5-2013

Exploiting Domain Structure in Multiagent Decision-Theoretic Planning and Reasoning

Akshat Kumar
University of Massachusetts Amherst, akshat.kumar@gmail.com

EXPLOITING DOMAIN STRUCTURE IN MULTIAGENT DECISION-THEORETIC PLANNING AND REASONING

A Dissertation Presented

by

AKSHAT KUMAR

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

DOCTOR OF PHILOSOPHY

May 2013

Computer Science

EXPLOITING DOMAIN STRUCTURE IN MULTIAGENT DECISION-THEORETIC PLANNING AND REASONING

A Dissertation Presented

by

AKSHAT KUMAR

Approved as to style and content by:

Shlomo Zilberstein, Chair

Daeyoung Kim, Member

Victor Lesser, Member

Sridhar Mahadevan, Member

Lori Clarke, Department Chair
Computer Science

To my parents and family

ACKNOWLEDGMENTS

First and foremost, this thesis would not have been possible without the direction of my advisor Shlomo Zilberstein. His faith in my ability, constant encouragement, both at the professional and the personal level and insightful technical advice were the key to the research directions of this thesis. Shlomo, in many wonderful ways, has been a great advisor for me and I hope to reach his high standards of advising in my career as well.

A special thanks is also due to my committee members. I met Victor on my first day in the department. In my every single subsequent meeting with him, his constant encouragement of my work and his passion for research in multiagent systems kept me motivated. His advice has been crucial in shaping the bigger picture of this thesis. Sridhar has been a great mentor as well as an inspiring teacher. I credit his machine learning course as the key enabler to bringing in the ML flavor of my thesis. Daeyoung Kim has been the source of many insightful pointers to some of the intricate machinery of my work.

I also wish to thank Boi Faltings and Marc Toussaint. Boi gave me my first break in AI and multiagent systems. His advice and encouragement early into my grad student life were instrumental in shaping up my research direction. Interactions with Marc were crucial in the clean formulation of technical aspects of some of my work.

I had been fortunate to be in the good company of fellow members of RBR and MAS labs. Thanks Marek for promoting the awareness of Microsoft products and trying to steer me to single agent systems, Chris for his energy and fantasy baseball discussions, Sid for being a source of much awaited dinner invitations to his home, Alan for being a travel companion and a source of American humor, Xiaojian for being my guide into the Chinese culture, both the Luises for bringing in the fresh energy to the lab and Rick for being such an energetic project partner.

MAS lab members were equally phenomenal—Hala was a constant presence in my discussions on multiagent systems to other late night cribs, thanks Yoonheui for defeating me

in many Racquetball games and making me work harder and introducing me to Korean cuisine, Chongjie and Xiaoxi constantly encouraged me even when things were not rosy. Will, another extended member of RBR and MAS lab, had been a great source of comic relief and a trusted colleague. My friends in the CS department were equally great. Rahul provided expert gym lessons for free, Uppi has been a trusted source of advice on all philosophical, spiritual and farcical matters, Naveen's focused enthusiasm had been inspiring, Parate taught me the art of wine tasting and Keen provided a roof over my head for my final few days. Ashish, Mohit, Rishi and Jabhi have always been the blast from the past! Without the support of such friends, my experience at UMass would have been incredibly dull.

I also owe a lot to the CS department for being such a friendly place to students. Leeanne was most helpful and resourceful in navigating the academic maze. Michele made conference travel so much smoother and efficient!

This thesis is dedicated to my parents and sister. Without their unflinching support and unwavering belief in my abilities over a number of years, I would have gone nowhere. They provided me the freedom to pursue my dreams even in the face of personal sacrifices. I stand forever in their debt for their support and encouragement.

ABSTRACT

EXPLOITING DOMAIN STRUCTURE IN MULTIAGENT DECISION-THEORETIC PLANNING AND REASONING

MAY 2013

AKSHAT KUMAR

B.Tech., INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Shlomo Zilberstein

This thesis focuses on decision-theoretic reasoning and planning problems that arise when a group of collaborative agents are tasked to achieve a goal that requires collective effort. The main contribution of this thesis is the development of effective, scalable and quality-bounded computational approaches for multiagent planning and coordination under uncertainty. This is achieved by a synthesis of techniques from multiple areas of artificial intelligence, machine learning and operations research. Empirically, each algorithmic contribution has been tested rigorously on common benchmark problems and, in many cases, real-world applications from machine learning and operations research literature.

The first part of the thesis addresses multiagent *single-step* decision making problems where a single joint-decision is required for the plan. We examine these decision-theoretic problems within the broad frameworks of distributed constraint optimization and Markov random fields. Such models succinctly capture the structure of interaction among different decision variables, which is subsequently exploited by algorithms to enhance scalability. The

algorithms presented in this thesis are rigorously grounded on concepts from mathematical programming and optimization.

The second part of the thesis addresses multiagent *sequential* decision making problems under uncertainty and partial observability. We use the decentralized partially observable Markov decision processes (Dec-POMDPs) to formulate multiagent planning problems. To address the challenge of NEXP-Hard complexity and yet push the envelope of scalability, we represent the domain structure in a multiagent system using graphical models such as dynamic Bayesian networks and constraint networks. By exploiting such graphical planning representation in an algorithmic framework composed of techniques from different sub-areas of artificial intelligence, machine learning and operations research, we show impressive gains in increasing the scalability, the range of problems addressed and enabling quality-bounded solutions for multiagent decision theoretic planning.

Our contributions for sequential decision making include **a)** development of efficient dynamic programming algorithms for finite-horizon decision making, resulting in significantly increased scalability w.r.t. the number of agents and multiple orders-of-magnitude speedup over previous best approaches; **b)** development of probabilistic inference based algorithms for infinite-horizon decision making, resulting in new insights connecting inference techniques from the machine learning literature to multiagent systems; **c)** development of mathematical programming based scalable techniques for quality bounded solutions in multiagent systems, which has been considered intractable so far.

Several of our contributions are some of the first for the respective class of problems. For example, we show for the first time how machine learning is closely related to multiagent decision making via a maximum likelihood formulation of the planning problem. We develop new graphical models and machine learning based inference algorithms for large factored planning problems. We also show for the first time how the problem of optimizing agents' policies can be formulated as a compact mixed-integer program, resulting in optimal solution for a range of Dec-POMDP benchmarks.

In summary, we present a synthesis of different techniques from multiple sub-areas of AI, ML and OR to address the scalability and efficiency of algorithms for decision-theoretic

reasoning and planning in multiagent systems. Such advances have already shown great promise to bridge the gap between multiagent systems and real-world applications.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

