Local Multiagent Control in Large Factored Planning Problems

by

Philipp Robbel

Master of Science, Massachusetts Institute of Technology (2007)
M.Sc., University of Edinburgh (2005)

Submitted to the Program in Media Arts and Sciences
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2016

Author_____
Program in Media Arts and Sciences
November 11, 2015

Certified by_____
Cynthia Breazeal
Associate Professor, MIT Program in Media Arts and Sciences
Thesis Supervisor

Certified by_____
Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
MIT Department of Aeronautics and Astronautics

Certified by_____
Mykel J. Kochenderfer
Assistant Professor of Aeronautics and Astronautics
Stanford University

Accepted by_____
Pattie Maes
Professor of Media Technology
Academic Head, MIT Program in Media Arts and Sciences

Abstract

Local Multiagent Control in Large Factored Planning Problems

by
Philipp Robbel

Submitted to the Program in Media Arts and Sciences
on November 11, 2015 in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at the Massachusetts Institute of Technology

Many problems of economic and societal interest in today's world involve tasks that are inherently distributed in nature. Whether it be the efficient control of robotic warehouses or delivery drones, distributed computing in the Internet of Things, or battling a disease outbreak in a city, they all share a common setting where multiple agents collaborate to jointly solve a larger task. The ability to quickly find effective solutions in such multiagent systems (MASs) forms an important prerequisite for enabling applications that require flexibility to changes in tasks or availability of agents.

This thesis contributes to the understanding and efficient exploitation of *locality* for the solution of general, cooperative multiagent Markov Decision Processes (MDPs). To achieve this, the proposed approximation architectures assume that the solution of the overall system can be represented with sparsely interacting (*i.e.*, local) value function components that—if found—approximate the global solution well. Locality takes on multiple interpretations, from its spatial sense to more general sparse interactions between subsets of agents, and the efficient representation of local effects in large planning problems.

Developed in the thesis are computational methods for extracting sparse agent coordination structure automatically in general, cooperative MDPs. Based on novel theoretical insights about factored value functions, the proposed algorithms automate the search for coordination via principled basis expansion in the approximate linear program (ALP). We show that the search maintains bounded solutions with respect to the optimal solution and that the bound improves monotonically.

Introduced then are novel solution methods that exploit "anonymous influence" in a particular class of factored MDPs. We show how anonymity can lead to representational and computational efficiencies, both for general variable elimination in a factor graph but also for the ALP solution to factored MDPs. The latter allows to scale linear programming to MDPs that were previously unsolvable.

Complex MAS applications require a principled trade-off between complexity in agent coordination and solution quality. The thesis results enable bounded approximate solutions to large multiagent control problems—*e.g.*, disease control with up to 50 agents in graphs with 100 nodes—for which previously only empirical results were reported.

Thesis Supervisor: Cynthia Breazeal
Title: Associate Professor of Media Arts and Sciences

Acknowledgements

There are countless people whom I have to thank for their direct and indirect support during my graduate studies at MIT.

First and foremost my gratitude goes to my thesis advisor and committee members, Prof. Cynthia Breazeal, Prof. Jonathan How, and Prof. Mykel Kochenderfer. I have been extremely lucky to work in *Cynthia*'s research group throughout graduate school. Cynthia accepted me into a group of wonderful labmates and kept everyone's research interests aligned. I am deeply thankful for the support over all these years, and in particular for her giving me the freedom to explore this research in multiagent planning in depth. My deep gratitude goes to *Jon* for his thoughtful comments, academic and non-academic guidance, for hosting me at ACL during my graduate career, as well as his pointed and realistic feedback about research content and timeline. I benefited tremendously from Jon's support and the collaboration with his research group at MIT—from the early MURI days to the last thesis year in particular. It is fair to say that without *Mykel* this thesis would not have happened. I owe him huge thanks for accepting me in his lab as a visitor and for giving my research direction in the times when it mattered most. His support and constructive feedback, along with the productive embedding in his research group at Stanford, made this thesis possible. Thank you, Mykel, for always making time to meet, for offering your personal advice as well, and for your detailed comments on various paper and thesis drafts.

My long-time collaborator and advisor, Frans Oliehoek, is the second person without whom this thesis would have been impossible. *Frans* has had a great influence on me and shaped my views on how to conduct research and how to distill and formulate thoughts crisply. His demands for constant progress were crucial in moving forward with my thesis research. I will miss our 7AM cross-continent research meetings.

My research has taken me to three labs over the years. I thank everyone at PRG for many insightful discussions. I thank ACL for kindly hosting me and the continued friendship with many alumni: Aditya Undurti, Alborz Geramifard, Luke Johnson, Kemal Ure, Georges Aoude, to name just a few. I thank SISL, in particular, for the priviledge to work with and get to know all of you. Everyone's optimisim, research interests, and GSD spirit helped my thesis progress tremendously.

My time in Cambridge and Boston has been some of the best of my life. This is largely due to my friends at MIT and (now) beyond: Alexis Turjman, Siggi Örn, Pankaj Sarin, Maria Carcolé, Craig Bonnoit, and many more. These friendships and shared experiences have defined my life inside and outside of graduate school over more than half a decade. Alexis has had a profound permanent positive impact on my life. I miss our regular hang-outs, be it at SidPac or 'downtown', our discussions on entrepreneurship and the other important topics in life. I thank Siggi for his close friendship, for being an inspiring colleague inside and outside of academia, and for his close support during and in particular the final stretches of the PhD. Pankaj has been a long-time friend throughout graduate school who always supported me unconditionally. I cherish our friendship. I thank Maria for the support over the last months in particular; her views and logical thought strengthened me in times of uncertainty and turmoil. Craig tought me that academic accomplishment is never enough and only a small aspect to a life well lived.

Thanks are also due to my colleagues in the Autonomous Driving group at Bosch, in particular Jan Becker and Jeff Johnson, for their advice and support during the final months of

the thesis. A kind acknowledgement also goes to Eminem for his 'rhymes that help get people through tough times'.

Last but not least, no one could be blessed with more wonderful parents and family. I thank them for their unconditional love and tireless support. This thesis is dedicated to them.

Contents

Although this may seem a paradox, all exact science is dominated by the idea of approximation.

BERTRAND RUSSELL (1872–1970)

Chapter 1

Introduction

Many problems of economic and societal interest in today's world involve tasks that are inherently distributed in nature. Whether it be the efficient control of robotic warehouses or delivery drones, distributed computing in the Internet of Things, or battling a disease outbreak in a city, they all share a common setting where multiple, distributed *agents* collaborate to jointly solve a larger task. The ability to quickly find effective solutions in such cooperative *multiagent systems* (MASs) forms an important prerequisite for real-world applications that require robustness and flexibility to changes in tasks or availability of agents.

Efficient solution algorithms that deliver on the full promise of MASs are still in their early stages. In particular, they rely on agent coordination that is predefined by the designer of the MAS, are limited in agent or task numbers, or heuristic in nature without any strong guarantees about their performance during deployment. Lifting these restrictions on problem size and type for general multiagent systems remains a largely unsolved problem. A fundamental reason for this is that large MASs suffer from well-known negative complexity results as problem sizes and agent numbers increase.

One major source of this complexity is due to the number of joint (*i.e.*, *global*) system states and actions which tend to increase exponentially with the number of agents. Consider, *e.g.*, the running example of a disease outbreak where 50 agents have to coordinate (binary) vaccination choices at their respective location in order to inhibit the further spread of the disease. The global action space is then spanned by 2^{50} distinct action choices for the team, and efficient vaccination policies raise complex multiagent planning problems. Unless the particular problem has some structure that may be exploited, solutions in such general, real-world settings will therefore likely remain approximate in nature.

In the decision-theoretic community, models like the Markov decision process (MDP) and its partially observable extensions have both seen widespread use to represent and solve complex

planning problems for single and multiple agents in stochastic worlds. These models provide a formalization of the domain in terms of *states*, *actions*, and *rewards*, and define the stochastic patterns governing the world. An optimal behavior of the agents in a system state follows from the maximum expected sum of rewards that a particular action affords. Determining a (joint) *agent policy* that optimizes this expected sum of rewards in every state is also known as solving the MDP. Value function representations are commonly used intermediaries that store these sums directly for every state or state-action pair in order to allow the principled extraction of agent policies. In general, however, solving an MDP, *e.g*. by obtaining the value function governing the process, suffers from the negative complexity results introduced above—both on the representational side (of storing exponentially many states or state-action pairs) but in particular also on the computational side. Many exact and approximate solution methods therefore attempt to exploit structure in the problem or value function under the assumption that *local interactions* describe the *global state* (or, *solution*) well.

This thesis contributes to the understanding and efficient exploitation of *locality* for the solution of general, cooperative multiagent MDPs, ultimately aiming to expand the size and problem classes for which MASs may be deployed. A fundamental underlying assumption of the thesis is that the global solution of the overall system is amendable to approximation with sparsely interacting (*i.e*., *local*) value function components. Locality takes on multiple interpretations throughout the thesis, from the immediate *spatial* sense to more general sparsity properties of the multiagent planning problem. A key result are computational methods for extracting sparse agent coordination structures in general MASs automatically, while maintaining *bounded solutions* compared to the optimal value function.

Many future applications will depend on efficient decision support systems as complexity of distributed MASs increases. Due to the sheer problem sizes, manual solutions by domain experts are inherently limited in scale; the guiding vision of this thesis is therefore to develop approximation architectures that allow the designer of a MAS a principled trade-off between complexity in agent coordination and solution quality. Our results enable bounded approximate solutions to large multiagent control problems – *e.g*., disease control with up to 50 agents in graphs with 100 nodes – for which previously only empirical results were reported.

1.1 Objectives and Research Questions

The main objective of this thesis is the detailed exploration of *locality* for computing efficient, generally approximate solutions to cooperative multiagent planning problems. These systems

have negative complexity results for the large state *and* action spaces usually associated with many agents. While significant effort has gone into developing theoretical insight and scalable solution methods for large planning problems over the past decade [40, 46, 51, 93], support for large discrete action spaces (as naturally posed by MASs) remains challenging and is the topic of ongoing research [15, 59, 84, 85].

Given the unfavorable complexity results associated with large MASs, many problem representations attempt to exploit structure in the domain for efficiency gains. The factored MDP (FMDP), along with its multiagent extension (FMMDP), provide a scalable method for representing the transition and reward models compactly. This is achieved by making local dependencies in state and action variables explicit in the transition and reward models, *e.g.*, by encoding the transition function as a dynamic Bayesian network [54]. These representational benefits of "locality", however, do not in general translate into gains during policy computation [12, 55]. The first research question therefore addresses the scalability of solution methods in the context of factored multiagent settings:

> *Question #1: Are there novel forms of "locality" that permit scaling FMMDP solution methods for cooperative MASs beyond the state-of-the-art?*

"Locality" is intentionally left general and approached in this thesis from three distinct directions. In its first, most immediate form, locality refers to its *spatial* interpretation, in particular in the context of stochastic task allocation problems modeled as FMMDPs. In these settings, the overall system state is often only affected locally by agents that are in spatial proximity to a task. Our focus is on formalizing a problem class that makes explicit these dependencies and on reviewing existing solution methods that exploit *independence of agent movement* and *locality of tasks* in these models for effective, albeit unbounded approximations.

Locality then takes on a second, more principled interpretation. Of particular interest are characterizations that are *not* domain-specific (*e.g.*, do not require a map or distance metric defined for the problem) and therefore apply to a wider class of factored multiagent planning problems. One such interpretation is a general notion of *locality of interaction* between agents that does not necessarily stem from spatial proximity. *Interaction structures* between agents arise from the (approximate) decomposition of value functions into local components, each defined over subsets of state and action factors. Many exact and approximate solution methods attempt to exploit such factorization of the value function for efficiency during planning [3, 36, 53, 69]. Multiagent settings in particular are known to suffer from an exponential increase in value component sizes as interactions become denser, meaning that approximation architectures are overly

restricted in the size and problem types they may address. One key research direction is therefore how finding and exploiting *sparsity properties* in agent interaction structures can enable approximate solution algorithms for MASs to scale.

Third, locality can refer to the *efficient summarization* of local effects in both single- and multiagent planning problems. Consider again the running example of controlling a disease process over a graph where *summary statistics*, such as the number of infected neighboring nodes, may enable computational efficiencies during planning. Exploring this observation for purposes of scaling up F(M)MDP solution methods defines another research objective spawned by question #1.

> *Question #2: How can interaction structures that specify coordination between cooperative agents in an FMMDP be determined automatically?*

Supporting different interaction structures between agents introduces a burden on the designer of the MAS, who has to decide which choice of agent coordination is effective in a particular problem setting. Because of the complexity of large MASs, the manual specification of such structures is not only unsatisfactory (due to the requirement of a domain expert) but also limited in scale. One research goal is therefore to investigate computational methods for automated coordination discovery between agents.

By making explicit the link between basis function choice and induced coordination structure, *basis selection* or *generation* methods apply in principle [57, 74, 76, 81, 82]. However, multiagent planning methods that scale to large state and action spaces place unique constraints on the basis function form in order to implement sparsity in agent interaction or to impose limits on the value function component sizes. A key research focus is thus the efficient search for basis functions that give rise to locally-scoped value function components that approximate the global value function well. Ideally, this search is conducted in a principled manner by a decision support system that not only draws on empirical evidence but supports performance guarantees of the discovered interaction structures.

> *Question #3: How can bounds on the FMMDP solution resulting from a particular agent interaction structure choice be derived?*

The search for agent coordination is generally driven by a trade-off between the desire for sparsity (*e.g.* due to computational limits in solving the joint agent policy or the goal to minimize agent communication overhead at runtime) and solution quality. Many algorithms address

this trade-off with domain-specific heuristics or implement unbounded approximations to the optimal value function, without strong guarantees on policy performance.

Of particular interest, however, are methods that maintain *bounds* on the value function error to offer stronger guarantees than only empirical evidence. A key research focus therefore falls on domain-*un*specific methods that enable bounded, approximate solutions to large factored planning problems. Bounded solutions will allow the MAS designer to make an educated trade-off between exact performance guarantees and complexity of agent interaction.

1.2 Summary of Contributions

The thesis contributes to the state-of-the-art in solution methods for large multiagent planning problems by:

1. *Formalizing the concept of "locality" in the context of FMMDPs that model spatial task allocation problems.*

   We formalize a general subclass of FMMDPs that model *spatial task allocation problems* (SPATAPs). SPATAPs are characterized by the spatial distribution of tasks over a map, the fact that tasks appear stochastically from exogenous events, and that a cooperative team of agents has to coordinate to address them effectively. The presented model exploits *locality* for avoiding any exponential dependencies on joint states or actions at the representational level. To illustrate the case for how domain-specific algorithms may exploit locality during solution, we review online, distributed planning methods for multiagent teams that resolve coordination in a decentralized fashion to tackle otherwise prohibitively large task assignment problems. All presented models remain unbounded approximations to the original SPATAP. The SPATAP model was co-developed by the thesis author; the reviewed class of subjective approximations are due to the other co-authors in [18].

2. *Providing novel theoretical insights on sparsity in multiagent interaction that enable the* exact *computation and manipulation of the Bellman residual in large FMMDPs.*

   The concept of "locality" is extended beyond the mere spatial sense to address a wider class of FMMDPs. Locality here refers to a general sparsity property in the interaction between agents that does not necessarily stem from spatial proximity. A novel theoretical result shows how the concept of *sparse agent interaction* permits the computation of the exact Bellman residual. An "action-connectivity" property is introduced that provides a sufficient condition for the Bellman residual to remain *factored* in MASs with large state and action spaces.

Based on this insight, efficient computations for Bellman error and marginals over subsets of variables are derived.

3. (Based on 2:) *Developing computational methods for the efficient discovery of interaction structures between agents that i) retain sparsity, ii) iteratively improve on the solution bound to the optimal value function, iii) apply to general FMMDPs.*

   At the core of this contribution are novel computational methods for automatic coordination discovery between agents in large FMMDPs, together with theoretical insights about the implied bound to the optimal solution V^*. A general assumption of the developed approximate methods is that there exists some form of sparse interaction between agents that—if found—allows to approximate the global value function well. Based on this we phrase the search for coordination as basis *selection* (through regularization) and as a principled *basis discovery* method. Previous work introduced Bellman error basis functions (BEBFs) for basis expansion in uncontrolled settings (*e.g.*, *policy evaluation* in reinforcement learning [31, 76]). We adapt this work to the full *policy computation* problem in the context of the approximate linear programming (ALP) solution to FMMDPs. A novel theoretical insight shows which basis function yields a guaranteed reduction of the error bound of the ALP solution in principle (referred to as the BEBF*). To retain *sparsity in agent coordination*, efficient approximations to the BEBF* are developed (using results from contribution 2) and implemented as an iterative basis expansion scheme from least to most complex. Basis discovery is shown to improve the bound to V^* monotonically (in a $\leq$ sense) and alleviates the need of a domain expert to specify a basis. Agent coordination follows as a by-product of achieving a desired bound on the solution to the FMMDP; this permits the designer of a MAS to trade-off complexity in coordination with solution guarantees where warranted. Our evaluation scales bounded solutions to problem sizes for which previously only empirical results were known.

4. *Providing theoretical and algorithmic insights of how "anonymity" in large propositional domains can be exploited during (exact) variable elimination for computational gain and scale.*

   The coordination discovery algorithms from contribution 3 retain the requirement that variable elimination (VE) may be carried out efficiently in the *factor graph* underlying the FMMDP. We present a novel concept of "anonymity" in propositional domains where only *local effects* of a set of variables, rather than their identity, are required to describe the domain exactly (albeit more compactly). This is naturally the case for stochastic propagation models over graphs (*e.g.* in models for disease or forest fire propagation). We show how under enforcement of a *variable consistency* property during elimination, VE computes the identical

result on the compressed factor graph. A novel theoretical result is proven that shows how under the same consistency property, "shattering" into disjoint variable counter scopes can be avoided so that VE retains compact representations during elimination. While related to "generalized counts" in first-order models [68, 89], our methods are novel as they summarize effects of variables that are not necessarily indistinguishable in the problem. In the running example of the disease control graph, nodes have a unique identity based on their distinct connections in the factor graph, but may exert "anonymous influence" together with other nodes on a target variable.

5. (Based on 4:) *Addressing the issue of scale for a class of large FMMDPs that model the control of dynamic processes on graphs, by virtue of exploiting "anonymity" properties in the problem.*

   The concept of "anonymity" from contribution 4 is extended beyond variable elimination to the ALP solution method for factored single- or multiagent MDPs (referred to as the *lifted ALP*). We show how anonymity (*i.e.*, aggregate counts over variable sets) at the *representational level* of the FMMDP translate into a compressed set of constraints that represent the original ALP formulation *exactly*. This algorithmic contribution has a natural application in a class of problems that control stochastic dynamics over large graphs. We show that anonymity properties in this domain translate into computational gains for the ALP solution method that allow to scale linear programming to factored MDPs that were previously *unsolvable* in practice with existing algorithms. Our results are shown for disease control domains over graphs with 50 nodes and dense neighbor connectivity.

1.3 Thesis Structure

Chapter 2 provides the necessary background on factored planning problems and their efficient solution methods. Known theoretical results (*e.g.*, solution bounds) are also shown. Chapter 3 then sets the stage for the general theme of this thesis by exploiting locality (in its immediate, *spatial* sense) for modeling and solving spatial task allocation problems (contribution 1). We then proceed to generalize locality beyond its spatial sense to general forms of *sparse interaction* in FMMDPs. Chapter 4 provides the theoretical foundations for efficient computation and manipulation of the Bellman residual in sparsely "action-connected" MASs (contribution 2). Based on these results, Chapter 5 presents a key contribution of this thesis with computational methods for coordination discovery that maintain bounded solutions (contribution 3). Chapter 6 exploits locality in a class of problems that support efficient representation with "anonymous influence".

We present novel theoretical and computational results for exploiting anonymity during general variable elimination (contribution 4) and the ALP solution to FMMDPs (contribution 5). Chapter 7 discusses our contributions and suggests further work.

Chapter 2

Background

This chapter reviews the fundamental background on decision-theoretic planning that forms the basis for the theoretical insights and solution algorithms developed in the remainder of the thesis. In the decision-theoretic view of the planning problem, a solution corresponds to a *course of action* (or, a policy) that maximizes expected *utility* in a stochastic world. Such problems of planning sequential actions under uncertainty have a natural formulation as a Markov decision process (MDP). In the following, we present a concise overview of MDPs, their factored and multiagent extensions, and introduce their basic exact and approximate solution methods. Special focus is then given to the approximate linear programming solution to multiagent MDPs due to its favorable scale in domains with large (factored) state *and* action spaces. Concluding the chapter is a summary of existing theoretical results for error analysis of the MDP solution. For a more in-depth treatment of these topics, the reader is referred to [50, 83].

2.1 Factored Markov Decision Processes

The Markov decision process framework formalizes the planning problem in terms of *states*, *actions*, and *rewards*, along with the stochastic processes governing the world. Future world states follow a known probability distribution determined by the current world state and the (joint) action of one or multiple decision makers (or, *agents*). Roughly speaking, the goal is to control the system over extended interactions towards desirable states, as described by a numerical reward measure. Solving the planning problem corresponds to computing a (joint) policy that optimizes some function of the rewards (*e.g.*, expected sum or average of rewards when executing the policy in the world).

An important property underlying the MDP is the *Markov assumption* (or *memoryless property*) which defines that the current state alone represents a sufficient statistic for acting opti-

mally in the world. Stated differently, future world states depend only on the current state and the joint action executed by the agents at the current time step. While, in general, the MDP framework supports continuous or infinite state and action spaces, we restrict our attention to the discrete finite-state and finite-action settings with both single and multiple decision makers.

2.1.1 Single-Agent Models

Definition 1. *A Markov decision process (MDP) is defined by the tuple* $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$*, where* $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$ *and* $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$ *are the finite sets of states and actions,* T *the (Markovian) transition probability function specifying* $P(s' \mid s, a)$*,* $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ *the reward function describing the immediate reward for executing an action in a given state, and* $\gamma \in [0, 1]$ *a discount factor.*

Factored MDPs (FMDPs) exploit structure in the state space $\mathcal{S}$ and define system state s by an assignment to the state variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ instead. The global state space is then spanned by the state variables, *i.e.*, $\mathcal{S} = X_i \times \ldots \times X_n$. Transition and reward function decompose into a two-slice temporal Bayesian network (2TBN) consisting of independent factors, each described by their scope-restricted conditional probability distributions (CPDs) [54]. Under a particular action $a \in \mathcal{A}$ the system described by a factored MDP transitions as:

$$P(\mathbf{x}' \mid \mathbf{x}, a) = \prod_i P(x'_i \mid \mathbf{x}[\mathrm{Pa}(X'_i)], a) \tag{2.1}$$

where $\mathrm{Pa}(X'_i)$ denote the parent nodes of X'_i in the 2TBN and the term $\mathbf{x}[\mathrm{Pa}(X'_i)]$ refers to the value of the parent variables extracted from the current state $\mathbf{x}$. A similar (additive) factorization holds for the reward function given state $\mathbf{x}$ and action a:

$$R(\mathbf{x}, a) = \sum_{i=1}^{r} R_i(\mathbf{x}[\mathbf{C}_i], a) \tag{2.2}$$

where each R_i is defined over some subset of state factors $\mathbf{C}_i \subseteq \{X_1, \ldots, X_n\}$. As shown in Figure 2.1, this yields one 2TBN *per action* in the single-agent case. An MDP utilizing factored transition and reward models is called a factored MDP [12].

2.1.2 Collaborative Multiagent Models

Both MDPs and FMDPS have a natural extension to collaborative multiagent settings which are the main focus of this thesis.

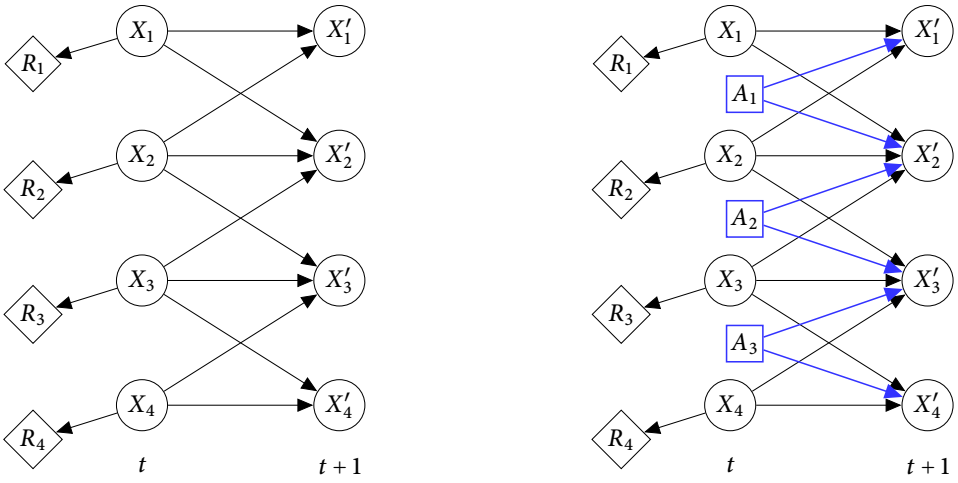

Figure 2.1: *Left:* Representation of transition and reward models of an example factored MDP (FMDP) as a two-slice temporal Bayesian network (2TBN). *Right:* A 2TBN representation of the collaborative factored multiagent MDP (FMMDP) with three agents, where parent node sets $\mathrm{Pa}(X'_i)$ include both state and action variables. Note that in the single-agent case, each action is associated with a unique 2TBN whereas the collaborative multiagent setting is fully described by a single 2TBN which includes action factors with local effects (formally a *dynamic decision network*).

Definition 2. *A Multiagent Markov decision process (MMDP) is defined by the tuple* $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$*, where* $\mathcal{D} = \{1, \ldots, g\}$ *is the set of* g *agents,* $\mathcal{S}$ *a finite set of states* s *of the environment,* $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_g$ *the set of joint actions* $a = \langle a_1, \ldots, a_g \rangle$*,* T *the (Markovian) transition probability function specifying* $P(s' \mid s, a)$*,* $R(s, a)$ *the immediate reward function, and* $\gamma \in [0, 1]$ *a discount factor.*

In the MMDP model, agents observe the full state of the environment, and transition and reward dynamics depend on the *global* state and *joint* actions of all agents. Collaboration arises from the fact that all agents share the identical reward function R.

Multiagent settings suffer from state and action spaces that generally grow exponentially with the number of agents. Representing global transition and reward functions with a tabular encoding is quickly rendered prohibitive in large multiagent domains. *Factored Multiagent MDPs* (FMMDPs) address this challenge by exploiting structure *at the representational level* of the decision process. An MMDP is called *factored* if its state and action spaces are spanned by a set of variables. For simplicity of notation, we assume one action variable per agent such that $\mathbf{A} = \{A_1, \ldots, A_g\}$ and $\mathbf{X} = \{X_1, \ldots, X_n\}$ span joint action and state spaces $\mathcal{A}$ and $\mathcal{S}$, respectively. The decomposition over the action space $\mathcal{A}$ further allows the direct representation of the *"locality of interaction"* that commonly arises in many realistic multiagent settings [39].

In particular, in the class of *collaborative FMMDPs* underlying this thesis, action variables may only affect a subset of state variables at the next time step [36]. By exploiting this decomposition of (generally exponential) state and action spaces, the collaborative FMMDP yields a tractable representation that introduces action variables into the 2TBN (see Figure 2.1). The global transition function experiences a factorization over local terms that depend only on subsets of state and action variables:

$$P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) = \prod_i P(x'_i \mid \mathbf{x}[\mathrm{Pa}(X'_i)], \mathbf{a}[\mathrm{Pa}(X'_i)]) \tag{2.3}$$

where $\mathrm{Pa}(X'_i)$ now include state *and* action variables and each local CPD is only defined over their relevant subsets. Collaborative FMMDPs further assume that each agent observes part of the global reward and is associated with (restricted scope) local reward function R_i, such that the global reward factors additively as:

$$R(\mathbf{x}, \mathbf{a}) = \sum_{i=1}^{g} R_i(\mathbf{x}[\mathbf{C}_i], \mathbf{a}[\mathbf{D}_i]) \tag{2.4}$$

for some subsets of state and action factors $\mathbf{C}_i$ and $\mathbf{D}_i$, respectively.

Additional representational efficiencies may be realized by exploiting other forms of structure that possibly exist in the decision process. Context-specific independence in the model, for example, can be exploited by choosing decision trees or algebraic decision diagrams for compactly encoding CPDs and reward functions [46].

2.2 Overview of Solution Methods

In this section we provide a concise treatment of relevant exact and approximate solution methods for (factored) MDPs. We begin by defining the single- or multiagent *control problem* as determining a (joint) policy π that optimizes some optimality criterion involving the rewards. Addressing the control problem is equivalently referred to as *solving* the decision process or the planning problem.

A fixed, *deterministic policy* $\pi : \mathcal{S} \mapsto \mathcal{A}$ is a function that maps each state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. For all solution methods considered in this thesis—apart from those in Chapter 3—we consider the expected *infinite-horizon discounted return* as the optimality criterion, defined by:

$$E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{2.5}$$

where E_π is the expectation of the sum of exponentially discounted rewards obtained by executing policy π in the stochastic world and $\gamma \in [0,1)$ ensures that this sum is finite. Chapter 3 considers an alternative criterion, $E_\pi\left[\sum_{t=0}^{h} r_t\right]$, commonly referred to as the expected *undiscounted return* obtained over finite (planning) horizon h. An *optimal policy* π^* maximizes the selected optimality criterion for any system state $s \in \mathcal{S}$ and returns the action that achieves the maximum expected return in a given state. A well-known result states that for any MDP there exists at least one optimal policy that is fixed and deterministic [83]. We will therefore only consider such policies in this thesis.

In cooperative multiagent settings, $\pi : \mathcal{S} \mapsto \mathcal{A}$ is the *joint policy* for the entire agent team, defined over joint states $s \in \mathcal{S}$ and joint actions $a \in \mathcal{A}$ of the MMDP, and can be written as:

$$\pi = \langle \pi_1, \ldots, \pi_g \rangle \tag{2.6}$$

where $\pi_i : \mathcal{S} \mapsto \mathcal{A}_i$ denotes the local policy of agent i, still defined over *joint states* $s \in \mathcal{S}$. Since the reward function is shared by all agents, π optimizes the total payoff achieved jointly by all agents. In all but degenerate cases, a good joint policy has to take the effects of agents on each other into account to achieve high return in the domain.

Solution methods for the MDP planning problem can be divided along multiple axes. One may broadly distinguish between exact and approximate solutions and within each class further separate algorithms that use *value functions* from those that use other representations of the solution. Another division is between online optimization for the current state only versus offline computation of the solution for all states $s \in \mathcal{S}$. The focus of this thesis is on exact and approximate *value-based* methods and our review considers common solution methods in this class. Complementary reviews of other methods, such as search in (restricted) policy space, are *e.g.* covered in [50]. Throughout our discussion, the focus is on the *planning problem* where models of the decision process are available; an introduction to the field of *reinforcement learning (RL)* where accurate models are generally unavailable can be found in [97].

2.2.1 Value Functions and Bellman Equations

Value functions provide a link between optimality criterion and solution to the decision process. They directly store the expected return for every state or state-action pair, thereby allowing the principled extraction of *agent policies*. In the following, we assume the infinite-horizon discounted optimality criterion from Equation 2.5 and generally use factored state and action spaces in our presentation.

The *(state) value function* V^π associated with policy π records the expected return when starting in state $\mathbf{x}$ and following π after:

$$V^\pi(\mathbf{x}) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \mathbf{x}_0 = \mathbf{x}\right]. \tag{2.7}$$

Value functions possess certain recursive properties. Formally, $V^\pi(\mathbf{x})$ is the unique fixed point to the *Bellman equation*:

$$V^\pi(\mathbf{x}) = R(\mathbf{x}, \pi(\mathbf{x})) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \pi(\mathbf{x}))V^\pi(\mathbf{x}'). \tag{2.8}$$

One may similarly consider the *state-action value function* Q^π which records the expected return when executing action $\mathbf{a}$ in $\mathbf{x}$ and following π thereafter:

$$Q^\pi(\mathbf{x}, \mathbf{a}) = R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a})V^\pi(\mathbf{x}). \tag{2.9}$$

The goal of solving an MDP is to find the best solution, *i.e.* an *optimal policy* π^* that maximizes the return from every state:

$$\pi^*(\mathbf{x}) \triangleq \operatorname*{argmax}_{\mathbf{a}} Q^{\pi^*}(\mathbf{x}, \mathbf{a}) \quad \forall \mathbf{x} \in \mathbf{X}. \tag{2.10}$$

The optimal value function $V^{\pi^*} \triangleq \max_{\mathbf{a}} Q^{\pi^*}(\mathbf{x}, \mathbf{a})$ corresponding to π^* can be shown to be the fixed point to the *Bellman operator* $\mathcal{T}^*$:

Definition 3 (Bellman operator)**.** *The Bellman operator $\mathcal{T}^*$ for any value function V is defined as:*

$$\mathcal{T}^* V(\mathbf{x}) \triangleq \max_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a})V(\mathbf{x}')\right] \tag{2.11}$$

The optimal value function V^{π^*} is the unique fixed point to $\mathcal{T}^*$, i.e.*, $V^{\pi^*} = \mathcal{T}^* V^{\pi^*}$.*

Both V^{π^*} and Q^{π^*} are commonly written as V^* and Q^*. An optimal policy π^* acts *greedily* with respect to an optimal value function. In the state value function case, acting greedily requires a one-step look-ahead to consider all possible transitions from the current state:

$$\pi^*(\mathbf{x}) = \text{Greedy}(V^*)(\mathbf{x}) \triangleq \operatorname*{argmax}_{\mathbf{a}} \left[R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a})V^*(\mathbf{x}')\right]. \tag{2.12}$$

State-action value functions are useful since the greedy policy $\text{Greedy}(Q^*)$ follows directly from

Equation 2.10 without the additional enumeration over successor states. It is important to note that, in general, structure in the transition or reward functions of a factored (M)MDP *does not imply* structure in the value function since "scopes of influence" of state and action factors tend to grow as the 2TBN is unrolled over time [55].

2.2.2 Exact Solution Methods

The Bellman operator $\mathcal{T}^*$ in Equation 2.11 corresponds to a system of non-linear equations ($|\mathcal{S}|$ equations in $|\mathcal{S}|$ unknowns) and can be solved as such in principle. Commonly used in practice are methods that employ linear programming or are implemented as *iterative*, dynamic programming-based algorithms. The *issue of scale* has to be considered for large (*e.g.*, multiagent) planning problems since exact solutions share polynomial (worst-case) time complexity in states and actions [12]. More fundamentally, mere *representation* of the policy for exponential state spaces is generally prohibitive giving rise to the approximate solutions reviewed in the following section.

Linear Programming

The (exact) linear programming (LP) solution to MDPs implements the non-linear Bellman operator with $|\mathcal{S}|$ variables—one per state value $V^*(\mathbf{x})$—and $|\mathcal{S}||\mathcal{A}|$ linear constraints [83]:

$$\begin{aligned} \min_{V^*(\mathbf{x})} \quad & \textstyle\sum_{\mathbf{x}} \alpha(\mathbf{x}) V^*(\mathbf{x}) \\ \text{s.t.} \quad & V^*(\mathbf{x}) \geq \quad R(\mathbf{x},\mathbf{a}) + \gamma \textstyle\sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^*(\mathbf{x}') \quad \forall \mathbf{x},\mathbf{a} \end{aligned} \tag{2.13}$$

where all variables $V^*(\mathbf{x})$ are unbounded. Note that each max in Equation 2.11 is transformed into $|\mathcal{A}|$ linear *"greater than or equal"* constraints in the exact LP. The optimal solution is computed for any choice of positive *state relevance weights* $\alpha(\mathbf{x})$.

Value and Policy Iteration

Value and policy iteration are iterative, dynamic programming (DP)-based solution methods to the planning problem.

Policy iteration starts with an (arbitrary) initial policy π^0 and iterates *policy evaluation* (computing V^{π^k}) and *policy improvement* (assigning $\pi^{k+1} = \text{Greedy}(V^{\pi^k})$) until convergence, *i.e.*, until $\pi^{k+1} = \pi^k$. Policy iteration can be shown to converge to the optimal policy π^* from any initial policy π^0 [83]. *Value iteration* uses the Bellman operator $\mathcal{T}^*$ directly and turns the right-hand

side of Equation 2.11 into an update rule that is iterated until the fixed point is achieved. Similarly to policy iteration, value iteration can be shown to converge to the optimal value function V^* in the limit from any initial value function choice V^0. The optimal policy π^* is then obtained as Greedy(V^*).

Structured DP

Representational benefits, *e.g.* in factored MDPs, do not in general translate into gains during policy computation [55]. Structured DP approaches are versions of classical value and policy iteration that assume structure *also in the solution* (*i.e.*, at the policy or value function level). In practice, descriptions such as trees or rules are used for compact representation and efficient operators that directly manipulate these structures form the basis of a class of *structured* (or *symbolic*) solution algorithms.

Compact descriptions can be viewed as implementing a form of *state aggregation* where multiple states with similar value or policy outcome are summarized within a particular rule or branch of the tree. Particularly compact are algebraic decision diagrams (ADDs) which correspond to tree structures that can share subtrees. The SPUDD algorithm operates directly on ADDs and has demonstrated scalability to domains with large factored state spaces [46].

Exact structured approaches suffer from a lack of guarantees that the representation of the solution remains compact. Additionally, they generally do not address large structured action spaces in multiagent settings (notable exceptions being [84, 85]).

2.2.3 Approximate Solution Methods

Exact solution methods are quickly rendered intractable as the sizes of state and action spaces increase. Approximate methods are therefore fundamental to scale solutions to general multiagent planning problems that underlie this thesis.

A commonly employed class of approximate solution methods is (parametric) *value function approximation* which fixes the structural description of the value function and reduces the planning problem to one of parameter estimation. Unlike the exact structural DP methods introduced previously, the value function is therefore guaranteed not to grow without limit and to retain a tractable representation stemming from the choice of the function approximator. This thesis focuses in particular on the *linear family* of value function approximators which has favorable theoretical guarantees, efficient solution methods for large multiagent settings, and is well-validated in practice [36, 97].

In recent years, online, sample-based planning methods that repeatedly estimate the value for the current system state have also demonstrated scalability to domains with large state spaces [48, 51]. They reduce planning complexity by sampling from the model of the decision process, avoiding the exhaustive enumeration of the state space. Sample-based planning has also been joined successfully with value function approximation in hybrid architectures (*e.g.*, [93]).

Linear Function Approximation

A particular class of *linear* value functions consists of those that can be written as a linear combination of pre-specified (possibly non-linear) basis functions.

Definition 4 (Linear value function). *A linear value function V given basis function choice $h_1, \ldots, h_k$ and associated parameters $w_1, \ldots, w_k$ can be written as:*

$$V(\mathbf{x}) = \sum_i w_i h_i(\mathbf{x}). \tag{2.14}$$

A linear value function can also conveniently be expressed in vector notation as $V = \mathbf{Hw}$ for parameter vector $\mathbf{w} = (w_1 \;\; w_2 \;\; \ldots \;\; w_k)^T$ and $\mathbf{H}$, the $|\mathbf{X}| \cdot k$ matrix of basis functions, defined as:

$$\mathbf{H} = \begin{pmatrix} | & | & & | \\ \mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{F}_k \\ | & | & & | \end{pmatrix} = \begin{pmatrix} — & \mathbf{f}_1^T & — \\ & \vdots & \\ — & \mathbf{f}_{|\mathbf{X}|}^T & — \end{pmatrix}.$$

Here, basis functions h_i span the columns of matrix $\mathbf{H}$, *i.e.* $h_i \triangleq \mathbf{F}_i$, so that each column vector has $|\mathbf{X}|$ elements. Equivalently, each row $\mathbf{f}_j^T$ in the matrix denotes the *features* assigned to state $\mathbf{x}_j$ by each of the basis functions $h_1, \ldots, h_k$. In general, $k \ll |\mathbf{X}|$ so that the number of *parameters* defining the value function is reduced compared to the direct, tabular enumeration of state values over the entire state space.

