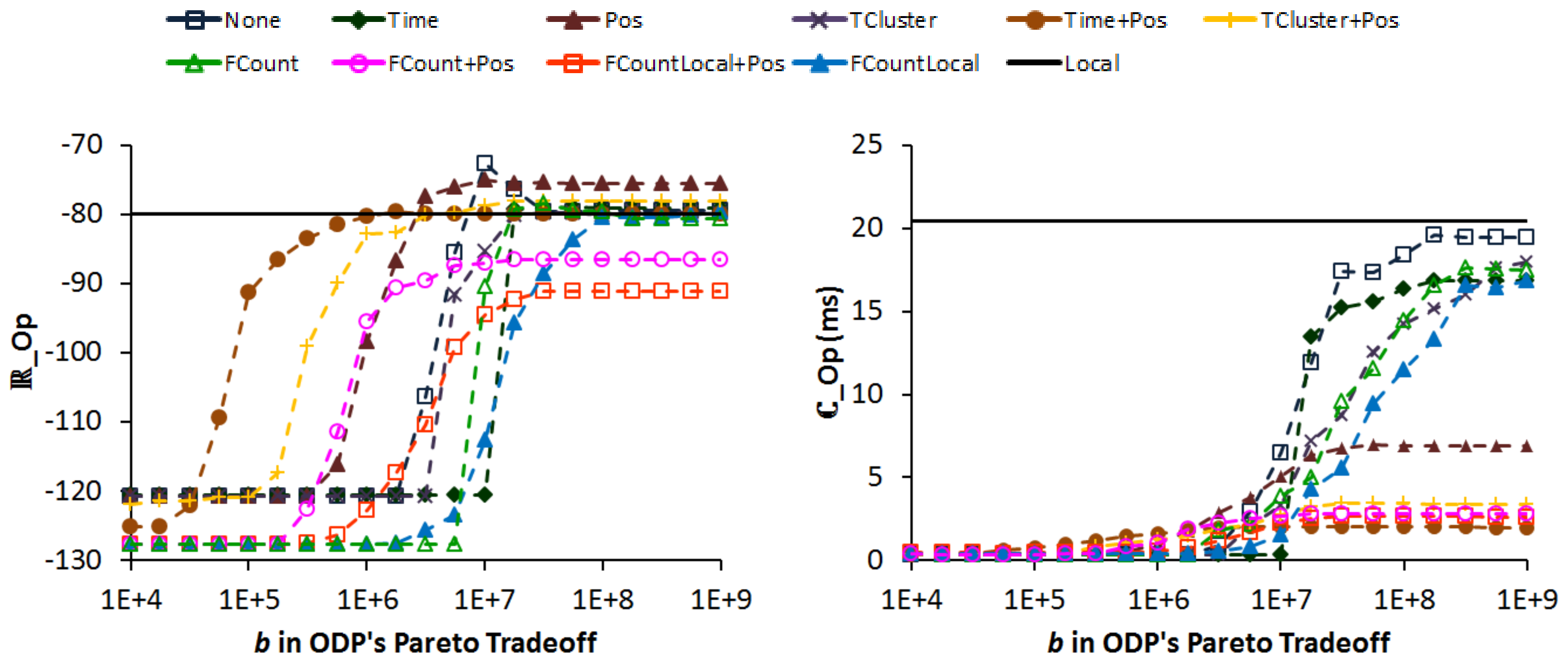


Figure 5.18: $\mathbb{P}_{Op}$ curves for each abstraction for different amounts of ODP information. Solid, dashed, and dotted curves correspond to organizations constructed from perfect information, 2/3 information, and 1/3 information respectively.



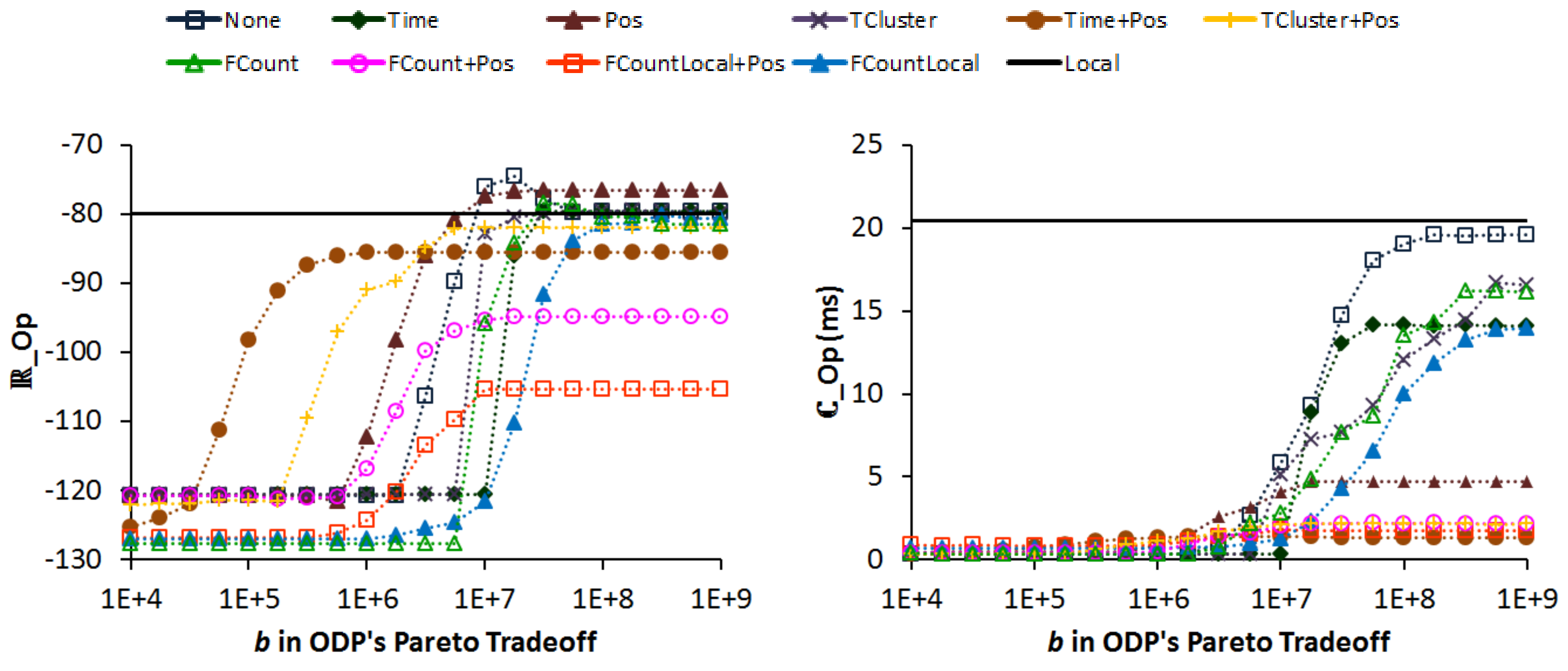
(a) (b)

Figure 5.19: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from perfect information.



(a) (b)

Figure 5.20: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from 2/3 information.



(a) (b)

Figure 5.21: $\mathbb{R}_{Op}$ and $\mathbb{C}_{Op}$ curves for organizations constructed from 1/3 information.

relatively poorly compared to the other abstraction choices (as a consequence of poor uniformity); however, the Time+Pos abstraction is arguably the best abstraction choice here (as a consequence of near perfect uniformity). Since the abstraction has such low inclusivity, the ODP is essentially overfitting the organization to the episodes that it sampled (which are identical to the test samples). In the previous experiments, overfitting yielded poor results because of uncertainty about which action was optimal for a specific location-time tuple (e.g., due to stochastic movement actions, and overpopulation of the abstraction space). Here, however, overfitting evenly distributes the information from the episode samples across the abstraction space, and comes close to simply dictating optimal policies to the agents (which is possible because of deterministic movements).

Finally, note that because of sub-par organizational performance, the local baseline performs relatively well compared to the organizations my ODP creates (e.g., in Figure 5.18). Moreover, there are many cases (when the agents' computational costs are not prohibitively expensive) where the MAS would be better off not using the organization and instead fell back to its baseline model. Unfortunately, my ODP is not designed to "realize what it doesn't know," so that it could (weakly-) dominate the baseline system by not specifying detrimental influences constructed from error-prone statistical estimates. I describe some initial thoughts for overcoming this issue more in Section 6.2.1.

5.8 Conclusions

In this chapter, I contributed a formal, agent-driven theory of abstract organizational influences. I precisely defined abstract organizational influences, and formally analyzed their dimensions in Section 5.2, before extending my existing automated ODP techniques to incorporate abstract influences (that modify agents' action spaces) in Section 5.3. I empirically evaluated the impact that such abstract influences have on both the ODP and resulting organizational performance in Sections 5.4, 5.6, and 5.7, and found that abstractions with high uniformity, moderate inclusivity, and low variance tend to reliably yield organizations that can smoothly follow the $\mathbb{R}_{Op}$, $\mathbb{C}_{Op}$ Pareto topology. Finally, I identified that constructing a provably optimal abstraction is computationally infeasible, but formulated a heuristic abstraction mechanism that decomposes the influence space into task-delineated segments, and validated the heuristic's effectiveness empirically (Sections 5.5, 5.6, and 5.7).