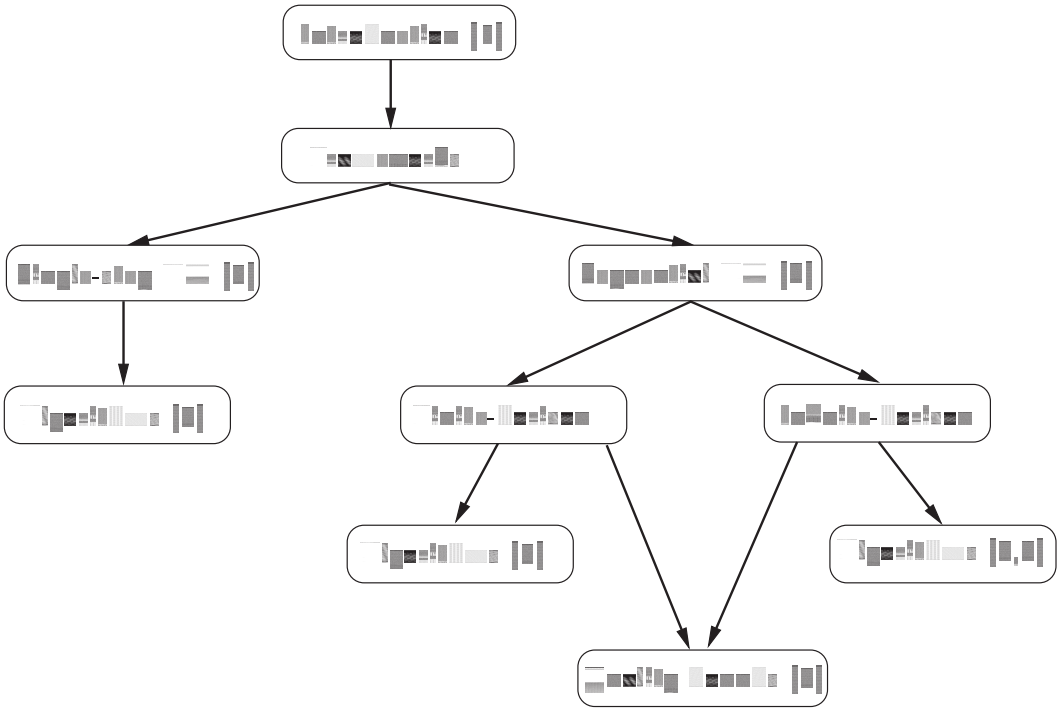


Figure 1. Chapter and concept dependencies. Numbers in (·) denote the chapter number. Text without (·) denote a concept.

CHAPTER 1

INTRODUCTION

Multiagent systems are one of the cornerstones of artificial intelligence. A key challenge in multiagent systems is how to achieve intelligent coordination among agents despite availability of only limited information to each agent and stochasticity in the environment. In this thesis, we address this problem of multiagent planning and coordination using decision-theoretic formal frameworks. When only a single decision maker is present in the environment, then the decision-theoretic framework of partially observable MDPs is a popular and well understood choice. However, extending decision theory to address multiagent systems presents unique challenges such as significantly high complexity barrier than single-agent approaches and the limited scalability of even approximate approaches. *The main contribution of this thesis is the development of effective, scalable and quality-bounded computational approaches for multiagent planning and coordination. This is achieved by exploiting the domain structure in an algorithmic framework developed using a synthesis of techniques from multiple areas of artificial intelligence, machine learning and operations research.*

When this thesis was started 5 years ago, research in decision-theoretic multiagent planning was mainly focused on developing scalable techniques for just 2-agent systems. There were attempts to handle larger multiagent systems using restricted models, but their scalability and the range of addressed problems were limited. Furthermore, the majority of the relatively scalable algorithms were approximate and could not provide quality bounds. Much of the research altogether avoided the question of quality bounds due to perceived high intractability. The already proven complexity result that finding the optimal policy just for 2-agents is NEXP-Complete hinted that despite the tantalizing promise of decision-theoretic multiagent planning, developing effective and scalable approaches was bound to be a fruitless effort.

The main insight which pushed us to move forward in addressing the challenges of multiagent coordination and planning was that despite the theoretical hardness, many real-world inspired problems have underlying structure that can be exploited to yield scalable approaches. Furthermore, instead of reinventing the wheel for scalable multiagent planning, one may fare significantly better by bringing algorithmic advances from different areas of artificial intelligence (AI), machine learning (ML) and operations research (OR) to the context of multiagent systems. Some of the big questions which this thesis addresses include: how to represent domain structure in a generic manner, how to exploit it in an algorithmic framework and how to synthesize algorithmic advances in different sub-areas of AI, ML and OR for scalable multiagent coordination and planning. We highlight this strategy below:

- ***Frameworks:*** We use the decision-theoretic frameworks of distributed constraint optimization (DCOP) and decentralized partially observable MDPs (Dec-POMDPs) to address multiagent decision making. DCOPs model single-step decision making, where agents need to take a single action in a coordinated manner to maximize the joint utility. Dec-POMDPs model scenarios where a team of agents need to take multiple actions in a sequential manner to achieve their goal. Developing optimal algorithms for DCOPs is NP-Hard and for Dec-POMDPs, it is NEXP-Hard.

- ***Representation:*** The first requirement for exploiting domain structure in a multiagent system is to find frameworks that can represent precisely the interdependencies among multiple agents in a generic manner. We use the well established language of *probabilistic graphical models* to represent structure within a large multiagent system. We use both directed models such as *dynamic Bayesian networks* and undirected models such as *Markov random fields* that lay bare the independencies present in a multiagent system for exploitation in an algorithmic framework.

- ***Reasoning:*** Once the structure in a multiagent system is established using a graphical model, we develop new algorithms that leverage techniques from AI, ML and optimization literature to exploit the structure of this graphical model. For example, we show that DCOPs can be handled by mapping them to Markov random fields and then solved using mathematical optimization based approaches that result in a dis-

tributed message-passing scheme over the underlying graph. Similarly, we show that a Dec-POMDP can be modeled using a mixture of dynamic Bayesian networks and thus, amenable to probabilistic inference based techniques from the ML literature.

Our main contributions are as following:

- For *single-step multiagent decision making* problems modeled using DCOPs, our approach has been to develop scalable algorithms using the framework of variational inference from machine learning. Our approach is the first that unifies two different existing frameworks based on quadratic and linear programming for DCOPs. We show that the resulting hybrid framework is analytically tighter than existing formulations. Furthermore, it combines desirable properties of existing formulations such as accuracy and convexity, while minimizing the impact of their weaknesses. We develop a scalable mathematical optimization based algorithm that efficiently exploits the domain structure using a message-passing scheme over the underlying graph. Our approach is always convergent, can provide tight quality bounds unlike other approximate approaches such as Max-Sum for DCOPs and empirically, provides near-optimal solutions for a number of synthetic and real-world benchmarks from multiagent systems, machine learning and operations research literature.

- For *sequential multiagent decision making* problems modeled using Dec-POMDPs, our approach has been to exploit techniques from machine learning and mathematical optimization literature for developing scalable algorithms. Our main contributions are as follows:

  - We show how to exploit the underlying problem structure for finite-horizon planning using constraint networks, which are a sub-class of graphical models. Using techniques from constraint optimization literature, our approach is more than an order-of-magnitude faster than previous approaches and scalable even for larger multiagent systems that could not be handled using previous approaches.

  - We show how the multiagent planning problem can be reformulated as that of probabilistic inference in a dynamic Bayesian network based graphical model.

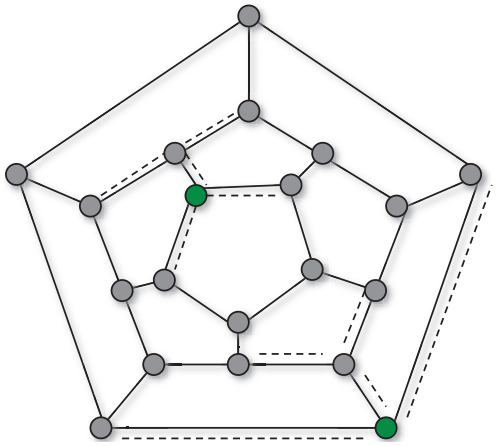
Using this insight, we develop new algorithms based on techniques from the ML literature such as Expectation-Maximization for multiagent planning. Our approach is the first to bring such inference-based view to multiagent planning. The insight of representing a planning problem using a graphical model exposes independencies among different problem parameters enabling scalable algorithms, and allows one to potentially address much richer class of problems such as factored and continuous planning models. We develop one such factored model based on value factorization and show how using such a graphical model based view results in a highly scalable message-passing algorithm.

– We develop the first mixed-integer linear programming (MILP) based approach for Dec-POMDPs than enables quality-bounded solutions. Furthermore, our approach is also the first that can formulate single-agent planning problems modeled using POMDPs as a MILP. Using such MILP-based formulations allows us to use highly efficient industrial strength solvers such as CPLEX for multiagent planning. We further show how one can exploit loosely-coupled multiagent systems in a Lagrangian relaxation based optimization framework to get scalable, quality bounded solutions. Empirically, our approach is more than an order-of-magnitude faster than previous approaches and provides provably near-optimal solutions for a number of benchmark problems for the first time.