Linear function approximation (LFA) applied to arbitrary value functions approximates the true value function V as a linear one for some choice of $\mathbf{H}$ and $\mathbf{w}$, *i.e.*, $V(\mathbf{x}) \approx \hat{V}(\mathbf{x}) = \mathbf{Hw}$. LFA therefore consists of two separate steps, namely the specification of a suitable basis (or, equivalently, feature set) $\mathbf{H}$, as well as the determination of parameter vector $\mathbf{w}$ that jointly yield a close approximation to the true value function. While $\mathbf{H}$ is frequently assumed given by a domain expert, different methods exist to compute $\mathbf{w}$. Our focus is on the approximate linear programming solution since it has a particularly efficient form even for domains with exponential state *and* action spaces, as will be covered later in Section 2.3.

Approximate Linear Programming

The approximate linear programming (ALP) solution to MDPs computes the best *linear value function approximation* (in a weighted L_1-norm sense) $\hat{V}$ to the optimal value function V^* in the space spanned by the given basis functions $\mathbf{H}$ [83]. For an infinite-horizon discounted MDP and given basis choice $h_1, \ldots, h_k$, the ALP formulation optimizes parameter vector $\mathbf{w}$ as per the following objective and constraints:

$$\begin{array}{lrcll} \min\limits_{\mathbf{w}} & \sum_{\mathbf{x}} \alpha(\mathbf{x}) \sum_i w_i h_i(\mathbf{x}) & & & \\ \text{s.t.} & \sum_i w_i h_i(\mathbf{x}) & \geq & [R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) \sum_i w_i h_i(\mathbf{x}')] & \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\ & w_i & & \text{unbounded} & \forall w_i \end{array} \tag{2.15}$$

Let $\hat{Q}(\mathbf{x},\mathbf{a}) = R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) \hat{V}(\mathbf{x}')$ denote the usual definition of the state-action value function but now including linear state value function *approximation* $\hat{V}$. Then the ALP constraints merely encode the inequalities:

$$\begin{array}{rrcll} & \hat{V}(\mathbf{x}) & \geq & \hat{Q}(\mathbf{x},\mathbf{a}) & \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\ \Rightarrow & \hat{V}(\mathbf{x}) & \geq & \max_{\mathbf{a}} \hat{Q}(\mathbf{x},\mathbf{a}) & \forall \mathbf{x} \in \mathbf{X} \\ \Rightarrow & \hat{V}(\mathbf{x}) & \geq & \mathcal{T}^* \hat{V}(\mathbf{x}) & \forall \mathbf{x} \in \mathbf{X} \end{array} \tag{2.16}$$

for approximate state and state-action value functions $\hat{V} = \mathbf{Hw}$ and $\hat{Q}(\mathbf{x},\mathbf{a})$, respectively. Unlike the case for the exact LP, the solution to the LP in equation 2.15 depends on the chosen state relevance weights $\alpha(\mathbf{x})$, essentially encoding how "tight" the value function approximation should hold at a particular state $\mathbf{x}$ (see, *e.g.*, [28] for details). Analogous to [40], state relevance weights are assumed uniform throughout the thesis.

Lemma 1 (ALP Feasibility)**.** *The ALP in equation 2.15 is guaranteed* feasible *if the constant basis function* $h_0(\mathbf{x}) = 1 \ \forall \mathbf{x} \in \mathbf{X}$ *is included among the basis functions and the reward function* R *is bounded by* $R_{\max}$.

Proof. We show feasibility with one valid solution. Let $w_{i \neq 0} = 0$ and w_0 denote the weight

associated with h_0. The ALP then corresponds to:

$$\begin{array}{lrcll} \min\limits_{w_0} & \sum_{\mathbf{x}} \alpha(\mathbf{x}) w_0 h_0(\mathbf{x}) = w_0 \sum_{\mathbf{x}} \alpha(\mathbf{x}) & & & \\ \text{s.t.} & w_0 & \geq & \left[R(\mathbf{x},\mathbf{a}) + \gamma w_0 \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a})\right] & \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\ \Rightarrow & (1-\gamma) w_0 & \geq & R(\mathbf{x},\mathbf{a}) & \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\ \Rightarrow & w_0 & \geq & R_{\max}/(1-\gamma) & \end{array}$$

It follows that $w_0 = R_{\max}/(1-\gamma)$ and $w_{i\neq 0} = 0$ is a valid solution of the ALP. □

Note, however, that this solution to the value function is hardly a useful one as it assigns the identical (optimistic upper bound) value w_0 to every state $\mathbf{x} \in \mathbf{X}$.

The approximate linear program computes a solution in time polynomial in $\mathcal{S}$ and $\mathcal{A}$ but both are exponentially large for general MASs. In Section 2.3 we return to the ALP and review a particularly efficient constraint generation method without any exponential dependencies in either state or action spaces.

Sample-based Planning

Unlike the solution methods introduced previously, *sample-based planning* does not compute a value-function approximation for the entire state space but instead continuously computes a value estimate for the current state only. In the planning setting considered in this thesis, value estimates are generated from full trajectories (or *roll-outs*) from the current state $\mathbf{x}_0$ by sampling from the given MDP model. Solution methods in this class are therefore also referred to as *simulation-based search*.

One can broadly distinguish between methods that *evaluate* a given policy π (*cf*. Equation 2.7) and those that approximate the *optimal value* (and optimal action $\pi^*(\mathbf{x}_0)$) in the current state. In both cases, simulation-based solutions avoid the exhaustive enumeration of the state space $\mathcal{S}$ (which may be exponentially large) and instead only utilize samples of successor states for value estimation.

Monte Carlo (MC) simulation is one central algorithm for *evaluating* $Q^\pi(\mathbf{x}_0, \mathbf{a})$, the mean return of all simulations in which $\mathbf{a}$ was selected in $\mathbf{x}_0$, under a fixed simulation policy π. Following the notation in [30], the MC equation is given by:

$$Q^\pi(\mathbf{x},\mathbf{a}) = \frac{1}{N(\mathbf{x},\mathbf{a})} \sum_{i=1}^{N(\mathbf{x})} \mathbb{I}_i(\mathbf{x},\mathbf{a}) R_i \tag{2.17}$$

where R_i is the return of the ith simulation, $N(\mathbf{x})$ denotes the total simulations from root $\mathbf{x}$, $\mathbb{I}_i$ is an indicator function returning 1 iff $\mathbf{a}$ was selected in $\mathbf{x}$ during the ith simulation, and $N(\mathbf{x}, \mathbf{a}) = \sum_{i=1}^{N(\mathbf{x})} \mathbb{I}_i(\mathbf{x}, \mathbf{a})$ counts the total number of simulations in which $\mathbf{a}$ was selected in $\mathbf{x}$. Via the Chernoff bound one can derive the number of required *samples per action* such that $Q^\pi(\mathbf{x}_0, \mathbf{a})$ is ϵ-accurate with probability at least $1 - \delta$, independent of the size of the state space. MC does not possess the characteristics of an anytime algorithm that monotonically improves on the estimate: the Q-value function estimate is only accurate in the limit of infinite samples [97].

For *optimal value and policy* estimation, (top-down) Monte Carlo Tree Search (MCTS) and real-time dynamic programming (RTDP) algorithms have enjoyed wide popularity in recent years [7, 48, 51, 92]. MCTS methods use MC simulation to evaluate the nodes of a search tree and continuously improve the simulation policy π based on the tree. In its most basic form, MCTS is *sequentially best-first*, selecting actions greedily within the tree (referred to as the *tree policy*) and randomly for unseen states until completion. After each simulation, each visited node (or, alternatively, only the first unseen node) is added to the tree. A number of methods exist which vary in how they choose actions at each state in their search (an overview can be found in [44]).

A frequently employed criterion for action choice is to focus sampling on the most promising actions based on *upper confidence bounds* on their return value: the UCT algorithm replaces uniform MCTS with a tree policy π_{UCB} that implements this criterion for trading-off exploration and exploitation in the search tree in a principled way [51]:

$$\pi_{\text{UCB}}(\mathbf{x}, d) \triangleq \operatorname*{argmax}_{\mathbf{a}} Q(\mathbf{x}, \mathbf{a}, d) + C_p \sqrt{\frac{\log N(\mathbf{x}, d)}{N(\mathbf{x}, \mathbf{a}, d)}} \tag{2.18}$$

for $Q(\mathbf{x}, \mathbf{a}, d)$, the current MC value of node $(\mathbf{x}, d)$ at depth d in the tree, counters $N(\cdot)$ denoting the respective visit or action counts for that node, and assuming normalized payoffs in $[0, 1]$. The upper confidence bound (UCB) action-selection criterion has a principled motivation in the theory of multi-arm bandits with stochastic rewards. Adapted to trees, UCT has been shown to be a *consistent algorithm*, *i.e.* the probability of selecting a non-optimal action converges to 0 at polynomial rate as the number of trajectories grows to infinity [51]. In the worst case of very delayed rewards, however, the trajectory number until ϵ-optimal behavior is found is hyperexponential in the horizon [19].

In summary, sample-based planning methods can address scale in $|\mathcal{S}|$ but generally require

continuous replanning from every state (although hybrid models that retain "memory" through value function approximation have appeared in the literature [13, 93]). Further, UCT must select *each action* at each node at least once before the tree policy π_{UCT} is well-defined. This is prohibitive in multiagent settings where $|\mathcal{A}|$ is exponential in the number of agents, requiring the development of additional approximations (*e.g.*, [3, 20]).

2.3 Efficient Solutions to Large FMMDPs

As illustrated above, coordinating a team of agents to maximize a shared performance measure (the long-term reward) is challenging because of exponential state and action space sizes. This applies at a computational level but more fundamentally also at a representational level since the exhaustive enumeration of all possible actions (for tabular storage or for retrieving the best action in the current state) is generally infeasible. *Factored value functions*—which are defined over local factors—offer a solution to these challenges and form a core approximation architecture for the complex multiagent systems considered in this thesis.

2.3.1 Factored Value Functions

Value function factorization assumes that smaller, localized value function components approximate the true value function well [36, 53]. This approach represents the global value function as a linear combination of locally-scoped terms, each of which addressing a part of the system and each covering potentially multiple, even overlapping, state factors. Following [36], we formalize the concept of a factored (linear) value function:

Definition 5 (Factored value function)**.** *A linear value function V given basis function choice $h_1, \ldots, h_k$ and associated parameters $w_1, \ldots, w_k$ is called* factored *if each basis function h_i is scope-restricted to some subset of variables. It can therefore be written as:*

$$V(\mathbf{x}[\mathbf{C}]) = \sum_{i=1}^{k} w_i h_i(\mathbf{x}[\mathbf{C}_i]) \tag{2.19}$$

where $\mathbf{C}_i \subseteq \mathbf{X}$ and $\mathbf{C} \subseteq \mathbf{X}$ denote (distinct) subsets of state factors.

Factored value functions address the representational challenge since they are defined over locally-scoped terms. They are further also key to efficient computation: for a given basis choice $h_1(\mathbf{c}_1), \ldots, h_k(\mathbf{c}_k)$, there exists an *approximate solution* method based on the ALP that retains no exponential dependencies in $|\mathcal{S}|$ and $|\mathcal{A}|$ [39, 40]. In the following, when it is clear from the

context that factored value functions are used, we may omit individual function scopes $\mathbf{C}_i$ for clarity in presentation.

2.3.2 Efficient Constraint Generation

Compared to the exact LP, the approximate linear programming (ALP) solution to general (multiagent) MDPs reduces the number of *variables* from $|\mathbf{X}|$ to the size of the basis function set. However, it still retains exponentially many *constraints* from the exhaustive enumeration of state and action spaces (*cf*. Equation 2.15). Guestrin *et al*. devise an efficient method to represent exponentially many constraints that applies if the basis functions have local scope and the transitions and rewards are factored [40]. The first realization is that *backprojections* of the local basis functions through the 2TBN transition model retain local scopes, such that the exhaustive enumeration of successor states $\mathbf{x}'$ in each constraint can be avoided:

$$\begin{aligned}
\hat{V}(\mathbf{x}) &\geq R(\mathbf{x},\mathbf{a}) + \gamma \textstyle\sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) \sum_i w_i h_i(\mathbf{x}'[\mathbf{C}_i]) && \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\
&= R(\mathbf{x},\mathbf{a}) + \gamma \textstyle\sum_i w_i \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) h_i(\mathbf{x}'[\mathbf{C}_i]) && \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\
&= R(\mathbf{x},\mathbf{a}) + \gamma \textstyle\sum_i w_i \sum_{\mathbf{c}_i'} P(\mathbf{c}_i' \mid \mathbf{x},\mathbf{a}) h_i(\mathbf{c}_i') && \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\
&= R(\mathbf{x},\mathbf{a}) + \gamma \textstyle\sum_i w_i g_i(\mathbf{x},\mathbf{a}) && \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A}
\end{aligned} \tag{2.20}$$

where each term possesses local scope so that *backprojections* g_i are computed efficiently. Note that functions g_i are themselves again defined over local scopes which are determined by the parents of variables $\mathbf{C}_i$ in the 2TBN (omitted for clarity).

The second insight is that the exponentially many constraints can be reduced to a *single nonlinear* constraint that, in turn, has an equivalent implementation with a small, *non-exponential* set of linear constraints [40]:

$$\begin{aligned}
& \hat{V}(\mathbf{x}) \geq R(\mathbf{x},\mathbf{a}) + \gamma \textstyle\sum_i w_i g_i(\mathbf{x},\mathbf{a}) && \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\
\Rightarrow \quad & 0 \geq R(\mathbf{x},\mathbf{a}) + \gamma \textstyle\sum_i w_i g_i(\mathbf{x},\mathbf{a}) - \sum_i w_i h_i(\mathbf{x}) && \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\
\Rightarrow \quad & 0 \geq R(\mathbf{x},\mathbf{a}) + \textstyle\sum_i w_i [\gamma g_i(\mathbf{x},\mathbf{a}) - h_i(\mathbf{x})] && \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\
\Rightarrow \quad & 0 \geq \max_{\mathbf{x},\mathbf{a}} \Big[\textstyle\sum_r R_r(\mathbf{x},\mathbf{a}) + \sum_i w_i [\gamma g_i(\mathbf{x},\mathbf{a}) - h_i(\mathbf{x})] \Big]
\end{aligned} \tag{2.21}$$

Every term on the right-hand side in the last row of Equation 2.21 has *local scope* (not shown) so that the maximization can be performed efficiently, akin to variable elimination in Bayesian networks. It follows that the non-linear max constraint has an exact implementation with a small set of linear constraints, removing any remaining dependencies on the exponential state

and action spaces from the linear program [36]. The total number of resulting linear constraints is only exponential in the *largest clique* formed during variable elimination.

Representation in General Form

Popular linear programming references (*e.g.*, [10]) commonly represent LPs in a particular matrix representation, referred to as the *general LP form*. It is instructive to show how the ALP with factored value functions can be represented using the same notation. We first note that efficient constraint generation via the max constraint is merely for computational reasons and that the result is equivalent to the (still exponential set of) constraints in row 3 of equation 2.21. The latter ones can immediately be translated to the general LP form given by:

$$\mathbf{A}\mathbf{w} \quad \geq \quad \mathbf{b}. \tag{2.22}$$

Let $\langle \mathbf{x}_k, \mathbf{a}_j \rangle$ denote a particular row index of the constraint matrix $\mathbf{A}$, corresponding to the constraint for state $\mathbf{x}_k$ and action $\mathbf{a}_j$. Then, for the ALP with factored value functions the rows of $\mathbf{A}$ can be written as:

$$\mathbf{a}^T_{\langle \mathbf{x}_k, \mathbf{a}_j \rangle} = (\underbrace{[h_0(\mathbf{x}_k) - \gamma g_0(\mathbf{x}_k, \mathbf{a}_j)]}_{\mathbf{f}_0(\mathbf{x}_k, \mathbf{a}_j)}, \underbrace{[h_1(\mathbf{x}_k) - \gamma g_1(\mathbf{x}_k, \mathbf{a}_j)]}_{\mathbf{f}_1(\mathbf{x}_k, \mathbf{a}_j)}, \ldots, \underbrace{[h_N(\mathbf{x}_k) - \gamma g_N(\mathbf{x}_k, \mathbf{a}_j)]}_{\mathbf{f}_N(\mathbf{x}_k, \mathbf{a}_j)}) \tag{2.23}$$

The LP basis is spanned by the columns of matrix $\mathbf{A}$, *i.e.*, $h_j = (\mathbf{f}_j(\mathbf{x}_0, \mathbf{a}_0) \quad \ldots \quad \mathbf{f}_j(\mathbf{x}_M, \mathbf{a}_L))^T$ for M and L total states and actions, respectively. The target vector $\mathbf{b}$ follows as $R(\mathbf{x}, \mathbf{a})$.

2.4 Solution Bounds

This section provides a brief overview of relevant bounds for the approximate linear programming solution and—more generally—any value function for which the *Bellman Error* can be computed. These bounds establish a limit on the error that a value function (or policy) approximation can have with respect to the *optimal solution*.

In our presentation we denote the optimal value function by V^* and its approximation by $\hat{V}$. Approximations that are based on *linear value functions* are also written as $\hat{V} = V_{\hat{\mathbf{w}}} = \mathbf{H}\hat{\mathbf{w}}$. We begin by outlining *a priori* bounds that illustrate the effect of basis choice $\mathbf{H}$ and are ALP specific, and follow up with *a posteriori*, Bellman Error-based bounds that extend to general value functions.

2.4.1 ALP Error Bounds

For a given basis choice $\mathbf{H}$ and state-relevance weights $\alpha(\mathbf{x})$, the ALP computes solution vector $\hat{\mathbf{w}}$ such that the approximation error to the optimal value function is minimized in a weighted L_1-norm sense [28]:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\| V^* - V_{\mathbf{w}} \right\|_{1,\alpha} \tag{2.24}$$

where the weighted L_1-norm of a vector is defined as $\|\mathbf{x}\|_{1,\alpha} \triangleq \sum_{i=1}^{N} \alpha_i |x_i|$. Under this norm, the ALP solution in Equation 2.24 yields the "best" approximation in the space *spanned by* $\mathbf{H}$ for the chosen state relevance weights $\alpha(\mathbf{x})$. From the same Equation it can be seen that if the span of $\mathbf{H}$ includes V^*, it will be the ALP solution. It is further interesting to note that for any fixed state-relevance weights, the addition of basis functions to the ALP *can never worsen the solution* $V_{\hat{\mathbf{w}}}$ (in the norm $\|\cdot\|_{1,\alpha}$).

Of general interest is the distance to the optimal value function V^* in a L_∞-norm sense, *i.e.*, the *maximum discrepancy* between value function estimate and the optimal value function. The ALP result can be related to this understanding of "best possible" approximation to V^* (in the norm $\|\cdot\|_\infty$) by the following bound [28]:

$$\underbrace{\left\| V^* - \hat{V} \right\|_{1,\alpha}}_{\text{"Error resulting from ALP"}} \leq \frac{2}{1-\gamma} \underbrace{\min_{\mathbf{w}} \left\| V^* - V_{\mathbf{w}} \right\|_\infty}_{\text{"Best possible approximation"}} \tag{2.25}$$

where the L_∞-norm of a vector is defined as $\|\mathbf{x}\|_\infty \triangleq \max_i |x_i|$. Increasing the *degree to which* V^* *may be approximated* using basis choice $\mathbf{H}$ (right-hand side of Equation 2.25) will similarly lead to bringing the ALP solution closer to V^*. Further, adding basis functions to $\mathbf{H}$ can never make the *bound* worse (in the norm $\|\cdot\|_\infty$). As mentioned above, in the limit that the optimal value function lies in the span of $\mathbf{H}$, the bound is reduced to 0 and the ALP solution is optimal.

The authors in [28] also develop a bound on the *quality of the policy* $\pi_{\hat{\mathbf{w}}} \triangleq \text{Greedy}(\hat{V})$, the *expected loss in return* when acting under $\pi_{\hat{\mathbf{w}}}$ instead of π^*:

$$\left\| V^* - V^{\pi_{\hat{\mathbf{w}}}} \right\|_{1,\nu} \leq \frac{1}{1-\gamma} \left\| V^* - \hat{V} \right\|_{1,(1-\gamma)\mu(\pi_{\hat{\mathbf{w}}},\nu)} \tag{2.26}$$

where ν is the initial state distribution and $\mu(\pi_{\hat{\mathbf{w}}}, \nu)$ the expected state visitation frequencies under policy $\pi_{\hat{\mathbf{w}}}$ conditioned on the initial state following distribution ν. A take-away is that if the states which a reasonable policy should cover are known beforehand, the state-relevance weights in the ALP, $\alpha(\mathbf{x})$, may be chosen accordingly to achieve a closer approximation in these

states (thereby also reducing the loss in expected return when acting under $\pi_{\hat{\mathbf{w}}}$).

2.4.2 Bellman Error-based Bounds

The bounds in the previous section are all *a priori* bounds based on the "approximability" of V^* with linear value functions under basis choice $\mathbf{H}$. The Bellman Error, on the other hand, enables the computation of *a posteriori* bounds when a particular value function $\hat{V}$ has been computed. The bounds in this section extend to arbitrary value functions and therefore do not depend on the ALP solution method or, more generally, linear value function approximations.

Definition 6 (Bellman residual). *The Bellman residual for value function $\hat{V}$ and any $\mathbf{x} \in \mathbf{X}$ is defined as:*

$$\text{BellmanResidual}(\hat{V})(\mathbf{x}) \triangleq \hat{V}(\mathbf{x}) - \mathcal{T}^*\hat{V}(\mathbf{x}). \tag{2.27}$$

The *Bellman Error* is then defined as the maximum absolute value of the Bellman residual vector:

Definition 7 (Bellman Error). *The Bellman Error for value function $\hat{V}$ is defined as:*

$$\text{BellmanError}(\hat{V}) \triangleq \|\hat{V} - \mathcal{T}^*\hat{V}\|_\infty = \max_{\mathbf{x}} |\hat{V}(\mathbf{x}) - \max_{\mathbf{a}} \hat{Q}(\mathbf{x}, \mathbf{a})|. \tag{2.28}$$

A well-known bound relates the Bellman Error of value function approximation $\hat{V}$ to the maximum error (or, *distance*) with respect to the optimal value function [112]:

$$\|V^* - \hat{V}\|_\infty \leq \frac{\gamma}{1-\gamma}\text{BellmanError}(\hat{V}) \tag{2.29}$$

Equation 2.29 gives the rationale for optimization techniques that aim to reduce the Bellman Error of a value function approximation as a means to bound the L_∞ approximation error to V^* (*e.g.* empirically for linear value functions in [82]). Even a single evaluation of the Bellman Error, however, is computationally infeasible for exponential state and action spaces since an exhaustive enumeration of both is generally required to implement the max operations in the same Equation. We will return to the topic of Bellman Error minimization for *"sparsely action-connected"* collaborative multiagent systems in Chapters 4 and 5.

2.5 Summary

This chapter covered the necessary background on factored models and their exact and approximate solution, arriving at the concept of *factored value functions* that form the basis for the

theoretical insights and novel solution methods in later chapters.

Chapter 4 revisits collaborative multiagent systems and introduces a novel "sparse action-connectivity" property in MASs that allows the efficient computation of the Bellman Error, even in domains with exponential state and action spaces. Based on these results, Chapter 5 develops a novel *coordination discovery* algorithm that incrementally constructs basis matrix $\mathbf{H}$ such that agent interaction remains sparse and bounded approximations to V^* can be computed. One of the fundamental assumptions in this background chapter, namely that $\mathbf{H}$ is provided by a domain expert, is relaxed in Chapter 5. Chapter 6 builds on the efficient constraint generation method for FMMDPs and identifies additional *locality* properties in the planning problem that allow to reduce the number of constraints further, scaling the method to planning problems that were previously infeasible.

We begin the treatment of *locality* in this thesis for a specific subclass of collaborative multiagent MDPs in the next chapter. A formal model for stochastic task allocation problems with many agents is introduced that makes the effects of *spatial location* explicit. While Chapter 3 focuses on *approximating the FMMDP model* for enabling efficient solutions, Chapters 4–6 introduce methods that approximate *the solution* to the full decision process and maintain bounded error guarantees.

Chapter 3

Approximate Models for Spatial Task Allocation Problems

Given the well-known unfavorable complexity results associated with large action and state spaces, many problem representations attempt to exploit structure in the domain for efficiency gains. The specific class of problems considered in this chapter are resource allocation settings under stochastic (generally, nonlinear) transition and reward models, which have a natural implementation as a collaborative multiagent MDP. Our exploration of structural leverage is in the context of a common subclass of these resource allocation settings where tasks are further characterized by their spatial distribution (*e.g.*, over a map), the fact that they may appear stochastically from external events, and that a cooperative team of robots has to coordinate to attend to them effectively. Jointly referred to as *spatial task allocation problems* (SPATAPs), these models afford particular representational benefits by exploiting spatial locality in the problem.

As seen in Chapter 2, representational benefits do not, in general, translate into computational gains during policy computation. However, task allocation problems lend themselves to particular domain-specific approximations given both *temporal duration* (tasks appear stochastically and disappear after servicing) and *spatial location* of tasks in the SPATAP model. One approach for computational leverage is therefore a model reduction to multiple smaller planning problems that jointly approximate the full SPATAP while enabling solutions in settings with large task and agent numbers.

In this chapter we consider two such classes of *domain-specific* model reductions for spatial task allocation problems. Considered first is an approximation that exploits the temporal characteristics of tasks by restricting the solution to the currently enabled tasks in the SPATAP. Solving this *phase approximation* reduces exponential dependence on tasks but does not, in general, enable scaling to large agent or (enabled) task numbers. To demonstrate how further domain-

specific model reductions may address exponential dependencies on these remaining factors, we review three existing distributed planning methods (originally in [18]) that jointly remove all exponential dependencies on task or agent numbers. Unlike in later chapters, these methods solve *approximate models* and do not offer solution bounds with respect to the optimal SPATAP solution. Their practical performance in large multi-robot planning domains compared to optimal (where available) and heuristic solutions has been validated in the extensive empirical study in [18] and is briefly summarized in the chapter.

This chapter therefore provides an introduction to the wider thesis topic of exploiting *locality* for the efficient solution of large factored multiagent problems. Revisited and generalized in later chapters, 'locality' here finds a *domain-specific interpretation* for a common class of problems where multiple robots have to address stochastic tasks in the environment.

3.1 Spatially Distributed Tasks in Multiagent MDPs

This chapter introduces a class of multiagent MDPs referred to as Spatial Task Allocation Problems (SPATAPs). SPATAPs are a general model for collaborative multiagent teams which jointly maximize utility by attending to and completing tasks in the world. Fundamental to a SPATAP is a *map*, *i.e.* a spatial representation of the world, together with a *distance metric* defining "closeness" between agents and tasks. One common example is the representation as a simple grid-world where tasks and robots are each assigned to individual locations (*i.e.*, *tiles*) and "closeness" is measured directly with the Euclidean distance.

A second key characteristic of SPATAPs is that they model stochastic task-assignment problems where tasks appear unpredictably due to *external events* outside the control of the agents. A scenario from the empirical study in [18], for example, is a *DirtWorld* domain where multiple robots coordinate to clean the world of stochastically appearing dirt spots. Another example is an incoming customer request in a ridesharing application, which modifies the current *planning phase* accordingly to include the new customer.

SPATAPs are fully described as a factored multiagent MDP (FMMDP). Loosely following the presentation in [18]:

Definition 8 (Spatial Task Allocation Problem)**.** *A SPATAP consists of the following elements:*

1. *A map defining possible task locations $\mathcal{L}$, distance metric $d_{\mathcal{L}}$, and agent movement actions $\mathcal{A}_M$,*
2. *A task structure $\mathcal{T}$ defining the set of* task types *in the world: $\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_{|\mathcal{T}|}\}$, where $\mathcal{T}_0$ refers to a special NULL task,*

3. *The set of* task states *for each task type $\mathcal{T}_k$, denoted $\mathbf{T}_k$, referring to the status of a task. Here, $\mathbf{T}_0$ is defined with exactly one state denoting the presence of no task, and $\mathbf{T} = \bigcup_k \mathbf{T}_k$ is the set of all task states,*

4. *One or more task actions $a_{\mathcal{T}_k}$ associated with task type $\mathcal{T}_k$ to perform that task,*

5. *A set of agents $\mathcal{D} = \{1, \ldots, n\}$ where each agent i may perform movement and (a subset of) task actions.*

A SPATAP has an implementation as a factored multiagent MDP $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, P, R \rangle$. Let agents $\mathcal{D}$ be as defined above and actions $\mathcal{A}$ include movement and task actions. Then the set $\mathcal{S}$ is defined by (joint) states $\mathbf{s} = \langle \lambda, \tau \rangle$ denoting agent location vector $\lambda = [\lambda_0, \ldots, \lambda_n]^T$ and task status vector $\tau = [\tau_0, \ldots, \tau_{|\mathcal{L}|}]^T$, where $|\mathcal{L}|$ denotes the size of the map. The transition model P naturally factors per (independent) agent movement and task states:

$$P(\lambda', \tau' \mid \lambda, \tau, a) = \prod_{x \in \mathcal{L}} p_x^T(\tau_x' \mid \tau_x, \lambda, a) \prod_{i \in \mathcal{D}} p_i^M(\lambda_i' \mid \lambda_i, a_i) \tag{3.1}$$

where p_x^T denotes the transition probability of the task at location x and p_i^M agent i's movement probabilities. The reward function factors additively into task rewards and movement costs:

$$R(s, a) = \sum_{x \in \mathcal{L}} R_x^T(\tau_x, \lambda, a) + \sum_{i \in \mathcal{D}} R_i^M(\lambda_i, a_i). \tag{3.2}$$

The FMMDP yields representational benefits by making the independence between agents and the locality of tasks explicit. In Equation 3.1, for example, tasks are assumed to transition independently given (joint) agent locations and their actions. A key theme that we will return to throughout this thesis is that of *exploiting locality* to translate these representational benefits also into efficiencies during solution of the planning problem. In this spirit, this chapter considers multiple domain-specific approximations to SPATAPs that are particularly suited for the spatial nature of the planning problem.

Note that the model is sufficiently general in principle to support different types of tasks with varying internal transitions, as well as heterogeneous agents with distinct capabilities. In the remainder of the chapter, however, we restrict our view to homogeneous agents that can *individually* service tasks with a single PerformTask action. Under this consideration, multiple agents servicing the same task is never beneficial and a key focus falls on efficient distribution of the team.

3.1.1 Locality assumptions

The SPATAP transition and reward models in Definition 8 encode the (conditional) independence between tasks that exists in the problem domain. In $P(\lambda', \tau' \mid \lambda, \tau, a)$, for example, tasks transition independently given their current state, *joint* agent locations and their action choices. This formulation does not address the unfavorable complexity results with many agents, however, since the terms pertaining to tasks remain *non-local*. One reasonable (domain-specific) assumption to make this problem class tractable is therefore to further exploit *spatial locality* inherent in agent and task locations, as measured by distance metric $d_{\mathcal{L}}$. Specifically, we assume that only agents in the vicinity of a task may influence its transition dynamics and introduce a task's *locality scope* $\mathbb{L}(x, \tau_x, \lambda, a)$ as the subset of agents within a specified distance d of task location x. Further criteria for inclusion in $\mathbb{L}$ may pertain to an agent's ability to service the task in state τ_x, or their action choice; the subset of included agent locations and actions is then denoted by $\lambda_{\mathbb{L}}$ and $a_{\mathbb{L}}$, respectively. With this assumption in place, all terms in transition and reward models are *local* and can be represented efficiently as $p_x^T(\tau_x' \mid \tau_x, \lambda_{\mathbb{L}}, a_{\mathbb{L}})$ and $R_x^T(\tau_x, \lambda_{\mathbb{L}}, a_{\mathbb{L}})$ in Equations 3.1 and 3.2.

3.2 Online Approximations

The locality assumptions developed in the previous section make the problem class representable for many agents but do not, in general, reduce the complexity for solving them. SPATAPs are general MMDPs and can be solved with a centralized planning method (*e.g.* value iteration or a factored planner like SPUDD [46]) for moderate problem sizes, yielding a policy in *joint* states $\mathbf{x}$ and actions $\mathbf{a}$. A central solution, followed by the distribution of policies to all agents, faces computational challenges in real-world systems as problems scale to many task and agent numbers, however. The approximate models in this section therefore seek to approximate the global solution well with a set of smaller, distributed planning problems, alleviating the need for a central node in the network.

All methods reduce the *model complexity* of the full SPATAP MMDP (with the locality assumptions from Section 3.1.1 in place) and solve the resulting reduced models exactly with an online, distributed planning algorithm. To enable this reduction, we first outline a *phase* approximation that exploits the distinct planning phases (*i.e.*, active tasks) in the SPATAP at the current time step. We then provide a review of existing, *subjective* approximations (originally in [18]) that compute the approximate best response given a summary statistic of task attendance by the other agents. Included in this review is an approximation that combines both phase and

subjective approximations to yield a solution algorithm without any exponential dependencies in (joint) state and action spaces. Table 3.1 summarizes the complexity of each model referred to in the remainder of the section.

	State space	Action space
MMDP:	$\lvert\mathcal{L}\rvert^{\lvert\mathcal{D}\rvert} \cdot \lvert\mathbf{T}\rvert^{\lvert\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert^{\lvert\mathcal{D}\rvert}$
Phase-MMDP:	$\lvert\mathcal{L}\rvert^{\lvert\mathcal{D}\rvert} \cdot \lvert\mathbf{T}\rvert^{\lvert p\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert^{\lvert\mathcal{D}\rvert}$
S-MDP:	$\lvert\mathcal{L}\rvert \cdot \lvert\mathbf{T}\rvert^{\lvert\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert$
SP-MDP:	$\lvert\mathcal{L}\rvert \cdot \lvert\mathbf{T}\rvert^{\lvert p\mathcal{L}\rvert}$	$\lvert\mathcal{A}_*\rvert$
k-SP-MDP:	$\lvert\mathcal{L}\rvert \cdot \lvert\mathbf{T}\rvert^{k}$	$\lvert\mathcal{A}_*\rvert$

Table 3.1: The SPATAP state and action space sizes, $|\mathcal{S}|$, $|\mathcal{A}|$, for the original MMDP formulation (first row) and the four approximations outlined in the chapter. $|\mathcal{A}_*|$ is an upper bound on the size of any agent's action set. Solution times are a polynomial function of these values; the *k-SP-MDP* retains no exponential dependencies (see text). Reproduced from [18].

3.2.1 Phase Approximations

Phase approximations exploit a domain-specific property of SPATAPs, namely their distinct division into *planning phases* of currently active tasks. A phase approximation ignores any task spawning likelihoods in the future and only considers active tasks after an external event has instantiated them. Phase approximations prohibit a proactive distribution of agents over the map (in anticipation of task appearance) but may yield a good approximation for the case that new tasks appear infrequently.

Phase approximations target one source of exponential dependencies in the SPATAP state, namely that pertaining to task locations $\mathcal{L}$. Planning is carried out in the SPATAP MMDP considering only the active tasks $p\mathcal{L}$. This reduces the second source of exponential dependencies but—in the limit—retains the dependency on the full set of tasks $|\mathcal{L}|$ (*cf*. Table 3.1). Additional unbounded approximations through model reduction of the (joint) SPATAP can be introduced to remove the exponential dependencies on both state and action space sizes.

3.2.2 A Review of Subjective Approximations

To address the issue of scale, the other co-authors in [18] introduce further, *subjective* approximations that are reviewed in the remainder of the section. Subjective approximations decompose a large planning problem into set of approximate, tractable problems, one for each agent

$i \in \mathcal{D}$. The distributed solution to the local planning problems then yields a global solution that is executed by the team. All *local MDPs* are referred to as *subjective MDPs* (S-MDPs) and consider only a single agent's action space $\mathcal{A}_i$. Agent i's S-MDP is given by the tuple $\langle \mathcal{S}_s, \mathcal{A}_i, P_i, R_i \rangle$ for some state (sub-)set $\mathcal{S}_s$, agent actions $\mathcal{A}_i$ and transition and reward models that only depend on agent i's actions. For SPATAPs, the solution complexity reduces as shown in Table 3.1.

The division of the global solution into a set of local value functions has a principled motivation as the *"best-response"* of an agent to all others in the planning problem. An approximation to the global MMDP is given by the *exact* best responses, which can be computed for an agent i under the (unrealistic) assumption that all other agents' policies are known and fixed over the planning horizon h. Formally, the best response is an MDP defined over local actions $\mathcal{A}_i$, and—in the fixed horizon case considered here—can be computed as:

$$\begin{aligned} Q_i^t(s, a_i) &= \sum_{a_{-i}} R(s, a_i, a_{-i}) p(a_{-i} \mid \pi_{-i}) + \sum_{s'} \sum_{a_{-i}} p(s' \mid s, a_i, a_{-i}) p(a_{-i} \mid \pi_{-i}) V_i^{t+1}(s') \\ &\equiv R_i^{\pi_{-i}}(s, a_i) + \sum_{s'} P_i^{\pi_{-i}}(s' \mid s, a_i) V_i^{t+1}(s') \end{aligned} \tag{3.3}$$

where the other agents are collectively referred to by the set $\{-i\}$, and further assumed to jointly execute known policy π_{-i}. That is, under an assumed behavior of the other agents, one can compute a best response as the solution to an MDP with (local) transition ($P_i^{\pi_{-i}}$) and reward ($R_i^{\pi_{-i}}$) functions. Note, however, that besides the assumption that π_{-i} is known, the resulting local value functions are still defined over the (generally exponential) *joint* state space.

Subjective MDPs implement an *approximate best response* defined over a local state space $\mathcal{S}_s$. For SPATAPs, the local state space is spanned by states $s_i = \langle \lambda_i, \tau \rangle$, *i.e.* agent i's position and the (global) task status vector. Additionally, agent i's S-MDP computes the approximate best response with an *approximate behavior* model of the other agents $\{-i\}$ during planning. Different such S-MDP implementations can be considered. One common baseline in this category is that of completely independent (or, *ignorant*) agents that ignore the presence of all other agents in the problem.

Ignorant Agent Baseline

A simple choice of S-MDP ignores the presence of other agents in the planning problem. It only models the elements of the SPATAP pertaining to its own location and assumes that the transitions (both movement *and* task status) only depend on its own actions (compare with

Equations 3.1 and 3.2 of the full SPATAP):

$$p_i^{SA}(s_i' \mid s_i, a_i) = \left(\prod_{x \in \mathcal{L}} p_x^{T,SA}(\tau_x' \mid \tau_x, \lambda_i, a_i) \right) p_i^M(\lambda_i' \mid \lambda_i, a_i)$$

$$R_i^{SA}(s_i, a_i) = \left(\sum_{x \in \mathcal{L}} R_x^{T,SA}(\tau_x, \lambda_i, a_i) \right) + R_i^M(\lambda_i, a_i). \tag{3.4}$$

To compute individual task transition ($p_x^{T,SA}$) and reward ($R_x^{T,SA}$) terms in Equation 3.4 from the full SPATAP, the model treats other agents' locations and actions as random noise or assumes a default behavior for the rest of the team. The resulting value functions $V_i^{SA,t}$ and $Q_i^{SA,t}$ disregard any coordination between agents and are likely to perform poorly where coordination is relevant.

An alternative is to compute the approximate best response via *agent prediction*. Prediction may yield a full policy π_{-i} of the remaining team over the planning horizon or take the form of a *summary statistic* over the aggregated remaining agents.

Empathic Reasoning through Agent Prediction

An S-MDP version of the (approximate) best response faces two issues; first, how to model the behavior of the other agents, and second how to integrate the model into the local planning of agent i. For general problems, a domain-specific form of policy prediction for every individual agent in the team may be necessary to obtain (time-dependent) $p_i^t(a_{-i} \mid \pi_{-i})$, *i.e.* the expected joint behavior from i's perspective over planning horizon h.

For *task-assignment problems* (and SPATAPs with *negative interactions* in particular), a summary statistic suffices that ignores agent identities and instead maintains a belief over (any) agent presence at locations on the map as a proxy during planning. In particular, agents that are close to active tasks are likely to service them with high probability in the future and vice versa. Under the assumption of such a reasonable policy, one may compute an aggregate statistic over future agent locations given current planning state s^0 and horizon h, represented by distributions $P(\lambda_{-i}^t \mid s^0)$, and then choose the best response for agent i.