My formal definitions and framework for analyzing abstract influences contribute the first precise, systematic investigation of abstract organizational influences to the organizational reasoning community, whereas the field previously lacked a mathematical understanding of how abstraction choice could impact the ODP and resulting outcomes. Additionally, these contributions provide the Dec-MDP research community with a theory of organizational abstractions that informs adoption of organizational techniques in Dec-MDP frameworks. My task-delineated abstraction heuristic contributes a theoretically-derived, empirically-validated (for influences that modify agents' action spaces) heuristic for identifying an effective influence abstraction, which provides the organizational reasoning and Dec-MDP communities with an overarching strategy for selecting an influence abstraction mechanism.

CHAPTER 6

Conclusion

The focus of this dissertation has been on the development of an agent-driven approach to organization, where decisions about how to design and represent an organization for a MAS stem from how the organization is expected to impact the agents' reasoning and behaviors. I begin in this chapter in Section 6.1 by summarizing the contributions of my research, and then in Section 6.2 describe some open questions that my research raises.

6.1 Summary of Contributions

In this section, I revisit my claimed contributions from Section 1.4, and provide specific instances of where each is realized in Chapters 3, 4, and 5.

6.1.1 Organizational Specification Language

In Chapter 3, I developed an agent-driven strategy for defining the organizational design space.

- In Section 3.1, I defined an organizational influence as a modification to an agent's local decision problem, and an organization as a set of organizational influences. After committing to a particular Dec-MDP based reasoning framework for the agents' decision processes, I then enumerated the ways in which the agents' models can be organizationally influenced to define the agent-driven organizational specification language for this Dec-MDP based reasoning framework. As is, my specification language contributes to Dec-MDP based reasoning techniques since it provides a mathematically-sound approach for incorporating organizational reasoning techniques into the agents' operational reasoning processes.

- In Section 3.1, I showed how the agents can natively understand and directly incorporate their organizational influences into their local decision problems, rather than requiring additional middleware to translate an organizational specification into unambiguous information that agents can act upon. In doing so I also defined the semantics of organizational influences associated with modifications to each construct of the agents' modeling representation, which systematically defines the organizational design space for the Dec-MDP based framework I adopted. This agent-driven definition of the organizational design space contributes to the body of organizational reasoning techniques, and provides a well-defined, systematic method for understanding how an organization can relate to a MAS given the agents' reasoning frameworks, whereas prior research (Section 2.2) only informally understood this relationship.

- In Section 3.2, I leveraged the mathematical foundations of the agents' decision frameworks to formally analyze the theoretical properties of my organizational specification language. I proved that the number of expressible organizational designs in my specification language (with distinct impact on the MAS's joint policy) is $O(|\boldsymbol{\pi}|!)$, where $|\boldsymbol{\pi}|$ is the cardinality of the joint policy space. Additionally, I proved that my specification language is both necessary and complete for the Dec-MDP based reasoning framework I have adopted. This analysis contributes a mathematical characterization of the organizational design space, and provides the Dec-MDP research community with theoretically-sound, agent-driven organizational specification language. Moreover, my analytical methodology contributes a general, agent-driven strategy for analyzing the theoretical properties of organizational specification languages to the organizational reasoning community.

- In Section 3.5, I empirically demonstrated the effectiveness of specifying influences in each of the constructs of my organizational specification language. In that evaluation, I was able to hand-specify organizations that impart effective coordination patterns on a MAS, which illustrates the practical applicability (beyond the theoretical capability) of deploying of my agent-driven approach for specifying an organization. These experiments (along with my other empirical experiments throughout this dissertation) contribute evidence of my agent-driven approach's practical effectiveness to the organizational reasoning community, and also contribute an illustration of the potential benefits of organizational techniques to the Dec-MDP research community.

6.1.2 Organizational Performance Metrics

In Chapter 3, I used my agent-driven approach to construct principled metrics of organizational performance based on how the MAS is expected to perform when conforming to the organization, and used these metrics to formulate a well-defined organizational design problem.

- In Section 3.3, I leveraged the agents' reasoning framework to define the performance of an organization in terms of the MAS's joint reward and the agents' computational costs when conforming to the organization, and framed a well-defined organizational design problem from these metrics. My agent-driven definition of the organizational design problem contributes to the body of organizational reasoning techniques, and provides a formal, mathematical foundation for selecting between alternative organizations. Moreover, my problem formulation establishes the basis for computational ODP techniques that create an organization from first-principles (e.g., like I develop in later chapters), in contrast to prior research that relies on an expert human to identify appropriate organizational constructs (e.g., Section 2.2.2) that may subsequently be configured/adapted (e.g., Section 2.2.4).
- In Section 3.4, I systematically formulated of a baseline MAS against which to compare organizational performance by constructing an aligned-but-uninformed MAS. This methodology for constructing a baseline organization contributes a performance benchmark to the organizational research community that can be used to evaluate the effectiveness of an ODP technique to identify beneficial organizational influences.

6.1.3 Automated Organizational Design

In Chapter 4, I developed an agent-driven strategy for automated organizational design, and implemented that strategy to create a computational ODP for the Dec-MDP I have adopted throughout this dissertation. Also in Chapter 4, I discussed how to extend my ODP implementation to explicitly consider the agents' metareasoning regime. Then, in Chapter 5, I constructed a formal, agent-driven theory of abstract organizational influences, and showed how to extend my ODP techniques to abstract influences.

- In Section 4.2.1, I contributed a general-purpose, agent-driven methodology for constructing a computational ODP consisting of three stages, namely: compute a quantitative description of organizational patterns; select organizational influences; and influence agent decision making. First, the ODP computes a quantitative description of organizational patterns, which serves to inform the ODP of which joint interactions are worthwhile for the MAS to pursue. Second, the ODP selects organizational influences by identifying patterns in its quantitative description of organizational patterns. Third, the agent's incorporate their organizational influences and solve for their local policies as previously described (Chapter 3).

- In Section 4.3, I showed how to extend my ODP representations and algorithms to explicitly reason about and balance the Pareto tradeoff between the agents' computational costs ($\mathbb{C}_{Op}$) and the quality of their joint policy ($\mathbb{R}_{Op}$). Also in that section, I developed techniques for efficiently estimating the incremental impact (both to the agents' computational costs and quality of their joint policy) of an organizational influence, and embedded those computations in a parameterized, incremental search algorithm to create an organization that imparts the approximately optimal metareasoning regime upon the MAS. My problem formulation and resulting algorithm contribute the first ODP techniques that intentionally, explicitly optimize the metareasoning regime imparted on a MAS by an organization.

- In Chapter 5, I formulated a framework for analyzing the effects of alternative abstract organizational influences, and showed how to extend my agent-driven techniques to incorporate an abstraction mechanism. This framework contributes the first in-depth study of the effects of alternative abstraction choices both on the organizational design process and on the MAS's performance using the designed organizations. Moreover, my results in this section contribute a systematic understanding of how to design and/or choose from among influence abstraction mechanisms, namely to employ an abstraction with high uniformity, moderate inclusivity, and low variance.

- In Sections 4.2.5, 4.3.3, 5, and 5.7, I performed empirical evaluations of my ODP techniques, and found that my ODP implementation is able to construct effective organizations for the MAS that: (approximately-) optimize the metareasoning regime imparted by the organization; utilize abstract organizational influences; and scale to complex MASs even when the ODP's domain information is diminished. Individually, these experiments demonstrate the effectiveness of my ODP techniques to cope with the associated challenges of the respective sections (e.g., metareasoning, abstractions, and scaling). Moreover, together these experiments contribute compelling evidence for the efficacy of my automated ODP techniques as well as my agent-driven approach to organizations in MASs more broadly.

6.1.4 Influence Selection Heuristics

In Chapters 4 and 5, I identified heuristics for guiding a human in designing an effective organization for a MAS and also for guiding usage of my automated ODP.

- In Section 4.1, I analyzed the organizational design space to identify that a well-designed organization should influence only the factors of agents models that are associated with agent interactions, which for the Dec-MDP-based agents I employ corresponds to factors stemming from reward-/transition-dependencies. My empirical evaluation of this heuristic confirms its effectiveness, especially in MASs where the agents possess local expertise that is imprecisely modeled by the ODP. As such, this heuristic contributes an overarching strategy for focusing an ODP's efforts on the aspects of organizational design where it can best contribute to the MAS's performance, namely on influencing the agents into coordinated patterns of joint interactions rather than attempting to micromanage the details of how the agents' achieve their respective portions of those joint interactions.

- In Section 5.5, I used my influence abstraction framework to identify task-delineated abstractions as a heuristic for selecting an effective influence abstraction for the ODP, since it tends to result in abstractions with high uniformity, moderate inclusivity, and low variance. I empirically evaluated this heuristic, and found that it led the ODP to robustly construct effective organizations that smoothly follow the Pareto topology. This heuristic contributes not only a tool for guiding selection of a significant parameter of my ODP techniques (i.e., choice of abstraction mechanism), but also a broader strategy for an ODP to efficiently reason about abstracting organizational influences.