1.1 Examples of Multiagent Decision Making

Multiagent decision making is applicable in several different domains such as multi-robot coordination [12, 100], broadcast channel protocols [13] and target tracking by a team of sensors [99]. Some emerging application areas of multiagent decision making include smart power grids [118], collaborative sensing of the atmosphere [91, 69] and cloud computing [167] among others. Smart grids present a radical new design from its current monolithic structure to being distributed and autonomous in nature. The presence of multiple renewable sources of power and the uncertainty in supply and demand present a fertile ground for multiagent decision making under uncertainty. My previous work has addressed the important task

(a) The DARPS ANTS project [83]



(b) An abstraction of the problem

Figure 1.1. Target tracking in sensor networks.

of reconfiguring a power grid after line failures using a distributed message-passing based approach [74]. Another example of multiagent decision making is the distributed collaborative sensing initiative of the NSF that focuses on a distributed sensing of the atmosphere, rather than using a few, high cost radars [91] . My work has developed new models and tractable algorithms for early outbreak detection of stochastically evolving events, such as tornadoes in the atmosphere [69].

We next describe the single-step and sequential version of a multiagent sensor network problem.

1.1.1 Single-Step Multiagent Decision Making

Fig. 1.1(a) shows the target tracking problem in a sensor network for the DARPA ANTS project [83]. The tracks in the figure show where mobile targets can move. The task for the stationary sensors is to track the targets by coordinated scanning of the target. An abstraction of this problem is represented in Fig. 1.1(b). Each node in the graph is a sensor agent. The edges of the graph represent the possible locations where a target can be. There are multiple targets which can be present on the different edges of the graph. First, consider the single-step reasoning problem where all targets are stationary. To track a target, a joint scan of the edge where the target is present is required. When such coordination happens, a positive joint-reward is given to the scanning agents. If agents scan in an uncoordinated

manner or they scan an edge where a target is not present, a penalty is given to each agent. There is no central controller which knows the location of all the agents and targets. Each agent has a strictly local view of the problem which involves knowledge about only those sensors that are its *immediate* neighbors. The goal for all the agents is to come up with a scan strategy such that the joint reward is maximized. Furthermore, agents can only communicate with their immediate neighbors using message-passing. Such decision making problems can be modeled as a distributed constraint optimization problem (DCOP) [95].

Distributed meeting scheduling presents another multiagent decision making domain [109]. In this problem, an agent represents a person in an organization. Each agents needs to participate in multiple meetings with a required set of attendees and range of start times. The goal for such automated personal agents is to find a joint meeting schedule that satisfies the local constraints of each meeting attendee and the global constraints for each meeting. Such multiagent distributed decision making has also found applications in several other domains such as distributed power supply reconfiguration in smart grids [74], synchronized traffic lights [60] and truck route planning [103].

1.1.2 Sequential Multiagent Decision Making

Now consider a much more realistic sequential decision making version of the sensor network problem described earlier. This version takes into account the case when the sensors can be faulty and targets are mobile. This problem can be formulated as a decentralized partially observable MDP (Dec-POMDP) [13]—the standard model for multiagent sequential decision making. In this problem, targets are *mobile*. All targets have independent, stochastic trajectories, which means that either the target can stay at the same location with some (low) probability or move to an adjacent location. Sensors also have an *internal state*, an indication of the battery level of the sensor. Each scan action depletes the battery. In addition to scanning, sensors have two additional actions *sensor off* and *recharge*. The *sensor off* action allows sensors to conserve energy by remaining idle. When the battery is completely depleted, a sensor must perform the *recharge* action, which has a cost. Each sensor can receive one of the following observations after performing a scan action: *target present*, *target absent* and *sensor idle*. False positives/negatives are allowed for the first

two observations. Sensors are encouraged to coordinate as simultaneous scans by multiple sensors are required to get the reward, otherwise a penalty is given. At runtime, sensors operate in a *decentralized* manner without a central controller. This implies that each sensor must select its next action based only on its local observation history. It seems that there is no communication among sensors during execution time. However, this is not true. The observations sensors receive are correlated by a *joint observation* function, that facilitates information exchange during execution time.

This particular sensor network domain presents several interesting characteristics as a multiagent planning benchmark. There are multiple agents ($\gg 2$) involved in the decision making, which requires for scalability in the planning approach. There is significant structure present in the domain such as local interaction among sensors, which can be potentially exploited for scalability. Furthermore, the requirement for coordination is significant as uncoordinated scanning provides no benefit and incurs negative penalty. Other applications for multiagent planning which can be formulated using the Dec-POMDP model include coordinating the operation of planetary exploration rovers [12], coordinating firefighting robots [100] and broadcast channel protocols [13].

1.2 Summary of Contributions

We now describe the main contributions of this thesis, which address multiagent decision making problems that arise in different contexts. We address decision making problems in which a single joint-decision is required and refer this as *single-step* decision making. We also address *sequential* decision making problems in which agents need to perform several actions in a sequential manner to achieve a common goal. Finally, we address the problem of computing quality bounded solutions for sequential decision making problems in a scalable manner, that has been particularly challenging so far in multiagent planning.

1.2.1 Single-Step Decision Making

Main research question: The main research question we address is how can multiple agents coordinate in an intelligent manner to achieve a common goal by using only *local*

communication protocols. As this is an NP-Hard problem, we also address the question of finding quality bounds for a given problem instance.

Framework: We model the single-step multiagent decision making problem using the framework of distributed constraint optimization (DCOP) [95]. We address a broader version of this problem by showing that the problem of finding the maximum-a-posteriori (MAP) assignment in Markov random fields [153] is equivalent to solving the DCOP under certain conditions. We thus develop approaches which are equally applicable to DCOP and the MAP problem in Markov random field, which is commonly used in machine learning applications.

Technical contributions: We address the MAP problem by using the framework of variational inference, which analyzes the inference problem through the lens of mathematical optimization. There are two main variational formulations for the MAP—Quadratic programming (QP) formulation and linear programming (LP) formulation. The main advantage of the LP formulation is that it is convex and computationally tractable, thus there are no local optima. However, this formulation is *inexact*, implying the optimum of the LP does not solves the MAP problem optimally. The QP formulation, on the other hand, is *exact*. However, it is non-convex. Thus, globally optimizing the QP is intractable. We present a new variational framework which combines the benefits of both the QP and the LP formulation, while minimizing the impact of their weaknesses. This new hybrid formulation presents a series of relaxations of the MAP problem, which on one extreme is identical to the QP formulation and on the other extreme, it is identical to the LP formulation. It can also represent an arbitrary combination of the QP and LP formulations, thus providing high flexibility.

The main benefit of this formulation lies in settings where the LP formulation is loose for a given instance. That is, the global optimum of the LP does not provide a good approximation to the MAP problem. We show analytically as well as empirically that the new hybrid formulation provides a tighter approximation to the MAP problem than the LP formulation, which translates into a better solution quality. By judiciously choosing how to introduce the QP constraints in the hybrid formulation, we can control the non-convexity of

the hybrid formulation, which otherwise can lead to a poor local optima. Thus, optimization over the hybrid formulation can avoid getting stuck in such poor local optima, which is the main obstacle for the QP formulation.

We also develop a message-passing algorithm called *Hybrid Belief Propagation* (HBP) that solves the hybrid variational formulation. This message-passing approach is developed by exploiting the connection between the MAP problem and the difference-of-convex function programming. This approach is also guaranteed to converge unlike other approaches such as max-sum [107, 41]. Such message-passing approach is also ideal for large multiagent systems as it only requires exchanging local, fixed-size messages among neighboring agents. Empirically, we show that our approach provides much better solution quality than the state-of-the-art approaches in both multiagent systems community and the machine learning community on a number of large synthetic and real-world benchmarks.