To compute the likelihood of task servicing by any other agent, one option is to assume that other agents approximately (*e.g.*, under a Boltzman action selection criterion [97]) follow the self-interested policy from the previous section starting from their current state in s^0, *i.e.* $\pi_j \sim \text{Greedy}(V_{SA})\ \forall j \neq i$. Note that V_{SA} is identical for every agent j in the team.

The second question pertains to how agent i may integrate the belief over other agent po-

sitions in its own local planning of a best response. As outlined previously for the exact best response in Equation 3.3, maintaining a joint value function that takes into account other agents' predictions is generally intractable for large problems. An alternative is a fully distributed value function (DVF) [65, 91] in the SPATAP that uses the model of other agents' task attendance to *discount* the value of tasks to planning agent i. Under negative interactions where co-location at a task is never beneficial, this intuitively corresponds to a *reward sharing* within the team when a task is attended to by multiple agents [65]:

$$V_i^t(s_i) = \max_{a_i} Q_i^t(s_i, a_i)$$

$$Q_i^t(s_i, a_i) = R_i(s_i, a_i) + \sum_{s_i'} P(s_i' \mid s_i, a_i) \left[V_i^{t+1}(s_i') - \sum_{j \neq i} f_{ij} P(s_j^{t+1} = s_i') V_j^{t+1}(s_i') \right] \tag{3.5}$$

where all terms are *local*, j refers to agents other than i, $P(s_j^{t+1} = s_i')$ denotes the likelihood of agent j being co-located with agent i at the next time step, V_j^{t+1} the value function used for discounting its presence, and f_{ij} a weighting factor.

Intuitively, value functions V_j^{t+1} permit an *empathic reasoning* over other agents j and describe how much value will fall on the remaining (collaborative) team members in state s_i'. Note that in a fully decentralized team, agents do not communicate their valuation V_j^{t+1} to the other agents. Instead, every agent computes a value function from j's perspective and uses it locally for discounting. In the original formulation in [65] for example, each agent uses $V^{SA,t+1}$, *i.e.* the self-interested value function at $t+1$ (identical for all agents), for discounting of every agent j in Equation 3.5. A simpler choice is suggested in [18] where agents may directly use their own valuation at the next time step, V_i^{t+1}, also for discounting the presence of the other (homogeneous) team members. This assumes that their valuation of a task is identical to one's own without the need to attribute another, self-interested valuation $V^{SA,t+1}(s_i')$.

In summary, the approximate best response for agent i reviewed in this section first computes a likelihood of task attendance of all remaining agents (unique computation for each agent i) and joins this task attendance likelihood with the (common) valuation V_i^{t+1} to discount other agents' behavior at the next time step. As outlined initially, the goal is to avoid agent presence in conflicting states in a distributed planning algorithm without a central node for coordination; planning is repeated continuously in a receding horizon fashion to respond to changes in agent behavior and task status at every time step. *Social laws*, such as an imposed ordering over agents, can be used to break identical action choices should two agents end up in the identical map location [18].

Table 3.1 shows that S-MDPs reduce planning complexity to local robot state and actions; the resulting approximate models generally still retain an exponential dependency in the number of tasks, however. Merging a restricted version of the previously introduced phase approximations with the subjective models removes this dependency fully.

Subjective Phase Approximations

Consider first the direct combination of both subjective and phase approximations (referred to as *SP-MDP* in Table 3.1). SP-MDPs are subjective models with (local) value functions V_i^t and Q_i^t that only consider the active tasks at the current planning iteration.

A further approximation to render SPATAPs tractable is to assume an even distribution of agents over the map and to discard tasks beyond a certain distance from agent i's planning at the current time step [18]. Given the local perspective of agent i, the *k-SP-MDP* only considers the k closest tasks during computation of the local best response policies.

Phase and subjective approximations are complementary in their reduction of different sources of exponential dependencies in the SPATAP. The combination of both with an imposed limit k on the considered number of tasks yields an approximation *without exponential dependencies* in state *and* action spaces. The complexities of all discussed models are summarized in Table 3.1.

3.2.3 Experimental Summary

Since phase and subjective approximations are all unbounded reductions of the original SPATAP, it is of key interest how they perform in practical multi-robot task-assignment problems. In the extensive empirical study led the first author in [18], the k-SP-MDP is compared to the optimal solution (where feasible) and to other baselines for a range of multi-robot cleaning tasks in large gridworlds. The *DirtWorld* problem consists of up to 6 homogeneous robots (with four movement and one PerformTask action), each initialized in randomly selected starting states, along with a small task spawning likelihood ($p = 0.05$) defined over *all* tiles on the map. The robot team has to react to external events (*e.g.*, a coffee spill event instantiating a new task on the map) and maintain a clean world efficiently. As outlined above, planning is fully decentralized in the k-SP-MDP but agents are assumed to keep track of the other agents' location in their planning, *e.g.* via (local) communication.

For the small worlds with up to 3 agents that remain solvable optimally with a factored solver (Spudd [46]), the k-SP-MDP with each agent considering the $k = 4$ nearest tasks performs within 97% of optimal. The evaluation is then scaled up to large office worlds with up to $|\mathcal{S}| =$

$6.1 \cdot 10^{30}$ joint states and $|\mathcal{A}| = 15{,}625$ actions for which no optimal solutions are known. The k-SP-MDP outperforms the self-interested model (Greedy(V_{SA})) by a wide margin and also an *a priori* partition algorithm that assigns fixed regions to every agent after it has been initialized on the map [94]. A (loose) upper bound for an optimistic estimate of maximum team performance in the world is one where each task is completed within exactly two time steps, essentially modeling a teleport followed by a successful PERFORMTASK action. For the largest office world problems over a map with 66 locations, the k-SP-MDP achieves within 70% of this loose bound. For the full empirical study, we refer the reader to [18].

3.3 Related Work

The restriction of the local problem of each agent to a subset of state factors is reminiscent of converting the problem to a Dec-MDP with limited observation, but is in fact fundamentally different, since *the observation of the global state* is used to construct the agents' k-SP-MDPs. Moreover, despite recent advances, *e.g.*, [23], Dec-MDP solution methods do not nearly scale to problems of the size considered here, or are suitable only for transition and observation independent settings [8, 22] (which our setting is not).

While there have been other approximate methods for *solving general MMDPs*, these typically depend on pre-specifying the fixed, or context-dependent coordination structure to scale to large problem sizes [42, 53, 95]. For SPATAPs, however, fixed coordination structures may be a poor choice and the number of contexts infeasibly large. In addition, these methods are by default not aimed at exploiting the particular structure present in SPATAPs, namely independence of movement and locality of tasks. To overcome the problem of pre-specifying interaction structures previous work has attempted to learn them [21, 67], but the premise underlying these methods is that there are only few states in which the agents need to coordinate. In contrast, in SPATAPs, the agents need to coordinate their task selection in *all* states.

Approaches for assigning agents to tasks based on *auctions* [2, 9, 14] are closely related, but either do not reason about subsets of tasks [14], or do not properly address the sequential nature of the task in SPATAPs [2, 63]. Furthermore, after the tasks are auctioned off, agent policies are fixed and remain unchanged until the next bidding circle. SPATAPs also relate to more general resource allocation problems, since agents may be interpreted as resources. [113] uses a MILP formulation to solve the resource assignment and policy optimization problem jointly. The methods proposed here, however, allow reallocation at every time step and consider spatially distributed tasks and travel times.

The idea of interacting with the *aggregate effect* of other agents is studied in detail in the field of mean-field games [35], where the focus lies on characterizing equilibria. A few approaches have tried to extend these ideas to settings such as taxi-fleet optimization [1, 108] and theme park crowd management [34] via off-line planning. The 'aggregate effect' in these approaches typically consists of the number of agents present in different zones which directly affects utility of the planning agent. In the methods reviewed here, the *predicted* future agent locations serve as a proxy for their behaviors, which in turn affects the utility of the planning agent. This is reminiscent of agent aggregation in Dec-MDPs, *e.g.*, the Group Aggregated Decentralized MMDP (GA-Dec-MMDP) in [86, 104].

Finally, the *subjective approximations* can also be interpreted as online planning for a special instance of a level 1 interactive POMDP [26, 32]. The solution methods reviewed in this chapter are specific to SPATAPs and use location as the proxy for the other agents' policies.

3.4 Contributions

The approximate models and results summarized in this chapter are based on previously published work [17, 18]. The thesis author co-developed the SPATAP model and contributed optimal full- and Phase-MMDP solutions to the evaluation. The subjective approximations are due to the other co-authors in [18]. Their presentation here is modified slightly to emphasize the relation to the (exact) best response.

Introduced in this chapter were thus SPATAPs, a general subclass of MMDPs suitable for spatially distributed problems that a team of agents or robots needs to address. Such tasks are characteristic of many realistic multi-robot systems, such as mobile sensor nets, distributed transportation systems, and multi-robot exploration. SPATAPs make the independence between agents and the locality of tasks explicit in order to avoid any exponential dependencies on joint states or actions at the representational level.

We then introduced two classes of domain-specific *model reductions* that exploit spatial locality and temporal duration of tasks for approximate solutions to SPATAPs. Model reductions correspond to multiple, individually tractable MDPs that jointly approximate the global problem. *Phase approximations* considered only the currently active tasks; the reviewed class of *subjective approximations* further implemented a fully distributed coordination method that predicted other agent locations as a proxy for discounting the value of these tasks in one's own planning. Combining both approximate models yielded the k-SP-MDP model (one per agent) without any remaining exponential dependencies on either state or action factors.

Locality entered the SPATAP through restricted *locality scopes* per task at the representational level, then as a proxy for predicting the task attendance likelihood for the remainder of the robot team, and lastly for distributing the computation of the value function across all agents. The developed approximations generally remain unbounded but have been validated empirically in the extensive studies in [17, 18]. Distributed value functions, which form the basis of the work, have further demonstrated their suitability for deployment in realistic multi-robot settings [66].

In the following chapters we continue with the exploration of locality in factored multiagent planning problems. Our focus is extended significantly to address *general FMMDPs*, the automated coordination discovery between sub-groups of agents, and the maintenance of *bounded approximate solutions* even in large multiagent domains.

Chapter 4

Generalizing Locality in Multiagent Control

Many multiagent settings of interest do not necessarily possess a spatial element that lends itself to the approximate solutions introduced in the previous chapter. Even in problems that do, there may exist other forms of coordination structure that prove more effective for tackling a particular planning task (*e.g.*, by taking specific roles of agents in the team into account). In this chapter we therefore expand the interpretation of *locality* beyond the mere spatial sense to address general factored multiagent MDPs.

The notion of *generalized locality* introduced in this chapter applies to collaborative factored multiagent MDPs and equates to a form of sparse interaction between agents that is agnostic to domain-specific details, such as spatial proximity. Key to the principled definition of generalized locality is the concept of *factored* (*i.e.*, locally-scoped) *value functions* from Chapter 2. We first review how a particular basis choice induces a coordination factor graph (CFG) between agents that defines observation and communication requirements within the team. CFGs permit a particularly efficient computation of the jointly maximizing action and can be stored in a decentralized fashion across the agents.

We then introduce a *sparse action-connectivity* property for CFGs and outline a novel theoretical insight that shows how the Bellman residual factors over the state space if agent interaction retains the sparsity property. In turn, the factored residual enables the efficient computation of the Bellman Error and the derivation of value function error bounds even in large multiagent settings. In later chapters we translate these theoretical insights into novel planning algorithms with bounded performance guarantees.

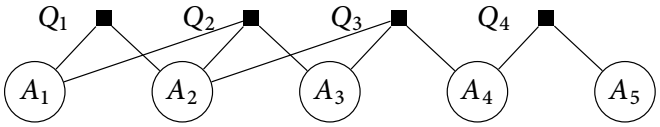

Figure 4.1: An example *coordination factor graph* (CFG) with five agents ($A_1, \ldots, A_5$) and local payoff functions $Q_1, \ldots, Q_4$. Edges indicate which agent participates in which Q-function component.

4.1 Factored Q-value Functions

In this section we introduce factored Q-value (or, equivalently, state-action value) functions and their key role in computing the jointly maximizing action in large multiagent settings efficiently. A factored value function $V(\mathbf{x}) = \sum_{j=1}^{k} w_j h_j(\mathbf{x}[\mathbf{C}_j])$ for given basis choice $h_1(\mathbf{c}_1), \ldots, h_k(\mathbf{c}_k)$ and parameters $w_1, \ldots, w_k$ induces a state-action value function that also factors into *local terms*:

$$\begin{aligned} Q(\mathbf{x}, \mathbf{a}) &= R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, \mathbf{a}) \sum_{j} w_j h_j(\mathbf{x}'[\mathbf{C}_j]) \\ &= \sum_{r} R_r(\mathbf{x}, \mathbf{a}) + \gamma \sum_{j} w_j g_j(\mathbf{x}, \mathbf{a}) \\ &= \sum_{i} Q_i(\mathbf{x}, \mathbf{a}) \end{aligned} \tag{4.1}$$

where all functions R_r, g_j are again locally-scoped (omitted for clarity) and $g_j(\mathbf{x}, \mathbf{a})$ is the *expectation* of an individual basis function h_j, which is computed efficiently via backprojection of h_j through the 2TBN (*cf*. Chapter 2). Local Q-functions Q_i follow by associating disjoint subsets of local reward and backprojection functions with each Q_i. Payoff functions Q_i and agents $A_1, \ldots, A_g$ then span a *coordination factor graph*.

Definition 9 (Factor graph)**.** *A factor graph (FG) [54] over variables* $\mathbf{X} = \{X_1, \ldots, X_N\}$*, factors* $\Phi = \{\phi_1, \ldots, \phi_M\}$*, and a function* $F : \mathbf{X} \mapsto \mathbb{R} \triangleq \sum_{i=1}^{M} \phi_i(\mathbf{x}[\mathbf{X}_i])$*,* $\mathbf{X}_i \subseteq \mathbf{X}$*, is an undirected graph with N variable nodes (ovals) and M factor nodes (squares) and edges only between each variable node* $X_i \in \mathbf{X}_i$ *and corresponding factor node* ϕ_i*, for all* i*.*

Definition 10 (Coordination factor graph)**.** *A coordination factor graph (CFG) is a FG in which variables correspond to agents* $A_1, \ldots, A_g$*, factors to local Q-functions* Q_i*, and function F is the global Q-function* $Q(\mathbf{x}, \mathbf{a}) = \sum_i Q_i(\mathbf{x}, \mathbf{a})$*.*

An example factorization of a global Q-value function into locally-scoped terms is shown as a CFG in Figure 4.1.

4.1.1 Distributed Q-functions

A factored Q-value function can be stored in a fully distributed fashion among the agents in a collaborative multiagent team, reducing state observation and action coordination (*i.e.*, *communication*) requirements at runtime. Following [39], a subset of basis and reward functions is associated with each agent $A_1, \ldots, A_g$ such that their union covers the entire set. Define by $\mathbf{H}_i$ the set of basis functions associated with agent i such that $\cup_i \mathbf{H}_i = \mathbf{H}$ and $\mathbf{H}_i \cap \mathbf{H}_j = \varnothing$ for $i \neq j$. Similarly, let the reward function for agent i be given by R_i. Then the Q-component associated with agent A_i,

$$Q_i(\mathbf{x}, \mathbf{a}) = R_i(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, \mathbf{a}) \sum_{h_j \in \mathbf{H}_i} w_j h_j(\mathbf{x}'[\mathbf{C}_j]),$$

has local state and action factor scope given by $Scope(Q_i) = Scope(R_i) \cup \bigcup_{h_j \in \mathbf{H}_i} Scope(g_j)$ where $Scope(f) \subseteq \{\mathbf{X}, \mathbf{A}\}$ denotes state and action variables that span the domain of a function f. Let $Q_i(\mathbf{x}[\mathbf{C}_i], \mathbf{a}[\mathbf{D}_i])$ denote the Q-component associated with agent i, making the local state and action factor scopes explicit. Note that variables $\mathbf{C}_i$ define *the subset of the state space* that agent i needs to observe. To compute the globally maximizing *joint action* of the agent team in a state $\mathbf{x}_0$, *i.e.* $\text{argmax}_{\mathbf{a}} \sum_i Q_i(\mathbf{x}_0, \mathbf{a})$, each agent instantiates Q_i in its (locally observed) state $\mathbf{x}_0[\mathbf{C}_i]$, yielding a *coordination factor graph* of instantiated Q-functions with mere action dependencies. As detailed in the following section, the coordination factor graph then enables an efficient computation of the max operation based on variable elimination or via a message passing scheme in the case of a fully distributed agent team [40]. Approximate implementations of the max operation over the CFG, akin to loopy belief propagation in Bayesian networks [54], have also appeared in the literature [53].

4.1.2 Variable Elimination

The variable elimination (VE) algorithm may be used for computing the max over a set of locally-scoped functions in a factor graph efficiently. Similarly to *maximum a posteriori* (MAP) estimation in Bayesian networks, VE maximizes over single variables at a time rather than enumerating all *joint states* $\mathbf{x} \in \mathbf{X}$ and picking the maximizing one [54]. VE performs two operations, Augment and Reduce, for every variable X_l to be eliminated. In our treatment of VE, we consider Augment to implement the sum operation over functions, and Reduce to be the maximization over X_l in the result. The VariableElimination$(\mathcal{F}, \mathcal{O})$ algorithm in Figure 1 implements VE over a given set of functions $\mathcal{F}$ and elimination order $\mathcal{O}$.

Adopted to the *coordination factor graph* setting, VE has an immediate application for com-

Algorithm 1: VARIABLEELIMINATION$(\mathcal{F}, \mathcal{O})$

Input: $\mathcal{F}$ is a set of functions
Input: $\mathcal{O}$ is an elimination order over variable indices
Output: The result of the maximization over all variables referred by $\mathcal{O}$
for $i = 1, \ldots, |\mathcal{O}|$ **do**
  $l = \mathcal{O}(i)$;
  `// Collect set of functions that depend on` X_l
  $\mathcal{E} = \textsc{Collect}(\mathcal{F}, X_l)$;
  `// Construct intermediate function (the sum) and compute the max`
  $f = \textsc{Augment}(\mathcal{E})$;
  $e = \textsc{Reduce}(f, X_l)$;
  `// Update the function set`
  $\mathcal{F} = \mathcal{F} \cup \{e\} \setminus \mathcal{E}$;
end
`// Return sum of empty-scope functions`
return $\textsc{Augment}(\mathcal{F})$;

Figure 4.2: The VARIABLEELIMINATION algorithm computing the maximum value of $\sum_{f \in \mathcal{F}} f$ over the state space [36, 54].

puting the globally maximizing *joint action* in a given state $\mathbf{x}_0$, *i.e.*, $\mathbf{a}^* = \operatorname{argmax}_{\mathbf{a}} \sum_i Q_i(\mathbf{x}_0, \mathbf{a})$, whenever the enumeration of all joint actions would be prohibitive. Every agent i first instantiates its local Q-component Q_i in the locally observed state $\mathbf{x}_0[\mathbf{C}_i]$, yielding an *instantiated CFG* with only action dependencies.

VE then proceeds as in Figure 4.2 but operates on *action* (or, *agent*) rather than state variables. The algorithm eliminates agents one-by-one from the instantiated CFG and performs *maximizations* and *summations* over local terms only. After all agents have been eliminated, a consistent maximizing joint action can be computed in a backwards pass [54].

Example 1. *Consider removal of agent A_4 from the CFG in Figure 4.1, instantiated in a particular state $\mathbf{x}_0$. Then the global Q-function can be written as (omitting the state factor scopes):*

$$Q(\mathbf{a}) = Q_1(a_1, a_2) + Q_2(a_1, a_2, a_3) + Q_3(a_2, a_3, a_4) + Q_4(a_4, a_5).$$

To remove agent A_4, a call to VARIABLEELIMINATION$(\{Q_1, Q_2, Q_3, Q_4\}, \{A_4\})$ *yields:*

$$\begin{aligned}
\max_{a_4} Q(\mathbf{a}) &= \max_{a_4} \left[Q_1(a_1, a_2) + Q_2(a_1, a_2, a_3) + Q_3(a_2, a_3, a_4) + Q_4(a_4, a_5) \right] \\
&= Q_1(a_1, a_2) + Q_2(a_1, a_2, a_3) + \max_{a_4} \left[Q_3(a_2, a_3, a_4) + Q_4(a_4, a_5) \right] \\
&= Q_1(a_1, a_2) + Q_2(a_1, a_2, a_3) + e(a_2, a_3, a_5)
\end{aligned}$$

where the intermediate function $e(a_2, a_3, a_5)$ denotes the result of the maximization over a_4. Agent A_4 can consequently be removed from the CFG.

The VE algorithm computes the optimal joint action in an instantiated CFG independent of the chosen elimination order. However, the algorithm's execution time is exponential in the size of the largest intermediate term formed during elimination which does depend on the chosen elimination order. In the distributed implementation of VE via message passing, communication only needs to occur whenever a link between agents is introduced in the CFG during elimination [36].

4.2 Sparse Coordination Factor Graphs

In this section we define a sparsity property for coordination factor graphs. As we will see later, *sparse CFGs* admit particularly efficient computation of the Bellman residual, even in large multiagent settings with exponential state and action spaces. The sparsity property is based on the connectivity structure of the CFG, specifically on how agents may exert influence on each other in a factored (*i.e.*, locally-scoped) Q-value function $Q(\mathbf{x}, \mathbf{a}) = \sum_i Q_i(\mathbf{x}, \mathbf{a})$.

Definition 11 (Action-connectivity). *Define two scopes $Scope(Q_i)$ and $Scope(Q_j)$ as "action-connected" iff $A_k \in Scope(Q_i) \wedge A_k \in Scope(Q_j)$ for some action variable $A_k \in \mathbf{A}$. Further, given a set of scopes $\mathcal{Z} = \{Scope(Q_1), \ldots, Scope(Q_N)\}$, define a* partition *$P$ over $\mathcal{Z}$ that consists of the disjoint sets $\mathbf{C}_1, \ldots, \mathbf{C}_k$ of within action-connected scopes: $\mathbf{C}_i \triangleq \{Scope(Q_1^i), \ldots, Scope(Q_M^i)\}$ where any scope in $\mathbf{C}_i$ is action-connected to another in $\mathbf{C}_i$ but to no other scope in $\mathbf{C}_j$, $j \neq i$.*

Let $\mathcal{Z}$ collect the scopes of the Q-components of a factored Q-value function. Intuitively, P partitions the corresponding CFG into disconnected agent chains.

Example 2. *Consider the CFG in Figure 4.3 where partition P is given by the two disjoint sets $\mathbf{C}_1 = \{Scope(Q_1), Scope(Q_2)\}$ and $\mathbf{C}_2 = \{Scope(Q_3), Scope(Q_4)\}$. The first set is action-connected through agents A_1, A_2, A_3, and the second set through A_4, A_5.*

Partition $P = \{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ defines the "action-connected subcomponents" of a factored Q-value function stemming from basis choice $\mathbf{H}$, *i.e.* those agents that interact directly in the scopes of either reward or backprojected basis functions. The action-connectivity property does not postulate independence between agents in different sets $\mathbf{C}_i, \mathbf{C}_j$, $j \neq i$, since they may still coordinate and exert influence on each other through mutually observed state factors.

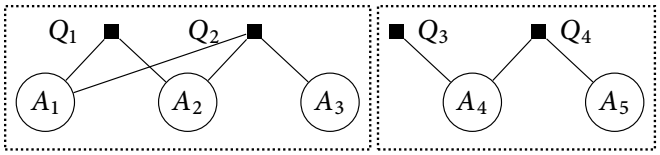

Figure 4.3: Two *action-connected* sets form a partition P of a coordination factor graph. The CFG fulfills a $\kappa_{\mathbf{A}}$-sparsity bound of 3 since the largest agent chain in P is A_1, A_2, A_3.

We can now define a sparsity property for coordination factor graphs. Denote by $\mathbf{C}_i.\mathbf{A}$ the union of action factors of all scopes in set $\mathbf{C}_i$, *i.e.*, $\mathbf{C}_i.\mathbf{A} \triangleq \bigcup_j Scope(Q_j^i).\mathbf{A}\ \forall Scope(Q_j^i) \in \mathbf{C}_i$, and similarly by $\mathbf{C}_i.\mathbf{X}$ the union of state factors corresponding to $\mathbf{C}_i$.

Definition 12 (Sparse coordination factor graph)**.** *Let $P = \{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ be the partition of a CFG into action-connected scopes. Then, the CFG is $\kappa_{\mathbf{A}}$-sparse iff the maximum number of action variables in any of the sets in P is less than $\kappa_{\mathbf{A}}$,* i.e. *if* $\max_i |\mathbf{C}_i.\mathbf{A}| \leq \kappa_{\mathbf{A}},\ \forall \mathbf{C}_i \in P$. *Analogously, the CFG is $\kappa_{\mathbf{X}}$-sparse if* $\max_i |\mathbf{C}_i.\mathbf{X}| \leq \kappa_{\mathbf{X}},\ \forall \mathbf{C}_i \in P$.

Example 3. *Consider the CFG in Figure 4.3 and its partition $P = \{\mathbf{C}_1, \mathbf{C}_2\}$ into two disjoint action-connected scope sets. The CFG fulfills a $\kappa_{\mathbf{A}}$-sparsity bound of 3 since the largest agent chain in P is formed by A_1, A_2, A_3,* i.e. $|\mathbf{C}_1.\mathbf{A}| \leq 3$.

Sparse coordination factor graphs possess interesting properties that can be exploited computationally, *e.g.* for the efficient computation of the Bellman Error in large multiagent planning problems.

4.2.1 Factored Bellman Error

As introduced in Chapter 2, the Bellman Error gives rise to a bound on the maximum error between a value function approximation $\hat{V}$ and the optimal solution V^*. *Bounded approximate solutions* have the key advantage that they admit strong performance guarantees beyond results from empirical evaluation alone. Unfortunately, evaluating the Bellman Error is computationally infeasible for exponential state and action spaces since an exhaustive enumeration of both is generally required to implement the max operations in the Bellman Error equation.

As introduced in the previous section, *factored functions* (which span a factor graph of locally-scoped terms) have a particularly efficient implementation of the max operation, even in settings with exponential state and action spaces. In this section we show that *sparse CFGs* retain a Bellman residual that is *factored* over the state space, in turn permitting the computation of solution bounds in large multiagent problems.

Recall the definition of the Bellman Error from Equation 2.28 and consider the full expansion of the L_∞-norm into the steps required to compute it:

$$\begin{aligned}\text{BellmanError}(\hat{V}) &= \|\hat{V} - \mathcal{T}^*\hat{V}\|_\infty \\ &= \underbrace{\max}_{\text{the } |\cdot|} \begin{pmatrix} \max_{\mathbf{x}} \Big(\sum_i w_i h_i(\mathbf{x}) - \max_{\mathbf{a}} [R(\mathbf{x},\mathbf{a}) + \gamma \sum_i w_i g_i(\mathbf{x},\mathbf{a})] \Big), \\ \max_{\mathbf{x}} \Big(\max_{\mathbf{a}} [R(\mathbf{x},\mathbf{a}) + \gamma \sum_i w_i g_i(\mathbf{x},\mathbf{a})] - \sum_i w_i h_i(\mathbf{x}) \Big) \end{pmatrix}. \end{aligned} \tag{4.2}$$

where the outer max implements the absolute value in the definition of the L_∞-norm. We first note that for the approximate linear programming solution *only one* of the two inner maximizations has to be computed:

Corollary 1. *For the ALP solution, the LP constraints enforce that* $\hat{V} \geq \mathcal{T}^*\hat{V}$ $(\forall \mathbf{x} \in \mathbf{X})$ *at the solution,* i.e.*, only the first cost network in Equation 4.2 needs to be computed. We may therefore define* $BellmanError_{ALP}$ *as:*

$$\begin{aligned} BellmanError_{ALP}(\hat{V}) &= \max_{\mathbf{x}} \left(\hat{V} - \mathcal{T}^*\hat{V} \right) \\ &= \max_{\mathbf{x}} \Big(\sum_i w_i h_i(\mathbf{x}) - \max_{\mathbf{a}} [R(\mathbf{x},\mathbf{a}) + \gamma \sum_i w_i g_i(\mathbf{x},\mathbf{a})] \Big). \end{aligned} \tag{4.3}$$

For general CFGs, the inner maximization over actions $\mathbf{a}$ *does not* necessarily result in a function that is factored over the state space. Evaluating the Bellman error exactly for these CFGs becomes infeasible in the exponential state and action spaces presented by collaborative multiagent systems. We now show a key result that for *sparse CFGs* the Bellman residual remains a factored function in $\mathbf{X}$ so that the maximization over the state space is carried out efficiently via variable elimination.

Lemma 2. *Let* $Q(\mathbf{x},\mathbf{a})$ *denote a factored Q-function and* $P = \{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ *be the partition of its CFG into action-connected scopes. Then* $(\mathcal{T}^*\hat{V})(\mathbf{x})$ *is a factored function over state variables* $\mathbf{X}$ *and can be written as* $\sum_{i=1}^{k} \phi_i(\mathbf{x}[\mathbf{C}_i.\mathbf{X}])$.

Proof. The result follows directly from the definition of $\mathcal{T}^*\hat{V}$. Note that we can maximize over each $\mathbf{C}_i.\mathbf{A}$ individually and that the resulting function consists of local terms, each defined over local scopes $\mathbf{C}_i.\mathbf{X}$. The operation's execution time is exponential in the largest clique formed, *i.e.* in the largest value $|\mathbf{C}_i.\mathbf{X}| + |\mathbf{C}_i.\mathbf{A}|$. □

Corollary 2. *Since both* $\hat{V}$ *and* $\mathcal{T}^*\hat{V}$ *are factored functions in* $\mathbf{X}$ *for sparse CFGs, so is the Bellman residual vector defined as* $\text{BellmanResidual}(\hat{V}) = \hat{V} - \mathcal{T}^*\hat{V}$.

The Bellman residual can therefore be computed efficiently if the sparsity bounds on the CFG, defined by $\kappa_{\mathbf{X}}$ and $\kappa_{\mathbf{A}}$, remain below a threshold (driven by computational considerations). To compute the Bellman Error for the ALP solution $\hat{V}$, an additional variable elimination procedure over the state space $\mathbf{X}$ is required to implement the max over the state space in Equation 4.3.

4.2.2 Bellman Residual Marginal

We have seen above that sparse CFGs enable the efficient computation of the Bellman residual by leveraging the factorization over *action-connected scopes* in the partition P of the CFG. The result is a factored function over the state space $\mathbf{X}$ so that max (*i.e.*, the Bellman Error) and min operations can be carried out in principle via variable elimination. VE retains an exponential dependency on the size of the largest intermediate term formed during elimination.

In this section we show how other functions of the Bellman residual can be computed efficiently, in particular *without the need for VE*. As will become apparent in the next chapter where novel solution methods to FMMDPs are presented, we focus on the computation of *marginals* of the factored Bellman residual. Marginals refer to sums of the Bellman residual *over subsets of variables*, resulting in new functions with reduced state factor scope. We collectively refer to these functions as "Bellman marginal functions" and delay a discussion of their detailed role to the next chapter.

Consider again the Bellman residual corresponding to a sparse CFG. Since it is a factored function per Corollary 2, we can write it explicitly as the sum over locally scoped factors ϕ_k:

$$\begin{aligned} \text{BellmanResidual}(\hat{V})(\mathbf{x}) &= \sum_i w_i h_i(\mathbf{x}) - \max_{\mathbf{a}}[R(\mathbf{x},\mathbf{a}) + \gamma \sum_i w_i g_i(\mathbf{x},\mathbf{a})] \\ &= \sum_k \phi_k(\mathbf{x}[\mathbf{C}_k]) \end{aligned} \tag{4.4}$$

where variable sets $\mathbf{C}_k \subseteq \mathbf{X}$ span the local domains $Dom(\phi_k)$.

We first show how the sum over *all* $\mathbf{x} \in \mathbf{X}$ can be implemented efficiently with only local computations over domains $Dom(\phi_k)$. We then extend this result to general marginals of the Bellman residual. As noted above, Bellman marginal functions are defined over subsets of variables $\mathbf{Y} \subseteq \mathbf{X}$, stemming from summing the Bellman residual over all other variables $\mathbf{W} = \mathbf{X} \setminus \mathbf{Y}$.

Efficient Summation over the State Space

The sum of the Bellman residual over the entire state space (denoted $BRS_{\hat{V}}$) can be computed efficiently, avoiding the explicit sum over the (exponential) state space $\mathbf{X}$, and variable elimina-

tion, by recognizing:

$$\begin{aligned} BRS_{\hat{V}} \triangleq \sum_{\mathbf{x}} \text{BellmanResidual}(\hat{V})(\mathbf{x}) &= \sum_{\mathbf{x}} \sum_{k} \phi_k(\mathbf{x}[\mathbf{C}_k]) = \sum_{k} \sum_{\mathbf{x}} \phi_k(\mathbf{x}[\mathbf{C}_k]) \\ &= \sum_{k} \lambda_{Dom(\phi_k)} \sum_{\mathbf{c}_k \in Dom(\phi_k)} \phi_k(\mathbf{c}_k) \end{aligned} \tag{4.5}$$

where $\lambda_{Dom(\phi_k)} = \frac{|Dom(\hat{V})|}{|Dom(\phi_k)|}$. The term $\lambda_{Dom(\phi_k)}$ accounts for the fact that in the sum over all $\mathbf{x}$, the summation over the *local* domain $\left(\sum_{\mathbf{c}_k \in Dom(\phi_k)} \phi_k(\mathbf{c}_k)\right)$ occurs exactly $\lambda_{Dom(\phi_k)}$ times. In the last line of Equation 4.5, summations are only over local domains, *i.e.* tractable in otherwise infeasibly large state spaces.

Example 4. *Consider a factored* $\text{BellmanResidual}(\hat{V})(\mathbf{x}) = \phi_1(x_1, x_2) + \phi_2(x_2)$*, defined over two factors given by:*

X_1	X_2	$\phi_l(x_1, x_2)$	$\phi_2(x_2)$
0	0	0	0
1	0	1	
0	1	2	3
1	1	3	
0	0	4	6
1	0	5	
0	1	6	9
1	1	7	

Then $BRS_{\hat{V}}$ *can be computed via exhaustive summation over the state space as:*

$$BRS_{\hat{V}} = \phi_1(0,0) + \phi_2(0) + \phi_1(1,0) + \phi_2(0) + \ldots + \phi_1(1,1) + \phi_2(1) = 64.$$

Alternatively, $BRS_{\hat{V}} = \lambda_{Dom(\phi_1)}(0+1+2+3+4+5+6+7) + \lambda_{Dom(\phi_2)}(0+3+6+9)$ *for* $\lambda_{Dom(\phi_1)} = \frac{|Dom(\hat{V})|}{|Dom(\phi_1)|} = 1$ *and* $\lambda_{Dom(\phi_2)} = \frac{|Dom(\hat{V})|}{|Dom(\phi_2)|} = 2$*, resulting in the same value 64. The summation over the (local) domain of* ϕ_2 *occurs two times in the sum over the global state space, and is accounted for in the result by multiplication with* $\lambda_{Dom(\phi_2)}$*.*

Efficient Marginals over the State Space

Summing the Bellman residual over a *subset of variables* $\mathbf{W} = \mathbf{X} \setminus \mathbf{Y}$ yields a *Bellman (residual) marginal function* defined over $\mathbf{Y}$. Efficient computation of these functions is possible with an

analogous method to the previous one which only sums each local factor ϕ_k over the space spanned by $\mathbf{C}_k \setminus \mathbf{Y}$. Note that after summing out $\mathbf{W}$, each local term $\phi'_k(\mathbf{y}[\mathbf{C}'_k])$ is now a *function* with scope $\mathbf{C}'_k \triangleq \mathbf{C}_k \cap \mathbf{Y}$.

The Bellman residual marginal function over $\mathbf{Y}$ can be computed efficiently as:

$$\begin{aligned}\text{BellmanMarginal}(\hat{V})(\mathbf{y}) &= \sum_{\mathbf{w}\in\mathbf{W}} \sum_k \phi_k(\mathbf{w}[\mathbf{C}_k \cap \mathbf{W}], \mathbf{y}[\mathbf{C}'_k]) = \sum_k \sum_{\mathbf{w}} \phi_k(\mathbf{w}[\mathbf{C}_k \cap \mathbf{W}], \mathbf{y}[\mathbf{C}'_k]) \\ &= \sum_k \lambda_k \sum_{\mathbf{z}_k \in (\mathbf{C}_k \setminus \mathbf{Y})} \phi_k(\mathbf{z}_k, \mathbf{y}[\mathbf{C}'_k]) = \sum_k \lambda_k \, \phi'_k(\mathbf{y}[\mathbf{C}'_k])\end{aligned} \tag{4.6}$$

where coefficients λ_k again account for how many times each function $\phi'_k(\mathbf{y}[\mathbf{C}'_k])$ occurs in the sum over $\mathbf{W}$ and each summation in the last row of Equation 4.6 is only over local subdomains.

Let $\mathbf{W}_k = \mathbf{W} \setminus \mathbf{C}_k$ denote the variables that are summed over at a local factor ϕ_k. Then λ_k is equivalent to the size of the space spanned by all variables in $\mathbf{W}_k$, *i.e.*, $\lambda_k = |\times \mathbf{W}_k|$ with $|\times \varnothing| \triangleq 1$.

Equation 4.5 is merely a special case of Equation 4.6 where the summation is over the entire state space, *i.e.*, $\mathbf{W} = \mathbf{X}$. In that special case it follows that $\mathbf{W}_k = \mathbf{X} \setminus \mathbf{C}_k$ and λ_k, the size of the space spanned by variables in $\mathbf{W}_k$, equates to $\lambda_{Dom(\phi_k)} = \frac{|Dom(\hat{V})|}{|Dom(\phi_k)|}$, as shown previously.

4.3 Generalized Locality

The previous section developed the concept of *sparse CFGs* and the efficient operations that they enable in domains with otherwise prohibitively large state and action spaces. For the remainder of the thesis, we take the notion of *generalized locality* to refer to (generic) sparsity in agent coordination stemming from a *sparse CFG*. There are no further assumptions in this understanding of generalized locality, *e.g.*, whether it is due to a spatial division between agents, different roles in the multiagent team, or any other domain-specific characteristics.

The approximation architectures for solving general, collaborative FMMDPs in the upcoming chapter merely assume that a sparse CFG—where $\kappa_{\mathbf{X}}$ and $\kappa_{\mathbf{A}}$ remain below a pre-defined threshold—exists whose associated value function approximation $\hat{V}$ approximates V^* well.

4.4 Related Work

Generalized locality in this chapter takes on the meaning of *factored (Q-)value functions* and the coordination factor graphs that they span. CFGs have been used in multiagent, decision-theoretic planning to compactly encode interactions between subsets of agents, usually for a pre-

specified coordination structure [39, 69]. They have also found application in multiagent *reinforcement learning (RL)* settings in order to distribute the observed rewards over (pre-specified) value function components [41, 53]. An extended overview of these RL methods can be found in [109].

A major computational advantage afforded by CFGs is their efficient computation of the jointly maximizing action. While we introduced variable elimination as an exact method, other, *approximate inference-based* approaches have been introduced to implement the max approximately in large CFGs [29, 53]. Further, while the presentation in this chapter focused on fixed CFGs, *context-based extensions* exist that enable varying CFGs in different states. Context-specific structure is commonly implemented with rule-based representations [36, 52].

Outside of the decision-theoretic realm, coordination factor graphs have a long history in decentralized coordination of actions to optimize a global utility function, *e.g*. in *distributed constraint optimization problems (DCOPs)*. These commonly focus on static domains without state (*e.g*., [80]) or repeated, independent problems although extensions to Markovian state signals exist [71]. A survey of these approaches can be found in [114].