6.2 Open Questions

My work in this dissertation has focused on developing core representations and algorithms for an agent-driven approach to organizational reasoning. Unsurprisingly, however, there are several other challenging aspects of the organizational design problem (both identified from prior organizational reasoning research as well as uncovered by my work here) that could be interesting directions for further study.

6.2.1 Organizationally Adept Agents

Recalling from Section 2.2.4, the premise of organizationally adept agents (OAAs) is that the agents explicitly make adaptations to their organization because they are aware of the broader context of how their organization is expected to relate to the MAS. This is accomplished via including *annotations* in the organizational specification, where an annotation provides second-order information about the context of the organizational influences. The agents can monitor the environment as they experience it, and compare their actual experiences to the expected context for which their organizational influences were designed. If there is a mismatch, then the agents can use the annotations as a springboard from which to adapt their organization, for example dropping influences whose expectations are not being met in the environment, altering influences whose expectations are close but not perfect, and/or adding new influences to better match actual environmental conditions.

Stepping back, OAAs provide an elegant solution for dealing with uncertainty and/or information deficiencies in my agent-driven organizational reasoning techniques. For example, in the experiments in Section 5.7 where organizations were constructed from insufficient information, the MAS would have been better off by not using some of the organizational influences and instead had fallen back to the local baseline. If the agents in the MAS were organizationally adept, they could recognize that the expectations on which their organizational influences were based were not being met, and adapt their organization as appropriate (e.g., ignore unjustified influences).

Another perspective on these ideas is to provide the OAAs with several alternative organizations (e.g., sub-organizations each tailored for a specific subset of problem episodes the MAS could encounter), and empower the OAAs to multiplex among these alternatives. This perspective could be used, for example, as a concession to uncertainty about the distribution of problem episodes the MAS will experience.

In the remainder of this section, I perform some initial experiments to illustrate that my agent-driven approach is amenable to OAA techniques. To perform these experiments, I use the 10-agent firefighting domain described in Section 3.5.3 and consider two different models of how fires arise: having an increasingly higher probability of arising toward the east end of the grid; and having an increasingly higher probability of arising toward the west end. Note that the optimal organizational influences are significantly different between the two environments; in the eastEnvironment the organization should designate more agents to the eastern region (and *vice versa* for the westEnvironment). I hand-designed a specialized organization for each case, which are analogous to fullOrg from Section 3.5.3 except that the primary

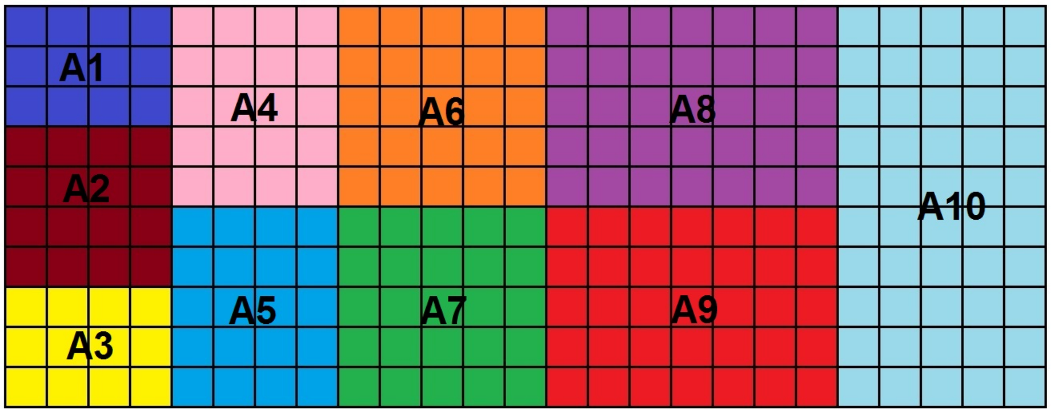


Figure 6.1: Illustration of the westOrg.

areas of responsibility (PARs) are non-uniformly sized to compensate for the biased fire distributions. For example, in the westOrg (Figure 6.1), 3 agents are responsible for the western 4 columns (4×3, 4×4, and 4×3 PARs). Working eastward, the PARs get progressively larger, starting with two 4×5 PARs (stacked vertically), then two 5×5 PARs, then two 6×5 PARs. Finally, a lone agent is responsible for the eastern edge with a 5×10 PAR. The eastOrg is a symmetric copy of the westOrg. Associated with each organization is a set of annotations informing each agent that the ODP (which was me in this case) expected one fire, on average, to be in its PAR (for that organization).

I provided the agents with both of these annotated organizations, in addition to an annotated fullOrg, which has uniformly sized and evenly distributed PARs. As such, the fullOrg is not intended to perform well for either environmental parameterization, but should be mediocre in both environments. The agents all initially adopt (based on the ODP's directives) fullOrg, to reflect the ODP's uncertainty about the environment. As episodes are experienced, the agents monitor the number of fires that are located in their respective PARs. They then jointly aggregate this observational evidence, e, and perform Bayesian inference to calculate the likelihoods that each of the possible expected environments (as captured within the annotations) is the actual environment being observed, which are used to estimate the expected reward of following each available organization. The agents then collectively and greedily adopt the organization with the highest anticipated expected reward. Formally, they adopt $\tilde{\Theta}$ as follows:

$$\tilde{\Theta} = \arg\max_{\Theta} E[\mathbb{P}_{Op} \,|\Theta, e] - c(\Theta_c, \Theta)$$

$$E[\mathbb{P}_{Op} \; |\Theta, e] = \sum_j Pr(\mathcal{M}_j|e) E[\mathbb{P}_{Op} \; |\Theta, \mathcal{M}_j]$$

where $c(\Theta_c, \Theta)$ is the cost of switching from the current organization Θ_c to Θ, and serves to temper the agents' adaptations. I assume there is no cost for remaining in the same organization, $\forall i \; c(\Theta_i, \Theta_i) = 0$. $Pr(\mathcal{M}_j|e)$ is the likelihood of environmental model $\mathcal{M}_j$ being the actual model given e, which the agents calculate via Bayesian inference. $E[\mathbb{P}_{Op}|\Theta, \mathcal{M}_j]$ is the expected performance of following organization Θ in $\mathcal{M}_j$, which I assume is provided by the ODP in the annotations. For these experiments, I estimated $E[\mathbb{P}_{Op} \; |\Theta, \mathcal{M}_j]$ by *a priori* simulating Θ on a training set of episodes created from $\mathcal{M}_j$, and $\mathbb{P}_{Op} = \mathbb{R}_{Op}$ (i.e., metareasoning concerns are ignored).

My experiments present the agents with episode batches where the true environment model, $\mathcal{M}^*$, is selected uniformly randomly from the two environments every 20 episodes (all organizations face the same episodes in the same order). The agents are allowed to collectively adopt whichever organization they deem best at the beginning of each episode before they observe the initial state. Since $\mathcal{M}^*$ is dynamic, I allow the ODP to set a decay rate in the annotations, which the agents use to decay the importance of their past observations of how many fires were in their PARs. I evaluated several organizations on this problem set: statically using the east/west/fullOrg for every episode; and several parameter settings of the OAA process described above. OAAX refers to the OAA process above where the organizational switching cost is X.

My results are summarized in Figure 6.2, which confirms several intuitions. Firstly, statically following either specialized organization is generally undesirable since they perform poorly when used in the environment they were not intended for (i.e., using the westOrg in the eastEnvironment); however, statically following fullOrg makes a significant improvement since it is weakly suited to both environments. Secondly, by allowing the agents to react to the shifting environment, the OAA capability (in general) can yield a large performance gain. Finally, if the organizational switching cost is low, the agents should maintain sufficient observational evidence history in order to prevent the agents from switching organizations due to a transient episode, such as when an episode from the eastEnvironment happens to "look" like an episode from the westEnvironment due to unlikely fire locations.

These results provide an initial demonstration that my agent-driven approach is extensible to OAAs, but leave many questions unanswered. Most significantly, how does an ODP create annotations to accompany its organizational influences (which was done by hand in my experiments above)? In principle, my ODP has an extensive number of expectations about the domain and/or agents, for example, as captured

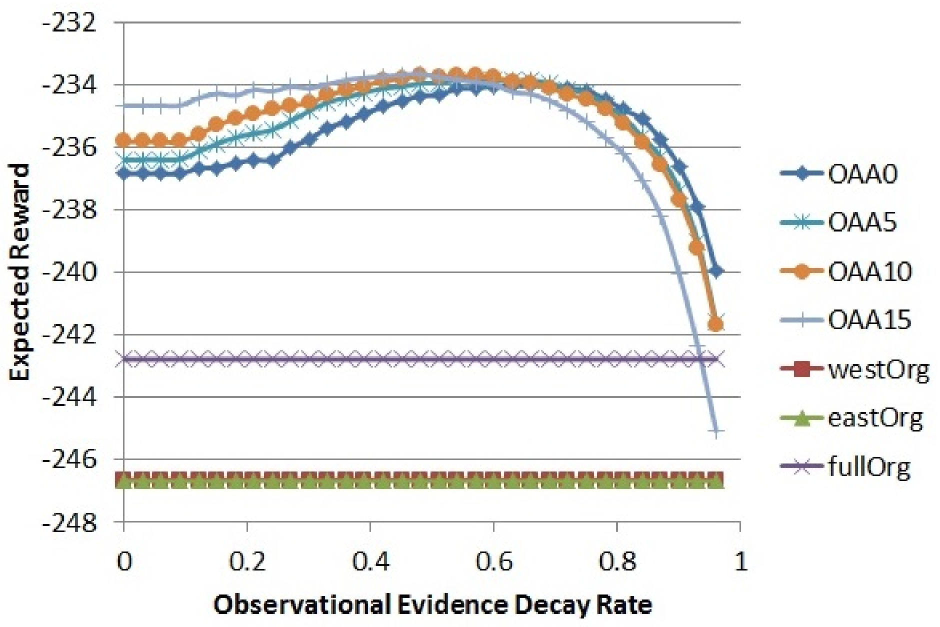


Figure 6.2: Expected reward vs. the observational evidence decay rate.

in its quantitative description of organizational patterns (Section 4.2.2); however, providing this full set of expectations as annotations seems at best excessive or at worst counterproductive (e.g., could lead agents to fixate on less significant expectations with little practical importance). An open challenge is thus identifying a subset of (possibly abstract) organizational annotations that summarize the significant expectations upon which the organization is vitally dependent.