1.2.2 Sequential Decision Making

Main research question: The main research question we address is how to develop scalable planning algorithms that allow multiple agents to operate collaboratively in a decentralized manner and under the partial observability of the environment. In this setting, multiple rounds of decision making are required. In each round, agents take an action based on their individual history of observations received from the environment so far. They perceive a new observation after taking the action. We address the problem of increasing the scalability of planning algorithms w.r.t. both the number of time steps in the plan (also called planning horizon) and the number of agents.

Framework: We use the framework of decentralized POMDPs (Dec-POMDPs) to model multiagent sequential decision making problems [13]. We also use restricted sub-classes of the Dec-POMDP model that can represent problems with structured interactions, such as the Network-Distributed POMDP (ND-POMDP) [99]. We also develop a new sub-class of the Dec-POMDP model called *value-factorization* based multiagent planning [76].

Technical contributions: Our technical contributions are along the two main sub-classes of sequential decision making—finite-horizon planning and infinite-horizon planning.

1.2.2.1 Finite-Horizon Planning

In *finite-horizon* planning problems, agents operate over a finite number of time steps without any discounting of the reward they receive. Computing optimal policies under this framework even for *two* agents is NEXP-Hard [13]. Therefore, approximation algorithms are commonly used. We focus on the bottleneck dynamic programming step of several approximate algorithms, also known as the *backup* problem. We investigate the computational characteristics of the backup problem for 2-agents and show that it is NP-Hard. Despite this negative result, we present an efficient and scalable optimal algorithm for the decentralized backup problem. We show how this problem can be mapped to a weighted constraint satisfaction problem (WCSP). We then use state-of-the-art WCSP solvers to solve the backup problem. Our results show that bringing the perspective of constraint optimization helps solve the backup problem more than an order-of-magnitude faster than state-of-the-art solver PBIP [36]. We also investigate the backup problem in other restricted sub-classes of the Dec-POMDP model, such as ND-POMDPs, that can model larger multiagent systems. We again show how to solve this problem efficiently using WCSP solvers. This approach provides magnitudes of speedup in the policy computation and generates better quality solution for all test instances than earlier approaches [146, 88, 70].

1.2.2.2 Infinite-Horizon Planning

In the *infinite-horizon* planning problems, agents operate continuously with a discounting of the reward. We address the problem of optimizing the joint-policy of a team of agents represented as finite-state controllers (FSC) under the Dec-POMDP framework for infinite-horizon planning problems [15, 6]. Relatively few algorithms existed before our work for solving this planning class due to the intrinsic difficulty of infinite-horizon planning. We present a promising new class of algorithms for the infinite-horizon case for 2-agent Dec-POMDPs, which recasts the optimization problem as inference in a mixture of dynamic Bayesian networks (DBNs), which represent directed graphical models. An attractive feature of this approach is the applicability of existing inference techniques in DBNs for solving Dec-POMDPs and supporting richer representations such as factored or continuous states and actions. We perform the inference in such DBNs by using the well known Expectation

Maximization (EM) algorithm from the machine learning literature to optimize the joint policy. Experiments on benchmark domains show that EM compares favorably against the state-of-the-art solvers.

We also try to answer the following question: *Is there a general characterization of the interaction among agents that when present in a multiagent planning model leads to a relatively scalable approximate algorithm?* We identify such conditions based on a joint-value factorization assumption, that when present in a planning model, enhances the scalability w.r.t. the number of agents. We also develop a scalable, message-passing algorithm that can solve such planning problems using the EM framework. The EM framework is particularly suited for developing algorithms for such restricted models as the EM approach works directly on the DBN representation of the planning problem. Such graphical representation allows for exploiting the independencies present in the planning model than the existing nonlinear programming based approach.

1.2.3 Bounded Optimality for Sequential Decision Making

Main research questions: The main question we address in this part is how to develop approaches that can provide boundedly optimal solutions for multiagent sequential decision making problems in a scalable manner. Since planning in the Dec-POMDP framework is NEXP-Hard [13], an optimal approach would naturally be highly inefficient. However, we can alternatively address the question that what is the optimal joint-policy when each agent's policy is restricted to a particular sub-class of all the allowed policies. This sub-class corresponds to the case when the agents' policies are represented as finite-state controllers [6]. We address the question of how to find optimal finite-state controllers of a given size for all the agents. This work is the first to propose a scalable technique for finding optimal controller based policies for multiagent planning.

Framework: We use the the Dec-POMDP model as the multiagent planning framework. We also use some restricted sub-classes of Dec-POMDPs, such as the ND-POMDP model [99]. We use the mixed integer linear programming (MILP) framework [22] to model the problem of finding optimal joint controllers. We also use the Lagrangian relaxation

technique [16] in conjunction with the MILP framework to extend this approach to large multiagent systems with structured interactions.

Technical contributions: Our main technical contribution is to show how to formulate the problem of optimizing the joint-policy of agents represented as fixed-size finite state controllers in the Dec-POMDP model as a compact mixed integer linear program (MILP). Once we have a MILP representation of the planning problem, we can use off-the-shelf and highly efficient MILP solvers such as CPLEX, which can solve the MILP to optimality or provide non-trivial upper bounds. Previously, several attempts have been made to formulate both the single-agent and multiagent planning problems under uncertainty using mathematical programming [114, 24, 3, 4, 15, 6]. However, most of these approaches either result in a non-convex program, which suffers from the problem of local optima or they approximate the policy optimization using a convex program, which does not guarantee accuracy. These difficulties arise due to the highly non-linear nature of the planning problem. We resolve these problems by interpreting the *multiagent* planning problem under *partial observability* as a *single-agent* planning problem under *full observability*. We then incorporate a small number of constraints into the MILP which guarantee that the resulting policy will work for the multiagent setting under partial observability.

Such connections to the MILP formulation also lays the groundwork for applying advanced mathematical programming techniques to planning. The mixed integer programming is one of the heavily researched area in the mathematical optimization community unlike non-convex programming. There are several techniques that can provide an approximate solution to large mixed integer programs, while also providing a worst case quality guarantee. We show how to apply one such technique called Lagrangian relaxation [16] when there are large number of agents ($\gg 2$) involved in the planning problem. These techniques can be easily implemented in a message-passing manner, implying their suitability for multiagent systems.

Area	Contributions	Techniques
Single-step decision making	Efficient message-passing algorithms for DCOP	Mathematical optimization—Linear prog., quadratic prog. and difference-of-convex function prog.
Sequential decision making–Finite-horizon	Scalable dynamic programming based algorithms	Constraint optimization, branch-and-bound search
Sequential decision making–Infinite-horizon	Inference based approximate planning algorithms	Machine learning—Graphical models, probabilistic inference, likelihood maximization
Sequential decision making–Bounded optimality	Techniques for quality bounded solutions for finite state controller based policy optimization	Mixed integer linear prog., Lagrangian relaxation, dual optimization

Table 1.1. Summary of contributions

1.3 Discussion

To conclude, our main contributions lie along two broad contexts in multiagent systems—single-step decision making and sequential decision making. For the single-step case, we address the question that how can multiple agents coordinate in a decentralized manner when the coordination problem is described using the distributed constraint optimization framework. We develop efficient message-passing algorithms using techniques from mathematical optimization.

For the finite-horizon sequential decision making problems, we address the bottleneck decentralized backup step of several approximate dynamic programming algorithms. The resulting techniques based on constraint optimization provide more than an order-of-magnitude speed up over previous best approaches. For the infinite-horizon case, we explore and exploit the connection between multiagent planning and probabilistic inference in graphical models to develop scalable approximate planning algorithms. Finally, we present several techniques that can provide quality bounded solutions for multiagent planning. This is achieved by reformulating the policy optimization problem as a mixed integer program, which can be solved using highly efficient off-the-shelf solvers such as CPLEX. Table 1.1 summarizes the main contributions of this thesis.