A second focus of this chapter was the definition of "*sparsity in interaction*" between agents. Sparsity assumptions commonly enter planning through pre-specified structure or heuristics, *e.g*. via the spatial separation of agents [94], or distributed value functions [65, 91]. We define sparsity as a general property of CFGs (an upper bound on the action-connected factors) motivated by the efficient computation of the Bellman residual. Sparse CFGs admit *bounded solutions* to the optimal value function. Our work can be seen as a generalization of previous approaches that *restrict policies* to achieve the same effect. Guestrin *et al*. show for the single-agent case—based on Koller and Parr's earlier results that greedy policies derived from factored value functions have the form of a *decision list* [56]—that the computation of the Bellman Error can be made tractable by assuming default transition and reward models in the MDP [40]. We do not restrict the type of policies in our approach.

For CFGs that are not sparse, the Bellman residual is generally not factored and approximations need to be introduced to compute it. Common methods for approximating the *Bellman Error* are sampling-based approaches (using simulations of the system), or those that use known, canonical states for estimation [11]. These can be joined with regression methods to attempt generalization in otherwise prohibitively large state spaces [31, 76]. We retain the *exact* solution of the Bellman Error.

Finally, the concept of *locality* is just one form of the wider topic of *utilizing structure* for representing or solving complex planning problems. Exploiting structure has long tradition in

both single- and multiagent planning and learning settings (see [107] for an overview). Popular choices for structural leverage are *parallel decompositions*—splitting the MDP into several independently solved ones [37]—or *hierarchical decompositions*. The latter compute the solution from a hierarchy of smaller sub-MDPs, *e.g.* by defining sub-goals or independent regions that agents may transition among [24, 43]. The *options framework* is closely related by introducing temporally extended actions (sub-skills) during planning [64, 103] or learning [58, 98]. Automatically determining a hierarchical decompositions for the efficient solution of large MDPs remains an active field of research [5, 6].

4.5 Contributions

This chapter expanded the notion of locality beyond the spatial sense to general *sparse interaction* in collaborative FMMDPs. We initially reviewed factored Q-value functions, their relation to coordination factor graphs, and their key role in computing the jointly maximizing action in large MASs efficiently.

We then introduced *sparse coordination factor graphs (sparse CFGs)*, a novel contribution for encoding "sparsity of interaction" between agents, agnostic to its concrete, domain-specific source. We developed theoretical insights about the computational benefits of factored value functions that span a sparse CFG, in particular that the Bellman residual remains factored, allowing the efficient computation of the Bellman Error and other functions of the residual.

Sparse CFGs form the basic approximation architecture for the solution methods developed in the forthcoming chapter. A key focus will be on finding "good" sparse CFGs whose associated value function approximation $\hat{V}$ approximates V^* well.

Chapter 5

Bounded Approximate Methods for Sparse Coordination Discovery

This chapter returns to the theme of solving large factored planning problems efficiently by exploiting *locality* present in the problem. We saw earlier in Chapter 3 how *spatial locality*, as commonly found in multi-robot task assignment problems, can be leveraged for developing approximations that have no exponential dependencies in state and action spaces. This chapter extends this work along two main lines: first, the assumption of spatial locality is relaxed towards the more general understanding of *generalized locality* developed in the previous chapter. The solution methods presented here therefore apply to general factored multiagent problems (FMMDPs) that do not necessarily possess a spatial component. Second, unlike the domain-specific (and generally unbounded) methods developed previously, our focus here is on approximate methods for which exact solution bounds can be derived.

The planning methods we develop are based on a general assumption that there exists some form of sparse interaction between agents that—if found—allows to approximate the global value function well[1]. Our goal is to discover this structure automatically (without any prior domain knowledge) and to bound the value function solution arising from the discovered agent coordination structure. The developed approximation architectures are value-based methods that divide the global solution into locally-scoped value function components. As introduced previously, these methods are promising for large multiagent problems since solution algorithms without exponential dependencies in state and action spaces apply in principle (see Chapter 2). However, existing solution methods frequently assume a pre-specified coordination structure between agents and therefore replace the search for "good" agent coordination with input from a domain expert. In this chapter we make the link between coordination discovery and basis

[1]The coordination structure is assumed to be fixed throughout the problem.

choice explicit and phrase the search for coordination as a principled basis determination problem.

The search for agent coordination is generally driven by a trade-off between the desire for sparsity (*e.g.* due to computational limits in solving the joint agent policy or the goal to minimize agent communication overhead at runtime) and solution quality. We show how this trade-off is implemented in the context of the approximate linear programming solution (ALP) to general factored multiagent MDPs.

In order to do so, we first establish the link between basis choice and resulting agent coordination structure. We then show how a regularized version of the ALP addresses the search for sparse coordination graphs in principle through *basis selection* from an 'overcomplete' set. Based on our results for efficient operations with the Bellman residual in the previous chapter, we then move to incremental *basis generation* schemes that address computational issues with the previous method by selectively expanding a basis set. A key contribution is a Bellman error-based basis discovery scheme (referred to as BEBF*) that expands a basis in regions where the current value function error is largest. We develop an exact solution bound under the sparse "action-connectivity" assumption introduced in Chapter 4 and show how a sorting over basis functions implements the search from simple to complex coordination in a principled way.

Unlike methods that separate coordination discovery and solution of the actual multiagent control problem, we treat coordination as a by-product of achieving a desired bound on the solution to the FMMDP. This allows the designer of a multiagent system to trade-off complexity in coordination with solution quality where warranted. Alongside, we address a common criticism of the ALP solution method, namely the requirement of a pre-specified (and domain-specific) basis. Our bounded solutions scale to problems that could previously be evaluated empirically through policy simulation only (such as 50-agent *SysAdmin* or *DiseasePropagation* domains).

5.1 The Link between Basis and Coordination Discovery

Both state and action spaces in multiagent problems generally scale exponentially in the number of agents. Besides *computational* challenges (*i.e.*, computing a policy in the first place), this also introduces *representational* difficulties for storing the joint value function or retrieving a maximizing joint action efficiently.

Chapter 4 introduced locally-scoped state-value functions, $V(\mathbf{x}) = \sum_i w_i h_i(\mathbf{x}[\mathbf{C}_i])$, as a solution and showed that in the multiagent setting, V induces a coordination factor graph (CFG) with *only local* state observation and action coordination requirements. Operations like de-

termining the jointly maximizing action are then carried out efficiently under certain sparsity assumptions on the graph. Specifically, (exact) variable elimination is exponential in the size of the largest clique formed during elimination, which is affected by both state and action factor scopes in the CFG. The relevance of sparsity for both algorithmic and representational reasons underlies the notion of 'locality' used this chapter. Locality takes on the meaning of sparsity in the coordination factor graph, covering both sparsity in each factor's *state scope* (*i.e.*, locality in observation), as well as in *action-connectivity* between agents (*i.e.*, locality in interaction).

For factored linear value functions, the scope of each factor in the CFG is determined by the Backprojection operator applied to each *basis function* $h_k \in \mathbf{H}$ (*cf*. Chapter 2). This establishes a direct link between basis choice and coordination structure in a multiagent MDP.

Example 5. *Consider an exhaustive indicator basis* $\mathbf{H}_1$ *with one indicator function* $I_i : \mathbf{X} \mapsto \{0, 1\}$ *centered on each (joint) state* $\mathbf{x}_i \in \mathbf{X}$. *Note that the optimal value function lies in the span of* $\mathbf{H}_1$ *and—by Equation 2.24—will be the solution to the ALP. The CFG is fully connected and each factor is defined over the joint state and action spaces* $\mathbf{X}$, $\mathbf{A}$, *resulting in an optimization problem that is equivalent to solving all Bellman optimality equations exactly.*

While the optimal value function can be spanned by an exhaustive indicator basis in principle, this is quickly rendered impractical for the problem sizes of interest here. Underlying the approximation architectures we develop in this chapter is the assumption that there exists a mode of sparse interaction between agents that—if found—yields a good approximation to the global value function. We arrive at an interesting trade-off of constructing a set of basis functions that admits a good value function approximation while maintaining localized state and action factor scopes in the CFG. Coordination discovery is therefore treated as an *optimization of the basis set* that maintains the aforementioned properties in the CFG.

In the following sections we develop two classes of algorithms to tackle this problem: first, we formulate basis choice as a selection from a given, 'overcomplete' basis set by introducing a suitable regularization term in the ALP. We then move to an iterative basis generation scheme that implements a search over coordination structures from least to most complex while at the same time affording bounded value function solutions.

5.2 Basis Function Selection

Basis function selection methods, such as Matching Pursuit (MP) [47, 74], assume that a domain expert has provided an exhaustive (or, 'overcomplete') set of functions from which to select an

optimal subset given the specific problem requirements. Translated to the ALP, the initial basis set may include a large number of functions of varying complexity (*e.g.*, up to a maximum scope size) with the goal to find a sparse, scope-restricted subset of functions that reconstructs the value function well.

Subset selection in the ALP corresponds to finding a sparse solution vector $\hat{\mathbf{w}}$ where only a few basis weights are *enabled*, *i.e.* different from 0. Such constraints on the solution vector can be imposed with a regularization measure that penalizes number (and type) of the basis functions. The ALP is then solved *once* as before and agent coordination follows from the CFG spanned by the enabled basis set. Considered here are two such implementations of regularization that trade off basis complexity with expected performance.

5.2.1 The Regularized ALP

We saw previously that a large set of complex basis functions may generally lead to better approximations of the true value function (recall that in the limit, an exhaustive basis over the *joint state factor scope* yields exactly V^* as the ALP solution). The two regularization measures introduced here therefore address both number and complexity of the basis functions enabled at the solution.

First, we address sparsity in the ALP solution vector $\hat{\mathbf{w}}$. The L_0 pseudonorm of a vector, $\|\mathbf{w}\|_0$, is commonly taken to denote the number of elements in $\mathbf{w}$ that are not equal to 0. It is well established that it has an approximation with the L_1 (or, Manhattan) norm $\|\mathbf{w}\|_1 \triangleq \sum_i |w_i|$ which is often used for regularization in sparse feature selection problems, for example in compressive sensing [25]. The L_1-regularized ALP (RALP) has been introduced previously for feature selection purposes in a reinforcement learning setting in [81]. It is implemented with a straightforward modification to the ALP objective (compare with Equation 2.15):

$$\min_{\mathbf{w}} \quad \sum_{\mathbf{x}} \alpha(\mathbf{x}) \sum_i w_i h_i(\mathbf{x}) + \lambda \|\mathbf{w}\|_1 \tag{5.1}$$

or, alternatively, by enforcing a bound on each $|w_i|$ in the constraints instead of modifying the objective. Note that besides enforcing sparsity in the solution, L_1 regularization may have the additional positive effect of reducing the overfit of the ALP solution to the L_1 norm and actually improve on the value function solution (in terms of Bellman error, *i.e.* the distance to V^* in a L_∞ sense) [82].

Merely enforcing sparsity in $\hat{\mathbf{w}}$ may be sufficient to bias the solution towards a non-complex basis set as well. This is because computing an "everywhere-good" approximation to V^* (under

uniform $\alpha(\mathbf{x})$) with a sparse basis will naturally involve features that are active in large portions of the state space, *i.e.* those features with large coverage and, respectively, small scope size. Still, regularization of basis function scope size $|Dom(h_i)|$ may be made explicit for each h_i appearing with non-zero weight in the solution vector $\hat{\mathbf{w}}$. Here, $|Dom(h_i)|$ refers to the cardinality of h_i's domain; a regularization penalty that increases proportionally with scope size follows as $\delta_i \triangleq |Dom(h_i)|$ and can be included in a revised objective:

$$\min_{\mathbf{w}} \quad \sum_{\mathbf{x}} \alpha(\mathbf{x}) \sum_i w_i h_i(\mathbf{x}) + \lambda \|\mathbf{w}\|_1 + \beta \sum_i I(w_i)\, \delta_i \tag{5.2}$$

where $I(w_i)$ is an indicator function that is 1 iff $w_i \neq 0$ and 0 otherwise. In the linear program, this objective can be implemented with two additional variables with the following procedure:

1. For every w_i introduce variable z_i denoting the absolute value of w_i,

2. For every z_i introduce *integer variable* k_i that is 1 iff $z_i \neq 0$ and 0 otherwise.

The revised approximate linear program with regularization terms for sparsity in both solution vector *and* basis scope then follows as (compare with Equation 2.15):

$$\begin{array}{lrcll}
\min\limits_{\mathbf{w}} & \sum_{\mathbf{x}} \alpha(\mathbf{x}) \sum_i w_i h_i(\mathbf{x}) + \lambda \sum_i z_i + \beta \sum_i k_i\, \delta_i & & & \\
\text{s.t.} & \sum_i w_i h_i(\mathbf{x}) & \geq & [R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) \sum_i w_i h_i(\mathbf{x}')] & \forall \mathbf{x} \in \mathbf{X}, \forall \mathbf{a} \in \mathbf{A} \\
& w_i & & \text{unbounded} & \forall w_i \\
& w_i & \leq & z_i & \forall z_i \\
& -w_i & \leq & z_i & \forall z_i \\
& k_i & \in & \{0,1\} & \forall k_i \\
& k_i & \geq & \frac{z_i}{N} & \forall k_i
\end{array} \tag{5.3}$$

with λ, β non-negative, N an upper bound on $z_i\ \forall i$ (*e.g.*, $|R_{\max}/(1-\gamma)|$), and $\delta_i = |Dom(h_i)|$. Note that this reformulation is now an integer linear program and that λ and β are new parameters to the algorithm denoting the degree of penalization in the objective. The efficient solution method for representing exponentially many constraints outlined in Chapter 2 remains directly applicable.

5.2.2 The Issue of Scale

The regularized ALP presents a solution method for computing a value function approximation spanned by a sparse basis from an initial, overcomplete set. Each basis function (whether it appears in the solution or not) affects the number of variables and constraints in the ALP. In the efficient FMMDP solution method introduced in Chapter 2 for example, it is assumed that variable elimination can be carried out efficiently in the scope of all backprojected basis functions. It is conceivable that this assumption will be violated if basis functions are inserted into the ALP without further restrictions, as provided for example by a domain expert. Even if VE can be carried out efficiently, computation time may be better spent discovering a sparse basis that will likely be active in the solution.

A second drawback of the regularized ALP is that a bounded solution may be hard to obtain if the initial basis set is unconstrained. This is because the sparse *'action-connectivity'* property derived for the purpose of efficient Bellman residual computations in Chapter 4 can, in general, not be enforced through settings of λ and β alone. Further approximations to the Bellman error at the solution are necessary for an *a posteriori* bound on $\hat{V}$.

A better approach appears to be a basis generation scheme that inserts complexity only if and where (in state space) it is warranted. In the next section we outline the main contribution of this chapter with an iterative method that selectively expands the basis in regions where the current value function error is largest. We will further see how this alleviates the need for a domain expert while maintaining efficient computation of bounds on the solution $\hat{V}$.

5.3 Basis Function Generation

Delayed column generation has a long history as an efficient solution method for large linear optimization problems [10]. These methods do not start with an exhaustive set of variables but instead iteratively insert and *re-solve* the original (or, *master*) optimization problem. Determining the next best column to insert is formulated as a slave optimization problem that is interleaved between each iteration of the master problem.

In the context of the ALP solution to large MDPs, an interesting approach implements a similar iterative basis construction where *full recovery* of V^* is possible (in the limit) but where the process can be halted early, ideally with a bound on the obtained solution. As introduced in Chapter 2, the ALP solves for the best approximation to V^* in a weighted L_1 norm sense given basis set $\mathbf{H}_k$ and weights $\alpha(\mathbf{x})$ (*cf*. Equation 2.24). Of main interest, however, is the solution's distance to V^* in a L_∞ sense, *i.e*. the maximum distance at any state $\mathbf{x}$. Column generation methods

that directly optimize the ALP objective have only shown empirical reduction in Bellman error (which bounds the L_∞ distance to V^*) and are prone to overfitting the L_1 objective [82]. An ideal basis expansion, however, would reduce the L_∞ distance on the right-hand side of Equation 2.25 directly, therefore tightening the closeness of the ALP solution to V^* (the left-hand side) at every iteration. Intuitively, the "approximability" of V^* afforded by basis set $\mathbf{H}_{k+1}$ should increase "optimally" at each iteration (under any additional constraints on the basis functions, such as the "small scope" requirement).

In the following section we introduce a novel basis expansion method for the ALP that implements this intuition. Based on the theoretical results for efficient manipulation of the Bellman residual in sparsely 'action-connected' MASs from Chapter 4, we show how basis expansion can utilize the Bellman error metric directly to optimize insertion of a basis at the next iteration. Efficient computations with the Bellman residual therefore do not only enable error bounds at each iteration but also the insight *where in state space* the error integral is largest.

As outlined for basis selection in the previous section, basis generation should maintain certain properties of the coordination factor graph to scale the solution algorithm to large multiagent MDPs. Besides increasing the approximability of V^* at the next iteration, a practical basis construction method should therefore also consider sparsity in the CFG during expansion. Delayed column generation offers an additional, more general computational benefit by selectively including (and backprojecting) only those basis functions that likely appear enabled in the solution.

5.3.1 Bellman Error-based Coordination Discovery

Recent work in linear value function approximation in reinforcement learning (RL) has considered Bellman error-based metrics for incremental basis expansion [31, 57, 74, 76]. These methods aim to iteratively improve the bound to the true value function by introducing as next feature the *Bellman error basis function* (BEBF) $h_{k+1} \triangleq \mathcal{T}\hat{V} - \hat{V}$—or an approximation thereof—into the basis set. Here, $\mathcal{T}$ refers to the dynamic programming operator and $\hat{V} = \mathbf{H}_k\hat{\mathbf{w}}$ to the current estimate of the value function at iteration k. It is easy to see that setting $\mathbf{H}_{k+1} = [\mathbf{H}_k, h_{k+1}]$ is equivalent to introducing $\mathcal{T}\hat{V}$ into the span of $\mathbf{H}_{k+1}$, where $\mathcal{T}\hat{V}$ improves on the bound to the true value function by at least γ due to the contraction property of the dynamic programming operator in L_∞. For linear fixed point methods, such as TD or Least Squares Policy Iteration (LSPI) [61], the fact that $\mathcal{T}$ is *also* a contraction in the weighted $L_{2,\rho}$ norm (with its associated inner product space) allows to construct a *geometrical argument* for feature insertion: if both

ρ, the stationary distribution induced by policy π, and the Bellman error vector are known, an *approximation* to the BEBF is also guaranteed to improve the error bound to the true value function V^π if it does not exceed a bound on the *angle* to the Bellman error vector [76]. Stated differently, for linear fixed point methods, basis function choice may actually worsen the error bound to the true value function if this criterion for the angle is violated.

Note that the previous analysis justifies the use of Bellman error basis functions for *policy evaluation* (or with uncontrolled Markov chains) but does not immediately extend to the controlled case since $\mathcal{T}^*$, the Bellman optimality operator, does not possess a contraction property in $L_{2,\rho}$. Further, since the Bellman error vector and steady-state distribution ρ are unavailable in the Batch-RL analysis of [31, 76], approximations to both need to be computed in practice, *e.g.* with a regression over sampled state-action pairs. If policy *evaluation and improvement* steps are interleaved to compute the optimal value function V^*, there also exists a norm incompatibility since policy improvement as per the usual max-norm analysis again only possesses a contraction property in L_∞.

The BEBF*

We now proceed to combine the advantages of the ALP solution method (*i.e.*, its suitability for solving the *full control problem* in domains with large state *and* action spaces) with a principled, Bellman error-based basis expansion method. We begin by formally introducing the BEBF*, the equivalent to the BEBF for the Bellman optimality operator $\mathcal{T}^*$:

Consider the *controlled case*: it is well known that the Bellman operator $\mathcal{T}^*$ is a contraction in L_∞ [83]:

$$\|V_1 - V_2\|_\infty = \epsilon \Rightarrow \|\mathcal{T}^* V_1 - \mathcal{T}^* V_2\|_\infty \leq \gamma\epsilon \tag{5.4}$$

where γ is the discount factor. Now consider the basis set $\mathbf{H}_k$ at the current iteration: including $\mathcal{T}^*\hat{V} = \mathcal{T}^*\mathbf{H}_k\hat{\mathbf{w}}$ in the span of $\mathbf{H}_{k+1}$ at the next iteration of the ALP is guaranteed to improve the bound on the right-hand side of Equation 2.25 by at least a factor of γ unless the optimal value function has been obtained. This is because the "best possible" approximation on the right-hand side of the same equation at iteration $k+1$,

$$\min_{\mathbf{w}} \|V^* - V_{\mathbf{w}}\|_\infty \triangleq \min_{\mathbf{w}} \|V^* - \mathbf{H}_{k+1}\mathbf{w}\|_\infty \tag{5.5}$$

then includes $\mathcal{T}^*\hat{V}$ as a solution which reduces the bound accordingly. Including this basis in the span of $\mathbf{H}_{k+1}$ is trivially achieved by defining h_{k+1} to be the BEBF* $\triangleq \mathcal{T}^*\hat{V} - \hat{V}$ and letting $\mathbf{H}_{k+1} = [\mathbf{H}_k, h_{k+1}]$ (then, precisely vector $\mathbf{w} = [\hat{\mathbf{w}}\ 1]^T$ reconstructs $\mathcal{T}^*\hat{V}$).

In our setting, the Bellman residual can be computed efficiently since it is factored over the state space under a sparse 'action-connectivity' assumption in the MAS (*cf*. Chapter 4). The BEBF* can therefore be computed at a given iteration in principle. Every *component* $1, \ldots, M$ of the BEBF* can then be appended as a (locally-scoped) basis function to $\mathbf{H}_k$ to form $\mathbf{H}_{k+1}$: note that $\mathcal{T}^*\hat{V}$ is then trivially constructed in $\text{span}(\mathbf{H}_{k+1})$ with the weights $w_1, \ldots, w_k$ from the previous iteration and $w_{k+1:M} = 1$, guaranteeing an improvement of the bound by a factor γ. However, executing this procedure N times—$\mathcal{T}^{*^N}$—leads to ever-increasing scopes of the basis functions since every new basis is in turn backprojected through the DBN at the next iteration.

The derivation above serves therefore foremost as a theoretical result – as mentioned previously, there are reasons to impose constraints on the basis function choice h_{k+1} (such as "small scope" to maintain sparsity in the CFG) while remaining a close approximation to the BEBF*. In the following sections we implement this trade-off between closeness to the BEBF* and favorable agent coordination as a principled search from least to most complex CFG for binary basis functions. Binary basis functions have strong practical relevance since associated manipulations are computationally cheap and a linearly-independent basis is easily constructed [31]. The procedure is divided into *basis generation* (*i.e*., construction of a candidate set $\boldsymbol{\chi}$) and *evaluation* (*i.e*., a sorting operation on the candidate set), which are detailed below. In the remainder of the chapter, we use the terms (binary) *feature function* and *basis function* interchangeably.

It is worthwhile to point out that unlike the analysis for linear fixed point methods in [31, 76], adding arbitrary columns to $\mathbf{H}$ can never worsen the bound in equation 2.25. This, however does not mean that the ALP solution cannot worsen *within* this bound: as for the case of linear fixed point methods, only the bound is guaranteed to improve monotonically. Corollary 3 below restates this fact.

Feature Evaluation

In this section we introduce approximations to the BEBF* by leveraging the efficient computation that the factored Bellman residual admits. Note that adding the $\text{BEBF}^* = \mathcal{T}^*\hat{V} - \hat{V}$ to $\mathbf{H}_k$ is equivalent to adding the negative Bellman residual from $\hat{V}$ as h_{k+1} to the basis. Following the notation of [31], given a set of candidate features $\boldsymbol{\chi}$, we therefore seek the feature $h_{k+1} \in \boldsymbol{\chi}$ that most closely approximates the residual $\hat{V} - \mathcal{T}^*\hat{V}$ (or, equivalently, the BEBF*), such that its addition to $\mathbf{H}_k$ reconstructs $\mathcal{T}^*\hat{V}$ in $\text{span}(\mathbf{H}_{k+1})$ well. The exact procedure for constructing the candidate set $\boldsymbol{\chi}$ is treated in the next section.

Corollary 3. *Trivially, for any choice* h_{k+1}, $\min_{\mathbf{w}} \| V^* - \hat{V}_{\boldsymbol{\chi}} \|_\infty - \min_{\mathbf{w}} \| V^* - \hat{V}_{\boldsymbol{\chi} \cup \{h_{k+1}\}} \|_\infty \geq 0$ *(one*

may always choose w_{k+1}, the weight associated with h_{k+1}, to be 0). The bound in Equation 2.25 is therefore monotonically decreasing (in the $\leq$ sense).

For the uncontrolled (*policy evaluation*) case, under the $L_{2,\rho}$ analysis mentioned previously, a sufficient condition for h_{k+1} that yields a strictly improving bound (in the $<$ sense) can be derived based on a geometrical argument [76]. The argument yields a maximum-angle bound between the BEBF and the candidate feature in the inner product space associated with $L_{2,\rho}$.

Definition 13 (Binary Feature Function). *Denote by $f : \mathbf{X} \mapsto \{0,1\}^N$ a binary basis function (or,* feature function*) that maps each state $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbf{X}$ to a feature vector $[\psi_f(\mathbf{x}_1), \ldots, \psi_f(\mathbf{x}_N)]^T$ indicating whether binary feature $\psi_f : \mathbf{X} \mapsto \{0,1\}$ is* enabled *in each state.*

For the choice f_{k+1} from a given candidate set $\boldsymbol{\chi}$, Geramifard et al. show in [31] that the same geometrical analysis can be extended to find the feature $f^*_{k+1} \in \boldsymbol{\chi}$ with the maximum guaranteed error bound reduction from the candidate set. Again, in practice two approximations to f^*_{k+1} are evaluated since the steady state distribution ρ and Bellman error are not available exactly.

For the analysis of the BEBF*, *i.e.* the controlled case of interest here, the "next best" feature to select from a candidate set $\boldsymbol{\chi}$ escapes a geometrical argument, precisely because the L_∞ norm is not associated with an inner product as is $\|\cdot\|_{2,\rho}$ and the notion of angles between vectors is undefined. Still, one may hope that approximations to the BEBF* that retain "closeness" may yield a (strict) decrease in the error bound to V^*. Note that a key difference to the uncontrolled implementation above is therefore that the Bellman residual is available exactly for constructing candidate basis functions (*i.e.*, does not have to be approximated) but that feature "closeness" escapes an exact geometrical analysis. Second, since basis insertion may not worsen the error bound to V^* for the ALP, insertion of a suboptimal basis may only delay improvement of (but not worsen) the bound.

We now turn to approximations to the BEBF* and show how they can be evaluated efficiently for the *binary feature function* case, inducing an ordering over features in $\boldsymbol{\chi}$ for iterative basis expansion. The following general properties of candidate basis functions are considered for evaluating approximations to the BEBF*:

Definition 14 (Feature Coverage). *Consider the binary feature function $f : Dom(f) \mapsto \{0,1\}^M$ defined over state variables $\mathbf{Z} \subseteq \mathbf{X}$ of size $|Dom(f)| = M$. Define its* coverage *as the region of the state space $\mathbf{X}$ where it is* active*,* i.e.*:*

$$Coverage(f) \triangleq \{\mathbf{x} \mid \psi_f(\mathbf{x}[\mathbf{Z}]) = 1, \mathbf{x} \in \mathbf{X}\}$$

where $\mathbf{x}[\mathbf{Z}]$ *denotes the value of* $\mathbf{x}$ *for the state variables in* $\mathbf{Z}$.

Corollary 4. *The sum of the* Bellman residual *within the region of the (global) state space where* f *is* active (i.e., *within Coverage*(f)*), can be computed efficiently for sparsely 'action-connected' MASs. Similarly for its maximum (*i.e., *the Bellman error) and minimum over Coverage*(f).

Proof: Chapter 4 introduced the *Bellman residual marginal* over $\mathbf{Y}$, BellmanMarginal$(\hat{V})(\mathbf{y})$, as the sum of the Bellman residual over $\mathbf{X} \setminus \mathbf{Y}$, and showed its efficient computation for sparsely 'action-connected' MASs. The sum of the Bellman residual marginal where binary feature f is active, denoted $BRS(f)$, can then be computed efficiently by realizing:

$$BRS(f) = \sum_{\mathbf{x}} |\text{BellmanResidual}(\hat{V})(\mathbf{x}) \cdot f(\mathbf{x}[\mathbf{Z}])| = \sum_{\mathbf{z}_i} |\text{BellmanMarginal}(\hat{V})(\mathbf{z}_i) \cdot f(\mathbf{z}_i)| \quad (5.6)$$

where BellmanMarginal$(\hat{V})(\mathbf{z}_i)$ is the Bellman residual marginal function over $Dom(f)$ evaluated at $\mathbf{z}_i$. As shown in Equation 4.6, this is computed efficiently without a variable elimination step. For the max (*i.e.*, the Bellman error), *variable elimination* yields BellmanError$(\hat{V})(\mathbf{z})$, *i.e.* the Bellman error function over $Dom(f)$ and Equation 5.6 applies again with the sum replaced by a max. The case of the min follows analogously. In the following, we denote the result of max and min computations by $BE(f)$ and $BM(f)$, respectively. □

Intuitively speaking, one may hope that including features in the ALP that subsume a large amount of the Bellman residual 'integral' will be active in the solution and move $\hat{V}_{k+1}$ "closer" to V^*. A second property of a canddiate feature can be referred to as its *range*:

Definition 15 (Feature Range)**.** *Define by Range*(f) *the difference between maximum and minimum Bellman residual in Coverage*(f)*: Range*$(f) \triangleq BE(f) - BM(f)$.

Corollary 5. *Adding the binary feature* f_{k+1} *to the basis,* i.e. $\mathbf{H}_{k+1} = [\mathbf{H}_k, f_{k+1}]$*, is equivalent to introducing a value function in span*$(\mathbf{H}_{k+1})$ *that is* ϵ*-close to* $\mathcal{T}^*\hat{V}$ *in Coverage*(f_{k+1})*, where* $\epsilon \triangleq \frac{Range(f)}{2}$.

Proof: Associate feature f_{k+1} with weight $w_{k+1} = -BM(f) - \frac{\text{Range}(f)}{2}$ and all other weights $w_1, \ldots, w_k$ as in the previous iteration. Then span$(\mathbf{H}_{k+1})$ includes the value function where the Bellman error $BE(f)$ has been reduced within Coverage(f) so that the new values lie in the band $\pm\frac{\text{Range}(f)}{2}$ of $\mathcal{T}^*\hat{V}$. □

While again no formal guarantees on bound reduction can be made for introducing an ϵ-close function to $\mathcal{T}^*\hat{V}$ (over a particular coverage) in span$(\mathbf{H}_{k+1})$, a feature's range may present another criterion for candidate evaluation.

Given a set of candidate features $f_{k+1} \in \boldsymbol{\chi}$, there are therefore multiple criteria under which the next feature f^*_{k+1} to add to $\mathbf{H}_k$ may be selected:

1. The sum of the (absolute) Bellman residual within Coverage(f): $BRS(f)$. Note that this increases monotonically with feature coverage and will naturally prefer "coarser" features (*i.e.*, those that cover larger portions of the state space).

2. The achieved *error band* around $\mathcal{T}^* \hat{V}$ in Coverage(f): ϵ. Note that ϵ is, in general, smaller for small feature coverage and vice versa. In the extreme case, a feature that is active at a *single state* can remove the Bellman error entirely at this state ($\epsilon = 0$).

3. The *absolute reduction* in Bellman error in Coverage(f), $BE(f) - \epsilon = \frac{1}{2}(BE(f) + BM(f))$.

There is therefore a clear dependency between all criteria and the size of f's state space coverage, $|\text{Coverage}(f)|$. A good feature selection criterion likely needs to trade off a choice of $1-3$ with the coverage measure. In the max-norm analysis pursued here, without an option for a deeper geometrical argument, this trade-off has to be evaluated empirically.

Interestingly, in the $\|\cdot\|_{2,\rho}$ analysis for policy evaluation (or uncontrolled Markov chains), the binary feature $f_{k+1} \in \boldsymbol{\chi}$ that yields the maximum guaranteed error bound reduction can be shown to be $f^*_{k+1} = \text{argmax}_{f \in \boldsymbol{\chi}} \frac{BRS(f)}{\sqrt{|\text{Coverage}(f)|}}$, showing a similar trade-off, albeit with additional approximations to both numerator and denominator in practice [31].

Incremental Feature Generation

Incremental feature generation addresses the question which feature set $\boldsymbol{\chi}$ to construct for evaluation at each iteration of basis discovery. Here, we seek to implement a coordination graph search from least to most complex through the restriction to locally-scoped basis functions in the ALP. For binary basis functions, one basis sorting operator that achieves this expansion ordering is the *binary feature conjunction* operator, which can be defined as [31]:

$$\texttt{pair}(\boldsymbol{\chi}) \triangleq \{f \cup g \mid f, g \in \boldsymbol{\chi}, f \cup g \notin \boldsymbol{\chi}\} \tag{5.7}$$

with $(f \cup g)$ denoting the feature function mapping each $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbf{X}$ to the vector $[\psi_f(\mathbf{x}_1) \wedge \psi_g(\mathbf{x}_1), \ldots, \psi_f(\mathbf{x}_N) \wedge \psi_g(\mathbf{x}_N)]^T$. This operator has the favorable theoretical properties that it expands a basis of linearly independent columns, that the expanded basis spans V^* in the limit, and that the maximum number of expansions is bounded (under the mild assumptions given in [31]). Further, if $\boldsymbol{\chi}$ is initialized with a set of "small-scope" base features, the operator naturally

implements an expansion from least to most complex basis via binary conjunctions of existing features.

The Sparse Coordination Discovery (SCD) Algorithm

In this section we summarize the complete Sparse Coordination Discovery algorithm based on the individual components described previously.

1. *Initialization:* Basis set $\boldsymbol{\chi}$ is seeded with an exhaustive indicator basis centered on single state factors X_i: Let $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$. One indicator $I^k_{X_i}$ is inserted for all factors X_i and factor values k: $I^k_{X_i} = 1$ iff $X_i = k$, 0 otherwise.

2. *Feature generation:* At every iteration of the algorithm, the `pair` operator constructs up to $\binom{|\boldsymbol{\chi}|}{2}$ *valid* binary feature conjunctions from the existing features in $\boldsymbol{\chi}$.

3. Given the current ALP solution vector $\hat{\mathbf{w}}$, $\text{BellmanResidual}(\hat{V})(\mathbf{x}) = \hat{V} - \mathcal{T}^* \hat{V}$ (factored over state factors $\mathbf{X}$) is computed, involving the efficient maximization over the joint action space detailed in Chapter 4.

4. *Feature evaluation:* Each candidate feature f_{k+1} is scored with an evaluation function $\textsc{FeatureEval}(f_{k+1})$ (see below). The largest 'action-connected' scope resulting from including f_{k+1} in the basis set is determined via backprojection of the candidate through the DBN (Chapter 4). If the *sparsity bound* of the CFG is violated, *i.e.* exceeding a limit on the complexity of interaction between agents, f_{k+1}'s score is set to $-\infty$. Otherwise, the highest scoring feature, $f^*_{k+1} = \text{argmax}_{f_{k+1}} \textsc{FeatureEval}(f_{k+1})$, is determined.

5. The basis set is expanded greedily with the highest scoring feature, $\boldsymbol{\chi}_{k+1} = \boldsymbol{\chi} \cup \{f^*_{k+1}\}$, and the ALP re-solved.

6. *Error bound computation:* The Bellman error (and bound on value function error) at the new solution $\hat{\mathbf{w}}_{k+1}$ is computed efficiently via variable elimination (Chapter 4).

7. Steps 2-6 are repeated until a pre-specified bound on the value function error has been achieved, a plateau in Bellman error is observed, or no valid feature (with a score $\neq -\infty$) remains. After the stopping criterion is met at iteration p, solution vectors $\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_p$, associated coordination factor graphs, and error bounds are returned.

Note that at every iteration of the algorithm a bound on the distance to the optimal value function is available. Let the `pair` operator generate a set of candidate features f_{k+1} from $\boldsymbol{\chi}$ at

the current iteration. We consider two evaluation functions to stand in for FeatureEval(f_{k+1}) above:

1. FeatureEvalBRS(f_{k+1}): Expand first the feature $f^*_{k+1} \in \texttt{pair}(\boldsymbol{\chi})$ that accumulates the maximum sum of the (absolute) Bellman residual where it is active, relative to its coverage. Note that this is analogous to the expansion criterion for the *uncontrolled* case, $\frac{BRS(f_{k+1})}{\sqrt{|\text{Coverage}(f_{k+1})|}}$.
2. FeatureEvalEpsilon(f_{k+1}): Expand first the feature $f^*_{k+1} \in \texttt{pair}(\boldsymbol{\chi})$ that achieves the tightest ϵ-band around $\mathcal{T}^*\hat{V}$ in Coverage(f). Intuitively, these features attempt to approximate $\mathcal{T}^*\hat{V}$ well, even if restricted to small coverage of the state space (a "discriminant feature").

5.3.2 Summary

This section detailed a method for incremental basis generation in the ALP and established the link to coordination discovery in a multiagent FMMDP. Based on the theoretical results for efficient manipulation of the Bellman residual in Chapter 4, more complex functions are continuously introduced into the basis while maintaining a bounded distance of the ALP solution to the optimal value function V^*. Replicated is a search for coordination factor graphs that admit "good" solutions while enforcing sparsity of agent interaction. Unlike previous methods, coordination search is therefore treated as a by-product of achieving a desired bound on the solution to the full control problem. Since *the bound* is monotonically decreasing at every iteration (in the $\leq$ sense), the algorithm can further be interrupted at any time, allowing a trade-off between coordination complexity and value function solution bound.

5.4 Experimental Evaluation

In this section we evaluate the two classes of approximation algorithms developed in this chapter, *i.e.* the *regularized ALP* initialized with an 'overcomplete' basis set (*selection*) and the incremental basis expansion method (*generation*). Due to the aforementioned issues of scale with the former, focus of the evaluation is on basis generation, which allows computing bounded solutions to problems that could previously only be evaluated empirically through policy simulation. Of particular interest are the *coordination structures* discovered by the algorithm and their intuitive interpretation for the particular domain.

Outlined first are the performance metrics used during evaluation which depend on the availability of the optimal solution V^*. Second, we introduce the evaluation domains in detail. Of particular interest is a large *disease control* domain where multiple agents attempt to control

a stochastic disease process over a graph through application of vaccinations. We then show the principal applicability of the regularized ALP solution in domains where we retain an optimal solution for comparison. Experiments are then scaled up to significantly larger domains for evaluation of iterative coordination discovery.

5.4.1 Performance Metrics

One can broadly distinguish between metrics for *empirical evaluation* and those that analyze the error with respect to the optimal value function V^* analytically. In the latter category, if V^* is available, is the direct computation of the *relative error* between both value functions, $\|V^* - \mathbf{H}\hat{\mathbf{w}}\|_\infty / \|V^*\|_\infty$, *i.e.*, the maximum difference in value between approximation $\mathbf{H}\hat{\mathbf{w}}$ and V^* at any state $\mathbf{x}$. Normalization allows comparison across different domains. Note that this requires an exhaustive enumeration over all states $\mathbf{x}$. In our analysis for general factored MDPs, we use Spudd [46] to obtain the optimal reference value function and policy where problem sizes permit.

The second entry in this category—if V^* is not available or cannot be enumerated exhaustively—is a bound with respect to the optimal value function. As outlined in Chapter 2, the Bellman error offers a bound on the approximation error to V^*. The *normalized Bellman Error*, $\text{BellmanError}(\mathbf{H}\hat{\mathbf{w}})/R_{\max}$, for $R_{\max}$ denoting the maximum reward in any state $\mathbf{x}$, generally requires an exhaustive enumeration over all $\mathbf{x}$ but is computed efficiently for our approximation methods that enforce the sparse action-connectivity property at the solution (Chapter 4).

In the category of empirical measures fall *policy simulation* results. These refer to the (discounted) return computed by simulating policy $\hat{\pi}$ over an evaluation horizon, usually averaged over multiple runs. Policy simulation allows comparison with both the optimal policy π^* (where available), or other upper bounds or problem-specific heuristics. For empirical evaluation, we frequently refer to the online sample-based planner Prost [48], the winner of the International Planning Competition (IPPC) in 2011 and 2014 for *binary state/action domains*, for comparison in domains with moderate branching factor (*i.e.*, moderate $|\mathcal{A}|$). Where available, we refer to published results for other (approximate) solvers for a subset of the domains below.