Another significant challenge is how to maintain organizational consistency across the OAAs as they adapt their organization (I forced agents to make collective adaptations in my experiments above). Organizations are constructed to be utilized as a whole (i.e., the influences to the various agents are codependent), and unilateral OAA adaptations may result in mismatched influences with poor aggregate operational performance. Further, it is not known under which conditions locally-motivated organizational adaptations will converge (assuming a stationary distribution of episodes), or the speed of such a convergence (should it exist). As such a possible direction for future research would be to develop strategies for decentralized organizational adaptations that ensure the currently adopted organization is internally consistent.

6.2.2 Other Influence Mechanisms

My organizational specification language (Section 3.1) identifies that each of the constructs in the agents' decision making framework could be used as a lever for influencing the agents. Throughout this dissertation, I have performed empirical experiments with:

- Hand-identified influences to the agents' state factors, action spaces, transition function factors, and reward function factors in Sections 3.5 and 4.1.
- Influences identified by my automated ODP (without regard to metareasoning and/or abstraction concerns) to the agents' state factors, action spaces, and transition function factors in Section 4.2.
- Influences identified by my automated ODP (accounting for metareasoning and/or abstraction concerns) to the agents' action spaces in Section 4.3 and Chapter 5.

As mentioned in the respective sections, I did not evaluate influences to the agents' initial state distribution factors and/or time horizon because they are generally inappropriate for the firefighting domain. An open question, in this regard, is if there are domains where influences to these factors have practical significance.

I did not evaluate influences that modify agents' reward function factors with my automated ODP techniques, because as I identified in Section 3.5, reward influences do not have the capacity to affect the agents' computational costs ($\mathbb{C}_{Op}$). Thus, when considering the ODP in the context of metareasoning issues, reward influences are a less interesting mechanism to study. As mentioned in Section 4.2.3, however, reward influences could be a useful fallback mechanism if the ODP cannot formulate hard-constraint influences (i.e., states, actions, and transitions).

A more significant open challenge is extending my automated ODP techniques to create influences to factors other than the agents' action spaces, while still accounting for metareasoning and abstraction concerns. As mentioned in Section 4.3, I elected to focus on influences to the agents action spaces because these types of influences are well studied in prior research and have efficiently-computable impact on the agents' local decision problems. However, as illustrated with my hand-designed influences (Sections 3.5 and 4.1) and when the ODP neglected metareasoning/abstraction concerns (Section 4.2), influences to the other components of the agents' decision problems can also be effective, and moreover provide additional expressive power for organizationally influencing the MAS (as shown in Appendix A). The primary difficulty in creating influences to the transition and state (accounting for metareasoning/abstraction) is that the impact of these influences to the agents' decision problem can be far reaching, rather than with limited scope as in action influences. For example, an influence to remove a_i from consideration in s_i^t is created because a_i is not a Pareto-valuable action for agent i to consider, and the ODP can compute the influence's impact by directly examining agent i's policy at s_i^t. In contrast, an influence to alter $P_i(f_k^{t+1}|s_i^t, a_i)$ is created because the ODP wants to change agent i's local policy, potentially at various states, at various decision points (e.g., at s_i^{t-1}, s_i^0, etc.), and thus the ODP cannot directly compute the influence's impact by directly examining the i's policy at a single state but rather must examine the policy more broadly. As a result, computing the $\Delta_i \cdot \frac{d\mathbb{C}_{Op}}{d\Theta^j}(\Theta^j)$ and $\Delta_i \cdot \frac{d\mathbb{R}_{Op}}{d\Theta^j}(\Theta^j)$ terms of Equation 4.2 is complex.

Another aspect of my specification language that I have not explored is the ability to create new state factors within the agents' local representations; rather, my experiments dealt only with constraining the agents' existing state factors. Intuitively, however, the capability to add new state factors to an agent's model is incredibly powerful as an organizational influence mechanism. For example, consider an influence that adds a new state factor to agent i's model that represents which task (from among a set of organizationally provided tasks) i is currently pursuing. Such a state factor is related to my task-delineated abstraction heuristic, but because it would be explicitly represented in i's state representation, the agent could use it while solving its decision problem and coordinating with other agents (as opposed to being simply a reasoning mechanism for the ODP). As another example, consider an influence that adds a new state factor to multiple agents' representations to serve as a sentinel flag for coordinating operational decisions. This type of state factor could allow the ODP to directly inform the MAS of which non-local information is critical to joint performance. Unsurprisingly, creating new state factors is an exceptionally challenging problem both since the space of possible factors is intractably large (i.e., infinite) and the expected impact of adding a new state factor is challenging to compute. As such, developing bounds on new state factors that are worth the ODP's consideration and/or heuristics for selecting from among viable candidate factors would be important topics for pursing this direction in future work.

6.2.3 Biasing the ODP's Statistics to Encourage Patterns

A somewhat subtle artifact of my ODP methodology is that not only does my ODP's effectiveness depend on the existence of interaction patterns within the problem episodes it samples (which intuitively is necessary for any ODP), but those patterns must be consistently expressed the same way in order for the ODP to have good statistical information. For example, in the firefighting domain, the agents' movements throughout the grid to a fire could arbitrarily follow any Manhattan path (assuming equally restrictive cell delays). If the ODP's policy solver arbitrarily decided among these alternatives for each episode sample, then the ODP's statistical estimates would be inconsistent, and essentially devalue the ODP's view of how important it is for the agent to fight that fire. Another example of this is if two agents are exact replicas (e.g., in the firefighting domain, if they begin in the same initial location with the same local model like in the clustered variant of experiments in Section 3.5.3), where the ODP could arbitrarily allocate a fire to either of those agents. More generally, this effect occurs if the ODP's policy solver arbitrarily breaks ties between equally good policies.

I have identified four fundamental ways for overcoming this issue, which I describe now.

- Break ties in a consistent fashion. That is, rather than break ties arbitrarily, the ODP's policy solving process (i.e., as part of computing a quantitative description of organizational patterns in Section 4.2.2) can consistently break ties in the same way. For some policy solving algorithms (e.g., value-iteration), this solution is straightforward, but for others (e.g., the linear program representation I use, Equation 2.4) would require comprehensive knowledge of the solution technique. Of course, this solution also implies that the interaction patterns that the ODP eventually identifies will be biased by whatever tie-breaking procedure is selected.

- Prevent ties by making policies distinctly valued. If ties between equally good policies are impossible (or exceedingly unlikely), then this issue is insignificant. Throughout this dissertation, my experiments have tended to adopt this solution when possible to avoid confounding results. For example, stochastic cell delays decrease the likelihood of equally viable Manhattan paths, and having the agents begin adjacent to each other (but not stacked) differentiates the optimal joint policy. However, naturally this solution is not generally applicable to all domains.

- Use abstractions such that the statistical information from all alternative policies maps to the same part of the abstraction space. For example, in Chapter 5, the issue of alternative Manhattan paths to a fire is less significant if the agent's location does not contribute to the abstraction mechanism (although could still matter to an extent because of system time, for example). This solution is broadly applicable to all domains and policy solving algorithms, but could result in an abstraction mechanism with other undesirable characteristics (e.g., poor uniformity).

- Iteratively construct an organizational design, where the organizational patterns identified from earlier sampled episodes bias the policy solver for subsequent samples. For example, if the agent's policy in an earlier episode took a Manhattan path that goes north then east, the policy solver could be biased to take Manhattan paths that go north before east in subsequent samples. This solution is also general purpose, and moreover has theoretical basis in experience-driven organizational techniques (Section 2.2.3). An open challenge, however, is how to temper the reinforcement process so that the ODP can avoid getting trapped in local optima based on its early samples.