By bringing together such diverse perspectives from a number of different sub-areas of AI, ML and operations research, we show how efficient and rigorous algorithms can be

developed for some of the most computationally challenging problems in multiagent systems. Such synergy among multiple sub-disciplines of computer science holds a signicant promise to make multiagent planning scalable in real-world settings.

Single-Step Decision Making

CHAPTER 2

DISTRIBUTED CONSTRAINT OPTIMIZATION FRAMEWORK

In several scenarios, multiple agents need to coordinate with each other so as to optimize a joint objective function. Such problems can be formulated as distributed constraint optimization problems (DCOPs) [95, 109]. Intuitively, each agent in these settings has its own internal optimization problem, which may reflect its private utility on assignments to decision variables modeled as constraints. Additionally, agents have shared constraints with other agents, which may reflect, for example, the need to share common resources [39]. Because internal optimization problems are private, agents only have knowledge about shared decision variables. A consensus among agents optimizing the joint utility must be reached through a distributed message-passing scheme rather than a centralized planning approach. Such problems arise in several application areas of growing importance such as stream processing and cloud computing [39]. In these settings, agents represent users who have to perform computation on several data streams in an overlay network. The network provides computing resources, which have limited capacity and are shared across different users. Users may have budget constraints and different preferences over the computing resources that process their queries, defining their internal optimization problem. The task for different users is to coordinate in a distributed manner so that the system efficiency is maximized for a given amount of shared resources. The DCOP formulation has been used to model several other practical multiagent problems such as coordinating unmanned aerial vehicles [125, 161], coordinating sensor management [83], distributed power supply management [74] among others.

While distributed constraint optimization is a relatively new framework dating back to mid 90s [162, 164], a closely related model in the machine learning community, that of Markov random field (MRF) [18], has been actively researched for more than thirty years. We show that the decision-theoretic problem of finding the maximum-a-posteriori

assignment (MAP) in MRFs is equivalent to that of DCOP with minor modifications [75]. Consequently, we build upon the wealth of research that already exists in the MRF literature and show that it offers an effective framework for solving DCOPs. This has the additional advantage that the resulting algorithms are equally applicable to several practical problems that are modeled using MRFs in such areas as bioinformatics (protein-prediction), computer vision (image denoising) among others. We first describe the MAP estimation problem in MRFs, followed by its connection to the DCOP problem.

2.1 Markov Random Fields and MAP Estimation

A Markov random field is a class of undirected graphical models which define a probability distribution over a collection of random variables [63]. Markov random fields are particularly useful in modeling inference problems for structured practical applications as they marry probability theory with graph theory. Inference in such graphical models can be made highly efficient by exploiting the local interaction structure among random variables.

Definition 1. *A pairwise Markov random field (MRF) is described using an undirected graph $G = (V, E)$ such that*

- *There are n nodes in the graph.*
- *A discrete random variable x_i is associated with each node $i \in V$ of the graph.*
- *A random variable can take values over a finite domain with maximum size k.*
- *Associated with each edge $(i, j) \in E$ is a potential function $\theta_{ij}(x_i, x_j)$.*
- *The complete assignment $\boldsymbol{x}$ to all the variables in the MRF defines the following joint probability:*

$$p(\boldsymbol{x}; \theta) \propto \exp\left(\sum_{ij \in E} \theta_{ij}(x_i, x_j) \right) \tag{2.1}$$

Although we have only described pairwise MRFs, this is without loss of generality as any m-ary MRF can be converted to a pairwise MRF. Figure 2.1 represents a grid shaped

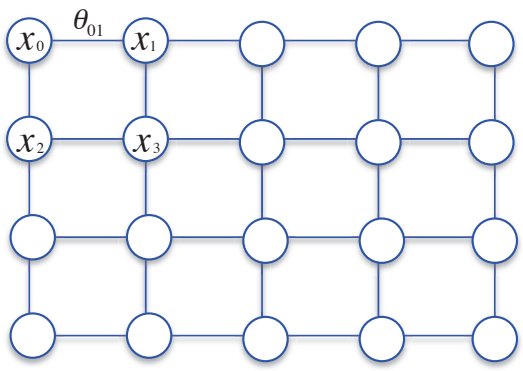


Figure 2.1. An example of a MRF.

MRF. Each node has a random variable associated with it and edges represent the potential functions. Next, we describe the MAP problem.

Definition 2. *The maximum a posteriori (MAP) problem consists of finding the most probable assignment to all the variables of the MRF under* $p(\boldsymbol{x};\theta)$*. Formally, it is given as:*

$$\operatorname*{argmax}_{\boldsymbol{x}} \exp\left(\sum_{ij\in E} \theta_{ij}(x_i, x_j)\right) \tag{2.2}$$

The above problem is also equivalent to the following optimization problem, which is mathematically a more convenient representation:

$$f(\boldsymbol{x};\theta) = \sum_{ij\in E} \theta_{ij}(x_i, x_j) \tag{2.3}$$

We assume w.l.o.g. that each θ_{ij} is nonnegative, otherwise a constant can be added to each θ_{ij} without changing the optimal solution.

2.2 The DCOP Model

Like MRFs, a distributed constraint optimization problem (DCOP) with binary constraints can also be visualized by an undirected graph $G = (V, E)$, commonly called a *constraint graph*. It is formally defined as:

- A set of agents $\mathbf{X} = \{x_i \mid \forall i \in V\}$, where each agent has a finite domain of possible values that it can take on. Each agent x_i is associated with a node $i \in V$ of the graph.

- A set of constraint functions $\boldsymbol{\theta} = \{\theta_{ij}(x_i, x_j) \mid \forall (i,j) \in E\}$. Each constraint $\theta_{ij}(x_i, x_j)$ is associated with an edge $(i,j) \in E$.[1]

Therefore agents and constraint functions in a DCOP correspond to random variables and potential functions in an MRF, respectively. Similar to the MAP estimation problem, the objective in a DCOP is to find the complete assignment $\boldsymbol{x}$ that maximizes the function $f(\boldsymbol{x}; \boldsymbol{\theta})$ in Equation (2.3). The main difference between MAP estimation problems and DCOPs is that the former are centralized problems while the latter are decentralized problems. In MAP estimation problems, a single agent has complete knowledge of all the potential functions and controls the value assignments of all the random variables. On the other hand, in DCOPs, each agent has knowledge of *only the constraint functions that it is involved in* and chooses its own value only. That is, an agent can compute based only on its local information about its immediate neighbors in the constraint graph. Nonetheless, many (centralized) MAP estimation algorithms are implemented using message-passing along the edges of the MRF. These messages are computed based only on the local information available to a variable (or an agent) and the immediate neighbors of a variable in the MRF (or the constraint graph). Thus, message-passing algorithms for MAP estimation in MRFs can be *directly* adopted to solving DCOPs [75]. A significant advantage of such message-passing algorithms is that they exploit the structure of the underlying MRF very efficiently and can scale well to large problems. Thus, developing novel message-passing algorithms based on rigorous principles from mathematical optimization will be the focus of our work.

2.3 Applications

We now describe some common application domains of DCOPs and MAP estimation.

- **Multiagent systems:** As the DCOP framework can naturally model problems where distributed decision making is required, they have found numerous applications in multiagent systems. They have been applied in distributed meeting schedul-

[1] Although the typical notation of a constraint function is F_{ij} or c_{ij} in the DCOP literature, we use the notation in the machine learning literature to better illustrate the mapping between MAP estimation problems and DCOPs.

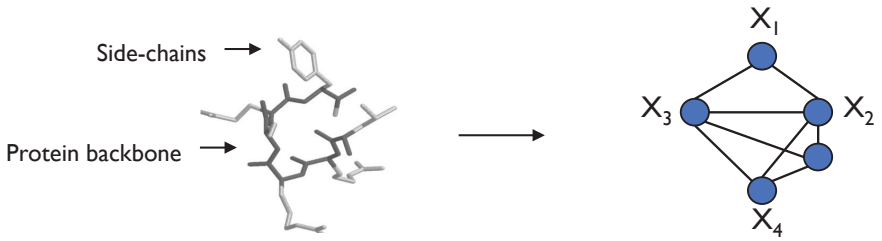


Figure 2.2. A protein backbone structure and its corresponding MRF [130]

ing [86, 108, 160], coordinating unmanned aerial vehicles [125], coordinated sensor networks [82, 83, 168, 95, 58, 171], distributed power supply reconfiguration in smart grids [74], synchronized operation of traffic lights [60] and truck route planning [103].