5.4.2 Domains

Five domains are considered in our evaluation (detailed specifications for each are given in Appendix B). First, variants of the *SysAdmin problem*, originally introduced in [36] for predetermined basis functions, are considered, which commonly serve as baselines in other pub-

lished work. Evaluation with 10 agents in a star configuration (referred to as `sysadmin-star-10`) serves as a test case for which optimal solutions are available. Comparisons to other published approximate solutions are drawn and UCT-based solvers (such as Prost) are applicable for additional comparisons. The state and action space sizes are $|\mathcal{S}| = |\mathcal{A}| = 2^{10}$.

We then look at the same domain with more agents arranged in a different configuration (`sysadmin-ring-50`). This is a significantly more difficult domain for which no optimal policy is available. Without further approximations, UCT-based algorithms fail (or do not work well) in this domain due to the large branching factor in $|\mathcal{A}|$. Recent extensions to UCT that attempt to discover state-action symmetries fail in this domain as well [4]. We are interested in how automatic coordination discovery fares in this problem compared to the smaller 10 agent version. We are the first to report bounded solutions for this problem. State and action space sizes are $|\mathcal{S}| = |\mathcal{A}| = 2^{50}$.

A *ResourceProtect* domain is introduced that is a minor modification to the previous problem and assigns vastly more reward to a specific computer in the network. Our goal is to evaluate whether the discovered coordination structures change significantly based on this modification.

A *TaskNetwork* domain that is not binary and significantly more difficult than the previous domains is introduced, for both 6 and 20 agents. In a task network, each computer i is associated with two variables, one indicating its load ($|L_i| = 3$) and the other its health ($|S_i| = 3$) and the goal is to move tasks to completion in the network. To avoid confusion with the previous *SysAdmin* domain, we refer to this as a *TaskNetwork* instead of *Multiagent-SysAdmin* under which name it has also appeared [36]. The `tasknetwork-6` problem is the last instance for which an optimal policy can be obtained with Spudd. We are the first to present bounded solutions in `tasknetwork-20`. Additionally, we demonstrate how important basis selection (and coordination choice) is to achieve good policy performance in simulation. The larger of the domains has approximately $1.3 \cdot 10^{25}$ state-action pairs.

We then move to the significantly larger *DiseasePropagation* domain [45, 87] with up to 50 agents in a graph of 100 nodes. Here, stochastic dynamics that model a disease propagation over a graph are to be controlled through vaccinations. The evaluated domain has up to $1.4 \cdot 10^{45}$ state-action pairs.

Lastly, a targeted version of the above problem is considered where the disease is intended to be contained in a certain sub-region of a graph. We refer to this domain as *GraphContainment*. This domain is identical in complexity to *DiseasePropagation* and is detailed further in Appendix B.

5.4.3 Basis Function Selection

In this section we consider "exhaustive" insertion of basis functions into the ALP by a domain expert, followed by a sparse selection with the *regularized ALP*. The setting is the *TaskNetwork* domain with 6 computers arranged in a ring (each with ternary Status$_i$ and Load$_i$ variables), and 6 administrators that may reboot individual computers to avoid errors in task processing (*cf*. Appendix B). In total, there are $|\mathcal{S}|$ = 531,441 states and $|\mathcal{A}|$ = 64 actions in the domain. An optimal reference policy can be computed with Spudd [46] (solved for tolerance = 0.0001).

For exhaustive basis insertion, we consider the following sets of basis functions (arranged from "least" to "most complex" in terms of number and scope sizes):

H$_1$: An exhaustive set of indicator functions per individual state variable (referred to as *"individual basis"*). Since $|X_i| = 3 \ \forall i$ in the *TaskNetwork* problem, this corresponds to 3 functions per variable indicating that a particular state value is active. The value function and policy resulting from the ALP solution with this basis are referred to as $\hat{V}_1$ and $\hat{\pi}_1$.

H$_2$: The first half of computers is equipped with a *"pairwise basis"*, the second half with the individual basis. In a pairwise basis, (Status$_i$, Load$_i$) are considered jointly for each computer i and one indicator is inserted for every instantiation (*i.e.*, 9 per computer). The resulting value function and greedy policy are referred to as $\hat{V}_2$ and $\hat{\pi}_2$.

H$_3$: All computers are equipped with a pairwise basis. The resulting value function and greedy policy are referred to as $\hat{V}_3$ and $\hat{\pi}_3$.

H$_4$: Both pairwise and individual basis functions are inserted exhaustively for all computers. The resulting value function and greedy policy are referred to as $\hat{V}_4$ and $\hat{\pi}_4$.

Basis set choices 1-3 are all proper subsets of 4; the latter corresponds to a possible "exhaustive" basis choice of pairwise and individual features by a domain expert. We initially obtain the ALP solution for all sets 1-4 and then consider basis *selection* from the exhaustive set. Note that for the basis set choices here, the resulting Q-value components $Q_i(\mathbf{z}_i[\mathbf{x}], \mathbf{z}_i[\mathbf{a}])$ are all defined over individual action variables A_i (by virtue of backprojection in the *TaskNetwork* DBN). Hence, coordination between agents happens through jointly observed subsets of the state space.

Results for value function error measures and policy simulation for sets 1-4 are summarized in Table 5.1 and Figure 5.1, respectively. Both relative error (in terms of L_∞) and L_1 distance to V^* are monotonically decreasing for more complex basis choices in the example. Policies $\hat{\pi} \triangleq \text{Greedy}(\hat{V})$ are simulated for 50 trials of 200 steps each in Figure 5.1. Empirical policy

performance increases monotonically (in a $\geq$ sense) for basis choices 1-4. Note that sets 3 and 4 both perform equivalently to the (optimal) Spudd policy.

The Regularized ALP

In this section we use the *regularized ALP* together with exhaustive basis set 4 (90 basis functions in total) to evaluate the impact of different regularization factors λ on the solution. In addition to the value function metrics above, we show the number of enabled basis functions, $\|\hat{\mathbf{w}}\|_0$, and its (regularized) approximation $\|\hat{\mathbf{w}}\|_1$ for each ALP solution. Distinguished is also between the number of *pairwise* and *individual* basis functions active at the solution. Table 5.2 summarizes the effects of varying $\lambda \in [0,1]$ in the `tasknetwork-6` domain ($\beta = 0$). For tested choices of λ beyond 1.0, no further reduction in $\|\hat{\mathbf{w}}\|_1$ or change in solution quality were recorded. It is noticeable that with higher regularization penalty λ, more smaller-scope functions ($h_{|Dom(h)|=1}$) are enabled. This is in line with the previous intuition that spanning an "everywhere-close" approximation to the value function with fewer basis function biases the solution to those with larger coverage of the state space. In the example, the state space coverage of an *individual basis* function is 3 times as high as that of a pairwise function (given that $|X_i| = 3 \; \forall i$). Therefore, as the L_1 norm of solution vector $\mathbf{w}$ is penalized, a solution composed out of individual basis functions will cover a larger state space while encountering less penalty than the identical state space covered with pairwise basis functions. This trade-off between solution quality and basis function choice can be seen for increasing choices of λ in Table 5.2. The same pattern for enabled basis functions $h_{|Dom(h)|=1}$ and $h_{|Dom(h)|=2}$ could be confirmed with the alternative implementation of L_1 regularization through the constraints $|w_i| \leq c \; \forall i$.

For the policy simulation results in Figure 5.2, the most interesting cases are summarized by settings $\lambda = 0.1$ and 0.2 from Table 5.2. The former yields optimal policy performance during simulation while disabling a number of larger-scope basis functions. The latter results in near-optimal performance (shown on the right of the Figure) with only a third of the larger-scope functions active at the solution compared to the unregularized solution. In practice, regularization parameters λ and β require a parameter search (*e.g.* with a cross-validation method) to avoid overfit to a specific problem instance.

5.4.4 Basis Function Generation

In this section we move to evaluation of the iterative basis construction scheme in all five domains introduced above. All experiments are reported for the FeatureEvalBRS evaluation

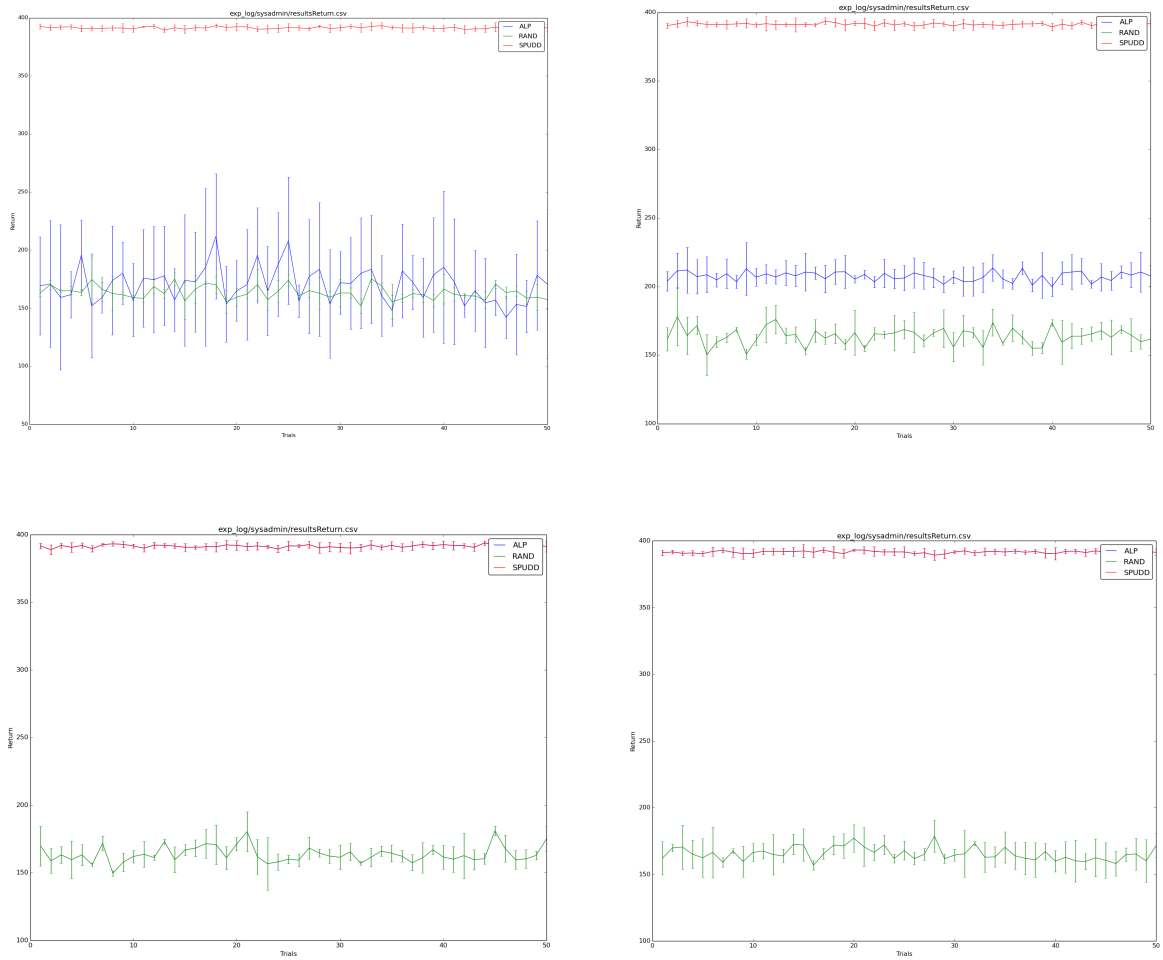

Figure 5.1: *From top left to bottom right:* Discounted values of policies $\hat{\pi}_1 - \hat{\pi}_4$, corresponding to ALP solutions with individual, partially pairwise, pairwise, and both individual and pairwise basis function sets (see text). Shown are results for 50 trials of 200 steps each with 95% confidence intervals computed from 5 runs, in comparison with random and optimal Spudd policies. Note that for both ALP solutions in the second row, the policy performance coincides with that of the optimal solution.

Basis set:	$\mathbf{H}_1$	$\mathbf{H}_2$	$\mathbf{H}_3$	$\mathbf{H}_4$
L_∞ w.r.t. V^*	3.966	2.420	0.756	0.756
L_∞ w.r.t. $V^* / \|V^*\|_\infty$	0.181	0.110	0.034	0.034
L_1 w.r.t. V^*	572,079	325,047	78,014	78,014
t_{comp}	0.09 s	0.12 s	0.25 s	0.46 s

Table 5.1: Value function error measures for ALP solutions corresponding to basis sets 1-4 (see text). For comparison, Spudd's computation time (tolerance = 0.0001) is t_{comp} = 5260 s on the same machine.

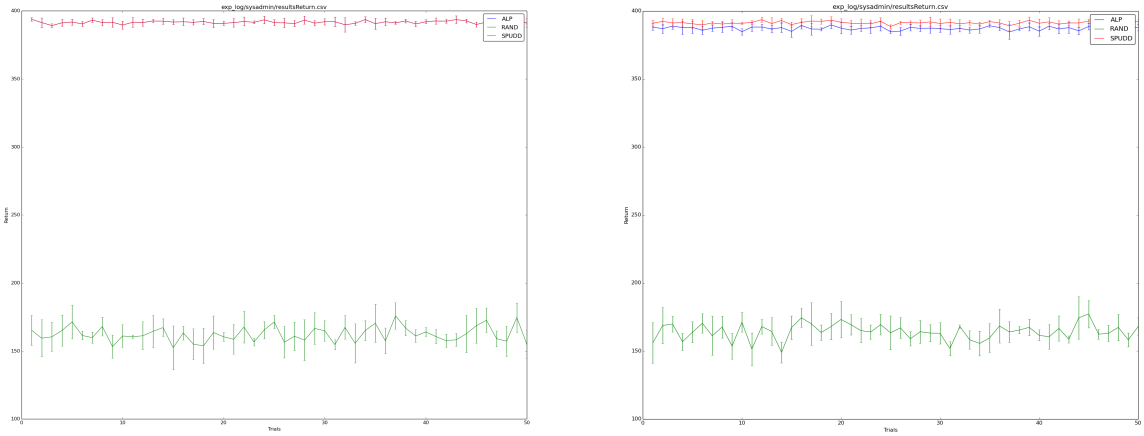

Figure 5.2: Discounted value of $\hat{\pi}$ from *regularized ALP* solutions with $\lambda = 0.1$ (*left*) and $\lambda = 0.2$ (*right*). Shown are results for 50 trials of 200 steps each, averaged over 5 runs, in comparison with random and optimal Spudd policies. Note that the former coincides with the optimal policy whereas the latter is near optimal with one third of the pairwise basis functions enabled in the solution vector $\hat{\mathbf{w}}$.

Regularization λ:	**0**	**0.1**	**0.2**	**1.0**
L_∞ w.r.t. V^*	0.756	0.756	1.664	1.664
L_1 w.r.t. V^*	78,014	78,014	217,582	217,582
$\|\hat{\mathbf{w}}\|_1$	74.693	61.716	59.531	59.531
$\|\hat{\mathbf{w}}\|_0$ (*Note:* $\|\hat{\mathbf{w}}\| = 90$)	46	49	37	40
$\|\hat{\mathbf{w}}_{\|Dom(h_i)\|=2}\|_0$, $\|\hat{\mathbf{w}}_{\|Dom(h_i)\|=1}\|_0$	$\{33, 13\}$	$\{24, 25\}$	$\{12, 25\}$	$\{15, 25\}$

Table 5.2: Comparison of different regularization factors λ in `tasknetwork-6` and their effect on the *regularized ALP* solution seeded with the exhaustive basis set $\mathbf{H}_4$ of both individual and pairwise basis functions (see text).

criterion after it outperformed the FeatureEvalEpsilon function consistently in our tests.

Sysadmin-star-10

We consider an application of the coordination discovery method to the *SysAdmin* baseline instance with 10 computers arranged in a star configuration. When applying the coordination discovery method, we obtain the optimal policy after 9 iterations (see value function error measures in Table 5.3, simulation results in Figure 5.3, and the *discovered coordination structures* shown as a succession of CFGs in Figure 5.4 on page 83). Bounded solutions, and value function error measures with respect to the optimal Spudd policy V^*, are available for this problem and

shown in Table 5.3 for every iteration of the algorithm. The optimal policy is achieved with 29 locally-scoped (at most pairwise) factors. *Empirically*, the policy in this simple problem is optimal even before the first iteration of basis construction (*i.e.*, with the basis centered on individual state factors only), as can be seen in the policy simulation results in Figure 5.3. In summary:

- No domain knowledge for basis construction was inserted into the problem. Discovered coordination structures mimic pairwise coordination with the center node in the problem and follow the intuition that this is the most important node in the network (if it goes down, all computers are affected)
- The process can be halted with a bounded solution as per the desired degree of agent coordination.
- Reported in [85] are policy simulation results for approximate symbolic dynamic programming (SDP) methods in the `sysadmin-star-10` domain. All are outperformed by the basis discovery method. Prost achieves equivalent results to the optimal policy during simulation, albeit without a bound on the solution.
- Solution times (not shown) outperform Prost in practice. Further, the obtained value function and policy is planned offline over the entire state space and requires no search at runtime.

Iteration	Bellman error	ALP Objective	Basis choice	Bound $\|V^* - \mathbf{Hw}\|_\infty$	Actual $\|V^* - \mathbf{Hw}\|_\infty$
0	2.48031	84.0909	$h_{10} = X_0 \wedge \overline{X}_2$	22.32279	3.77192
1	3.01435	83.8496	$h_{11} = \overline{X}_0 \wedge X_5$	27.12915	4.12898
2	2.63756	83.6083	$h_{12} = \overline{X}_0 \wedge X_3$	23.73804	3.61286
3	2.26077	83.367	$h_{13} = \overline{X}_0 \wedge X_6$	20.34693	3.09674
4	1.88397	83.1257	$h_{14} = \overline{X}_0 \wedge X_1$	16.95573	2.58062
5	1.50718	82.8844	$h_{15} = \overline{X}_0 \wedge X_8$	13.56462	2.06449
6	1.13038	82.6431	$h_{16} = \overline{X}_0 \wedge X_9$	10.17342	1.54837
7	0.753588	82.4018	$h_{17} = \overline{X}_0 \wedge X_7$	6.782292	1.03225
8	0.376794	82.1605	$h_{18} = \overline{X}_0 \wedge X_4$	3.391146	0.516123
9	4.57412E-14	81.9192	$h_{19} = \overline{X}_0 \wedge \overline{X}_2$	4.116708E-013	0
10	4.57412E-14	81.9192	—	4.116708E-013	0

Table 5.3: (`sysadmin-star-10`) Iteration 0 refers to the ALP before incremental basis construction. Denoted in the *Basis choice* column is the conjunctive basis added at the next iteration. It can be seen in iteration 1 that the Bellman error increases, which verifies that the ALP optimization of the L_1 norm does not, in general, monotonically decrease the Bellman error. Bounds and resulting factor graphs are visualized in Figures 5.3 and 5.4, respectively.

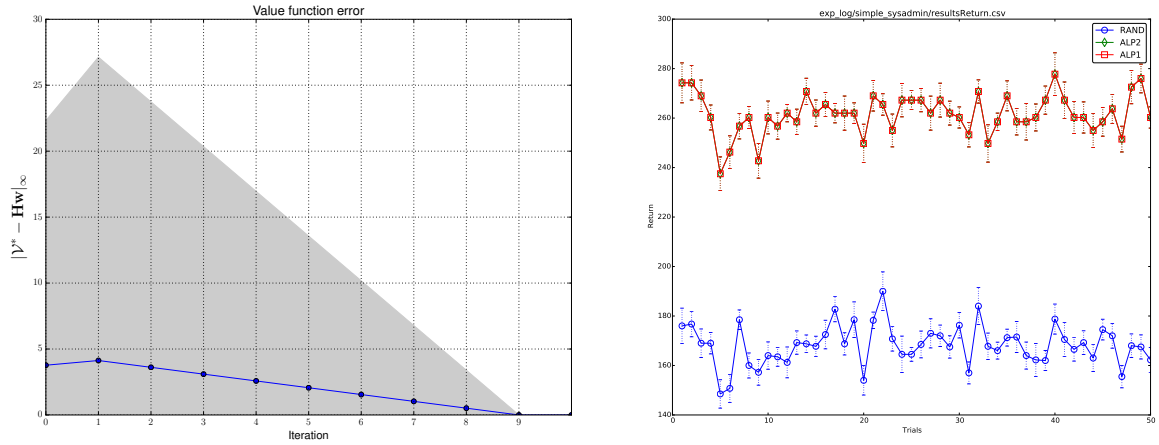

Figure 5.3: (`sysadmin-star-10`) *Left:* Value function error for 10 iterations of basis construction. Plotted is the *actual* error $\|V^* - \mathbf{Hw}\|_\infty$ (in blue) versus the Bellman errror bound on the same metric (in gray). Note that the maximum return is bounded by $R_{\max}/(1-\gamma) = 100$ in this problem setup. The optimum policy is obtained after 9 iterations. *Right:* Mean return for 50 trials of 30-step policy simulation for a random policy and two ALP policies: ALP1 and ALP2 coincide in the figure and show policy performance before and after 10 iterations of incremental basis construction. Both yield the same (*i.e.*, optimal) performance during simulation. All results are averaged over 10 runs and are shown with 95% confidence intervals.

It is interesting to see that the bound shown in Figure 5.3 is rather pessimistic compared to the actual error achieved with respect to the known optimal value function in this problem.

Sysadmin-ring-50

This is a larger domain for which no optimal policy is available. 50 agents are arranged in a ring structure without a central node. UCT-based methods do not apply without further approximations over the exponential action space ($|\mathcal{A}| = 2^{50}$). Prost cannot serve as a baseline in this experiment since it runs out of memory (specifically, during the conversion step into its proprietary representation).

We apply our coordination discovery method to this scenario. For the Bellman error to remain computable *exactly*, we restrict the maximum number of 'action-connected' state factors to 12 (*cf*. Chapter4), enforcing a *sparsity property* on the coordination graph as per our working assumption that such a sparse interaction structure approximates the true value function well. For problems where the optimal value function is not available, the actual error $\|V^* - \mathbf{H}\hat{\mathbf{w}}\|_\infty$ cannot be computed. Based on our efficient computation of the Bellman error, we do maintain the ability to *bound the solution* with respect to V^* and present the first such solutions for this

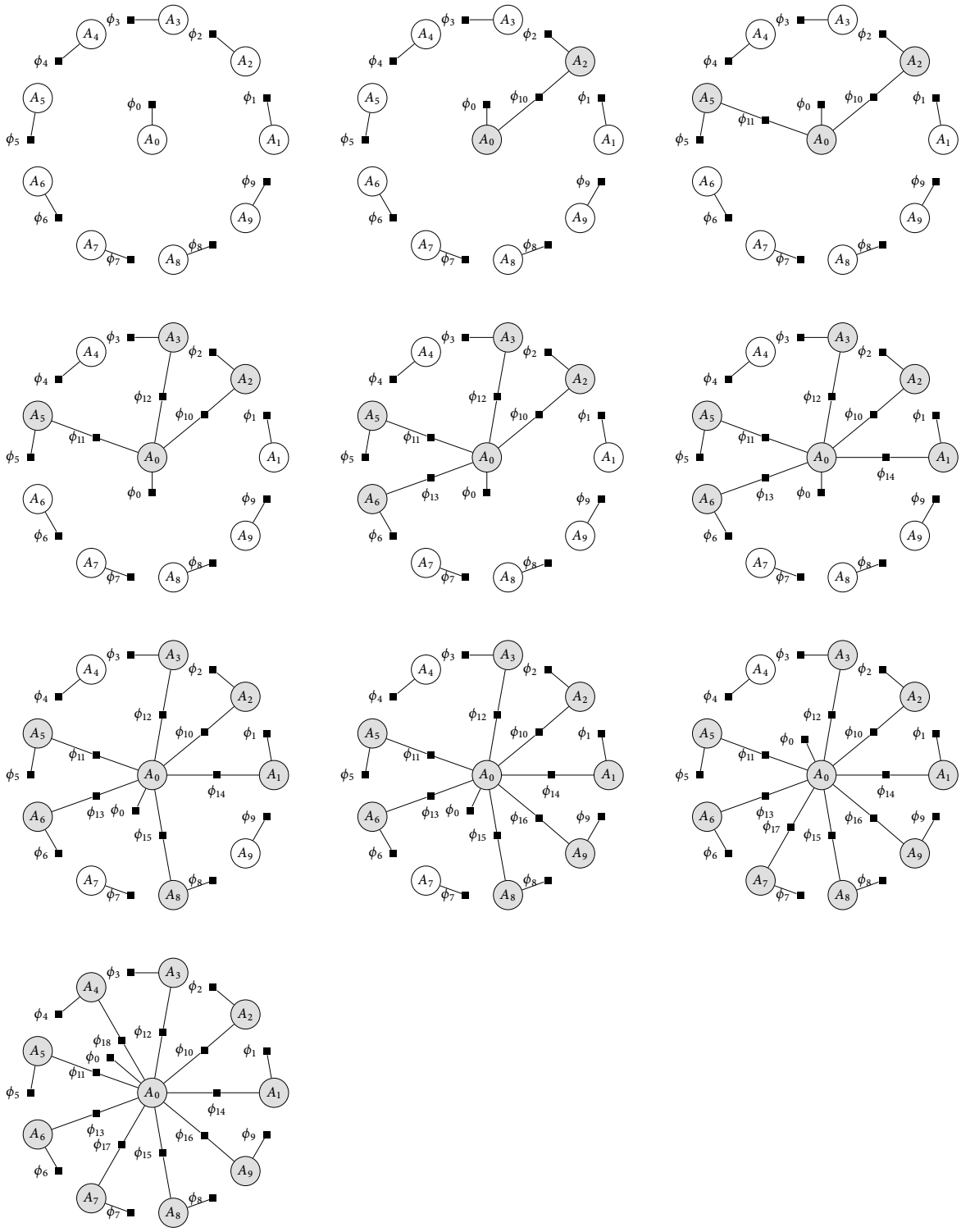

Figure 5.4: (`sysadmin-star-10`) Obtained factor graphs from 10 iterations of basis construction in the *SysAdmin* domain with 10 computers connected in a star layout. Agents participating in *pairwise factors* are highlighted in gray. Note that the last iteration is omitted since the policy has reached optimality in the previous iteration.

problem.

A common metric to assess the results when the optimal policy is not available is the *normalized Bellman Error (BE)*, defined as $\text{BellmanError}(\mathbf{H}\hat{\mathbf{w}})/R_{\max}$. The BE is commonly not available for large problems due to the max over exponentially many states – we compute it efficiently during coordination discovery with a variable elimination procedure. Note that the loss in return with respect to the optimal value function V^* in any state is then bounded by $\|V^* - \hat{V}\|_\infty \leq \frac{\gamma}{1-\gamma}\text{BellmanError}(\hat{V})$, which can similarly be compared to the upper bound on the (infinite, discounted) return in any state, $R_{\max}/(1-\gamma)$. For the evaluation in simulation over h timesteps, $R_{\max} \cdot h$ yields an optimistic upper bound if all computers were constantly running and no reboot action ever occurred.

Table 5.4 shows 50 iterations of coordination discovery in this domain at increments of 10 iterations. Adjacent nodes in the ring are continuously joined into 'action-connected' subsets while enforcing the upper bound on the maximum partition size. Within each partition, *pairwise factors* join adjacent nodes into a connected chain, intuitively corresponding to "subchains" of the original 50-node ring (without having provided such domain knowledge).

Iteration	Bellman error	Bound $\|V^* - \mathbf{Hw}\|_\infty$	Node partitions in ring (link 0-49 closes the ring)
0	10.7667	96.9003	$\{0\}, \ldots, \{49\}$
1	9.12724	82.14516	$\{0\}, \ldots, \{17, 18\}, \ldots, \{49\}$
10	7.57585	68.18265	$\{0\}, \ldots, \{8-18\}, \ldots, \{49\}$
20	6.52235	58.70115	$\{2-7\}, \{8-18\}, \{36, 37\}, \{38, 39\}, \{44, 45\}, \{47, 48\}$ (Rest: single)
30	4.65677	41.91093	$\{0-7\}, \{8-18\}, \{28, 29\}, \{30-39\}, \{44, 45\}, \{47, 48\}$ (Rest: single)
40	2.79094	25.11846	$\{8-18\}, \{20-29\}, \{30-39\}, \{44, 45\}, \{47-7\}$ (Rest: single)
50	2.04514	18.40626	$\{8-18\}, \{19-29\}, \{30-39\}, \{40-46\}, \{47-7\}$

Table 5.4: (`sysadmin-ring-50`) Iteration 0 refers to the ALP before incremental basis construction with only single basis functions per state factor. Shown in the last column are the sets of nodes joined by conjunctive basis functions up to that iteration. Note that within each set, pairwise factors connect the nodes into sub-chains of the original ring.

After 50 iterations this approximation architecture reaches a plateau for the Bellman error of 2.05 (compare to $R_{\max} = 50$) and a bound on the value function error of 18.41 (compare to $R_{\max}/(1-\gamma) = 500$). Figure 5.5 shows the *relative Bellman error* for all 50 iterations of coordination discovery and Figure 5.6 policy simulation results over 30 timesteps. As for the previous smaller domain, only the bound is significantly decreased; the performance in our simulations is identical before and after coordination discovery in this problem.

In summary, introducing collaborating neighboring agents as per the partitions shown in Table 5.4 reduces the bound on the value-function loss by a factor of 5.

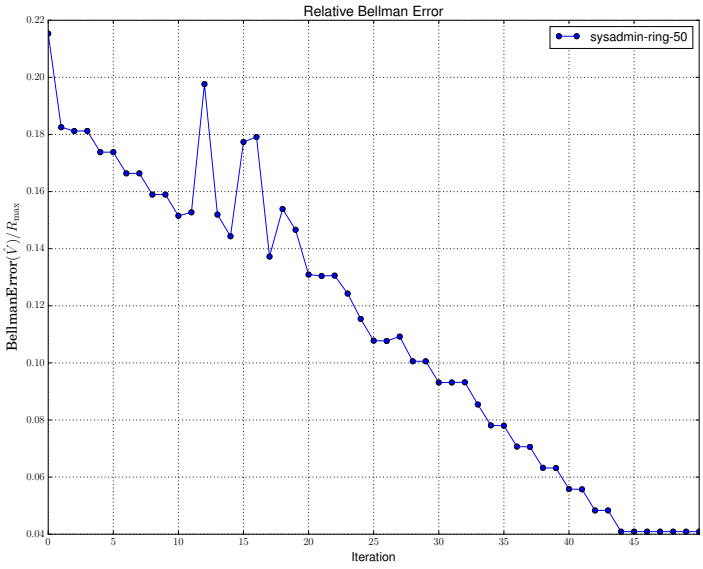

Figure 5.5: (`sysadmin-ring-50`) Relative Bellman error for 50 iterations of coordination discovery. Note the intermediate increases in Bellman error where the ALP solution is not the best possible approximation (in a L_∞ sense) given the basis: the optimization of the weighted L_1 norm does not, in general, *monotonically* decrease the Bellman error.

Resourceprotect-ring-50

We briefly experimented with a variation of the previous domain where the first computer in the network (node 0, which is connected to nodes 49 to the left and 1 to the right) is more important than all others (by a factor of 5). This domain has the same complexity as the previous one; of interest here is only a possible change in the agent coordination structure to account for the more valuable resource.

After the first 10 iterations, node 0 is covered by the largest connected chain formed: $\{45-1\}$ (besides connections $\{15-18\}$ and $\{36,37\}$). At iteration 22, that sub-chain has been further expanded to cover nodes $\{41-1\}$. Based on our domain knowledge, nodes $n-1$ (*i.e.*, neighbors to the left of a node n in the ring) affect the status of node n at the next iteration. The discovered coordination structure resembles that property of the domain. It is intuitive that a good policy in this domain (*i.e.*, one associated with low Bellman error) should yield agent coordination around the important resource to maximize its uptime as is confirmed by this experiment.

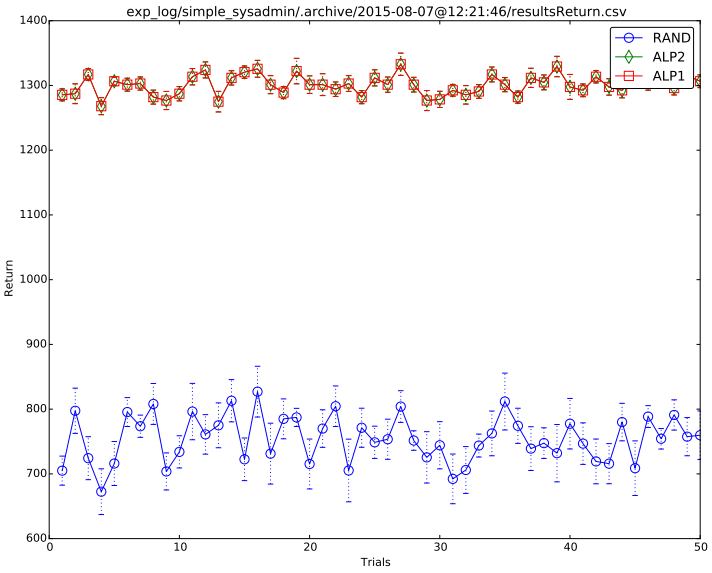

Figure 5.6: (`sysadmin-ring-50`) Mean return for 50 trials of 30-step policy simulation for a random policy and two ALP policies; no optimal reference policy is available: ALP1 and ALP2 coincide in the figure and show identical policy performance before and after 50 iterations of incremental basis construction. All results are averaged over 10 runs and are shown with 95% confidence intervals. Note the (optimistic) upper bound of 1500 in this setup for constantly running computers and no reboot actions.

Tasknetwork-6

In the *TaskNetwork* domain, a computer network completes jobs based on both status and current load at each machine (each represented with ternary variables as detailed in the full domain description in Appendix B). The load variable at each computer indicates task progress and a reward of 1 is collected only if a task continues uninterrupted over two processing steps.

As shown for the optimal policy performance over $h = 200$ steps in Figure 5.9, it is much harder to achieve high reward in this domain (average reward of approximately 400 compared to the optimistic upper bound $R_{\max} \cdot 200 = 1200$). The case of $N = 6$ machines, where state and action spaces have size $|\mathcal{S}| = 3^{12}$ and $|\mathcal{A}| = 2^6$, respectively, is the last one for which we could obtain an optimal policy with SPUDD.

Basis discovery is run for 20 iterations after which the Bellman error measure reaches a plateau (stopped manually after 30 iterations). As shown in Table 5.5, all constructed features increase complexity over the *same computer* through pairwise factors over the S_i, L_i variables. Only during the last iterations are pairwise factors across other machines' status variables introduced, interestingly between equally distant machines in the network (see Figure 5.8). Given the

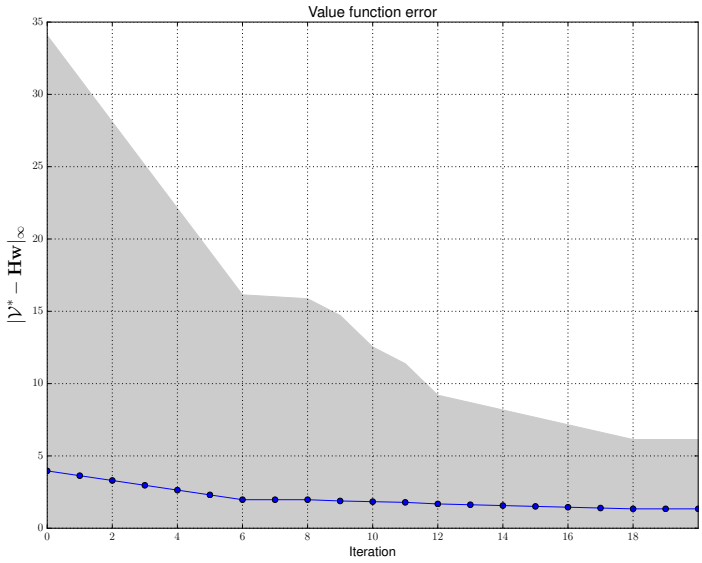

Figure 5.7: (`tasknetwork-6`) Actual (blue) versus bound (gray) on value function error for 20 iterations of basis construction. Note that here $R_{\max}/(1-\gamma) = 60$.

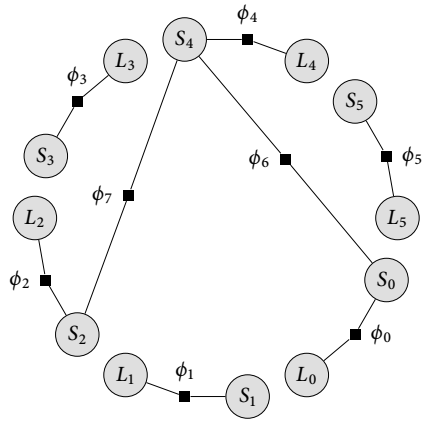

Figure 5.8: (`tasknetwork-6`) Discovered coordination structure at iteration 20. The last two factors ϕ_6, ϕ_7 are added in the final two iterations but do not improve the empirical (and theoretical) performance of the policy during simulation (see Figure 5.9).

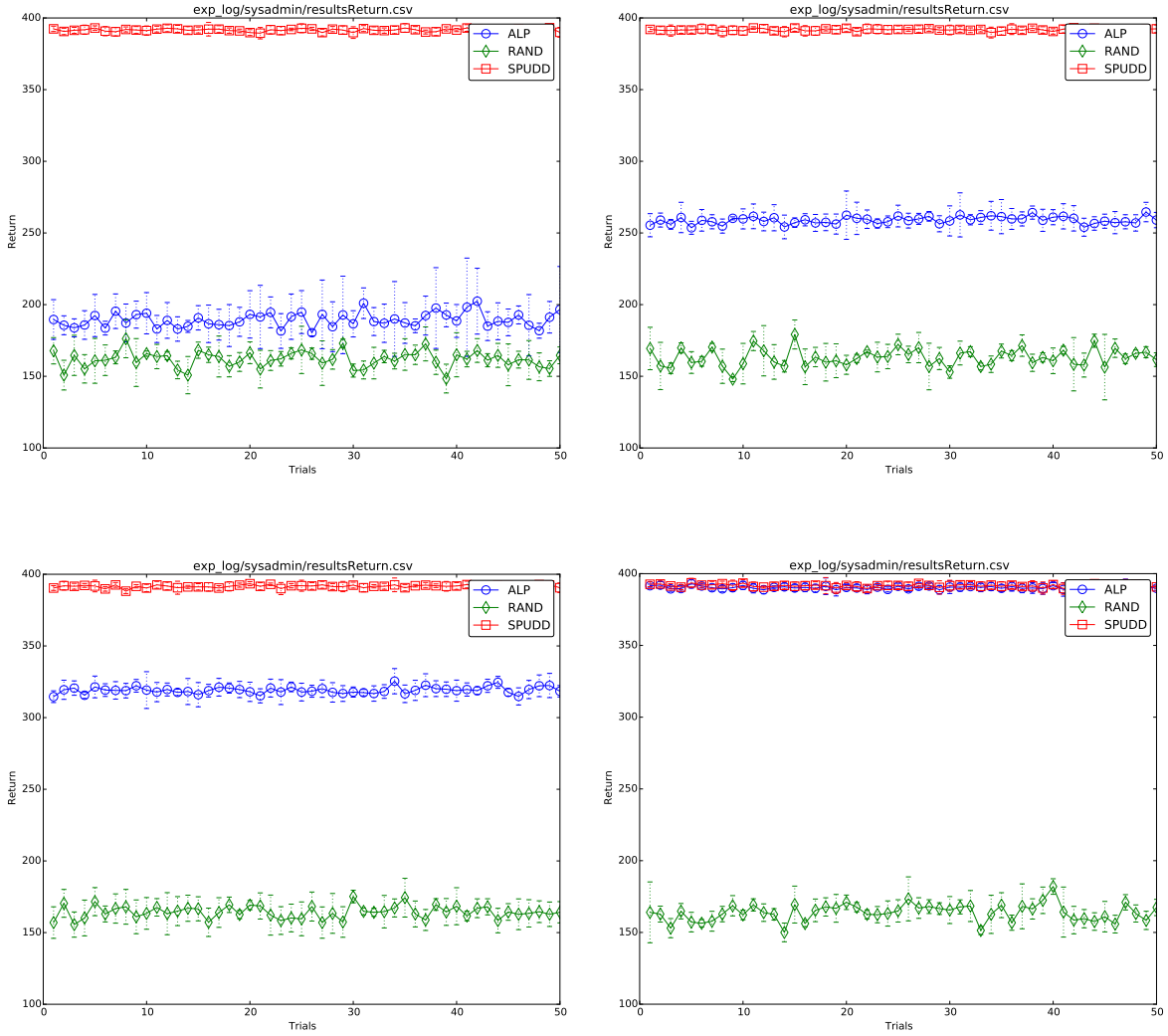

Figure 5.9: (`tasknetwork-6`) *From top left to bottom right:* mean return for 50 trials of (here) **200**-step policy simulation comparing random and optimal policies to the ALP solution after a selection of 1, 10, 11, and 17 iterations of basis discovery. All results are averaged over 5 runs and are shown with 95% confidence intervals.

bounds on the value function error shown in Figure 5.7, a sparse coordination graph *without any inter-agent connections* suffices to guarantee the same performance bound. Figure 5.9 shows the empirical policy performance compared to the optimal Spudd policy, confirming this intuition (approximately optimal performance is achieved from 15 iterations onward).