6.2.4 Advanced Statistical Representations and Abstraction Choice

Throughout this dissertation, I computed the impact of an organizational influence by taking the expectation, which carries an implicit assumption of a uni-modal distribution. However, as mentioned in Section 5.4.3, with abstract influences it is possible for an ODP to be reasoning about multimodal distributions, which fundamentally violates the assumptions of my primitive statistical representation, and as a consequence, results in the ODP mis-applying influences. An open challenge is to develop a strategy for identifying an appropriate statistical representation in response to the information the ODP needs to model, as well as to extend the ODP's search algorithm to reason appropriately with the more advanced statistics.

Stepping back, the statistical representation problem above is but one manifestation of a more fundamental question of how to best represent and compose the ODP's limited information. That is, the information the ODP possesses should be used to guide both the choice of influence abstraction mechanism as well as the underlying statistical representation. For example, in Section 5.7, I identified that my task-delineated abstraction mechanism did not evenly distribute the ODP's information across the abstraction space for the four agent domain, and as a result the ODP mis-identified influences at critical points of the agents' decision problems. A comprehensive solution to these issues, however, could dynamically adapt the ODP's abstraction mechanism in response to the information it obtains from each sampled episode, so as to avoid over-/under-populating any portion of the abstraction space. A challenge, however, is efficiently determining the best abstraction from among the intractably many alternatives.

6.2.5 Relaxing the Restrictions to Organizational Influence Mappings

Throughout this dissertation, I limited my discussions to organizational influences with many-to-one mappings, and the ODP specified influences using a single abstraction mechanism for the entire organization. That is, an (abstract) organizational influence could modify agent i's local model at several $s_i^t \times a_i \times s_i^{t+1}$ tuples, but any specific $s_i^t \times a_i \times s_i^{t+1}$ tuple could only be modified by a single organizational influence.

More generally, an organization could consist of influences with different abstractions, for example corresponding to the amount of information the ODP has about that section of the influence space as just described in Section 6.2.4. Even more generally, an organization could consist of hierarchical influences, where for example the influences at the top level of the hierarchy map to every state and broadly modify the agents' reasoning to align with organizational objectives, and influences at the bottom level of the hierarchy map to singular states and rigidly modify the agents' reasoning to ensure effective coordination at critical junctures. This type of hierarchical influence specification is closely related to how organizational modeling languages specify organizations (Section 2.2.2).

Extending the agents' reasoning mechanisms to permit hierarchical specifications of organizational influences seems rather straightforward (indeed abstract influences do not provide additional expressive power in the organizational specification language), although some edge cases merit further consideration. For example, should all the layers in a hierarchical influence specification have their modifications applied to the agents' models (assuming they are compatible with each other), or just the bottom-most layer, or perhaps just some of the layers? Additionally, how should the agents respond if the layers contain conflicting influences (supposing such conflicts are permitted)? Arguably, such decisions could be made by the ODP and included as part of organizational specification (at the expense of further complicating the ODP). The larger challenges to relaxing the restrictions of influence mappings lie on the ODP side, where the flexibility to consider influences within the context of multiple and/or arbitrary abstraction mechanisms could greatly increase the computational complexity of the ODP.

6.2.6 Scaling the ODP

I have identified five dimensions, corresponding to Properties 1.1– 1.5 respectively, that intuitively could stress the ODP techniques I have developed in this dissertation:

Number of Agents. Increasing the number of agents in the MAS increases the number of possible coordination patterns that an ODP must consider. Additionally, those patterns could contain more agents, which could make the patterns more complex. This dimension's impact on my ODP can be seen empirically in Section 5.7.

As previously shown in Section 4.3.2.3, the complexity of Algorithm 4.2 is

$$O\left(|\mathbf{\Delta_i}|^2\left(|S_i^{successor}| + |A||S|\right) + |A_i|^{|S_i|^n}\right)$$

which depends on the number of agents (n) both directly and indirectly (since the size of the influence space is linear in the number of agents). The $|A_i|^{|S_i|^n}$ term reflects the computational complexity for the ODP to compute the optimal joint policy for a sampled problem instance (as described in Section 4.2.2), and presents a serious challenge in scaling my ODP implementation. For example, state of the art Dec-(PO)MDP research (see Section 2.1.4) often evaluates algorithms in domains with only two or three agents and short time horizons (i.e., < 10) due to the general intractability of the reasoning framework. As such, scaling this portion of my ODP algorithm up beyond a few agents is theoretically intractable for Dec-MDP based agents, unless additional assumptions about problem structure are made (e.g., coordination locales or specific forms of inter-agent dependencies). In the experiments in Section 5.7, I scale the firefighting domain to four agents, which is the largest system for which I could solve a problem episode for the optimal joint policy.

Stepping back, however, computing joint policies for sampled problem episodes is a particular implementation of my more general ODP methodology for computing a quantitative description of organizational patterns. As mentioned in Section 4.2.1, if a quantitative description of organizational patterns can be obtained from some other source (e.g., an external expert or the agents themselves), then the other stages of my ODP can use those provided statistics instead of the ODP computing them itself. Examining the theoretical scaling capabilities of my ODP for the other stages, we see that the ODP's organizational search costs scale quadratically in the number of agents, at least for the greedy hill-climbing search (Algorithm 4.2).

Beyond the ODP's computational costs, it is also important that the ODP continue to produce organizations that perform well, even for larger MASs. Unsurprisingly, the quality of an organization is determined by a myriad of factors besides the number of agents, and cannot be formally disentangled without making further (unreasonable) assumptions about the domain (e.g., the degree and form of inter-agent dependencies). To provide a high-level intuition, however, larger MASs present the possibility of larger and/or more complex coordination patterns, which could stress an ODP's ability to identify appropriate organizational influences. Along these same lines, the larger space of joint interactions also stresses the ODP's limited information (e.g., for computing a quantitative description of organizational patterns), since information is dispersed across a larger space of influences, and moreover, a fixed episode sample size represents a smaller subset of the possible coordination patterns the MAS could exhibit.

MAS Iterdependencies. Tighter coupling among agents increases the importance of correctly identifying interaction patterns and/or makes those patterns more complex. This dimension's impact on my ODP can be seen, for example, in the experiments from Section 5.6, where the agents beginning adjacent to each other increases the effective interdependency between them. Thus, as shown in those experiments, my ODP seems robust in this scaling dimension, and if anything creates organizations with larger benefit to the MAS as the MAS becomes more interdependent. Recognize, however, that these improvements could be the result of simply having more opportunities for organizing the MAS, and my ODP is actually scaling poorly in this dimension (i.e., a decreasing percentage relative to the optimal organization) despite an increase in absolute effectiveness. Thus, further investigation is necessary to definitively determine how my ODP scales in terms of the MAS's interdependencies.

Information Deficiencies. With less information, the ODP cannot correctly influence the MAS as aggressively, and must instead rely on the agents to appropriately exercise their local expertise. This dimension's impact on my ODP can be seen, for example, in the experiments from Section 4.1 where the ODP's model was smoothed, and/or experiments from Section 4.3 where the ODP had fewer sample episodes from which to compute a quantitative description of organizational patterns. As shown in those experiments, by following my organizational design principle of only influencing the agents' interaction patterns, and using a task-delineated abstraction mechanism, my ODP is relatively robust to information deficiencies. Although, as shown in Section 5.7, the ODP becomes susceptible to mis-applying influences when it has severely insufficient information. It is challenging to formally disentangle exactly how information deficiencies affect the quality of the organizations the ODP creates; however, at a high level, information deficiencies degrade the ODP's statistical estimates of how the agents should act. At certain thresholds (that depend on the domain), these data errors lead the ODP to mis-identify which influences are appropriate for the MAS, and can result in an under-performing, even detrimental, organization.

Coordination Overhead. As the agents are increasingly unable to coordinate their local policies operationally, an ODP must compensate by providing an organization to influence the agents into effective coordination patterns. Recalling my experimental methodology described in Section 3.5.1, throughout my experiments, I have forced all coordination to stem from the organization rather than permit the agents to coordinate their local polices with operational reasoning. As such, my empirical results throughout this dissertation already reflect the worst case of this scaling dimension.

Temporal Scope of the Organization. To create an equally performing organization but over a more diverse set of problem episodes, the ODP must identify either broader organizational influences or a larger set of narrow influences. This dimension's impact on my ODP can be seen across the empirical results throughout this dissertation; for example, as the chapters progressed, I intentionally added addition parameters and complexity to the firefighting domain (e.g., stochastic cell delays, additional fires, and more agents). As the results in the respective sections show, my ODP is able to adapt to a more diverse space of problem episodes.

6.2.7 Open Systems

As discussed in Section 2.2.2, organizations (especially from a problem-driven approach) are often viewed as part of an open system, where agents can join and leave the organization at their will, but the organization remains comparatively static. Since my agent-driven approach leverages the agents' decision models as a basis for organizational reasoning, open systems are a somewhat unnatural premise in my approach, unless another agent with the same capabilities can be recruited to directly substitute for an agent that leaves the organization. While that assumption could be reasonable in some domains (e.g., with homogenous agents), in other domains (e.g., with heterogeneous agents and finely tuned local expertise) such direct substitutions may not be feasible. Of course, the ODP could redesign an organization for the new MAS every time agents come and go, but doing so could be computationally demanding and/or excessive, especially if the net change to the organization from different agents is small. The challenge, thus, is to develop efficient techniques for designing an organization for a MAS, given an existing organization for a similar MAS, and resembles the transfer learning problem (Pan & Yang, 2010). Alternatively, the ODP could assume agents are OAAs, and so will adapt their organizational roles to fit their local expertise, in which case the challenge is for the ODP to identify and account for such adaptations.