- **Bioinformatics:** Predicting the protein structure from its amino acid sequence is crucial for drug design and biotechnology. The side-chain prediction problem involves finding the three-dimensional configuration of rotamers given the backbone structure of protein [134, 62]. This problem can be mapped to finding the MAP in an appropriately constructed MRF [134]. Figure 2.2 shows a schematic representation of this problem.

- **Computer vision:** Recovering the original digital image from a given image that has been contaminated with Gaussian noise is an important challenge in computer vision [19]. This problem can also be handled by using MAP estimation in MRFs. Computer stereo vision, which entails extracting 3D information from a set of images, can also be addressed using this framework [68].

- **Information theory:** Given observations over a noisy channel, the decoding problem is to determine the most likely codeword. In the context of low-density-parity-check (LDPC) codes, which have proven to be quite useful in noisy communication channels, the decoding task can again be translated to that of MAP estimation over a codeword factor graph [43].

2.4 Algorithmic Approach—Message-Passing

As most algorithms for DCOPs require iterative message-passing among the agents, we now describe some of the desirable characteristic of such message-passing approaches.

- **Convergence:** An algorithm is said to have this property if it is guaranteed to converge to a fixed point after a *finite* number of iterations. For message-passing algorithms, the fixed point is the point where the content of every message no longer changes. This property can be used as a basis to guarantee termination.

- **Anytime Behavior:** An algorithm is said to have this property if it finds solutions whose qualities are monotonically non-decreasing.

- **Error Bounded:** An algorithm is said to have this property if it has an error bound on the solution quality. This bound can provide a worse case guarantee about how far the current solution is from the optimal.

- **Adaptive:** An algorithm is said to have this property if it is able to solve dynamic DCOPs, that is, DCOPs that change over time. In such dynamic DCOPs, a message-passing approach must be able to incorporate the situations in which new agents may join the system or some of the agents become inactive over time. A desirable property is that in spite of such dynamism, the message-passing approach must be able to warm start the computation from previously computed messages, rather than throwing all the work performed so far.

- **Privacy Preserving:** This a relevant property to DCOP algorithms which is not applicable in the MAP estimation case. In particular, this involves answering questions such as what private information is revealed by such message-passing algorithms and can such message-passing approaches be anonymized.

In section 3.5, we present message-passing algorithms that have most of the above properties.

2.5 Discussion

In this chapter, we have formally introduced the framework of distributed constraint optimization as a model for multiagent single-step decision making. We have established a

close relationship of this problem with that of MAP estimation in Markov random fields. Our work is also the first that formally establishes such connections between DCOPs and MAP estimation [75]. As a result, we can easily leverage approaches from machine learning literature under the umbrella of message-passing algorithms for multiagent systems.

CHAPTER 3

MESSAGE-PASSING ALGORITHMS FOR DCOPS

In this chapter, we provide an overview of message-passing approaches for multiagent systems and related approaches in the Markov random field (MRF) literature. We start with a review of complete and approximate algorithms for DCOPs, followed by variational inference based techniques commonly used in the machine learning literature. We then present our contributions that leverages techniques from the machine learning and optimization literature to develop novel algorithms for DCOPs.

A key contribution of our work is the unification of two different existing variational frameworks based on quadratic programming (QP) and linear programming (LP) for MAP and DCOPs [77]. The main benefit of the unified framework lies in its ability to combine desirable properties of existing formulations such as the accuracy of the QP formulation and the convexity of the LP formulation, while minimizing the impact of their weaknesses, namely the non-convexity of the QP formulation and approximate nature of the LP formulation. We show that the resulting unified framework is also analytically tighter than existing formulations. We then develop a scalable algorithm based on difference-of-convex functions (DC) programming that efficiently exploits the underlying graph structure using a message-passing scheme. This approach is always convergent, can provide tight quality bounds unlike other approximate approaches such as Max-Sum and empirically, provides near-optimal solutions for a number of real-world and synthetic benchmarks from the multiagent systems, machine learning and operations research literature. Extensive empirical results show that our approach significantly outperforms state-of-the-art approaches both in the multiagent systems and machine learning literature.

3.1 Related Work

3.1.1 DCOP Algorithms

DCOP algorithms have been typically categorized as *incomplete* algorithms and *complete* algorithms. Incomplete algorithms generally refer to an approximation scheme, whereas complete algorithms solve the DCOP problem optimally. It should be noted that the DCOP problem is NP-Hard [34]. Therefore, any complete algorithm requires exponential computational time/space in the worst case in the number of agents, unless P=NP.

Some of the incomplete DCOP algorithms include the distributed breakout algorithm (DBA) [163], distributed stochastic algorithm (DSA) [45] and maximum gain message [85]. These algorithms have the common property that they consider changing the assignments of a small group of agents while keeping the assignments of the rest of the agents fixed. Hence, these are hill climbing algorithms that converge quickly, but are prone to getting stuck in poor local optima. There is another class of message-passing algorithms based on the classical max-product algorithm [107], called max-sum [42]. Several extensions and application domains of this method have been presented [137, 138, 139]. Another class of algorithms include k-optimal algorithms [106, 21, 52, 149] and t-optimal algorithms [61]. Other frameworks for incomplete DCOP algorithms include anytime local search [170] and divide-and-coordinate [148].

There has also been progress along complete DCOP algorithms. Some complete DCOP algorithms allow partial centralization in which local groups of agents can transfer their information to a mediator agent [87, 53]. There are also completely decentralized asynchronous search algorithms such as ADOPT [95], BnB-ADOPT [161], no commitment branch-and-bound search (NCBB) [25, 26] and SBB [55]. These algorithms have small memory requirements, but the number of messages exchanged can be exponential in the number of agents to reach optimality. Another direction for complete DCOP algorithms is based on dynamic programming. The distributed pseudo-tree optimization (DPOP) algorithm and its several variants have been influential in this direction [108, 109, 110, 111]. The dynamic programming approaches have the advantage that the number of messages exchanged remain linear in the number of agents. However, messages can have an exponential size related to the tree width of the constraint network.

3.1.2 MAP Estimation Algorithms

As the problem of MAP estimation in MRFs has proven to be useful in several different application areas such as computer vision and information theory, there exists tremendous literature on how to find the MAP in specialized cases that arise in these applications, such as binary graphs or graphs with submodular potential functions. We do not attempt to provide a comprehensive overview. Instead, we limit ourselves to general graphs and state-of-the-art algorithms that have message-passing structure, rather than having a combinatorial nature such as being based on graph cuts [66] or network flow [121]. This is because message-passing algorithms naturally satisfy the privacy and distributed computation requirements of multiagent decision making models such as DCOPs, and are therefore directly applicable to DCOPs.

One of the earliest message-passing algorithm for finding the the MAP assignment in MRFs is max-product belief propagation [107]. While this algorithm has been used successfully in a variety of information-theoretic applications [1, 152], it comes without any guarantee of either correctness or convergence on general graphs. As highlighted before, the message-passing nature of this algorithm has led to its successful adoption for DCOPs too [41]. As this algorithm has poor theoretical guarantees for general graphs, most of the contemporary work for finding the MAP has been focused on the linear programming (LP) relaxation of the MAP problem and its connection with the embedded spanning trees of the graph [151]. Another direction of work has focused on the quadratic programming (QP) relaxation of the MAP problem. We first provide an overview of LP-based approaches followed by QP-based techniques.