Iteration	Bellman error	Bound $\|V^* - \mathbf{Hw}\|_\infty$	Actual $\|V^* - \mathbf{Hw}\|_\infty$	Factor graph partition
0	3.78598	34.07382	3.96567	$\{L_0\}, \{S_0\}, \ldots, \{L_5\}, \{S_5\}$
5	2.12546	19.12914	2.30515	$\{L_0, S_0\}, \{L_1, S_1\}, \{L_2, S_2\}, \{L_4, S_4\}, \{L_5, S_5\}, \{L_3\}, \{S_3\}$
10	1.39284	12.53556	1.83688	$\{L_0, S_0\}, \{L_1, S_1\}, \{L_2, S_2\}, \{L_3, S_3\}, \{L_4, S_4\}, \{L_5, S_5\}$
15	0.851845	7.666605	1.51027	$\{L_0, S_0\}, \{L_1, S_1\}, \{L_2, S_2\}, \{L_3, S_3\}, \{L_4, S_4\}, \{L_5, S_5\}$
20	0.681476	6.133284	1.3399	$\{L_0, S_0, L_2, S_2, L_4, S_4\}, \{L_1, S_1\}, \{L_3, S_3\}, \{L_5, S_5\}$

Table 5.5: (`tasknetwork-6`) Iteration 0 refers to the ALP before incremental basis construction. Discovered partitions join S_i, L_i corresponding to each individual agent i.

Tasknetwork-20

This domain is significantly larger than the previous ones at $|\mathcal{S}| = 3^{40}$ and $|\mathcal{A}| = 2^{20}$ (approximately $1.3 \cdot 10^{25}$ joint state-action pairs). No optimal policy is available; we are the first to show bounded solutions to this problem. It also serves to demonstrate the importance of good basis choice, be it through automated discovery or manually by a domain expert.

The Bellman error reaches a plateau after 62 iterations at which time it has been reduced by a factor of 6 compared to the non-optimized basis. Interestingly, Figure 5.10 (bottom) shows a similar curve like the bound for 6 machines shown previously in Figure 5.7: the relative BE decreases linearly until all S_i, L_i are covered by pairwise factors (here at iteration 20 compared to iteration 6 before). The ALP policy converges to one with an average payoff of 180 ($h = 30$) after approximately 40 iterations of basis discovery. We re-ran the previous experiment for the same horizon (results on the left of Figure 5.11) and obtain an average reward of 54 which directly mimics the ratio in machine numbers. This leads us to believe that the optimal policy has in fact also been discovered for this domain.

To show the difficulty of the search for a good basis set, we shift the full set of pairwise factors to the right to cover adjacent state factors L_i, S_{i+1} instead, essentially introducing coordination links between adjacent agents. The lack of performance (*cf*. right of Figure 5.11) validates the criticism raised previously against manual basis selection for the ALP, *i.e*. that a domain expert is required to choose a problem-specific basis set. Coordination discovery resolves this concern with a principled method for basis discovery (respectively, coordination search in the multiagent setting) that maintains bounds on the value function error.

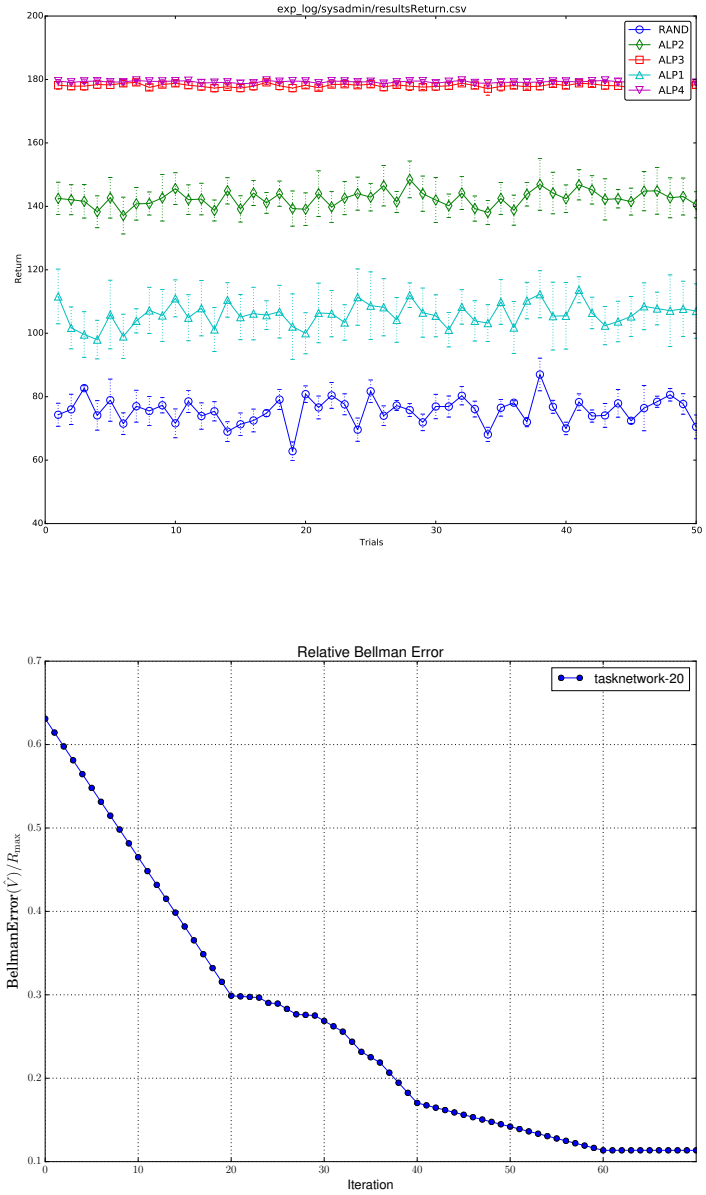

Figure 5.10: (`tasknetwork-20`). *Top:* Comparison of 4 ALP policies (after 0, 30, 40, 70 iterations) to the random policy (h = 30). Note that policies for iterations 40 and 70 coincide in the Figure. *Bottom:* relative Bellman error over 70 iterations of basis discovery. Note the similarity in shape to the bound for the smaller version of this problem in Figure 5.7 (see text).

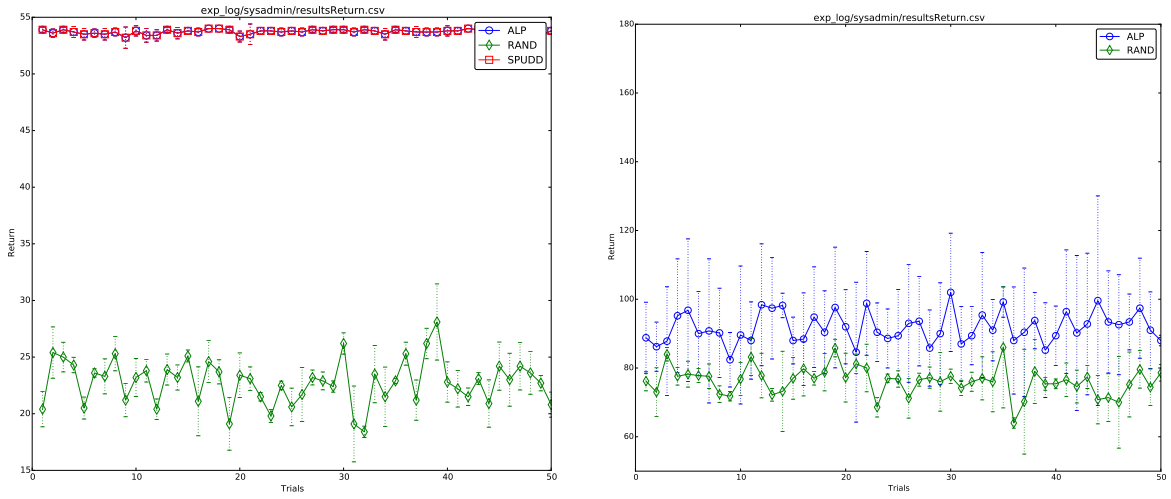

Figure 5.11: *Left* (`tasknetwork-6`): mean return for 50 trials of (here) 30-step policy simulation comparing random and optimal policies to the ALP solution at iteration 20. *Right* (`tasknetwork-20`): the effect of a suboptimal basis choice, outlining the need for a domain expert or automated basis discovery – here, pairwise factors are shifted to the right to cover L_i, S_{i+1} in the ring.

DiseasePropagation

We now apply the same method to a disease propagation setting with up to 50 agents in a graph of 100 nodes. This is the most complex of the domains considered with up to $1.4 \cdot 10^{45}$ state-action pairs (see Appendix B). Agents perform vaccinations in a generic network of nodes in order to quickly cancel out the effects of a disease process over a graph. Previous results (for up to 3 agents where MCTS yielded empirically the best policies) are available in [45]. Our contributions concern the extension to many more agents and the evaluation of coordination requirements between those, and the availability of bounds on the solution.

As outlined in the algorithm description in Section 5.3.1, three steps are iterated: *i)* Representing the constraints and solving the ALP given a basis set, *ii)* computing the Bellman error exactly given the (sparse) coordination graph, and *iii)* determining the next best basis to introduce. Steps *i)* and *ii)* require a variable elimination (VE) procedure (exponential in the largest clique size formed during VE) whereas *iii)* only requires the computation of the Bellman residual marginal "covered" by a candidate basis, which is, in general, less complex (exponential in the chosen limit on the 'action-connected' state factors; a parameter to the basis discovery algorithm which is fixed at 12 for the experiments here). Note that the issues of scale for *i)* and *ii)* are revisited in a later chapter on exploiting "anonymous influence" in large graphs with stochastic

Node ID	out-degree	controlled	Node ID	out-degree	controlled
35	6	–	69	4	✓
87	5	✓	45	4	✓
22	5	–	50	4	–
12	4	–	18	4	–
72	4	–	25	4	–

Table 5.6: The ten nodes with the highest out-degree in the *DiseaseProp* graph. Column "*controlled*" refers to whether an agent is present or not (*cf*. Figure 5.13).

dynamics. We provide timing results for each step of the algorithm in the experiments.

Figure 5.13 shows the graph used for the experiments. 15, 25, and 50 controlled agents are placed at random locations in the graph. In the first set of experiments, the goal is to quickly cancel out the effects of the disease. Since, to the best of our knowledge, no comparative results are available, we formulate a problem-specific heuristic (referred to as "`copystate`") which immediately administers a vaccination at each infected, controlled node. Note that this policy is reactive and cannot administer a vaccination proactively if nodes further up in the network are already infected. In the results we omit a random policy since it is outperformed by a wide margin in all experiments.

In problems of this scale, the ALP solution becomes a limiting factor (see the timing results for the comparable *GraphContainment* problem in Figure 5.15). As basis functions are inserted that span multiple nodes, (exact) variable elimination in the graph suffers from generally larger scope sizes during elimination. Note in the same Figure that computation times for *feature evaluation* (*i.e.* step *iii)* of the algorithm) is not affected due to the efficient computation of the Bellman marginal without a VE procedure. Given the increase in ALP solution times, we complete 24 iterations of basis discovery and evaluate the resulting policies for 15, 25, and 50 agents against the `copystate` heuristic.

No links between agents are introduced during 24 iterations of basis discovery. Similarly to the case for *GraphContainment* (see Table 5.7 in the next section), nodes appearing in pairwise factors during basis discovery resemble the "most important" vertices (as per their out-degree) in the graph.

The reduction in relative Bellman error of approximately 10% after 24 iterations does not have an effect on the empirical policy performance (see Figure 5.12). All ALP policies coincide and outperform the heuristic significantly: the mean return of the ALP policy with 15 agents matches that of the *heuristic with 50 agents*. We experimented with a second version of the problem (not shown) for a random re-shuffling of agents in the graph and note that the absolute

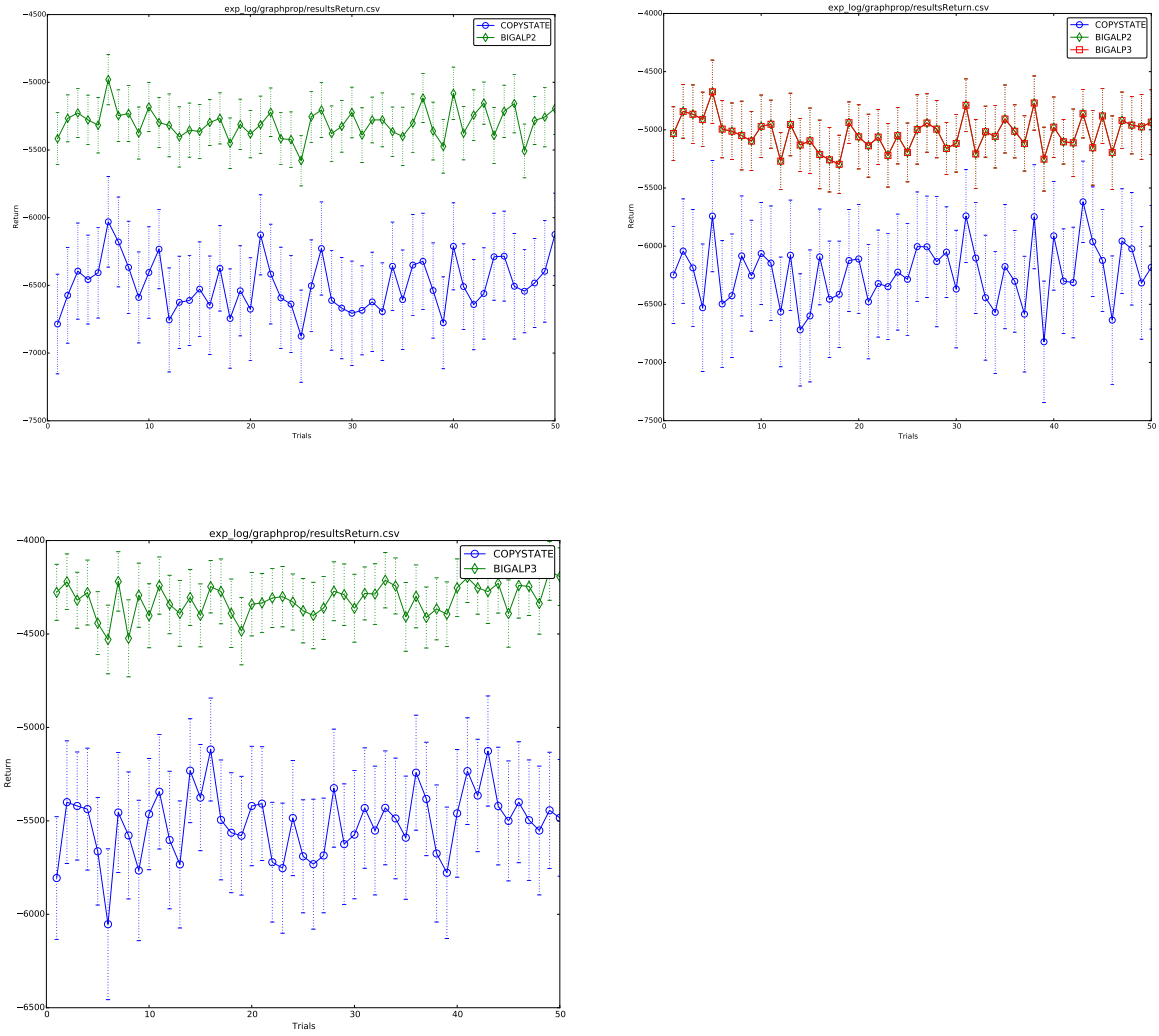

Figure 5.12: (`diseaseprop-100` with 50 agents). *From top left to bottom right:* Mean return for 50 trials of 30-step policy simulation for ALP policy and `copystate` heuristic for 15, 25, and 50 agents in the graph (see text). *Note that the mean return of the ALP policy with 15 agents matches that of the heuristic with 50 agents.*

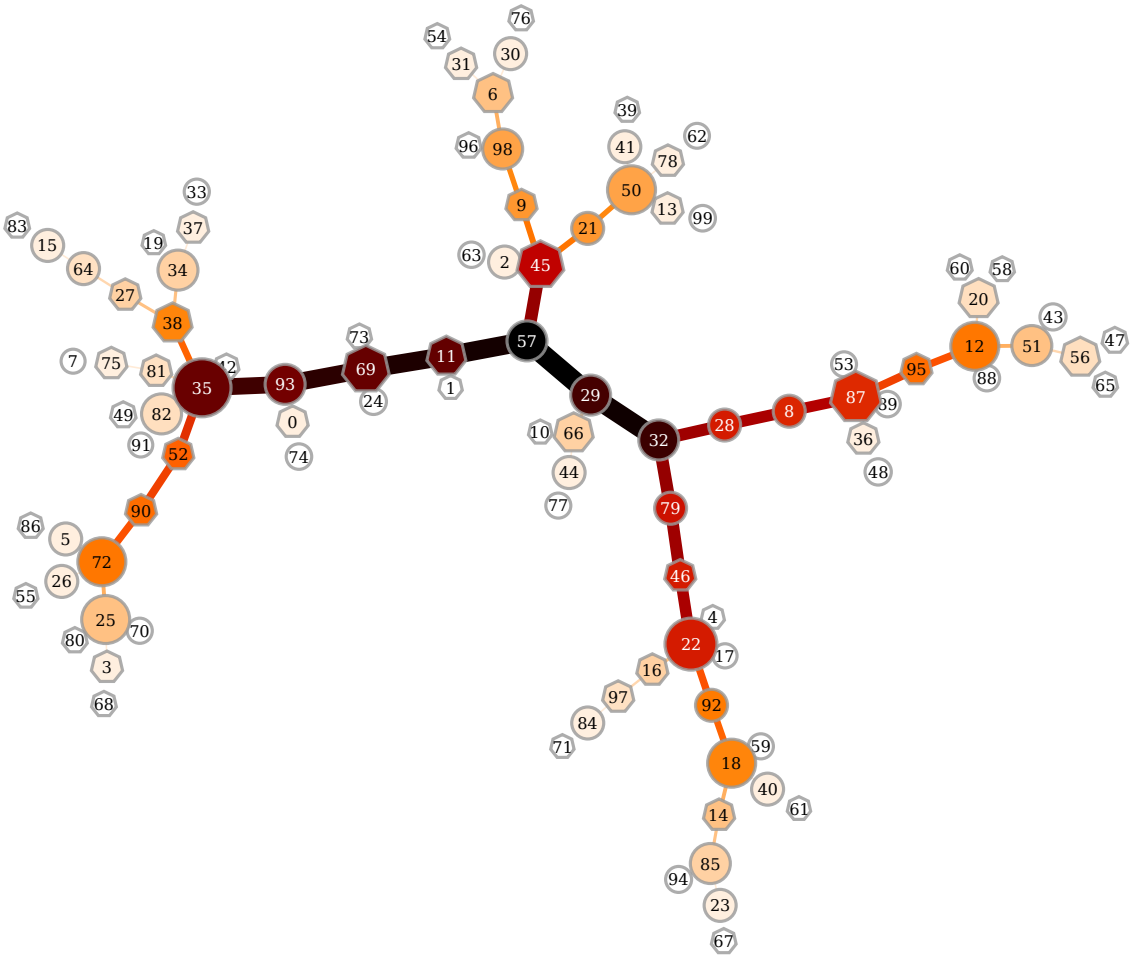

Figure 5.13: Example graph for *DiseaseProp* and *GraphContainment* domains with 100 nodes and 50 randomly placed agents ($|\mathcal{S}| = 2^{100}$, $|\mathcal{A}| = 2^{50}$). The following properties are visualized in the graph:

- controlled (*i.e.*, agent) nodes are surrounded with hexagon shapes, uncontrolled nodes (where the process may propagate undisturbed) with circles,
- each node's size is proportional to its out-degree in the network $\in [1, 6]$,
- edge width and color (heat map from black to white) indicate its "betweenness" centrality property, *i.e.* the number of shortest paths from all vertices to all others that pass through it,
- analogous for a vertex' color which indicates its "betweenness" centrality.

return depends strongly on agent placement in the graph: the re-shuffling resulted in a reduction of mean return by approximately 2000.

GraphContainment

In the *GraphContainment* domain, the process may propagate without penalty among all 50 *uncontrolled* nodes in the graph whereas the same penalty as previously occurs at infected *controlled* nodes. The goal is to contain the process in one half of the graph and controlled nodes have to trade off a vaccination action (at a cost) with letting the process pass through to "penalty-free" regions of the graph.

Table 5.7 shows the results for 20 iterations of basis discovery, along with Bellman error bounds on the value function and a summary of the pairwise factors that are introduced by basis discovery. The Bellman error and the value function bound can be compared to $R_{\max} = 0$ (respectively, the maximum cost per step of −2550), and analogously the maximum accumulated cost of −25500 in this domain ($\gamma = 0.9$), yielding a tight guarantee on the error.

Similarly to the previous domain, increasing ALP solution times only allow us to solve 20 iterations of basis discovery, yielding a reduction of the error bound by approximately 10%. To the best of our knowledge, a bounded solution to a problem of this size (without further restrictions on the policy) were not shown previously.

Iteration	Bellman error	Bound $\|V^* - \mathbf{Hw}\|_\infty$	Nodes in pairwise factors (and their occurrence count if > 1)
0	118.233	1064.097	–
5	118.181	1063.629	34, 35(3), 72(2), 93(2), 98(2)
10	110.872	997.848	12(2), 34, 35(**6**), 50(2), 72(2), 84, 93(2), 98(**4**)
15	109.645	986.805	12(**3**), 15, 34, 35(**10**), 50(**3**), 64, 72(2), 84, 85, 92, 93(2), 98(4)
20	106.165	955.485	12(**4**), 15, 22(**4**), 25, 34(**2**), 35(**11**), 50(3), 64, 72(2), 84(**2**), 85, 92, 93(2), 98(**5**)

Table 5.7: (`graphcontainment-100` with 50 agents). Iteration 0 refers to the ALP before incremental basis construction with single basis functions per state factor. Shown in the last column are the nodes that appear in pairwise factors up to that iteration, along with the number of times they appear (in brackets and highlighted in bold if changed from previous row).

Interestingly, nodes appearing in pairwise factors during basis discovery resemble the "most important" vertices (as per their out-degree) in the graph (*cf*. Table 5.6). Further, 20 iterations of basis discovery in this domain did not cause links between agents to be introduced in the coordination graph.

Results for policy simulation are summarized in Figure 5.14. The reduction in error bound did not result in an empirical performance increase during simulation: all ALP solutions coincide and outperform the `copystate` heuristic by a larger margin than in the previous domain.

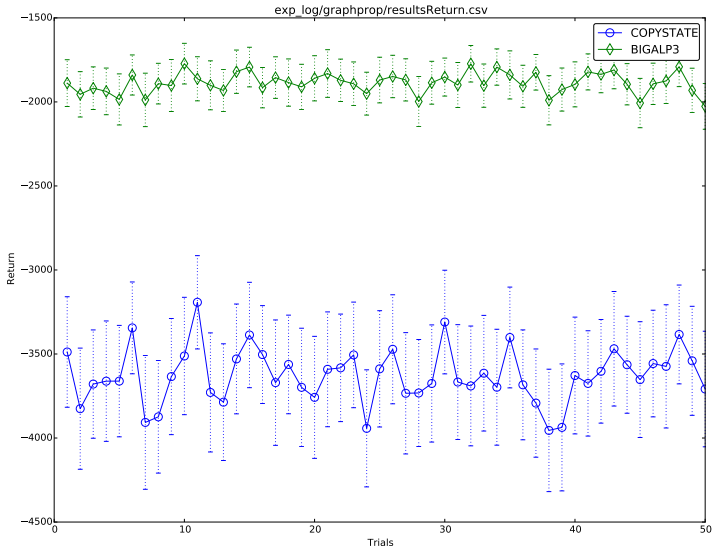

Figure 5.14: (`graphcontainment-100` with 50 agents). Mean return for 50 trials of 30-step policy simulation for ALP policy and `copystate` heuristic.

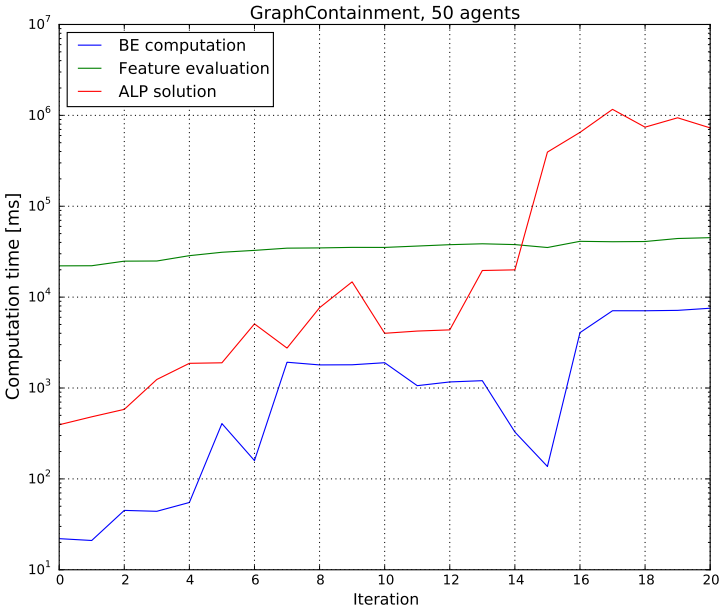

Figure 5.15: (`graphcontainment-100` with 50 agents) Computation time (in ms) for each of *Bellman error computation*, *Feature evaluation*, and *ALP solution* over 20 iterations (note the log scale). $\binom{n}{2}$ candidate features are evaluated per iteration (between 19900 and 24090 binary feature conjunctions and Bellman marginal evaluations over region where feature is active).

This reflects the intuition that a good policy in this domain should trade off vaccination cost with allowing a pass-through to "cost-free" regions in the graph. The results in Figure 5.14 further indicate that even in this large (binary) problem, an exhaustive singleton basis is sufficient to achieve a solution within the tight bound shown in Table 5.7. This is different from *e.g.* the previous *TaskNetwork* domain where only the introduction of pairwise factors reduced the relative Bellman error to comparable values.

5.4.5 Summary

In this section we have experimentally verified the regularized ALP approach to basis selection and the iterative basis discovery method in general factored multiagent MDPs. Based on the theoretical insight that "sparse agent interaction" enables efficient computations with the Bellman residual, we have shown *exact solution bounds* in domains for which such bounds were previously not available.

We also included comparisons with optimal (where available) and state-of-the-art planning methods, specifically, a sampling-based planner that won the IPPC 2011 and 2014 competitions [48]. For problems with 50 agents that are beyond the scope of these methods, we showed that basis discovery outperforms a domain-specific heuristic: the obtained policy in the *DiseaseProp* domain, for example, achieves comparable results with 15 agents to the heuristic with 50 agents.

Lastly, we noted that the ALP solution (and not basis discovery) becomes a limiting factor in large-scale experiments with graphs of up to 100 nodes and 50 agents.

5.5 Related Work

The specific, novel focus in this chapter has been on basis discovery methods that *retain a (bounded) solution* in large, collaborative FMMDPs. The methods introduced in this chapter may be broadly related to existing *basis discovery* methods with the caveat that these do not explicitly address the scale to large MASs. Multiagent settings have usually pursued a *coordination search and solve* strategy, separating coordination (or, *basis*) discovery from solving the value function. We first introduce general basis discovery methods and then return to the multiagent case below.

Among basis discovery methods, one can distinguish between methods for basis *selection* and *generation*. The regularized ALP (RALP) formulations from Section 5.2 fall into the former category while the Sparse Coordination Discovery (SCD) algorithm is an instance of the latter.

Regularization with the L_1-norm is frequently applied in regression or compressive sensing applications for sparse *feature selection* [25, 102]. Similar basis selection methods have also appeared for approximate dynamic programming (ADP) formulations to planning and reinforcement learning. They are usually based on *sparsity-inducing* regularization of the parameter vector [57, 70]. In the context of reinforcement learning, [75, 81] consider L_1-regularization for linear value function approximation. Our regularized ALP implementation is most closely related to the RALP variant in [81]. Different from their method, we do not rely on constraint sampling [28] but instead implement the constraints exactly with the method detailed in Chapter 2. We also consider a second regularization term on the domain size of the basis function to encourage locally-scoped features in the ALP. Recent work has introduced a non-parametric formulation of the ALP that skips feature selection and instead encodes a smoothness constraint of the value function with the constraints [78]. An evaluation of this method in the context of large multiagent settings is future work.

(Iterative) basis generation has a history in matching pursuit methods that greedily forward select features [77]. Versions of matching pursuit were evaluated in context of RL in [27, 47, 74]. Different from RL methods with basis function selection for *policy evaluation* we address the control problem. As mentioned above, our interest is in maintaining efficient solutions in the multiagent setting. However, existing methods for control disregard the fact that in order for a class of efficient MAS solution algorithms to apply in the first place, basis functions are additionally constrained to have *local scope*. Existing ALP basis generation methods, for example, do not prioritize sparsity in the CFG and do not scale to large MASs. These methods have also only shown *empirical reduction* in Bellman Error [82, 106]. Unlike these methods, we directly optimize for a Bellman Error-based basis selection criterion to avoid overfit to the objective in the ALP (observed, *e.g.*, in [82]).

Recent work in linear value function approximation in reinforcement learning (RL) has considered Bellman Error-based metrics for incremental basis expansion [57, 74, 76]. These methods aim to iteratively improve the bound to the true value function by introducing as next feature the Bellman Error Basis Function (BEBF) or an approximation thereof. The BEBF is derived for $\mathcal{T}$ operator (*i.e.*, policy evaluation or uncontrolled settings) for linear fixed point methods. Our BEBF* is developed for $\mathcal{T}^*$ (*i.e.*, the control problem) in context of the ALP. We also retain an efficient *exact* computation of the BEBF* in settings with sparse interaction but forego stronger guarantees about bound reduction for approximations to the BEBF* (our results only guarantee a monotonic decrease of the bound in the '$\leq$' sense). Our incremental basis expansion is most closely related to the method in [31] although additional constraints (the *sparsity assumption* for

CFGs) are necessary to scale the solution to the large factored action spaces of MASs.

Basis discovery in multiagent settings has usually pursued a *coordination search and solve* strategy, separating coordination (or, *basis*) search from solving the value function. For example, [105] considers a randomized search over structures that generates and evaluates coordination factor graphs online. Other work in RL has considered statistical tests for greedily selecting CFGs [52]. Our focus is on the *joint optimization* of value function and the CFG. Unlike previous methods, coordination discovery is treated as a by-product of achieving a desired bound to the optimal value function.

Planning in large discrete action spaces is challenging and frequently based on problem-specific heuristics (not considered here). Our SCD algorithm extends Guestrin's original work on efficient approximate linear programming (ALP) solutions to cooperative FMMDPs [36]. We extend the method for the multiagent case with sparse interactions to include automated basis discovery and develop *exact Bellman Error-based bounds* for the approximations.

Other general planning methods that are not based on linear value functions include *symbolic dynamic programming (SDP)*. Recent work on SDP with factored actions [84, 85] describes both exact and approximate methods that extend earlier work on SDP (the SPUDD algorithm [46]) to the factored action case.

Sample-based planning algorithms can be applied in principle even if the action space cannot be explored exhaustively [48, 51]. Recent extensions aim to address factored actions in particular but have thus far not demonstrated scale to the many agents considered here [3, 20].

Recent multiagent work phrases decision-theoretic planning as inference in Bayesian Networks [59, 60]. Policies are restricted to finite state controllers (versus no such restrictions on the policy in our approach). We note that in principle this method is promising for scaling to many agents. The same assessment holds for the variational framework in [15] which uses belief propagation (BP) and is exponential in the cluster size of the graph. Results are shown for graphs with up to 20 nodes and a restricted class of chain graphs. In contrast, we provide exact bounds on the solution for multiagent settings with sparse interactions.

5.6 Contributions

In this chapter we translated the theoretical insights about sparse CFGs from Chapter 4 into novel basis *selection* and *generation* methods for the ALP. We developed the Bellman Error Basis Function for $\mathcal{T}^*$ (BEBF*), which extends previous work on BEBFs for policy evaluation to the *control problem*, *i.e.*, the (approximate) solution to the optimal policy.

We then introduced a key algorithmic contribution with the Sparse Coordination Discovery (SCD) algorithm for general collaborative FMMDPs. Underlying SCD is the assumption that there exists some form of sparse interaction between agents that—if found—allows to approximate the global value function well. Based on the BEBF* and the efficient computation of the Bellman Error, SCD is implemented as an iterative algorithm that automates the search for sparse coordination via basis expansion in the ALP. SCD implements a coordination factor graph search from least to most complex by successively expanding more complex basis functions that fulfill a sparsity constraint on the associated CFG.

We further showed that the search maintains *bounded solutions* (with respect to the optimal solution V^*) and that it improves the bound monotonically (in a '$\leq$' sense). By utilizing the efficient constraint generation method for the ALP reviewed in Chapter 2, this *joint optimization* of CFG and value function approximation $\hat{V}$ further scales to large multiagent settings. It also addresses a common criticism of the ALP solution method, namely the requirement of a prespecified (and domain-specific) basis. SCD was validated experimentally across a number of large multiagent planning problems and enabled error bound analysis for the first time in the larger of the evaluated domains (*e.g.*, *SysAdmin* or *DiseasePropagation* with 50 agents). Our results aid the design of multiagent systems by allowing a principled trade-off between strong performance guarantees and associated complexity of coordination between agents.

Chapter 6

The Lifted Approximate Linear Program

The general class of cooperative multiagent systems (MASs) considered in this thesis possess negative complexity results as they scale to larger agent numbers. Previous chapters introduced different means to exploit *locality* in the problem to scale solution methods in both agent number and problem size. As shown in the previous two chapters in particular, many (general) exact and approximate solution algorithms that attempt to exploit structure in the problem are based on *value factorization*. Locally-scoped value functions induce a coordination factor graph (CFG), enabling efficient *representation* and *computation* of the joint policy, even in problems whose exponential state and action spaces would otherwise render a solution prohibitive. Especially multiagent settings, however, are known to suffer from an exponential increase in value component sizes as interactions among agents become denser, meaning that approximation architectures are overly restricted in the problem sizes and types they can handle. In the previous chapter we developed a coordination discovery method that enforces sparsity in agent interaction to maintain compact value function scopes. At the core of the method lies the assumption that variable elimination can continue to be carried out efficiently in the underlying graph, which presents a limit for general factored MDPs.

In this chapter we present an approach to mitigate this limitation for certain types of MASs, exploiting a property that can be thought of as *'anonymous influence'* in general, factored MDPs. In particular, we show how anonymity can lead to representational and computational efficiencies, both for general variable elimination (VE) in a factor graph but also for the approximate linear programming (ALP) solution to factored MDPs. The latter allows to scale linear programming to factored MDPs that were previously infeasible to solve.

This chapter therefore explores yet another interpretation of 'locality', specifically an efficient representation of *local effects* in large graphs. Our contribution has a natural application in a class of problems that can be summarized as the *control of stochastic dynamics* over large

graphs. As introduced in the previous chapter (and Appendix B in detail), one such example is the *disease propagation* setting where multiple agents have to control a disease outbreak through the targeted application of vaccinations. Our evaluation shows the computational benefits of an efficient representation of local effects in this class of problems, enabling the solution of instances with both high node count and dense graph connectivity.

6.1 Introduction

Given the well-known unfavorable complexity results associated with large action and state spaces, many problem representations and their solution methods attempt to exploit structure in the domain for efficiency gains. The factored (equivalently, "graph-based") MDPs (FMDPs) introduced in Chapter 2 offer representational advantages that do not, however, translate immediately into gains for policy computation. The approximate linear program (ALP) is one of the few general solution methods that has no exponential dependencies in $\mathcal{S}$ *and* $\mathcal{A}$ through the efficient computation of the constraints in the linear program based on a variable elimination method [40]. The method retains an exponential dependency on the tree-width (the largest clique formed during variable elimination) meaning that the feasibility of the approximation architecture is based on the connectivity and scale of the underlying graph.

In this chapter we present an approach to mitigate this limitation for certain types of MASs, exploiting a property that can be thought of as "anonymous influence" in the graph. Anonymity refers to the reasoning over *joint effects* rather than identity of the neighbors in the graph. In the disease control example, the joint infection rates of the parent nodes rather than their individual identity can fully define the behavior of the propagation model. Based on this observation we show how variable elimination—and the complete set of constraints in the ALP—can still be computed *exactly* for a larger class of graph-based problems than previously feasible.

The contributions of this chapter are as follows: first, we define "anonymous influence" for representing aggregate effects in a graph. Second, we develop the concept in a general variable elimination setting and show how it supports compact representation of intermediate functions generated during elimination. A key contribution is the insight how a property referred to as "variable consistency" during VE admits particularly compact representations without the need to "shatter" function scopes into disjoint subsets. Third, based on the results for VE, we move to the planning problem and establish how all constraints in the ALP can be represented exactly (albeit more compactly) for factored MDPs that support anonymous influence. Forth, we contrast the efficiency gains from exploiting anonymous influence on a set of random graphs that

can still be solved with the normal VE and ALP methods. We demonstrate speed-ups of the ALP by an order of magnitude to arrive at the identical solution in a sampled set of random graphs with 30 nodes. Last, we address the disease control problem in graph sizes that were previously infeasible to solve with the ALP solution method. We show that the ALP policy outperforms a hand-crafted heuristic by a wide margin in a 50-node graph with dense neighbor connections and 25 controlled agents.

6.2 Anonymous Influence

The FMMDPs described in Appendix B may not, in general, impose strong constraints on the connectivity of the underlying graph. In fact, many interesting disease propagation settings contain nodes with large in- or out-degrees yielding dense connectivity in some regions of the graph, rendering the ALP solution method intractable (see, for example, the graphs in Figure 6.5). In this section we develop a novel approach to deal with larger scope sizes than addressed previously.

At the core lies the assumption that in the graph-based problems above, only the *joint effects* of the parents $\mathrm{Pa}(X_i)$—rather than their identity—may determine the outcome at an individual node X_i. We show how under this assumption variable elimination can be run *exactly* in graphs with higher node and degree counts. The key insight is that the exponential representation of intermediate functions e may be reduced to some *subscope* when only the joint effects, rather than the identity, of some variables in the domain $Dom(e)$ need to be considered. In the following, we first address the representation of "joint effects" before turning to how it can be exploited at a computational level during VE (Section 6.3) and in the ALP (Section 6.4). In our exposition we default to binary variables but the results carry over to the more general, discrete variable setting.

6.2.1 Count Aggregator Functions

We define count aggregator functions to summarize the "anonymous influence" of a set of variables. In the disease propagation scenario for example, the number of active parents uniquely defines the transition model T_i while the identity of the parent nodes is irrelevant (for *representing* T_i).