APPENDIX

APPENDIX A

Formal Proofs of Specification Language Necessity

In this appendix, I provide formal proofs for the necessity of each construct in my organizational specification language (Section 3.1) as well as briefly discuss how each construct has been used in previous work to influence agents in a MAS.

Rewards. As discussed in Section 2.2.5, the idea of influencing reward functions is well studied in prior operational reasoning research. By influencing an agent's reward function, the organization provides extra incentive for the agent to execute actions that are expected to positively contribute to collective MAS performance (and *vice versa* for disincentives). In the firefighting domain, for example, an agent could have a reward associated with being located within a certain region of responsibility. As another example, reward influences can be used to lead an agent to establish conditions that have low uninfluenced local reward, but that enable other agents to then take actions that lead to high joint reward.

Theorem A.1. *Organizational influences to reward factors are a necessary specification language construct for the Dec-MDP framework I have adopted.*

Proof. **By Exhaustion.** Assume that agent i's local reward function is constant for all conditions, that is, $\forall s_i^t, s_i^{t+1} \in S_i, \forall a_i \in A_i, R_i(s_i^t, a_i, s_i^{t+1}) = c$ for some constant c. This implies agent i's Q-values are invariant with respect to the topology of agent i's transition function, and more precisely that $\forall s_i^t, \in S_i, \forall a_i \in A_i, Q_i^\pi(s_i^t, a_i) = c \cdot (T_i - t)$.

Now suppose that an ODP wants to reprioritize agent i's local policy space (to an organizationally desired total-ordering) but without adding or removing any policies from agent i's consideration. This could occur, for example, if the ODP knows that, given the influence abstraction mechanism it is using, agent i needs to be able to consider both action a_i and a'_i in the set of states that $\hat{\Delta}_i$ will map to. This type of influence cannot be expressed by modifications to agent i's state factors, actions, initial state distribution factors, or time horizon, because those types of modifications will alter the space of policies agent i considers. Additionally, modifying agent i's transition factors will not affect its policy because its Q-values are invariant to its transition function. The only remaining construct for expressing this influence is modifying agent i's reward factors, and indeed modifying agent i's rewards will impact its Q-values and subsequently re-prioritize its local policy space without adding or removing any policies from consideration. □

Transitions. Researchers have also identified how influences to an agent's local transition function can be a powerful tool for guiding agent policies (Section 2.1.4.1). By informing an agent of the expected non-local effects on its local model, the agent can create a local policy that accounts for the actions of the other agents. For example in the firefighting domain, an agent could be informed that fires will (probabilistically) be extinguished at certain times without the agent fighting them. This is due to other agents' efforts; however, the means of extinguishing, or even existence of other agents, need not be specified.

Theorem A.2. *Organizational influences to transition factors are a necessary specification language construct for the Dec-MDP framework I have adopted.*

Proof. **By Exhaustion.** Assume that, given the organization an ODP has identified, agent j stochastically exerts a non-local effect on agent i's local state space. As a consequence of this non-local effect, suppose agent i should prefer local policy π_i over local policy π'_i, for example, because π_i establishes necessary preconditions for agent i to capitalize on the non-local effect if it should occur. Further, assume that given the influence abstraction mechanism the ODP is using, the ODP does not want to use $\hat{\Delta}_i$ to directly prevent any of the actions of agent i, because those actions are Pareto-valuable over the space of states that $\hat{\Delta}_i$ maps to. In addition to actions, this type of influence cannot be expressed by modifications to agent i's state factors, initial state distribution factors, or time horizon, because those types of modifications cannot provide agent i with information about the non-local effect's transition dynamics. A $\hat{\Delta}_i$ that modifies agent i's reward factors could potentially induce agent i to prefer π_i over π'_i; however, if we additionally assume the ODP has imprecise knowledge of the agent's local rewards (Property 1.3), the ODP cannot use influences to reward factors to ensure agent i prefers π_i without overwriting i's local expertise, which

I have demonstrated as undesirable (Section 4.1). The only remaining construct for expressing this influence is modifying agent i's transition factors, and indeed, modifying the transition function factors can inform agent i of these non-local effects, and lead agent i to prefer π_i over π_i'. $\square$

Actions. Like reward and transition influences, influences to an agent's action space are also a familiar approach in the literature (e.g., see Sections 2.2.2, 2.2.4, and 2.2.5). The idea of action influences is to restrict the actions an agent can consider, for example, an organizational designer might associate different roles with different agents and thus induce agents to specialize in the possible actions they will exercise. Chosen well, such restrictions not only help agents pursue complementary policies, but simplify planning for each (Section 4.3). For example, in the firefighting domain, an agent might be assigned a region of responsibility, and thus not need to consider actions that would take it out of that region since other agents are responsible for other regions. Alternatively, an action influence could ensure that an agent considers an action that it might not have otherwise (e.g., a communicative action).

Theorem A.3. *Organizational influences to actions spaces are a necessary specification language construct for the Dec-MDP framework I have adopted.*

Proof. **By Exhaustion.** Assume the ODP wishes to constrain agent i's local policy space such that a certain action, a_i, can never be performed, or even considered, in state s_i^t, but otherwise leave i's local policy space unchanged. Influences to the agent's state factors, initial state distribution factors, transition function factors, and/or time horizon can only prevent a_i from consideration in s_i^t by preventing consideration of s_i^t entirely, which modifies agent i's policy space beyond what is permitted in the example and thus inappropriate. Influences to agent i's reward function factors could discourage the agent from performing a_i, but do not prevent i from considering a_i and thus are inappropriate in this example. The only remaining construct for expressing this influence is to modify agent i's action space, and indeed, modifying the action space can prevent agent i from performing and considering a_i in s_i^t. $\square$

States. While influences to an agent's state representation as a means to induce coordination has not appeared in prior work, state abstraction more generally has received a great deal of attention (Andre & Russell, 2002; Roy et al., 2005; Li et al., 2006). One underlying concept to organizationally influencing an agent's state factors is that, given the organization, there could be state factors that an agent can sense that are unnecessary to represent. For example, in the firefighting domain, an agent may not need to represent the intensity of distant fires because they are the responsibility of other agents. In addition to restricting an agent's local state space, an organization might purposely augment an agent's local state representation with new features, where the ODP has decided that those features are crucial to distinguishing between states that otherwise would look locally identical. Such augmentations can provide additional information that improves operational performance, but it also falls on the ODP to delineate the communication protocols and policies that would ensure an agent possesses up-to-date values for those features despite not being able to directly observe them. For instance, in the firefighting domain, to improve coordination, an ODP might insist that each firefighter tell the others which fire it is now working towards extinguishing.

Theorem A.4. *Organizational influences to state factors are a necessary specification language construct for the Dec-MDP framework I have adopted.*

Proof. **By Exhaustion.** Assume there are two distinct global states that appear locally identical to agent i, $\exists s^t, s'^t \in S, s_i^t, s_i'^t \in S_i$, s.t. $s^t \neq s'^t$, but $s_i^t = s_i'^t$, where the ODP has determined it is critical for agent i to perform distinct actions in those two states. Influences to agent i's local action space, reward function factors, transition function factors, time horizon, and/or initial state distribution factors will not allow agent i to distinguish between s^t and s'^t. The only remaining construct for expressing this influence is to modify agent i's state factors, and indeed, modifying i's state representation (and the necessary accompanying transition and initial state influences) could add another factor to agent i's state representation that allows it to distinguish between s^t and s'^t. □

Finite Time Horizon. Influences to an agent's finite time horizon have not been studied in prior research to the best of my knowledge. The central idea here is that an ODP might determine that the improved parallelism from better coordination means that agents can safely reason over shorter time horizons. Alternatively, the ODP might improve coordination by increasing the time horizon for the agents, effectively asking them to be less myopic.

Theorem A.5. *Organizational influences to state factors are a necessary specification language construct for the Dec-MDP framework I have adopted.*

Proof. **By Exhaustion.** Assume the ODP wishes to influence the number of decision points that agent i plans for. Influences to agent i's rewards function factors, action spaces, and/or initial state distribution factors clearly can not impact the number of decision points that i plans for. Influences to agent i's states and transitions, however, could meaningfully impact the number of decision points, for example, by adding a state factor to keep track of the number of prior decision points, and then influencing the transitions to deterministically (independently of the action taken) go to a terminal state when the number of decision points exceeds some threshold. This threshold, however, is precisely what the time horizon represents, and other than using this threshold, state and/or transition influences cannot impact the number of decision points. The only remaining construct for expressing this influence is to modify agent i's time horizon, and indeed, modifying the time horizon prompts agent i to plan for more/fewer decision points. □

As just illustrated, the time horizon is unnecessary to some extent in that an identical effect can be realized within the state and transition influences (and simply not explicitly labeled as the time horizon). In this case, however, I would also argue that there should not be a finite time horizon explicitly represented in the agents' local models either, but rather implicitly encoded within their local states and transition functions. Since I have presented the agent's local models as explicitly representing the finite time horizon, I advocate that any influences to the number of decision points that relies on a threshold should also be labeled as influences to the finite time horizon, which leads to the time horizon being a necessary construct given my agents' decision models as I have presented them.