An optimal solution of the LP relaxation provides an upper bound over the MAP as the LP solution can be fractional, whereas the MAP assignment is always integral. The LP formulation is also beneficial in proving optimality, which is guaranteed if the LP solution is integral. The LP formulation is at the core of most of the state-of-the-art MAP algorithms [151, 65, 154, 134, 131, 133, 120, 90, 68]. We now describe some of these approaches briefly.

One of the earliest studies that introduced the LP relaxation method and led to a flurry of developments in the field is due to Wainwright *et al.* [151]. They formally described

the LP relaxation and its properties, and established its connection with the underlying spanning trees of the graph. They also developed the tree-reweighted message-passing algorithm (TRW) which solves the *dual* of the LP relaxation.

For tree-structred graphs, the TRW algorithm reduces to the max-product algorithm, thus providing a formal grounding for max-product on trees. This algorithm has a particularly attractive property that at each iteration, it provides an *upper bound* on the MAP solution quality. This algorithm is designed to decrease this upper bound and upon convergence, it will provide the optimal LP solution. It can also provide a guarantee of optimality based on certain optimum specifications. However, it is not guaranteed to decrease the dual objective for every iteration and some damping of messages is required to guarantee convergence. The non-monotonicity of TRW was addressed by Kolmogorov [65], who presented the sequential TRW (TRW-S) algorithm that monotonically decreased the upper bound with each iteration. The TRW-S algorithm modifies the original TRW algorithm in that messages are updated in a specific sequential manner rather than in parallel. The TRW-S is guaranteed to converge, however its converged solution may not provide the optimal bound, a disadvantage compared to TRW. Werner [154] introduced another message-passing algorithm, Max-Sum diffusion (MSD), that is also guaranteed to decrease the upper bound. However, it is also not guaranteed to converge to the optimal bound based on the LP relaxation. Another approach which works on the dual of the LP relaxation is the max-product LP (MPLP) algorithm [49]. This approach also monotonically decreases the upper bound on MAP like TRW-S, but may get stuck in local optima.

The development of such theoretical advances led to the application of these approaches to many practical problems such as protein side-chain prediction and protein design in bioinformatics and in computer vision [159]. An empirical study of TRW-based algorithms showed their effectiveness against black-box solvers such as CPLEX, which were not able to solve LPs for some large problems [159], whereas message-passing algorithms still provided good solutions as they effectively exploited the underlying graphical representation. Combining such message-passing approaches with domain-dependent heuristics helped optimally solve some large MAP instances in computer vision [92].

Recently there has been an increasing interest in developing provably optimal algorithms for solving the LP relaxation. Most of these algorithms are based on the framework of *dual decomposition* [17]. In this framework, an optimization problem is decomposed into several *small* and easy-to-solve sub-problems. These sub-problems are then solved independently, but a simple composition of their solutions may be inconsistent. To make these sub-problems consistent, a master program coordinates their parameters iteratively. This method has been originally introduced in [67] for MRFs which also guarantees that the dual of the relaxed LP is solved in the limit, but is quite slow in practice [120]. Others have proposed to solve the dual LP faster using smoothing of discrete convex functions [59] or other specialized approaches from the convex optimization literature [90].

An interesting point to notice is that even though the LP relaxation may be solved optimally, it still does not provides the MAP solution as the LP relaxation is not exact. To remedy this, a number of techniques have been proposed which *tighten* this LP [134, 131, 155] using cutting planes approach or cycle inequalities based on graph cuts. In these techniques, additional constraints are added to the LP which restrict the feasible parameter space, but do not preclude the optimal solution. This process is performed iteratively until a satisfactory solution is found. A main concern with such approaches is that to make the relaxation tighter either the computational complexity is increased exponentially [134, 155] w.r.t. to the size of variable-cluster that are added to the LP relaxation or it requires the LP to be solved using a black-box solver like CPLEX as message-passing algorithms are not able to handle the new LP [131].

In our work, we will describe the outline of some of the techniques which can solve both the relaxed LP *optimally* and also tighten it *without* increasing the size of the optimization problem in stark contrast to previous approaches. Next we review an alternate variational formulation of the MAP problem that is based on quadratic programming and also has connection to the mean-field based variational inference.

A different formulation of MAP is based on *quadratic programming* (QP) [119, 78]. The QP formulation is an attractive alternative to the LP formulation because it provides a more compact representation of MAP: In a MRF with n variables, k values per variable, and $|E|$ edges, the QP has $O(nk)$ variables whereas the LP has $O(|E|k^2)$ variables. The large size

of the LP makes off-the-shelf LP solvers impractical for several real-world problems [159]. Another significant advantage of the QP formulation is that it is *exact*. However, the QP formulation is non-convex, making global optimization hard.

As the QP formulation is non-convex, several convex relaxations of the above QP are also proposed [119, 30]. Ravikumar *et al.* [119] convexify the above QP by making the potential funtion matrix (consisting of the potentials θ_{ij} in a matrix form) positive semidefinite. This is done by adding certain unary terms for each node of the MRF to the objective and subtracting a linear term for each edge. The convexified QP has the advantage that it can be *optimally* solved by black-box QP solvers such as CPLEX. However this modified QP is no longer exact and its optimal solution may not correspond to the MAP. Nonetheless, it showed good empirical performance [119]. Another convexification of the QP is proposed in [30] based on spectral techniques. However, the experiments do not provide enough evidence that using spectral techniques is scalable enough to solve large graphical models. The reason is that the algorithm in [30] does not involve message-passing and thus may have difficulty in solving large QPs.

3.2 Variational Methods—An Optimization-based View

As our work utilizes an optimization based view of the MAP and DCOP problems, we first introduce some existing variational techniques for this problem. In variational methods, the main idea is to express the inference problem of interest (in our case, DCOP or MAP inference) as the solution of a mathematical optimization problem. This does not make the problem easier by itself. However, analyzing the inference problem through the lens of mathematical optimization exposes the precise underlying structure of the problem. The optimization problem can then be relaxed by approximating the complicating constraints or the function to be optimized. This in turn provides a principled relaxation of the original problem, which is often computationally tractable.

We start by defining *marginal probabilities* associated with an MRF.

Definition 3. *Node marginals associated with each node i of the MRF are defined as:*

$$\mu_i(x_i) = P(X_i = x_i; \boldsymbol{\theta}) \; \forall x_i$$

Edge marginals associated with each edge (i, j) of the MRF are defined as:

$$\mu_{ij}(x_i, x_j) = P(X_i = x_i, X_j = x_j; \boldsymbol{\theta}) \; \forall x_i, x_j$$

We represent the set of all node and edge marginals as $\boldsymbol{\mu}$. Based on this, we define the following:

Definition 4. *The set of marginal vector $\boldsymbol{\mu}$ that arise from a valid joint distribution over all the variables of the MRF G is called the marginal polytope:*

$$\mathcal{M}(G) = \left\{ \boldsymbol{\mu} \mid \exists p(\boldsymbol{x}) : p(x_i, x_j) = \mu_{ij}(x_i, x_j),\ p(x_i) = \mu_i(x_i) \right\} \tag{3.1}$$

The MAP problem now becomes equivalent to the following LP:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} = \max_{\boldsymbol{\mu} \in \mathcal{M}(G)} \sum_{ij \in E} \sum_{x_i x_j} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j) \tag{3.2}$$

However, the above LP is intractable as representing the marginal polytope $\mathcal{M}(G)$ is prohibitively hard. Therefore, despite this variational view, the MAP problem is still intractable. However, this variational view highlights the main obstacle in solving the LP. The constraint in the marginal polytope that the edge and node marginals must arise from a valid joint distribution is generally prohibitive to represent. Therefore it is relaxed, yielding an *outer bound* on the marginal polytope, also called the *local polytope* $\mathcal{L}(G) \supseteq \mathcal{M}(G)$ as it enforces only the local consistency on the marginals.