Definition 16 (Count Aggregator Function)**.** *Let* $\#\{\mathbf{Z}\} : Z_1 \times \ldots \times Z_N \mapsto \mathbb{R}$, $Z_i \in \{0,1\}$, *define a* count aggregator function *(CAF) that takes on* $N+1$ *distinct values, one for each setting of* k

'enabled' factors Z_i (including the case that no factor is 'enabled'). Note that all permutations of k 'enabled' factors map to the same value.

Count aggregator functions can be used to summarize the effects of variable sets compactly whenever their exact identity is not required, as is the case in the following example:

Example 6. *Consider a monotonically increasing CAF, $\#_i$, that summarizes the number of infected parents of a node X_i in a disease propagation graph. Here, the codomain of $\#_i\{\mathbf{Z}\}$ directly corresponds to* $\{0, \ldots, N\}$, i.e. *the* number *of 'enabled' factors in* $\mathbf{z} \in \mathbf{Z}$.

We delay a discussion of conceptual similarities with generalized (or 'lifted') counters in first-order inference to the comparison with related work in Section 6.6.

6.2.2 Mixed-mode Functions

We now make the distinction between 'proper' variables and those variables that appear in a counter scope explicit.

Definition 17 (Mixed-Mode Function). *Let $f(\mathbf{X}, \#\{\mathbf{Z}\})$ denote a* mixed-mode function *over domain $\mathbf{X} \times \mathbf{Z} = X_1 \times \ldots \times X_M \times Z_1 \times \ldots \times Z_N$ if,* for every instantiation $\mathbf{x} \in \mathbf{X}$, *$f(\mathbf{x}, \#\{\mathbf{Z}\})$ is a count aggregator function. We refer to $X_i \in \mathbf{X}$ as* proper variables *and $Z_j \in \mathbf{Z}$ as* count variables *in the scope of f.*

A mixed-mode function can be described with (at most) $K^M(N+1)$ parameters where K is an upper bound on $|Dom(X_i)|$. A CAF is a mixed-mode function where $\mathbf{X} = \varnothing$.

Example 7. *Consider the conditional probability distribution $T_i(X_i \mid \mathrm{Pa}(X_i))$ of a (binary) node X_i and its parents in the disease propagation graph. Let x_i and $\bar{x}_i$ denote the case that node i is infected and not infected, respectively. Then $T_i(X_i \mid \#\{\mathrm{Pa}(X_i)\})$ is a* mixed-mode function *that induces two CAFs, one for x_i and one for $\bar{x}_i$.*

The definition of mixed-mode functions can be extended to allow for multiple count scopes $\#_i$:

Definition 18. *Let $f(\mathbf{X}, \#_i\{\mathbf{Z_i}\}, \ldots, \#_k\{\mathbf{Z_k}\})$ denote a* mixed-mode function *and assume (for now) that $\mathbf{Z}_i \cap \mathbf{Z}_j = \varnothing$ for $i \neq j$. Then, for each of the K^M instantiations $\mathbf{x} \in \mathbf{X}$, there is the induced CAF $f(\mathbf{x}, \#_i\{\mathbf{Z_i}\}, \ldots, \#_k\{\mathbf{Z_k}\})$ which is fully defined by the values assigned to $\#_i\{\mathbf{Z_i}\}, \ldots, \#_k\{\mathbf{Z_k}\}$.*

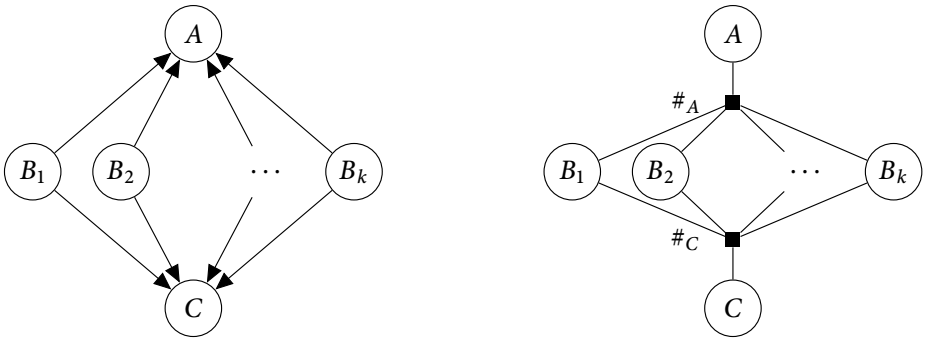

Figure 6.1: An example Bayesian Network (*left*) and factor graph (*right*). The latter defines two factors with *mixed-mode functions* $f(A, \#\{B_1, B_2, \ldots, B_k\})$ and $g(C, \#\{B_1, B_2, \ldots, B_k\})$. Here, variables B_i only occur in counter scopes # (not the general case).

Example 8. *Consider,* e.g., $g(X_1, X_2, \#_1\{\mathrm{Pa}(X_1)\}, \#_2\{\mathrm{Pa}(X_2)\})$ *and let* $|\mathrm{Pa}(X_1)| = |\mathrm{Pa}(X_2)| = 6$ *and* $\mathrm{Pa}(X_1) \cap \mathrm{Pa}(X_2) = \varnothing$. *While the naive representation has* 2^{14} *values, the mixed-mode function g can be represented with* $2^2 \cdot 7 \cdot 7$ *parameters.*

In the next section we show how mixed-mode functions can be exploited during variable elimination and during constraint generation in the ALP.

6.3 Efficient Variable Elimination

Variable elimination (VE) removes variables iteratively from a factor graph given an elimination ordering $\mathcal{O}$. Different *elimination operators* (*e.g. sum* or *max*) implement distinct operations on the graph (such as marginalization of a variable in a probability distribution or determining the maximizing action in a coordination factor graph (*cf*. Chapter 4). As part of this, VE performs maximizations and summations over local terms.

The complexity of variable elimination is exponential in the scope size of the largest factor generated during elimination, which generally depends on the chosen ordering $\mathcal{O}$ [54]. Consider, *e.g.*, the Bayesian Network on the left of Figure 6.1 and assume that variables $A, B_1, \ldots, B_k$ are to be eliminated with some operator Op. If variable A is eliminated first, the initial factor e comprises all terms dependent on A. Therefore, in order to implement Op, $e(A, B_1, \ldots, B_k)$ has to be constructed in full (exponential in k+1). In this section we show how the max operation is implemented efficiently when factors are *mixed-mode functions* (results for summation follow analogously). We begin with the special case that counter scopes in a mixed-mode function are disjoint and then address the general setting.

Consider again maximization over proper variable A under the same ordering $\mathcal{O}$ in the factor graph on the right of Figure 6.1. Denote the result by the intermediate (mixed-mode) function $f'(\#\{B_1, B_2, \ldots, B_k\})$ computed as:

$$f'(v) = \max\Big[f(a, v), f(\bar{a}, v)\Big] \tag{6.1}$$

for all $v \in \{0, \ldots, k\}$ that can be assigned to counter #. This operation is implemented with $k+1$ operations and has no exponential dependency on k like normal VE when computing f'.

Now consider the elimination of count variable B_i from f. The result is again a mixed-mode function $f''(A, \#\{B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_k\})$ where:

$$f''(a, v) = \max\Big[f(a, v), f(a, \mathbf{v} + \mathbf{1})\Big] \tag{6.2}$$

for all $a \in A, v \in \{0, \ldots, \mathbf{k} - \mathbf{1}\}$, since the eliminated count variable *may increase the count by at most 1*. Note that for monotonically increasing (or decreasing) CAFs the max in Equation 6.2 can be avoided.

Completing the coverage of all elementary operations performed during variable elimination, sums of mixed-mode functions can again be written compactly in mixed-mode form. Consider N additional factors $\phi_j(A, \#\{\mathbf{Z}_j\})$ in the factor graph and assume (for now) that all counter scopes are mutually disjoint. Then $l = f + \sum_{j=1}^{N} \phi_j$ can be represented as:

$$l(A, \#\{B_1, \ldots, B_k\}, \#_1\{\mathbf{Z}_1\}, \ldots, \#_N\{\mathbf{Z}_N\}) \tag{6.3}$$

which is computed without exponential expansion of any of the variables appearing in a counter scope as $l(a, v, z_1, \ldots, z_N) = f(a, v) + \sum_{j=1}^{N} \phi_j(a, z_i)$ for all valid assignments a, v, z_i.

Example 9. *We now consider a fully worked-out example comparing the elimination of variable A and C under $Op = \max$ in both graphs of Figure 6.1, for the case that $k = 2$ and that all variables are binary. Both 'regular' and mixed-mode functions are represented in tabular form and A is eliminated first from the graph. Assume the definitions for $f_l(A, B_1, B_2)$ (left graph) and $f_r(A, \#\{B_1, B_2\})$ (right graph) as shown in Table 6.1:*

To maximize over A, consider the last column (respectively, row) of f_l and f_r in Table 6.1. Note that in order to perform the max *operation, f_l requires the construction of exponentially many columns (with respect to its scope size). Further note that the result of* $\max_A f_r$ *is a CAF. We now*

f_l :

A	B_1	B_2	$f_l(A, B_1, B_2)$	$\max_A f_l$
0	0	0	0	0
1	0	0	0	
0	1	0	2	3
1	1	0	3	
0	0	1	2	3
1	0	1	3	
0	1	1	4	6
1	1	1	6	

f_r :

A \ #	0	1	2
0	0	2	4
1	0	3	6
$\max_A f_r$	0	3	6

Table 6.1: The definitions for 'regular' and mixed-mode functions f_l and f_r to go along with Example 9.

turn to elimination of C from the graph. The result—with the analogous operations to the previous case—is shown in Table 6.2.

g_l :

C	B_1	B_2	$g_l(C, B_1, B_2)$	$\max_C g_l$
0	0	0	9	9
1	0	0	2	
0	1	0	3	3
1	1	0	2	
0	0	1	3	3
1	0	1	2	
0	1	1	0	4
1	1	1	4	

g_r :

C \ #	0	1	2
0	9	3	0
1	2	2	4
$\max_C g_r$	9	3	4

Table 6.2: The definitions for 'regular' and mixed-mode functions g_l and g_r to go along with Example 9.

Finally, consider the result of eliminating one of the remaining variables from the graph, i.e. *let* $o_l = \max_{B_1} \left[\max_A f_l + \max_C g_l \right]$ *and* $o_r = \max_{B_1} \left[\max_A f_r + \max_C g_r \right]$. *Note that* B_1 *is a count variable in CAFs* f_r *and* g_r *and that both functions are defined over the same domain. It follows that* o_r *is a CAF with reduced scope,* $o_r(\#\{B_2\})$, *which can be computed efficiently analogously to Equation 6.2 (see Table 6.3 for the result).*

To compute o_l, *on the other hand, the full table for all instantiations of* B_1, B_2 *has to be constructed (see Table 6.4 for the result).*

o_r :

#	
0	$\max[f_r^*(\# = 0) + g_r^*(\# = 0), f_r^*(\# = 1) + g_r^*(\# = 1)] = 9$
1	$\max[f_r^*(\# = 1) + g_r^*(\# = 1), f_r^*(\# = 2) + g_r^*(\# = 2)] = 10$

Table 6.3: The result of maximizing over count variable B_1 after eliminating A and C from the graph. Note that $f_r^* = \max_A f_r$ and $g_r^* = \max_C g_r$ are both CAFs and $o_r = \max_{B_1}[f_r^* + g_r^*]$.

o_l :

B_1	B_2	$f_l^*(B_1, B_2) + g_l^*(B_1, B_2)$	$\max_{B_1}$
0	0	0 + 9 = 9	9
1	0	3 + 3 = 6	
0	1	3 + 3 = 6	10
1	1	6 + 4 = 10	

Table 6.4: The result of maximizing over variable B_1 after eliminating A and C from the graph. Note that $f_l^* = \max_A f_l$ and $g_l^* = \max_C g_l$ and $o_l = \max_{B_1}[f_l^* + g_l^*]$.

6.3.1 The General Case

Both proper and count variables are, in general, not uniquely associated with a single factor during VE. For example, in a disease propagation graph, variables may appear as *both proper and count* variables in different factors. We can distinguish two cases:

Shared proper and count variables

Consider factor $e(A, \#\{A, B_1, \ldots, B_k\})$ where A appears as both proper and count variable. Elimination of A requires full instantiation (*i.e.*, it can be considered proper) and it is removed from the counter scope: $e'(a, v) = e(a, \mathbf{a} + \mathbf{v})\ \forall a \in A, v \in \{0, \ldots, k\}$ in a variant of Equation 6.2 that *enforces consistency* with the choice of (binary) proper variable A, thereby avoiding the max operation in the same Equation. The resulting e' has a representation that is strictly smaller than that of e.

Example 10. *Consider the case of a mixed-mode function $f(A, \#\{A, B\})$ with a (binary) variable A appearing as both 'proper' and 'count' type. Removal of A from # is done as before in Equation 6.2, additionally ensuring that the setting of A in proper and counter scopes are consistent during elimination. Figure 6.2 shows the implementation for eliminating A from the counter scope of f.*

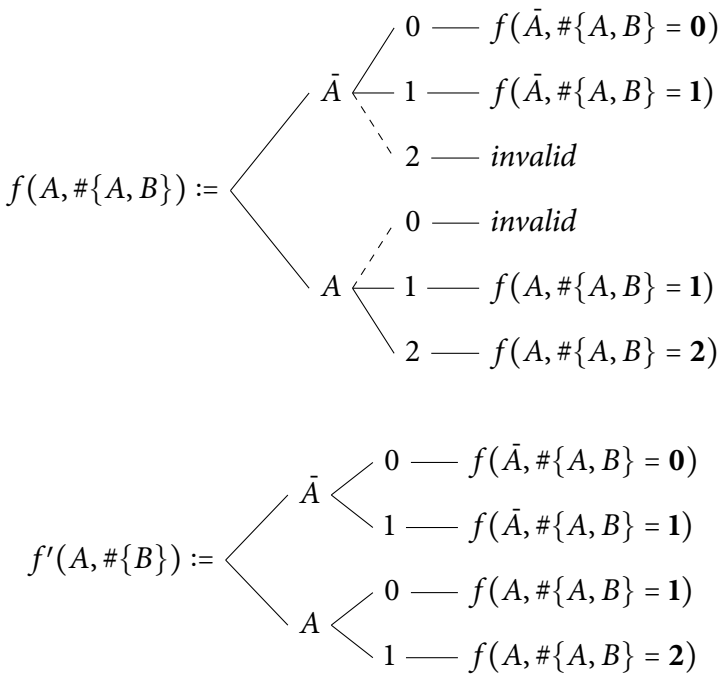

Figure 6.2: Tree representation of function $f'(A, \#\{B\})$ after eliminating A from the counter scope in $f(A, \#\{A,B\})$. Note how the value of (proper) variable A is consistent with that in each counter scope during elimination, removing the max operation in Equation 6.2. Further note that no *invalid* entries from $f(A, \#\{A,B\})$ appear in f' after variable elimination.

Non-disjoint counter scopes

Consider two non-disjoint count variable sets $\#_i, \#_j$ in a function f, *i.e.* $\mathbf{Z}_i \cap \mathbf{Z}_j \neq \varnothing$. Trivially, there always exists a partition of $\mathbf{Z}_i \cup \mathbf{Z}_j$ of mutually disjoint sets $\{\mathbf{Y}\}, \{\mathbf{W}_i\}, \{\mathbf{W}_j\}$ where $\mathbf{W}_i, \mathbf{W}_j$ denote the variables unique to $\#_i$ and $\#_j$, respectively, and $\mathbf{Y}$ are shared. Associate the counts $\#\{\mathbf{Y}\}$, $\#\{\mathbf{W}_i\}$, and $\#\{\mathbf{W}_j\}$. Then the mixed-mode function f can be written as $f(\mathbf{X}, \#\{\mathbf{Y}\}, \#\{\mathbf{W}_i\}, \#\{\mathbf{W}_j\})$. This observation extends to more than two non-disjoint count variable sets.

In the worst case, a partition of $\bigcup_{i=1}^{k} \mathbf{Z}_i$, $k \geq 2$ requires $p = 3 \cdot 2^{k-2}$ splits into mutually disjoint sets and the resulting representation of f is exponential in p. The next section shows that this 'shattering' into mutually disjoint sets can be avoided and representations be kept compact if *variable consistency is enforced during VE.*

Example 11. *Consider* $f(\#\{A,B,C,D,E\}, \#\{A,B,X,Y,Z\}, \#\{A,C,W,X\})$ *with non-disjoint counter scopes. The direct tabular encoding of* f *requires* $6 \cdot 6 \cdot 5 = 180$ *parameters but contains* invalid entries *due to overlapping counter scopes. The representation of the identical function,*

$f'(\#\{A\}, \#\{B\}, \#\{C\}, \#\{D, E\}, \#\{X\}, \#\{W\}, \#\{Y, Z\})$ *with no invalid entries requires* 288 *parameters.*

Note that, in general, the difference in size between shattered and un-shattered representations can be made arbitrarily large.

6.3.2 Compact Representation

As shown above, a representation that avoids invalid entries due to overlapping variable scopes is, in general, less compact than one that does not. We now state a key result that functions do not need to avoid overlapping counter scopes to guarantee valid solutions during variable elimination. This establishes that intermediate functions generated during elimination *do not suffer from an exponential increase in representational size* due to enforcing (disjoint) counter scopes $\mathbf{Y}, \mathbf{W}_i, \mathbf{W}_j$. For simplicity of proofs, we generally assume binary (proper) variables but all results extend equally to the more general, multi-valued case.

Theorem 1. *Given a mixed-mode function f with overlapping counter scopes, variable elimination will never include an* invalid value *in any of its operations Op (*e.g.*,* max*) on f. In particular, denote by f' the result of eliminating one variable from f and consider a valid entry v in f'. Then, the operation Op used to compute v only involves feasible (*i.e.*, consistent) values from f. This holds for any elimination ordering $\mathcal{O}$.*

Proof. The result is immediately clear for the case of eliminating proper and *non-shared* count variables. Let f' be the result of such elimination. In case of a proper variable, it follows from Equation 6.1 that any valid assignment to the variables in f' yields a max operation over valid entries in f. For non-shared count variables, the corresponding counter in f' is reduced by 1 and the max is over assignments $v + 0, v + 1$ to the *reduced counter scope* (Equation 6.2). Both count assignments are therefore valid in the *extended counter scope* of f. The full proof (in particular the case for *shared count variables* is given in Appendix A.

Illustration. Consider the elimination of the shared count variable B_1 from f in Figure 6.3 under $Op = \max$. Denote by $f'(\#_{g-\{B_1\}}, \#_{h-\{B_1\}})$ the result of variable elimination. Then, for all valid assignments a, b to its count functions we have that:

$$f'(\#_{g-\{B_1\}} = a, \#_{h-\{B_1\}} = b) = \max\left[f(\#_g = a, \#_h = b), f(\#_g = a + 1, \#_h = b + 1)\right]$$

with validity of the respective values in f established by Theorem 1. Another way to see this

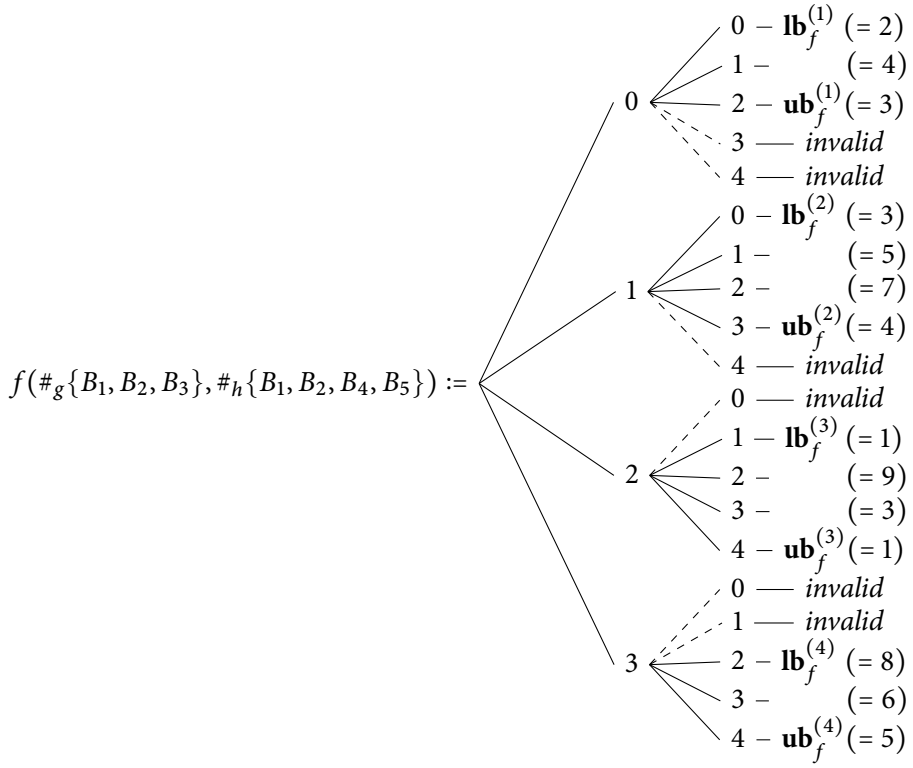

Figure 6.3: A valid tree representation for a mixed-mode function f with overlapping count variables B_1, B_2. The tree is expanded along $\#_g$ (level 1) and $\#_h$ (level 2) and invalid combinations of values are marked as such. $ub_f^{(k)}$ and $lb_f^{(k)}$ refer to upper and lower bounds, respectively, for all 4 subtrees corresponding to $\#_h$ in the tree (not shown for level 1 where all choices $0, \ldots, 3$ are valid).

is to consider the equivalent representation of f with disjoint (*i.e.*, "shattered") count variable sets, $l(\#\{B1, B2\}, \#_{g-\{B_1,B_2\}}\{B_3\}, \#_{h-\{B_1,B_2\}}\{B_4, B_5\})$, with no invalid values. Note that $l(\# = \mathbf{v}, \#_{g-\{B_1,B_2\}} = \mathbf{w}, \#_{h-\{B_1,B_2\}} = \mathbf{z})$ is just $f(\#_g = (\mathbf{w} + \mathbf{v}), \#_h = (\mathbf{z} + \mathbf{v}))$ which enforces the selection of valid values from f. Eliminating B_1 from l then corresponds to a removal of a *non-shared* count variable via $\max_{\mathbf{v},\mathbf{v}+\mathbf{1}} l \ \forall w, z$ (as previously in Equation 6.2). Note that this is equivalent to treating $\#_g$ and $\#_h$ *jointly* in the original function f, *i.e.* incrementing both count functions together.

Example 12. *Compute* $\max_{Dom(f)} f$ *for the function* f *defined in Figure 6.3 via variable elimination given an elimination ordering* $\mathcal{O} = B_1, B_3, B_2, B_4, B_5$. *Results are shown in Figure 6.4 below.*

Corollary 6. *The representation of mixed-mode function* $f(\mathbf{X}, \#_1\{\mathbf{Z}_1\}, \ldots, \#_N\{\mathbf{Z}_N\})$ *for proper variable set* $\mathbf{X}$ *and* N *counters is of* $O(K^{|X|} \cdot L^N)$ *where* K *is an upper bound on* $|Dom(X_i)|$ *and* L *the largest counter value.*

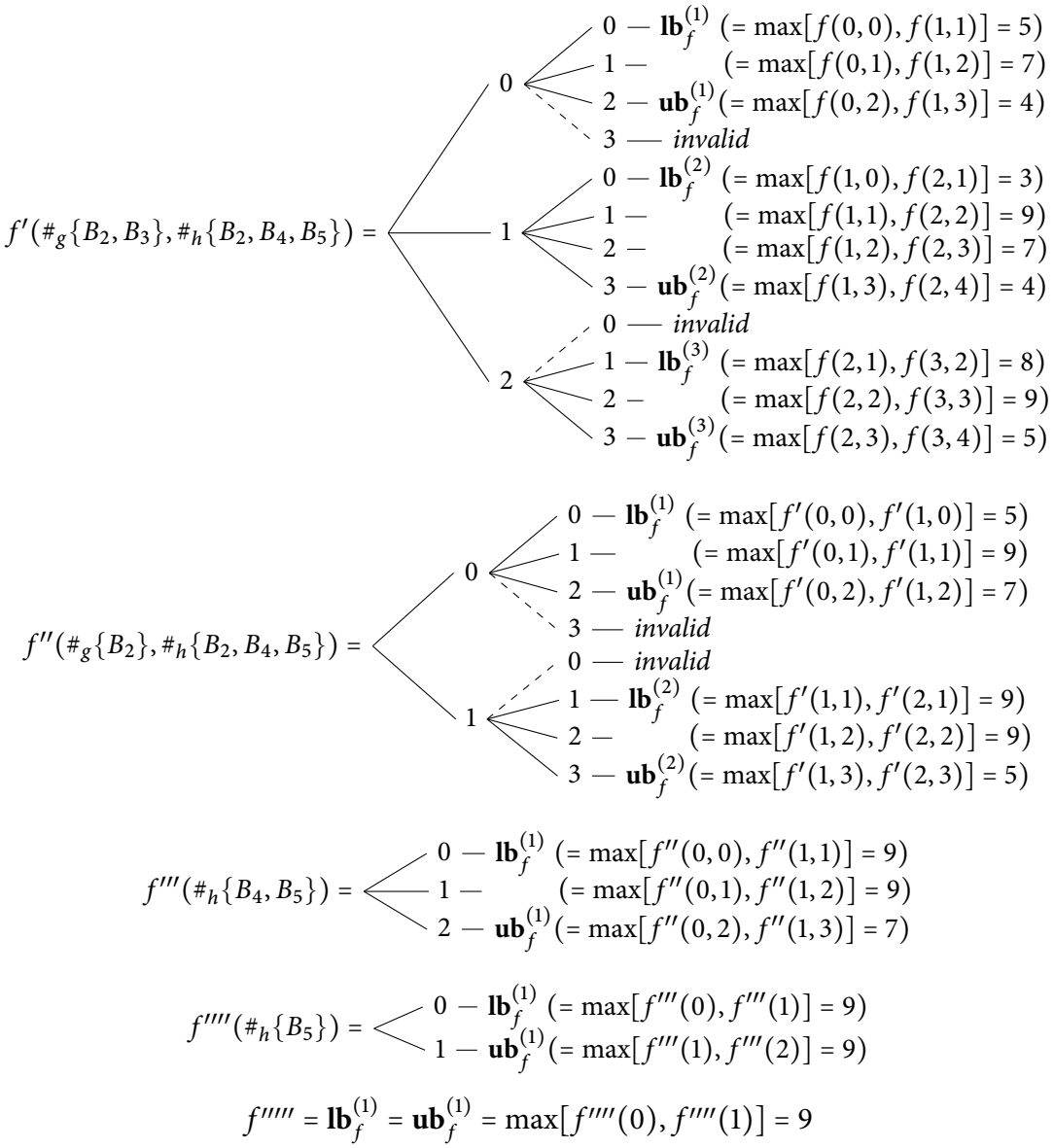

Figure 6.4: Results of variable elimination for function f from Figure 6.3 under Op = max and ordering $\mathcal{O} = B_1, B_3, B_2, B_4, B_5$ (see Example 12). $ub_f^{(k)}$ and $lb_f^{(k)}$ refer to upper and lower bounds, respectively (not shown for level 1 where all choices are valid). The max is computed with 22 rather than 2^5 operations.

The results above are for general variable elimination but can similarly be exploited in the ALP for efficient constraint generation in both single- and multiagent FMDPs.

6.4 Exploiting Anonymity in the ALP

In the previous section we showed how mixed-mode functions can be represented efficiently during variable elimination via *compact representations* that do not need to enforce mutually disjoint counter scopes. In this section we apply this intuition to the approximate linear program (ALP). For the ALP, the functions $c_i = \gamma g_i - h_i \ \forall h_i \in H$, along with reward functions R_j, $j = 1, \ldots, r$ define a factor graph corresponding to the max constraint in Equation 2.21 (*cf*. details for the efficient ALP solution method in Chapter 2). Note that the scopes of c_i are generally larger than those of h_i since parent variables in the 2TBN are added during backprojection. Factors are over state and action variables in the multiagent case.

A key insight is that for a class of factored (single- or multiagent) MDPs defined with count aggregator functions in the 2TBN, the same intuition as in previous section applies to implement the non-linear max constraint in the ALP exactly.

We first establish that basis functions $h_i \in H$, when backprojected through the 2TBN (which now includes mixed-mode functions), retain the counters in the resulting backprojections g_i. The backprojection operator is the expectation of basis function h_i defined as $g_i(\mathbf{x}, \mathbf{a}) = \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) h_i(\mathbf{x})$ and involves summation and product operations only (*cf*. Chapter 2). We have established previously that summation of mixed-mode functions preserves counters (Equation 6.3). The same result holds for multiplication when replacing the sum operation in Equation 6.3 with a multiplication. It follows that g_i (and c_i) preserve counters present in the 2TBN and share the results for compact representations derived in the previous section.

6.4.1 ALP Constraint Generation

The exact implementation of the max constraint via VE in Equation 2.21 proceeds as before with compact representations. Note that the domain $Dom(e)$ of an intermediate (mixed-mode) term $e(\mathbf{X}, \mathbf{Z})$ with proper and count variable sets $\mathbf{X}$, $\mathbf{Z}$, is reduced as established for general variable elimination in Corollary 6. In particular, the number of variables and constraints in the ALP is exponential only the *size of the representation* of the largest mixed-mode function formed during VE. Further, the reduction is exact and the ALP computes the identical value function approximation $\hat{V} = \mathbf{H}\hat{\mathbf{w}}$.

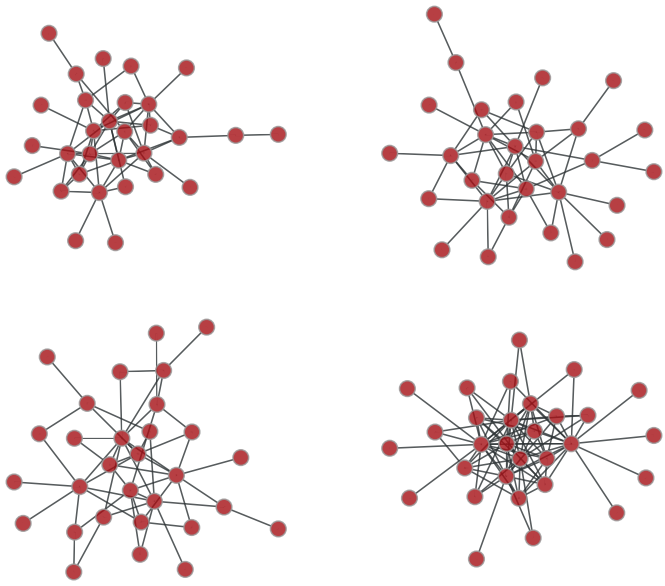

Figure 6.5: *From top left to bottom right:* sample of three random graphs in the test set with 30 nodes and a maximum out-degree of 10 (first three). *Bottom right:* Test graph with an increased out-degree sampled from $[1, 20]$.

6.5 Experimental Evaluation

We evaluate the method on random disease propagation graphs with 30 and 50 nodes. For the first round of random graph experiments, we obtain the value function for an uncontrolled disease propagation process and contrast runtimes of the normal VE/ALP method (where possible) with those that exploit "anonymous influence" in the graph. We then move to a controlled disease propagation process with 25 agents in a 50-node graph and compare the results of the obtained policy to two heuristics.

All examples implement the disease control domain from Section B.3. For the regular case (referred to as VE_1/ALP_1), the parent scope in T_i includes only 'proper' variables as usual. The alternative implementation (VE_2/ALP_2) utilizes count aggregator functions $\#\{\text{Pa}(X_i')\}$ in every T_i. We use identical transmission and node recovery rates throughout the graph, $\beta = 0.6$, $\delta = 0.3$. Action costs are set to $\lambda_1 = 1$ and infection costs to $\lambda_2 = 50$. All experiments implement the same greedy elimination heuristic for VE that minimizes the scope size at the next iteration.

6.5.1 Random Graphs

We use graph-tool [79] to generate 10 random graphs with an out-degree k sampled from $P(k) \propto 1/k$, $k \in [1, 10]$. Table 6.5 summarizes the graphs and Figure 6.5 illustrates a subset. 60 indica-

Mean/min/max degree:				
4.2/1/10	3.4/1/10	3.7/1/10	3.7/1/10	3.7/1/10
2.8/1/10	3.5/1/10	3.1/1/9	3.2/1/8	3.9/1/9

Table 6.5: Properties of the 10 random 30-node graphs.

$\lvert\mathbf{C}_1\rvert$	VE_1	ALP_1	$\lvert\mathbf{C}_2\rvert$	VE_2	ALP_2	$\lvert\mathbf{C}_2\rvert/\lvert\mathbf{C}_1\rvert$	VE_2/VE_1	ALP_2/ALP_1
131475	6.2s	**1085.8s**	94023	1.5s	**25.37s**	0.72	0.24	**0.02**
24595	1.1s	3.59s	12515	0.17s	1.2s	0.51	0.15	0.33
55145	3.5s	30.43s	27309	0.4s	8.63s	0.5	0.11	0.28
74735	3.0s	115.83s	41711	0.69s	12.49s	0.56	0.23	0.11
71067	4.16s	57.1s	23619	0.36s	8.86s	0.33	0.08	0.16
24615	**1.6s**	1.15s	**4539**	**0.07s**	0.35s	**0.18**	**0.04**	0.30
63307	2.2s	141.44s	34523	0.39s	4.03s	0.55	0.18	0.03
57113	0.91s	**123.16s**	40497	0.49s	**2.68s**	0.71	0.54	**0.02**
28755	0.54s	17.16	24819	0.36s	3.86s	0.86	0.67	0.22
100465	2.47s	284.75s	38229	0.62s	36.76s	0.38	0.25	0.13
					Average reduction:	0.53	0.25	0.16

Table 6.6: Results of random graph experiment. Shown are constraint set sizes, VE and ALP solution times for both normal implementation (columns 1-3) and the one exploiting anonymous influence (columns 4-6). Highlighted in bold are the maximal reductions for each of the three criteria.

tor basis functions I_{X_i}, $I_{\bar{X}_i}$ (covering instantiations x_i, $\bar{x}_i$ for all 30 variables X_i), along with 30 reward functions R_i, are utilized in the ALP. The *factor graph* consists of functions c_i that additionally span the parent scope of h_i. Table 6.5 shows minimum and maximum node degrees, which correspond to lower bounds on parent scope sizes since action and state factors from the previous time step are added in Equation B.2.

The results are summarized in Table 6.6. Recorded are the number of resulting constraints, the wall-clock times for variable elimination to generate the constraints, and the ALP runtime to solve the value function after constraint generation on the identical machine. The last three columns record the gains in efficiency per graph.

Lastly, we test with a graph with a larger out-degree (k sampled from the interval $[1, 20]$, shown at the bottom right of Figure 6.5). The disease propagation problem over this graph cannot be solved with the normal VE_1/ALP_1 because of exponential blow-up of intermediate

terms. The version exploiting anonymous influence can perform constraint generation using VE in 124.7s generating 5,816,731 constraints.

6.5.2 Disease Control

In this section we show results of policy simulation for three distinct policies in the disease control task over two random graphs (30 nodes with 15 agents, and 50 nodes with 25 agents with a maximum out-degree per node of 15 neighbors: $|\mathcal{S}| = 2^{50}$, $|\mathcal{A}| = 2^{25}$). Besides a random policy, we consider a heuristic (referred to as "copystate" policy) that applies a vaccination action at X_i if X_i is infected in the current state. It is reactive and does not provide anticipatory vaccinations if some of its parent nodes are infected. The "copystate" heuristic serves as our main comparison metric for these large scale graphs where optimal solutions are not available.

Modifications to the Domain

The modified disease control domain used for the experiments utilizes the efficient encoding of stochastic dynamics that exploits anonymity. Specifically, the transition model introduced in Equation B.3 is encoded with count aggregator functions $\#\{\mathrm{Pa}(X_i')\}$ instead of the exhaustive enumeration over distinct parent set instantiations. This reduction in representational size is lossless, *i.e.* both VE_2 and ALP_2 compute identical results as their regular counterparts.

Results

The ALP is solved with the exact max constraint by exploiting anonymous influence in the graph. It is not possible to solve this problem with the normal ALP_1 due to infeasibly large intermediate terms being formed during variable elimination. All nodes are covered with two indicator basis functions I_{X_i} and $I_{\bar{X}_i}$ as in the previous experiment. Results for the 30 and 50-node control tasks are shown in Figure 6.6 with 95% confidence intervals. The "copystate" heuristic appears to work reasonably well in the first problem domain but is consistently outperformed by the ALP solution which can administer anticipatory vaccinations. This effect actually becomes more pronounced with *fewer* agents: we experimented with 6 agents in the identical graph and the results (not shown) indicate that the "copystate" heuristic performs significantly worse than the random policy with returns averaging up to -20000 in a subset of the trials. This is presumably because blocking out disease paths early becomes more important with fewer agents since the lack of agents in other regions of the graph cannot make up for omissions later. Similarly, in the 50-node scenario the reactive "copystate" policy does not provide a statistically significant

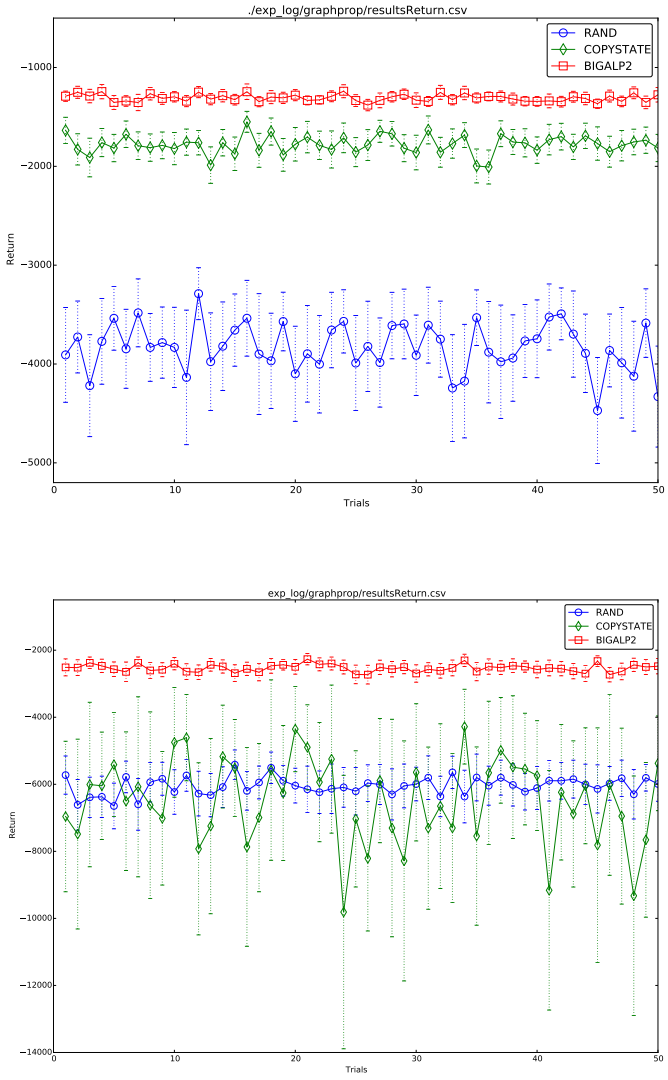

Figure 6.6: Mean return for 50 trials of 200 steps each in the 30-node disease control domain with 15 agents (top) and 50-nodes with 25 agents (bottom). All results are averaged over 50 runs and shown with 95% confidence intervals for each of *random*, *"copystate" heuristic*, and *ALP policy* (see text).

improvement over a random policy (Figure 6.6).

6.6 Related Work

Many recent algorithms tackle domains with large (structured) state spaces. For exact planning in factored domains, SPUDD exploits an efficient, decision diagram-based representation [46]. Monte Carlo tree search (MCTS) has been a popular online approximate planning method to scale to large (not necessarily factored) domains [48, 51, 93]. These methods do not apply to

exponential action spaces without further approximations. Ho *et al.*, for example, evaluated MCTS with 3 agents for a targeted version of the graph control problem [45]. Recent variants that exploit factorization may be applicable [3, 20].