Initial State Distribution. Like time horizon influences, I am unaware of any prior work that influences an agent's initial state distribution. If the ODP is permitted to impose initial states on the MAS (e.g., initially spread the agents throughout the grid in the firefighting domain), then influencing the MAS into more desirable, coordinated initial configurations could be a powerful organizational mechanism (e.g., as my experiments in Section 3.5 demonstrate). Other uses for influences to the agents initial state distribution factors include providing the necessary initialization information for new state factors that agent i cannot directly observe.

Theorem A.6. *Organizational influences to initial state distribution factors are a necessary specification language construct for the Dec-MDP framework I have adopted.*

Proof. **By Direct Proof.** As proved in Theorem A.4, adding a new state factor to agent i's initial state distribution is a necessary construct in my specification language. Since every state factor must be covered by an initial state distribution factor (Definition 2.3), influences to an agent i's initial state distribution factors are also a necessary construct. $\square$

BIBLIOGRAPHY

BIBLIOGRAPHY

Abdallah, S., & Lesser, V. (2007). Multiagent reinforcement learning and self-organization in a network of agents. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems*, (pp. 172–179).

Agogino, A. K., & Tumer, K. (2005). Multi-agent reward analysis for learning in noisy domains. In *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems*, (pp. 81–88).

Alexander, G., Raja, A., Durfee, E. H., & Musliner, D. J. (2007). Design paradigms for meta-control in multi-agent systems. In *Proceedings of AAMAS 2007 Workshop on Metareasoning in Agent-based Systems*, (pp. 92–103).

Amato, C., Konidaris, G. D., & Kaelbling, L. P. (2014). Planning with macro-actions in decentralized POMDPs. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems*, (pp. 1273–1280).

Andre, D., & Russell, S. J. (2002). State abstraction for programmable reinforcement learning agents. In *Proceedings of the 18th National Conference on Artificial Intelligence*, (pp. 119–125).

Babes, M., de Cote, E. M., & Littman, M. L. (2008). Social reward shaping in the prisoner's dilemma. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, (pp. 1389–1392).

Barto, A., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*(4), 341–379.

Becker, R., Zilberstein, S., & Lesser, V. (2004a). Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 302–309).

Becker, R., Zilberstein, S., Lesser, V., & Goldman, C. (2004b). Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, *22*, 423–455.

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

Bernstein, D., Amato, C., Hansen, E., & Zilberstein, S. (2009). Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research*, *34*, 89–132.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*(4), 819–840.

Bernstein, D. S., Hansen, E. A., & Zilberstein, S. (2005). Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conferences on Artificial Intelligence*, (pp. 1287–1292).

Bernstein, E. S. (2012). The transparency paradox: A role for privacy in organizational learning and operational control. *Administrative Science Quarterly*, *57*(2), 181–216.

Bichier, M., & Lin, K.-J. (2006). Service-oriented computing. *Computer*, *39*(3), 99–101.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford.

Bordini, R. H., Hübner, J. F., & Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, vol. 8. John Wiley & Sons.

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, (pp. 195–210).

Boutilier, C., Dearden, R., & Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, *121*, 49–107.

Bowling, M., & Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, *136*(2), 215–250.

Bratman, J., Singh, S., Sorg, J., & Lewis, R. (2012). Strong mitigation: Nesting search for good policies within search for good reward. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 407–414).

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multiagent Systems*, *8*(3), 203–236.

Brooks, C. H., & Durfee, E. H. (2003). Congregation formation in multiagent systems. *Autonomous Agents and Multiagent Systems*, *7*(1-2), 145–170.

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *38*(2), 156–172.

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*, vol. 39. CRC Press.

Butts, C. T., & Carley, K. M. (2007). Structural change and homeostasis in organizations: A decision-theoretic approach. *The Journal of Mathematical Sociology*, *31*(4), 295–321.

Castelfranchi, C. (1995). Commitments: From individual intentions to groups and organizations. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, vol. 95, (pp. 41–48).

Corkill, D., Durfee, E., Lesser, V., Zafar, H., & Zhang, C. (2011). Organizationally adept agents. In *Proceedings of the Coordination, Organization, Institutions and Norms in Agent Systems 2011 Workshop at AAMAS*.

Corkill, D. D. (1979). Hierarchical planning in a distributed environment. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, (pp. 168–175).

Corkill, D. D. (1991). Blackboard systems. *AI Expert*, *6*(9), 40–47.

Corkill, D. D., Garant, D., & Lesser, V. R. (2015). Exploring the effectiveness of agent organizations. Tech. rep., University of Massachusetts Amherst - School of Computer Science.

Corkill, D. D., & Lander, S. E. (1998). Diversity in agent organizations. *Object Magazine*, *8*(4), 41–47.

Corkill, D. D., & Lesser, V. R. (1983). The use of meta-level control for coordination in a distributed problem solving network. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*.

Corkill, D. D., Zhang, C., da Silva, B., Kim, Y., Zhang, X., & Lesser, V. R. (2012). Using annotated guidelines to influence the behavior of organizationally adept agents. In *Proceedings of the Coordination, Organization, Institutions and Norms in Agent Systems 2012 Workshop at AAMAS*.

Cox, M. T., & Raja, A. (2011). *Metareasoning: Thinking about thinking*. MIT Press.

Dastani, M. (2008). 2APL: A practical agent programming language. *Autonomous Agents and Multiagent Systems*, *16*(3), 214–248.

Davis, R., & Smith, R. G. (1983). Negotiation as a metaphor for distributed problem solving. *Artificial intelligence*, *20*(1), 63–109.

Dietterich, T. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Dignum, V. (2004). *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. Ph.D. thesis, Universiteit Utrecht.

Dignum, V., Vázquez-Salceda, J., & Dignum, F. (2005). Omni: Introducing social structure, norms and ontologies into agent organizations. In *Programming Multi-Agent Systems*, (pp. 181–198).

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *Computational Intelligence Magazine, IEEE*, *1*(4), 28–39.

Durfee, E. H. (1993). Organizations, plans and schedules: An interdisciplinary perspective on coordinating ai agents. *Journal of Intelligent Systems*, *3*(2-4), 157–188.

Durfee, E. H., Lesser, V. R., & Corkill, D. D. (1987). Coherent cooperation among communicating problem solvers. *Computers, IEEE Transactions on*, *100*(11), 1275–1291.

Durfee, E. H., & Montgomery, T. A. (1991). Coordination as distributed search in a hierarchical behavior space. *Systems, Man and Cybernetics, IEEE Transactions on*, *21*(6), 1363–1378.

Esteva, M., Padget, J. A., & Sierra, C. (2001). Formalizing a language for institutions and norms. In *Intelligent Agents VIII*, vol. 2333 of *Lecture Notes in Computer Science*, (pp. 348–366).

Esteva, M., Rosell, B., Rodriguez-Aguilar, J. A., & Arcos, J. L. (2004). Ameli: An agent-based middleware for electronic institutions. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 236–243).

Fox, M., Gerevini, A., Long, D., & Serina, I. (2006). Plan stability: Replanning versus plan repair. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling*, (pp. 212–221).

Fox, M. S. (1981). An organizational view of distributed systems. *Systems, Man and Cybernetics, IEEE Transactions on*, *11*(1), 70–80.

Fox, M. S., Barbuceanu, M., Gruninger, M., & Lin, J. (1998). An organizational ontology for enterprise modeling. In *Simulating organizations*, (pp. 131–152). Cambridge, MA, USA: MIT Press.

Fox, M. S., & Gruninger, M. (1998). Enterprise modeling. *AI Magazine*, *19*(3), 109.

Fox, M. S., & Smith, S. F. (1984). ISISa knowledge-based system for factory scheduling. *Expert Systems*, *1*(1), 25–49.

Galbraith, J. R. (1973). *Designing Complex Organizations*. Addison-Wesley.

Gaston, M., & desJardins, M. (2005). Agent-organized networks for dynamic team formation. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 230–237).

Gauci, M., Chen, J., Li, W., Dodd, T. J., & Groß, R. (2014). Clustering objects with robots that do not compute. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems*, (pp. 421–428).

Ghavamzadeh, M., Mahadevan, S., & Makar, R. (2006). Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, *13*, 197–229.

Goldman, C., & Zilberstein, S. (2004). Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, *22*, 143–174.

Goldman, C. V., & Zilberstein, S. (2003). Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 137–144).

Goldman, C. V., & Zilberstein, S. (2008). Communication-based decomposition mechanisms for decentralized MDPs. *Journal of Artificial Intelligence Research*, *32*, 169–202.

Guestrin, C., Koller, D., & Parr, R. (2001). Multiagent planning with factored MDPs. In *Proceedings of the 14th Conference on Advances in Neural Information Processing Systems*, (pp. 1523–1530).

Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, *19*, 399–468.

Hansen, E., Bernstein, D., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the the 19th National Conference on Artificial Intelligence*, (pp. 709–715).