Our work is based on Guestrin *et al.*'s earlier contributions on exploiting factored value functions to scale to large factored action spaces [36, 39]. Similar assumptions can be exploited by inference-based approaches to planning which have been introduced for MASs where policies are represented as finite state controllers [59, 60]. There are no assumptions on the policy in our approach. The variational framework in [15] uses belief propagation (BP) and is exponential in the cluster size of the graph. Results are shown for 20-node graphs with out-degree 3 and a restricted class of chain graphs. The results here remain exponential in tree-width but exploit anonymous influence in the graph to scale to random graphs with denser connectivity. A more detailed comparison with (approximate) loopy BP is future work.

First-order (FO) methods [89, 90, 100, 107] solve planning problems in lifted domains without resorting to grounded representations. Our ideas share a similarity with "generalized counts" [16, 49, 68, 99] in FO models that can eliminate indistiguishable variables in the same predicate in a single operation. Our contributions are distinct from FO methods. Anonymous influence applies in propositional models and to node sets that are not necessarily indistiguishable in the problem: even if nodes appear in a count aggregator scope of some X_i in the network, they are further *uniquely connected in the graph* and are unique instances. We also show that shattering into disjoint counter scopes is not required during VE and how this results in efficiency gains during VE.

Lastly, decentralized and partially-observable frameworks exist to model a larger class of MASs [33, 72, 73]. The issue of scalability in these models due to negative complexity results is an active field of research (see [3, 23] and references therein).

6.7 Contributions

The work presented in this chapter is an extended version of [87]. We introduce the concept of "anonymous influence" in large factored multiagent MDPs and show how it can be exploited to scale variable elimination and approximate linear programming beyond what has been previously solvable. The key idea is that both representational and computational benefits follow from reasoning about influence of variable sets rather than variable identity in the factor graph. These results hold for both single and multiagent factored MDPs and are exact reductions, yielding the identical result to the normal VE/ALP, while greatly extending the class of graphs that can be

solved. Potential future directions include approximate methods (such as loopy BP) in the factor graph to scale the ALP to even larger problems and to support increased basis function coverage in more complex graphs.

Chapter 7

Conclusions and Future Work

This chapter provides a high-level summary of our thesis results, reviewing and putting into context the individual contributions from parts of the thesis. We conclude the chapter with ideas for future work that are motivated by our results.

7.1 Conclusions

The main focus of this thesis has been the exploration of *locality*—in its various forms—for the efficient solution of large, cooperative multiagent planning problems. Characteristic for problems in this important field is that they possess large discrete state and action spaces, generally driven by an exponential dependency on the number of agents. This fundamental complexity makes existing exact solution methods intractable and illustrates the importance of developing principled approximate solutions that scale to realistic settings.

Approximate solutions that rely on the pre-specification of agent interaction require domain expertise, are problem-specific, and commonly heuristic in nature without strong performance guarantees. A reocurring theme in this thesis has therefore been the development of *bounded approximate solutions* that, in addition to empirical evaluation, permit an error bound analysis with respect to the optimal solution. We believe that future deployment scenarios for multiagent systems (MASs) will benefit from the availability of such performance guarantees, especially as other constraints, such as limits on the bandwidth of the communication channel between agents, need to be weighed in a principled fashion against expected performance. Bounded approximations could further widen the deployment of MASs to settings that require strict performance guarantees.

7.1.1 Summary

This thesis can be divided into three lines of work with their individual contributions under the *"local multiagent control"* umbrella. We began in Chapter 3 with the exploration of *spatial locality* in stochastic task assignment problems. Similarly to a human expert who may divide the allocation of tasks according to spatial proximity of agents to tasks, we introduced a novel problem class that formalizes this intuition in the context of FMMDPs. These spatial task allocation problems (SPATAPs) offer a tractable representation for many agents and tasks, and give rise to approximate solutions that exploit locality. We explored two classes of solution algorithms that compute exact solutions to *model approximations* of the SPATAP. First were methods that exploit the temporal characteristics of tasks by restricting attention to the currently active set of tasks (*phase approximations*). Second were fully distributed planning methods that restrict planning at every agent to its individual action space (*subjective approximations*). These latter models *predict the task attendance* of other agents and plan the approximate best response. Combining both approximations yields a model without exponential dependencies in either task or agent number (the k-SP-MDP) whose solution outperforms recent partitioning algorithms in the empirical analysis in [17, 18].

The second major line of work in this thesis generalizes the concept of locality beyond the spatial interpretation to *general collaborative FMMDPs* that do not necessarily possess a spatial component. Chapter 4 formalized the idea with the key contribution of *sparse coordination factor graphs (sparse CFGs)* that encode a "sparsity of interaction" between agents, agnostic to its concrete, domain-specific source. The same chapter also developed novel theoretical insights about the computational benefits of factored value functions that span a sparse CFG. We showed that the *Bellman residual* remains factored in these settings, allowing the efficient computation of the Bellman Error and other functions of the Bellman residual. These insights consequently led to the development of the Bellman Error Basis Function for $\mathcal{T}^*$ (BEBF*) in Chapter 5, which extends previous work on BEBFs for policy evaluation to the *control problem*, *i.e.*, (approximately) solving for the optimal policy.

Chapter 5 then introduced a key algorithmic contribution with the "Sparse Coordination Discovery" (SCD) algorithm for general collaborative FMMDPs. Underlying SCD is the assumption that there exists some form of sparse interaction between agents that—if found—allows to approximate the global value function well. Based on the BEBF* and the efficient computation of the Bellman Error, SCD is implemented as an iterative algorithm that automates the search for sparse coordination via basis expansion in the approximate linear program (ALP).

We showed that the search maintains *bounded solutions* (with respect to the optimal solution V^*) and that it improves on the bound monotonically (in a '$\leq$' sense). By utilizing the efficient constraint generation method for the ALP reviewed in Chapter 2, this *joint optimization* of CFG and value function approximation $\hat{V}$ further scales to large multiagent settings. It also removes the need for a domain expert to specify a basis function (or, *feature*) set. SCD was validated experimentally across a number of large multiagent planning problems and enabled error bound analysis for the first time in the larger of the evaluated domains (*e.g.*, disease control with 50 agents in Chapter 5).

In the third and final line of work in this thesis, we focused on the disease control example and—more broadly—problems in the general class of *controlling stochastic processes over graphs*. This problem class has broad application including in the management of electric power grids, battling forest fires, targeted drug delivery in biological networks, or detecting network intrusion. Chapter 6 introduced novel local summary statistics for controlled graphs that describe the *joint effects* of a set of variables compactly whenever variable identity does not have to be represented explicitly. We proved a key theoretical result that shows how variable elimination (VE) on compact representations can compute the identical result to general VE by enforcing a *variable consistency property* during elimination. These novel compact representations are (in the limit) exponentially smaller than existing flat or "shattered" representations, scaling VE to graphs that were previously computationally infeasible.

The Sparse Coordination Discovery (SCD) algorithm from Chapter 5 uses VE for efficient constraint generation in the ALP. Realizing this, Chapter 6 introduced a novel algorithmic contribution (the *"lifted ALP"*) that transfers the results for VE on compact representations to constraint generation in the ALP. The lifted ALP computes the *identical solution* but exploits variable anonymity for computational gain and scale (*i.e.*, the same optimization problem is implemented with fewer variables and constraints). In particular, we showed for the disease control setting in densely connected graphs how the lifted ALP solution scales to FMMDPs that were previously unsolvable due to intermediate terms in general VE exceededing computational limits.

7.2 Future Work

We now outline possible extensions of the thesis work. We believe that each of these could be a rewarding venue for follow-up research, further addressing scale and applicability of the models and algorithms proposed in this thesis.

7.2.1 Extending Sparse Coordination Discovery (SCD)

Context-dependent Coordination Discovery

The SCD algorithm determines coordination structure between agents that remains fixed throughout the problem (although an online formulation applies in principle). As illustrated for the spatial task allocation problems in Chapter 3, however, this may be a limiting assumption in cases where agents move and dynamically associate with "close" tasks or agents. A promising venue for future work is therefore the extension of the method to *context-specific* coordination. Existing methods rely on the pre-specification of contexts via rule-based formulations (*e.g.*, [40, 53]) although learning-based approaches based on statistical tests have been demonstrated in comparably small domains [52]. The SCD algorithm could in principle be extended to generate such context-specific rules for novel definitions of the basis generation operator (`pair` in Chapter 5). This would further enable interesting future applications, *e.g.* in vehicle collision avoidance.

Non-binary Basis Functions

Considered in this thesis were binary conjunctive features due to their efficient manipulation (*e.g.*, for *feature coverage* computations) and the associated ease in determining a linearly independent basis for the ALP. Along with other planning domains, extensions of the *disease control* problem which adds more fine-grained control of the flow through the network may benefit from non-binary feature functions. Many choices of basis functions apply in principle (see, *e.g.*, [107] for an overview). An extension of the SCD algorithm would require the definition of *basis conjunctions* for the non-binary case. Similarly, *locally-scoped* features must retain a well-defined meaning for other basis function choices.

7.2.2 Addressing the Scale of Solution Methods

Approximate Inference

The efficient ALP-based solution method from Section 2.3 relies on a variable elimination method to compute the (otherwise exponentially large) constraint set efficiently. However, VE possess theoretical limits that prohibit scale in densely connected graphs. While the lifted ALP of Chapter 6 has addressed some of these concerns for the control of stochastic processes on graphs, *models with large induced width* remain computationally challenging.

Future interesting work is in solving the ALP approximately, for example with an approximate constraint generation scheme based on loopy belief propagation (*e.g.*, *"Max-Plus"*, [29, 53, 54]), to allow further scaling of the solution method. An approximate, loopy BP-based step

could be introduced in the ALP for approximately representing the max constraint in Equation 2.21. This is related to existing constraint-sampling methods in the LP [28]. Approximate constraint generation could also permit the insertion of more basis functions into the ALP before computational limits are reached, thereby (in principle) enabling better approximations to the optimal solution.

Approximate methods could also enter the picture for estimating the Bellman Error if coordination graphs that violate the sparsity bound should be considered.

7.2.3 Planning with Structural Uncertainty

Changing Domains

An interesting class of planning and learning problems consists of those that permit variable 2TBN structure. In *generalized domains* (or, equivalently, *generalized environments*) the agent may face a distribution over environments [110, 111]. Similarly, planning with relational templates aims to compute value functions with *generalizable behavior* over a set of possible environments [38, 62]. Existing work optimizes performance in expectation given a known distribution over environments. Interesting follow-up work would allow the transition *between environments* in order to plan *e.g.* for failing agents.

The idea of "structural uncertainty" during planning is also highly applicable for the disease control setting as connectivity of the underlying graph changes. One idea to tackle these problems is to generate new plans from factored value function components, akin to transfer planning [38, 101].

Planning Agent Insertion

While the focus of generalized domains has historically not been in the context of multiagent settings (a notable exception being [38]), their principal relatedness has been suggested before (*e.g.*, in [101]). A particularly interesting venue for future work is the optimization of agent placement, *e.g.* in a disease control setting.

Heuristic solutions could evaluate which (local) value function component may benefit the most from having another agent available and perform greedy insertion. Given the factorization of the value function, it may be possible to only recompute a subset of the value function components rather than optimize the complete basis $\mathbf{H}$ again.

Appendix A

Proofs

A.1 Proofs for Results in Chapter 6

A.1.1 Preliminaries

For simplicity of proofs, we generally assume binary (proper) variables but all results extend equally to the more general, multi-valued case. Similarly to Figure 6.2, we choose to represent functions in tree-form for illustration. Note that this representation is not unique since variables may be expanded in arbitrary order along the tree (commutative property). It follows that showing the results for a single tree expansion suffices to establish the result.

Observation 1. *In a mixed-mode function f with overlapping counter scopes every counter setting $\#\{\mathbf{Z}_i\} = v_i$ establishes an* upper *and* lower bound *on the permissive values of the following counters $\#\{\mathbf{Z}_j\}$, $j > i$, in any tree expansion of f.*

In particular, no function f contains invalid values *inside* the upper and lower bounds established (at each level of the tree) by a complete assignment to all $\#\{\mathbf{Z}_i\} = v_i$ (see Figure 6.3 for an illustration).

Example 13. *Consider function f at the top of Figure 6.2. Note that the lower (lb) and upper bound (ub) established for $\#\{A, B\}$ given the choice of $A = 1$ are* 1 *and* 2*, respectively. In general, the lower and upper bounds for a counter $\# = 0, \ldots, k$ along a specific expansion (*i.e. *branch in the tree) are such that $lb \geq 0, ub \leq k, ub \geq lb$.*

A.1.2 Proof of Theorem 1

Let mixed-mode function f denote a canonical intermediate term generated during variable elimination which includes shared count variables:

$$f(X, Y, \#_1\{A, B\}, \#_2\{B, C\}) = g(X, \#_1\{A, B\}) + h(Y, \#_2\{B, C\}) \tag{A.1}$$

with $f(x, y, \#_1 = v, \#_2 = w) \triangleq g(x, \#_1 = v) + h(y, \#_2 = w)\ \forall x \in X, y \in Y, v \in \#_1, w \in \#_2$. As established before, f contains *invalid values* due to non-disjoint counter scopes, $\#_1 \cap \#_2 = \{B\}$.

At any iteration, variable elimination may eliminate a *proper*, *non-shared*, or *shared count* variable. We establish the result for all three cases. The case for overlap between proper and count variables was addressed in Section 6.3.1, resulting in the removal of the variable from the counter scope.

The result is immediately clear for the case of eliminating *proper* and *non-shared count variables*. Let f' be the result of such elimination. In case of a proper variable, it is easy to see from Equation 6.1 that any valid assignment to the variables in f' yields a max operation over valid entries in f. For non-shared count variables, the corresponding counter in f' is reduced by 1 and the max is over assignments $v + 0, v + 1$ to the *reduced counter scope* (Equation 6.2). Both count assignments are therefore valid in the *extended counter scope* of f.

For *shared count variables* recall that a function tree expansion of f corresponds to a particular expansion ordering of variables and counters of f. In such an expansion, let $tr(k)^f_{1:L} = \langle tr(k)^f_1, \ldots, tr(k)^f_L \rangle$ denote a particular path from root to a leaf, involving a choice of subtree (*i.e.*, variable assignment) at every level $1 - L$ in the tree. Denote by $(lb, ub)^{k,f}_{1:L}$ the lower and upper bounds encountered along $tr(k)^f_{1:L}$ in the tree. Without loss of generality, assume that two counters in f, $\#_1 = \{0, \ldots, |\#_1|\}$, and $\#_2 = \{0, \ldots, |\#_2|\}$, contain shared count variable B. For Op at a leaf in f' to be over valid values we have to show that *any valid trace* in f', $tr(k)^{f'}_{1:L}$ with $\#_1 = v$, $\#_2 = w$, is also valid in f *and further that a second trace* $tr(l)^f_{1:L}$ with $\#_1 = v + 1$, $\#_2 = w + 1$ is also valid in f.

First, given arbitrary trace $tr(k)^{f'}_{1:L}$, denote by $tr(k)^f_{1:L}$ the equivalent trace in f, *i.e.* the one where the assignments (subtrees) are chosen identically. Assume $tr(k)^{f'}_{1:L}$ is valid. Then, adding a (shared count) variable to both $\#_1$ and $\#_2$ (to form f) will never increase any of the $lb^{k,f'}_{1:L}$ or decrease any $ub^{k,f'}_{1:L}$ since the interval of permissible values at any level $1 - L$ in $tr(k)^f_{1:L}$ cannot be reduced by virtue of having an additional variable available. It follows that $tr(k)^f_{1:L}$ is also valid.

Second, it can be seen at the top of Figure 6.4 that removing a shared count variable from function f in Figure 6.3 has the effect of *bounding above* all corresponding upper bounds in

f' by $|\#_1| - 1$ and $|\#_2| - 1$ by virtue of removing a variable from both counters. Here, "bounding above" refers to either reducing the upper bound directly or removing a previously invalid entry. Further, consider f's tree expansion and observe that upper bounds within the same level of the tree *increase strictly by 1* up to the maximum value $|\#_1|$ (respectively $|\#_2|$). Combining all three results that *i)* validity of $tr(k)^{f'}_{1:L}$ implies validity of $tr(k)^{f}_{1:L}$, *ii)* new upper bounds $ub^{k,f'}_{1:L}$ are bounded above by $|\#_1| - 1$ and $|\#_2| - 1$, and *iii)* upper bounds increase monotonically up to $|\#_1|$ and $|\#_2|$ in f, establishes that $tr(l)^{f}_{1:L}$ is also valid in f. □

Appendix B

Evaluation Domains

B.1 The SysAdmin Domain

The *SysAdmin* domain corresponds to a multiagent version of the domain with the same name that originally appeared in [36]. The multiagent extension described here has been used in the Boolean track of the International Planning Competition (IPPC) [88] and serves as a baseline in much recent work on planning with large factored action spaces (*e.g.*, [20, 84, 85]).

The *SysAdmin* domain models a set of N computers arranged in either a star or ring topology. Individual machines can fail with a certain probability; the global state space is spanned by binary variables $\mathbf{X} = \{X_1, \ldots, X_N\}$ where $X_i = \{0, 1\} = \{\text{failed}, \text{running}\}$ denotes the state of computer i.

If a neighboring machine in the topology has failed, the probability of oneself failing at the next time step increases by a large amount. A team of system administrators (one per machine) is in charge of maintaining the computers and collects a fixed reward per running computer. System administrators may reboot individual machines to guarantee their uptime at the next time step, *i.e.* $\mathbf{A} = \{A_1, \ldots, A_N\}$ where $A_j = \{0, 1\} = \{\text{do nothing}, \text{reboot}\}$.

The reward model incentivizes running computers and penalizes reboot actions by the administrators:

$$R(\mathbf{x}, \mathbf{a}) = \|\mathbf{x}\|_1 - \lambda \|\mathbf{a}\|_1$$

where the L_1-norms implement a reward of 1 per running machine discounted by a *reboot penalty* of λ. The optimization problem is to minimize reboot interventions while ensuring high uptime for all computers in the network.

B.1.1 ResourceProtect

The *ResourceProtect* domain is a minor modification to the *SysAdmin* domain where a much higher reward r^+ is associated with a specific machine (or, resource) in the network, denoted by state factor X^+. The only modification to *SysAdmin* lies therefore with the reward model:

$$R(\mathbf{x}, \mathbf{a}) = \|\mathbf{x}[\mathbf{X} \setminus \{X^+\}]\|_1 + r^+ x^+ - \lambda_1 \|\mathbf{a}\|_1$$

where $\mathbf{x}[\mathbf{X} \setminus \{X^+\}]$ extracts from state $\mathbf{x}$ the state of the "regular" computers and x^+ refers to the state of the protected resource in $\mathbf{x}$. Of particular interest are the effects of the presence of X^+ on the policy for the system administrator team.

B.2 The TaskNetwork Domain

In the *TaskNetwork* domain, a computer network completes jobs based on both *status* and *current load* at each machine, *i.e.* $\mathbf{X} = \{\langle S_1, L_1\rangle, \ldots, \langle S_N, L_N\rangle\}$. Figure B.1 shows the factored model in plate notation which is instantiated for different numbers of computers N. As shown, each computer i is associated with two (ternary) variables, one indicating its health and the other its current load:

$$\langle S_i \in \{\text{running, faulty, failed}\}, L_i \in \{\text{idle, loaded, task complete}\}\rangle \qquad \forall i = 1, \ldots, N$$

A reward of 1 is collected for every task that reaches the *task complete* state in the network. A machine is then reset deterministically to the *idle* state and may process another task. Stochastic downtime effects, similar to the *SysAdmin* domain, are defined over the computers in the network. In this domain, *faulty* or *failed* states impact the status at the next time step for neighboring machines. A *faulty* machine is also less efficient at processing tasks than a *running* machine. The action space is again spanned by binary decision variables, *i.e.* $\mathbf{A} = \{A_1, \ldots, A_N\}$ where $A_j = \{0, 1\} = \{\text{do nothing, reboot}\}$.

The goal is to move as many tasks as possible to completion in the network. A good policy has to trade off the penalty of reboot actions (which additionally remove any tasks from processing at the respective machine) with the risk of propagating downtime effects in the network. To avoid confusion with the simpler *SysAdmin* domain, we refer to this problem as a *TaskNetwork* instead of *Multiagent-SysAdmin* under which name it has also appeared [36].

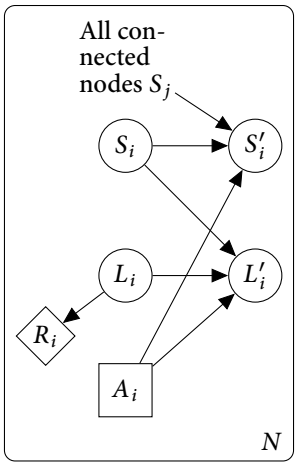

Figure B.1: The *TaskNetwork* domain in plate notation which can be instantiated for N computers. All variables take three values; the action choice at each node is binary [36].

B.3 The Disease Control Domain

We use the domain of controlling a disease outbreak over a graph to serve as the running example in this paper. Note that the control of stochastic dynamics over graphs has wider applications, e.g. in management of electric power grids and network intrusion [45]. Other domains aim to minimize collateral diffusion effects while actively targeting specific nodes in the graph (e.g. drugs in biological networks) [96].

Underlying the formulation as a FMMDP is a (directed or undirected) graph $G = (V, E)$ with controlled and uncontrolled vertices $V = (V_c, V_u)$ and the edge set $E \subseteq V \times V$. The state space $\mathcal{S}$ is spanned by state variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, one per associated vertex V_i, encoding the health of that node. The action set $\mathbf{A} = \{A_1, \ldots, A_{|V_c|}\}$ factors similarly over the controlled vertices V_c in the graph and each denotes a modulation of the *flow out of* node $V_i \in V_c$. Let $x_i = \mathbf{x}[\{X_i\}]$ and $a_i = \mathbf{a}[\{A_i\}]$ denote the state and action for a single node. Then reward and transition model factor on a per-node basis as:

$$R(\mathbf{x}, \mathbf{a}) = \sum_{i}^{n} R_i(x_i, a_i) \tag{B.1}$$

$$P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) = \prod_{i}^{n} T_i(x'_i \mid \mathbf{x}[\mathrm{Pa}(X'_i)], a_i) \tag{B.2}$$

where the set $\mathrm{Pa}(X'_i)$ includes variable X_i at the previous time step as well as all nodes that *flow into* X_i in G. The known infection transmission probabilities from node j to i, β_{ji}, and the

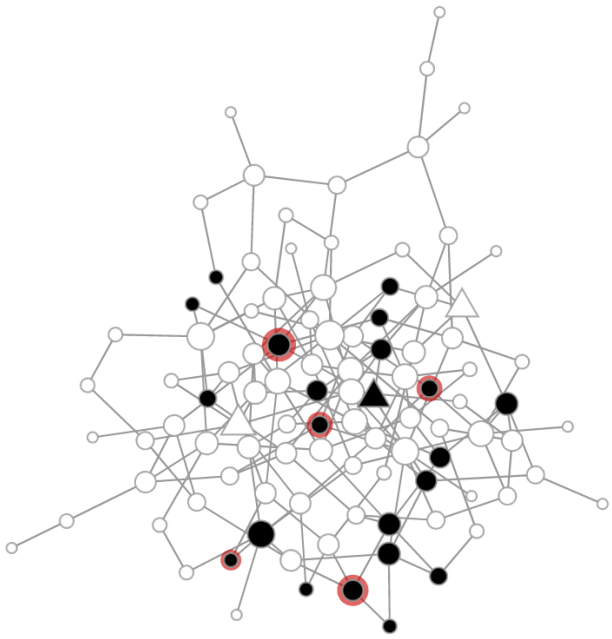

Figure B.2: An example disease propagation scenario over a random graph with 100 nodes. Infected nodes are shown in black and highlighted with a red halo in case of a new infection at the current iteration. Controlled nodes V_c are shown as triangles, those in V_u as circles.

recovery rate of node i, δ_i, define the transition function $T_i(x_i' \mid \mathbf{x}[\mathrm{Pa}(X_i')], a_i)$ as follows:

$$T_i \triangleq \begin{cases} (1-a_i)(1-\prod_j(1-\beta_{ji}x_j)) & \text{if } x_i = 0 \\ (1-a_i)\,(1-\delta_i) & \text{otherwise} \end{cases} \tag{B.3}$$

distinguishing the two cases that X_i was infected at the previous time step (bottom) or not (top). Note that this model assumes binary state variables $X_i = \{0,1\} = \{\text{healthy}, \text{infected}\}$, and actions $A_i = \{0,1\} = \{\text{do not vaccinate}, \text{vaccinate}\}$ and that $A_u = \{0\}$ for all uncontrolled nodes V_u. The reward function factors as:

$$R(\mathbf{x}, \mathbf{a}) = -\lambda_1 \|\mathbf{a}\|_1 - \lambda_2 \|\mathbf{x}\|_1 \tag{B.4}$$

where the L_1-norm records a cost λ_2 per infected node X_i and an action cost λ_1 per vaccination action at a controlled node. All our experiments are for the infinite horizon case on an undirected graph G of varying size and structure but with consistent transmission and recovery rates β, δ.

B.3.1 GraphContainment

In the *GraphContainment* domain, the disease process may propagate without penalty among all *uncontrolled* nodes V_u in the graph while a penalty cost applies only at the infected, *controlled* nodes V_c. The only modification to the *DiseasePropagation* domain is therefore in the reward model:

$$R(\mathbf{x}, \mathbf{a}) = -\lambda_1 \|\mathbf{a}\|_1 - \lambda_2 \|\mathbf{x}[\mathbf{X}_c]\|_1 \tag{B.5}$$

where $\mathbf{x}[\mathbf{X}_c]$ extracts the state of the controlled nodes V_c from $\mathbf{x}$.

The goal is to contain the stochastic process in one subset of the graph and controlled nodes V_c have to trade off a vaccination action (at a vaccination cost λ_1) with letting the process pass through to "penalty-free" regions of the graph.

Bibliography

[1] A. Ahmed, P. Varakantham, and S. Cheng. Uncertain congestion games with assorted human agent populations. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 44–53, 2012.

[2] S. Amador, S. Okamoto, and R. Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1384–1390, 2014.

[3] C. Amato and F. A. Oliehoek. Scalable planning and learning for multiagent POMDPs. In *Twenty-Ninth Conference on Artificial Intelligence (AAAI)*, pages 1995–2002, Jan. 2015.

[4] A. Anand, A. Grover, Mausam, and P. Singla. ASAP-UCT: Abstraction of state-action pairs in UCT. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1509–1515, 2015.

[5] J. L. Barry, L. P. Kaelbling, and T. Lozano-Pérez. Hierarchical solution of large Markov decision processes. In *ICAPS 2010 Workshop on Planning and Scheduling Under Uncertainty*, May 2010.

[6] J. L. Barry, L. P. Kaelbling, and T. Lozano-Pérez. Deth*: Approximate hierarchical solution of large Markov decision processes. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1928–1935, 2011.

[7] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, Jan. 1995.

[8] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized Markov decision processes. In *AAMAS*, pages 41–48, 2003.

[9] C. Bererton, G. J. Gordon, and S. Thrun. Auction mechanism design for multi-robot coordination. *Advances in Neural Information Processing Systems 16*, pages 879–886, 2004.

[10] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[11] B. Bethke and J. How. Approximate dynamic programming using Bellman residual elimination and Gaussian process regression. In *American Control Conference, 2009. ACC '09.*, pages 745–750, June 2009.

[12] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[13] S. R. K. Branavan, D. Silver, and R. Barzilay. Non-linear Monte-Carlo search in Civilization II. In *Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2404–2410, Barcelona, Spain, 2011.

[14] J. Capitán, M. T. J. Spaan, L. Merino, and A. Ollero. Decentralized multi-robot cooperation with auctioned POMDPs. *International Journal of Robotics Research*, 32(6):650–671, 2013.

[15] Q. Cheng, Q. Liu, F. Chen, and A. Ihler. Variational planning for graph-based MDPs. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2976–2984, 2013.

[16] J. Choi, R. de Salvo Braz, and H. H. Bui. Efficient methods for lifted inference with aggregate factors. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI-11)*, pages 1030–1036, 2011.

[17] D. Claes, P. Robbel, F. A. Oliehoek, D. Hennes, and K. Tuyls. Effective approximations for spatial task allocation problems. In *Proc. of the 25th Benelux Conference on Artificial Intelligence (BNAIC)*, pages 33–40, 2013. Best paper runner up.

[18] D. Claes, P. Robbel, F. A. Oliehoek, D. Hennes, and K. Tuyls. Effective approximations for spatial task allocation problems. In *Proc. of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 881–890, May 2015.

[19] P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. In *Proceedings of Uncertainty in Artificial Intelligence*, 2007.

[20] H. Cui, R. Khardon, A. Fern, and P. Tadepalli. Factored MCTS for large scale stochastic planning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3261–3267, 2015.

[21] Y.-M. De Hauwere, P. Vrancx, and A. Nowé. Learning multi-agent state space representations. In *AAMAS*, pages 715–722, 2010.

[22] J. S. Dibangoye, C. Amato, O. Buffet, and F. Charpillet. Exploiting separability in multiagent planning with continuous-state MDPs. In *Proceedings of the Thirteenth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1281–1288, 2014.

[23] J. S. Dibangoye, C. Amato, and A. Doniec. Scaling up decentralized MDPs through heuristic search. In *UAI*, pages 217–226, 2012.

[24] T. G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 118–126, 1998.

[25] D. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, April 2006.

[26] P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive POMDPs: Representations and solutions. *Autonomous Agents and Multi-Agent Systems*, 18(3):376–416, 2008.

[27] A. M. Farahmand and D. Precup. Value pursuit iteration. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1349–1357, 2012.

[28] D. P. D. Farias and B. V. Roy. The linear programming approach to approximate dynamic programming. *Oper. Res.*, 51(6):850–865, 2003.

[29] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, pages 639–646, May 2008.

[30] S. Gelly and D. Silver. Monte-Carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[31] A. Geramifard, T. J. Walsh, N. Roy, and J. P. How. Batch-iFDD for representation expansion in large MDPs. *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 242–251, 2013.

[32] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[33] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.

[34] G. J. Gordon, P. Varakantham, W. Yeoh, H. C. Lau, A. S. Aravamudhan, and S. Cheng. Lagrangian relaxation for large-scale multi-agent planning. In *2012 IEEE/WIC/ACM International Conferences on Intelligent Agent Technology, IAT 2012, Macau, China, December 4-7, 2012*, pages 494–501, 2012.

[35] O. Guéant, J.-M. Lasry, and P.-L. Lions. Mean field games and applications. In *Paris-Princeton Lectures on Mathematical Finance 2010*, volume 2003 of *Lecture Notes in Mathematics*, pages 205–266. Springer Berlin Heidelberg, 2011.

[36] C. Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Computer Science Department, Stanford University, August 2003.

[37] C. Guestrin and G. Gordon. Distributed planning in hierarchical factored MDPs. *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 197–206, 2002.

[38] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational MDPs. In *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, August 2003.

[39] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems (NIPS 2001)*, pages 1523–1530, Vancouver, Canada, 2002.

[40] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[41] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proc. of the International Conference on Machine Learning*, pages 227–234, 2002.

[42] C. Guestrin, S. Venkataraman, and D. Koller. Context specific multiagent coordination and planning with factored MDPs. In *AAAI*, pages 253–259, 2002.

[43] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 220–229, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[44] T. Hester and P. Stone. Learning and using models. In *Reinforcement Learning: State of the Art*. Springer Verlag, 2011.

[45] C. Ho, M. J. Kochenderfer, V. Mehta, and R. S. Caceres. Control of epidemics on graphs. In *54th IEEE Conference on Decision and Control (CDC)*, Osaka, Japan, 2015.

[46] J. Hoey, R. St-Aubin, A. J. Hu, and C. Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of Uncertainty in Artificial Intelligence*, Stockholm, Sweden, 1999.

[47] J. Johns. *Basis Construction and Utilization for Markov Decision Processes using Graphs*. PhD thesis, University of Massachusetts Amherst, 2010.

[48] T. Keller and P. Eyerich. PROST: Probabilistic planning based on UCT. In *International Conference on Automated Planning and Scheduling*, 2012.

[49] K. Kersting, B. Ahmadi, and S. Natarajan. Counting belief propagation. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 277–284, Arlington, Virginia, United States, 2009. AUAI Press.

[50] M. J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[51] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, volume 4212, pages 282–293, 2006.

[52] J. R. Kok, P. J. 't Hoen, B. Bakker, and N. Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 29–36, Colchester, United Kingdom, Apr. 2005.

[53] J. R. Kok and N. A. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[54] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[55] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1332–1339, 1999.

[56] D. Koller and R. Parr. Policy iteration for factored MDPs. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 326–334, June 2000.

[57] J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, 94305:1–8, 2009.

[58] G. Konidaris and A. G. Barto. Efficient skill learning using abstraction selection. In C. Boutilier, editor, *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI)*, pages 1107–1112, 2009.

[59] A. Kumar, S. Zilberstein, and M. Toussaint. Scalable multiagent planning using probabilistic inference. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2140–2146, Barcelona, Spain, 2011.

[60] A. Kumar, S. Zilberstein, and M. Toussaint. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research*, 2015. accepted.

[61] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[62] T. Lang, M. Toussaint, and K. Kersting. Exploration in relational domains for model-based reinforcement learning. *Journal of Machine Learning Research*, 13:3691–3734, 2012.

[63] H. lim Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 2009.

[64] T. Mann and S. Mannor. Scaling up approximate value iteration with options: Better policies with fewer iterations. In T. Jebara and E. P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 127–135. JMLR Workshop and Conference Proceedings, 2014.

[65] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib. Coordinated multi-robot exploration under communication constraints using decentralized Markov decision processes. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2017–2023, 2012.

[66] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib. Distributed value functions for multi-robot exploration. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1544–1550, May 2012.

[67] F. S. Melo and M. Veloso. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *AAMAS*, pages 773–780, 2009.

[68] B. Milch, L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. Lifted probabilistic inference with counting formulas. In *Twenty Third Conference on Artificial Intelligence (AAAI)*, pages 1062–1068, 2008.

[69] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and pomdps. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, pages 133–139. AAAI Press, 2005.

[70] A. Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. *Twenty-first international conference on Machine learning - ICML '04*, page 78, 2004.

[71] D. T. Nguyen, W. Yeoh, H. C. Lau, S. Zilberstein, and C. Zhang. Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, pages 1341–1342, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[72] F. A. Oliehoek. Decentralized POMDPs. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State of the Art*, volume 12 of *Adaptation, Learning, and Optimization*, pages 471–503. Springer Berlin Heidelberg, Berlin, Germany, 2012.

[73] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[74] C. Painter-Wakefield, R. Parr, P. Cs, and D. Edu. Greedy algorithms for sparse reinforcement learning. *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1391–1398, 2012.

[75] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 752–759, 2008.

[76] R. Parr, C. Painter-Wakefield, and L. Li. Analyzing feature generation for value-function approximation. *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 737–744, 2007.

[77] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44 vol.1, Nov 1993.

[78] J. Pazis. Non-parametric approximate linear programming for MDPs. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 459–464, 2011.

[79] T. P. Peixoto. The graph-tool Python library. *figshare*, 2014.

[80] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. In *IJCAI 05*, pages 266–271, Edinburgh, Scotland, Aug 2005.

[81] M. Petrik, G. Taylor, R. Parr, and S. Zilberstein. Feature selection using regularization in approximate linear programs for Markov decision processes. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 871–878, 2010.

[82] P. Poupart, C. Boutilier, R. Patrascu, D. Schuurmans, and C. Guestrin. Greedy linear value-approximation for factored Markov decision processes. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 292–299, 2002.

[83] M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, New York, 2005. A Wiley-Interscience publication.

[84] A. Raghavan, S. Joshi, A. Fern, P. Tadepalli, and R. Khardon. Planning in factored action spaces with symbolic dynamic programming. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[85] A. Raghavan, P. Tadepalli, A. Fern, and R. Khardon. Memory-efficient symbolic online planning for factored MDPs. In *Proceedings of the Thirty-First Annual Conference on Uncertainty in Artificial Intelligence (UAI-15)*. AUAI Press, July 2015.

[86] J. Redding, N. Ure, J. How, M. Vavrina, and J. Vian. Scalable, mdp-based planning for multiple, cooperating agents. In *American Control Conference (ACC), 2012*, pages 6011–6016, June 2012.

[87] P. Robbel, F. A. Oliehoek, and M. J. Kochenderfer. Exploiting anonymity in approximate linear programming: Scaling to large multiagent MDPs. In *Proceedings of the AAAI Fall Symposium: Sequential Decision Making for Intelligent Agents*, Washington, DC, November 2015.

[88] S. Sanner. ICAPS 2011 international probabilistic planning competition. `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011`, 2011. Accessed: 2015-08-01.

[89] S. Sanner and C. Boutilier. Practical solution techniques for first-order MDPs. *Artificial Intelligence*, 173(5-6):748–788, Apr. 2009.

[90] S. Sanner and K. Kersting. Symbolic dynamic programming for first-order POMDPs. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 1140–1146, 2010.

[91] J. G. Schneider, W.-K. Wong, A. W. Moore, and M. A. Riedmiller. Distributed value functions. In *Proc. of the International Conference on Machine Learning*, pages 371–378, 1999.

[92] D. Silver. *Reinforcement Learning and Simulation-Based Search in Computer Go*. PhD thesis, University of Alberta, 2009.

[93] D. Silver, R. S. Sutton, and M. Müller. Sample-based learning and search with permanent and transient memories. In *Twenty-Fifth International Conference on Machine Learning (ICML)*, pages 968–975, 2008.

[94] J. Sleight and E. H. Durfee. A decision-theoretic characterization of organizational influences. In *Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 323–330, 2012.

[95] M. T. J. Spaan and F. S. Melo. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *AAMAS*, pages 525–532, 2008.

[96] S. Srihari, V. Raman, H. W. Leong, and M. A. Ragan. Evolution and controllability of cancer networks: A boolean perspective. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 11(1):83–94, Jan 2014.

[97] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[98] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, pages 181–211, 1999.

[99] N. Taghipour, J. Davis, and H. Blockeel. Generalized counting for lifted variable elimination. *Springer Lecture Notes in Computer Science: Inductive Logic Programming*, 8812:107–122, 2014.

[100] N. Taghipour, D. Fierens, J. Davis, and H. Blockeel. Lifted variable elimination: Decoupling the operators from the constraint language. *Journal of Artificial Intelligence Research*, 47:393–439, 2013.

[101] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

[102] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[103] M. Toussaint, L. Charlin, and P. Poupart. Hierarchical POMDP controller optimization by likelihood maximization. In *Uncertainty in Artificial Intelligence (UAI 2008)*, pages 562–570. AUAI Press, 2008.

[104] N. Ure, G. Chowdhary, Y. F. Chen, M. Cutler, J. How, and J. Vian. Decentralized learning-based planning for multiagent missions in the presence of actuator failures. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 1125–1134, May 2013.

[105] N. K. Ure, J. P. How, and J. Vian. Randomized coordination search for scalable multiagent planning. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1793–1794, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[106] M. J. Valenti. *Approximate Dynamic Programming with Applications in Multi-Agent Systems*. PhD thesis, Massachusetts Institute of Technology, 2007.

[107] M. van Otterlo. *The Logic of Adaptive Behavior - Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*. Phd thesis, University of Twente, May 2008.

[108] P. Varakantham, S. Cheng, G. J. Gordon, and A. Ahmed. Decision support for agent populations in uncertain and congested environments. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

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

[110] S. Whiteson, B. Tanner, M. E. Taylor, and P. Stone. Generalized domains for empirical evaluations in reinforcement learning. In *ICML Workshop on Evaluation Methods for Machine Learning*, June 2009.

[111] S. Whiteson, B. Tanner, M. E. Taylor, and P. Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, April 2011.

[112] R. J. Williams and L. C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU–CCS-93-14, Northeastern University, Nov 1993.

[113] J. Wu and E. H. Durfee. Resource-driven mission-phasing techniques for constrained agents in stochastic environments. *Journal of Artificial Intelligence Research*, 38:415–473, 2010.

[114] W. Yeoh and M. Yokoo. Distributed problem solving. *AI Magazine*, 33(3):53–65, 2012.