Hansen, E. A., & Zilberstein, S. (2001a). LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, *129*(1), 35–62.

Hansen, E. A., & Zilberstein, S. (2001b). Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, *126*(1), 139–157.

Hoogendoorn, M. (2007). Adaptation of organizational models for multi-agent systems based on max flow networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, (pp. 1321–1326).

Horling, B., Benyo, B., & Lesser, V. (2001). Using self-diagnosis to adapt organizational structures. In *Proceedings of the 5th International Conference on Autonomous Agents*, (pp. 529–536).

Horling, B., & Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, *19*(04), 281–316.

Horling, B., & Lesser, V. (2008). Using quantitative models to search for appropriate organizational designs. *Autonomous Agents and Multi-Agent Systems*, *16*(2), 95–149.

Howard, R. (1971). *Dynamic Probabilistic Systems: Semi-Markov and decision processes*, vol. 2. John Wiley & Sons.

Hu, J., & Wellman, M. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, (pp. 242–250).

Hübner, J., Sichman, J., & Boissier, O. (2007). Developing organised multiagent systems using the MOISE$^+$ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, *1*(3), 370–395.

Hübner, J. F., Boissier, O., Kitio, R., & Ricci, A. (2010). Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multiagent Systems*, *20*(3), 369–400.

Hübner, J. F., Sichman, J. S., & Boissier, O. (2005). S-MOISE+: A middleware for developing organised multi-agent systems. In *Proceedings of the AAMAS05 Workshop on Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, (pp. 64–77).

Iba, G. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, *3*(4), 285–317.

IBM (2012). IBM ILOG CPLEX. See http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

Ishida, T., Gasser, L., & Yokoo, M. (1992). Organization self-design of distributed production systems. *Knowledge and Data Engineering, IEEE Transactions on*, *4*(2), 123–134.

Kallenberg, L. C. M. (1983). *Linear Programming and Finite Markovian Control*. Mathematical Centre Tracts.

Koenig, S., & Likhachev, M. (2002). D* Lite. In *Proceedings of the 18th National Conference on Artificial Intelligence*, (pp. 476–483).

Kota, R., Gibbins, N., & Jennings, N. (2009). Self-organising agent organisations. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 797–804).

Krogt, R. v. d. (2005). *Plan repair in single-agent and multi-agent systems*. Ph.D. thesis, Netherlands TRAIL Research School.

Lau, Q. P., Lee, M. L., & Hsu, W. (2012). Coordination guided reinforcement learning. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 215–222).

Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., Prasad, M. N., Raja, A., et al. (2004). Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, *9*(1-2), 87–143.

Li, L., Walsh, T. J., & Littman, M. L. (2006). Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*, (pp. 531–539).

Littman, M., Dean, T., & Kaelbling, L. (1995). On the complexity of solving Markov decision problems. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*, (pp. 394–402).

McGovern, E. (2002). *Autonomous Discovery of Temporal Abstractions from Interaction with an Environment*. Ph.D. thesis, University of Massachusetts Amherst.

Musliner, D. J., Carciofini, J., Durfee, E. H., Wu, J., Goldman, R. P., & Boddy, M. S. (2007). Flexibly integrating deliberation and execution in decision-theoretic agents. In *Proceedings of the 3rd Workshop on Planning and Plan Execution for Real-World Systems (held in conjunction with ICAPS-07)*.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D., & Marsella, S. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, vol. 18, (pp. 705–711).

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, (pp. 278–287).

Oliehoek, F. A., Whiteson, S., & Spaan, M. T. J. (2013). Approximate solutions for factored Dec-POMDPs with many agents. In *Proceedings of the 12th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 563–570).

Oliehoek, F. A., Witwicki, S. J., & Kaelbling, L. P. (2012). Influence-based abstraction for multiagent systems. In *Proceedings of the 26th Conference on Artificial Intelligence*, (pp. 1422–1428).

Omicini, A. (2001). SODA: Societies and infrastructures in the analysis and design of agent-based systems. In *Agent-Oriented Software Engineering*, vol. 1957, (pp. 311–326).

Pacheco, O., & Carmo, J. (2003). A role based model for the normative specification of organized collective agency and agents interaction. *Autonomous Agents and Multi-Agent Systems*, *6*(2), 145–184.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Papadimitriou, C., & Tsitsiklis, J. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, *12*(3), 441–450.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, (pp. 3–12). IEEE.

Parr, R., & Russell, S. (1998). Reinforcement learning with hierarchies of machines. *Proceedings of the 11th Conference on Advances in Neural Information Processing Systems*, (pp. 1043–1049).

Poupart, P., & Boutilier, C. (2003). Bounded finite state controllers. In *Proceedings of the 16th Conference on Advances in Neural Information Processing Systems*, (pp. 823–830).

Pynadath, D. V., & Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, (pp. 389–423).

Pynadath, D. V., & Tambe, M. (2003). An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems*, *7*(1-2), 71–100.

Raja, A., & Lesser, V. (2007). A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, *15*(2), 147–196.

Rao, A. S., & Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. *Knowledge Review*, *91*, 473–484.

Rivkin, J. W., & Siggelkow, N. (2003). Balancing search and stability: Interdependencies among elements of organizational design. *Management Science*, *49*(3), 290–311.

Roy, N., Gordon, G. J., & Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, *23*, 1–40.

Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd ed.

Sandholm, T. (1993). An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, vol. 93, (pp. 256–262).

Sardina, S., de Silva, L., & Padgham, L. (2006). Hierarchical planning in BDI agent programming languages: A formal approach. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 1001–1008).

Shoham, Y., & Tennenholtz, M. (1995). On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, *73*(1-2), 231–252.

Sims, M., Corkill, D., & Lesser, V. (2008). Automated organization design for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, *16*(2), 151–185.

Sims, M., Goldman, C. V., & Lesser, V. (2003). Self-organization through bottom-up coalition formation. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 867–874).

Sleight, J., & Durfee, E. H. (2013). Organizational design principles and techniques for decision-theoretic agents. In *Proceedings of the 12th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 463–470).

Sleight, J., & Durfee, E. H. (2014). Multiagent metareasoning through organizational design. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*.

Smallwood, R., & Sondik, E. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, *21*(5), 1071–1088.

So, Y. p., & Durfee, E. H. (1998). Designing organizations for computational agents. In *Simulating Organizations*, (pp. 47–64). Cambridge, MA, USA: MIT Press.

Stolle, M., & Precup, D. (2002). Learning options in reinforcement learning. In *Lecture Notes in Computer Science*, (pp. 212–223).

Stone, P., Kaminka, G., Kraus, S., & Rosenschein, J. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, (pp. 1504–1509).

Stone, P., Sutton, R., & Kuhlmann, G. (2005). Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, *13*(3), 165–188.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th Conference on Advances in Neural Information Processing Systems*, (pp. 1057–1063).

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Sycara, K., Paolucci, M., Van Velsen, M., & Giampapa, J. (2003). The RETSINA MAS infrastructure. *Autonomous Agents and Multi-Agent Systems*, *7*(1-2), 29–48.

Szer, D., Charpillet, F., & Zilberstein, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, (pp. 576–583).

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, *7*, 81–124.

Varakantham, P., Kwak, J., Taylor, M., Marecki, J., Scerri, P., & Tambe, M. (2009). Exploiting coordination locales in distributed POMDPs via social model shaping. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*, (pp. 313–320).

Vázquez-Salceda, J., Dignum, V., & Dignum, F. (2005). Organizing multiagent systems. *Autonomous Agents and Multiagent Systems*, *11*, 307–360.

Velagapudi, P., Varakantham, P., Sycara, K., & Scerri, P. (2011). Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 955–962).

Whiteson, S., & Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, *7*, 877–917.

Witwicki, S. J. (2010). *Abstracting Influences for Efficient Multiagent Coordination Under Uncertainty*. Ph.D. thesis, University of Michigan.

Witwicki, S. J., & Durfee, E. H. (2011). Towards a unifying characterization for quantifying weak coupling in Dec-POMDPs. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 29–36).

Witwicki, S. J., Oliehoek, F. A., & Kaelbling, L. P. (2012). Heuristic search of multiagent influence space. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 973–980).

Wolpert, D., & Tumer, K. (2001). Optimal payoff functions for members of collectives. *Advances in Complex Systems*, *4*(2/3), 265–279.

Wood, R., & Bandura, A. (1989). Social cognitive theory of organizational management. *The Academy of Management Review*, *14*(3), 361–384.

Wooldridge, M., Jennings, N., & Kinny, D. (2000). The GAIA methodology for agent-oriented analysis and design. *Autonomous Agents and Multiagent Systems*, *3*(3), 285–312.

Wooldridge, M. J. (2000). *Reasoning About Rational Agents*. MIT press.

Wu, J., & Durfee, E. H. (2007). Solving large TAEMS problems efficiently by selective exploration and decomposition. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 56–64).

Zhang, C. (2011). *Scaling multi-agent learning in complex environments*. Ph.D. thesis, University of Massachusetts Amherst.

Zhang, C., Lesser, V., & Abdallah, S. (2010). Self-organization for coordinating decentralized reinforcement learning. In *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multiagent Systems*, (pp. 739–746).