Definition 5. *The local polytope $\mathcal{L}(G)$ consists of all marginal vectors $\boldsymbol{\mu}$ which satisfy the following constraints:*

$$\begin{aligned} \sum_{x_i} \mu_i(x_i) &= 1 \; \forall i \in V; \\ \sum_{\hat{x}_j} \mu_{ij}(x_i, \hat{x}_j) &= \mu_i(x_i) \; \forall i \in V, \forall x_i, \forall j \in Nb(i) \end{aligned} \tag{3.3}$$

where $Nb(i)$ refers to the set of all the immediate neighbors of a node i in the MRF. There is another variational representation of the MAP problem which assumes the mean-field structure on the joint distribution [119, 153]. That is, the joint probability factorizes according to the nodes of the MRF: $p(\boldsymbol{x}) = \prod_{i=1}^{n} p_i(x_i)$. The resulting set of marginal vectors give rise to the inner bound $\mathcal{I}(G) \subseteq \mathcal{M}(G)$ on the marginal polytope:

Definition 6. *The inner bound $\mathcal{I}(G)$ consists of all marginal vectors $\boldsymbol{\mu}$ which satisfy the following constraints:*

$$\begin{aligned} &\mu_{ij}(x_i, x_j) = \mu_i(x_i)\mu_j(x_j) \; \forall (i,j) \in E \\ &\sum_{x_i} \mu_i(x_i) = 1 \; \forall i \in V \end{aligned} \tag{3.4}$$

The first constraint is non-linear in $\boldsymbol{\mu}$ and, in general, $\mathcal{I}(G)$ is non-convex. The relationship among these different formulations and the MAP is given by:

Proposition 1. *The MAP solution quality $f^\star$ satisfies*

$$f^\star = \max_{\boldsymbol{\mu} \in \mathcal{M}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} = \max_{\boldsymbol{\mu} \in \mathcal{I}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} \leq \max_{\boldsymbol{\mu} \in \mathcal{L}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} \tag{3.5}$$

This known result [151, 119, 153] shows that optimizing over $\mathcal{I}(G)$ is exact. Furthermore, $\mathcal{I}(G)$ can be compactly represented using only $O(n)$ normalization constraints for node marginals and $O(nk)$ variables for each $\mu_i(x_i)$.

The optimization problem $\max_{\boldsymbol{\mu} \in \mathcal{L}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta}$ is also know as the *linear programming* (LP) relaxation of the MAP problem. The above proposition also shows that the optimal value of the LP relaxation provides an upper bound on the MAP solution quality. Formally, it is given as:

$$\max_{\boldsymbol{\mu} \in \mathcal{L}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} = \max_{\boldsymbol{\mu} \in \mathcal{L}(G)} \sum_{ij \in E} \sum_{x_i x_j} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j) \tag{3.6}$$

The constraints which characterize the local polytope $\mathcal{L}(G)$ are given in Def. 5. The optimization over the inner bound is also called the *quadratic programming* (QP) formulation of the MAP and can be represented as follows:

$$\max_{\boldsymbol{\mu}\in\mathcal{I}(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta} = \max_{\boldsymbol{\mu}\in\mathcal{I}(G)} \sum_{ij\in E}\sum_{x_i x_j} \theta_{ij}(x_i,x_j)\mu_i(x_i)\mu_j(x_j) \tag{3.7}$$

The constraints which characterize the inner bound $\mathcal{I}(G)$ are given in Def. 6. Even though the above variational formulation of the MAP problem is exact, it is non-convex. Therefore, local optima are the main obstacle while solving the QP formulation. On the other hand, the LP relaxation is convex and can be solved optimally. However, it is *non-exact*. It does not provide an accurate representation of the MAP problem. In our experiments, we found that in several hard MAP problems, the LP relaxation is quite loose. Therefore, to remedy these drawbacks of these variational formulations, we next present our approach.

3.3 Our Approach—A Hybrid Variational Bound

We now present our approach, which combines the inner bound with the outer bound, resulting in the hybrid set of parameters $\mathcal{H}(G;Q)$. It is parameterized by a set of MRF edges Q. Intuitively, the non-convexity in the set $\mathcal{I}(G)$ arises from the mean-field constraint that $\mu_{ij}(x_i,x_j)=\mu_i(x_i)\mu_j(x_j)$. The set Q contains all the edges $e\in E$ for which the mean-field constraint is enforced. Let $L=E\setminus Q$ denote the rest of the edges, which intuitively correspond to the outer bound constraints. We denote the edges in the set Q as QP edges (short for quadratic) and the edges in L as LP edges. The hybrid bound for the marginal polytope is defined as:

Definition 7. *The hybrid bound $\mathcal{H}(G;Q)$ consists of all marginal vectors $\boldsymbol{\mu}$ which satisfy the following constraints:*

$$\begin{aligned}
&\mu_{ij}(x_i,x_j) = \mu_i(x_i)\mu_j(x_j)\ \forall (i,j)\in Q\ ;\\
&\sum_{x_i}\mu_i(x_i) = 1\ \forall i\in V;\\
&\sum_{\hat{x}_j}\mu_{ij}(x_i,\hat{x}_j) = \mu_i(x_i)\ \forall i,\forall x_i,\forall j\in Nb_l(i)
\end{aligned} \tag{3.8}$$

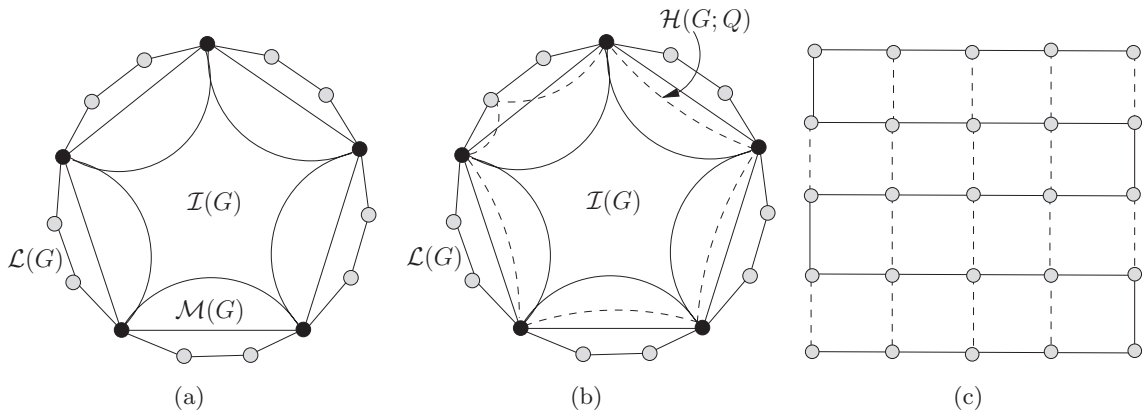


Figure 3.1. a) Relationship among marginal polytope, inner and outer bound; b) Hybrid bound; c) A grid graph with dotted edges denoting QP edges set Q.

where $Nb_l(i)$ denotes the *LP neighbors* of a node i: $Nb_l(i) = \{j : j \in Nb(i) \wedge (i,j) \in L\}$.

Proposition 2. *It holds that:* $\max_{\boldsymbol{\mu}\in\mathcal{I}(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta} \leq \max_{\boldsymbol{\mu}\in\mathcal{H}(G;Q)} \boldsymbol{\mu}\cdot\boldsymbol{\theta} \leq \max_{\boldsymbol{\mu}\in\mathcal{L}(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}$

Proof. First notice that the mean-field constraint $\mu_{ij}(x_i,x_j) = \mu_i(x_i)\mu_j(x_j)$ for an edge (i,j) automatically enforces the edge consistency constraint $\sum_{\hat{x}_i} \mu_{ij}(\hat{x}_i,x_j) = \mu_j(x_j)$ in both directions. Therefore, intuitively, the set $\mathcal{I}(G)$ is much smaller than the local polytope, which only enforces the edge consistency constraints. The first inequality holds because the mean-field constraint is not enforced for the LP edges in $\mathcal{H}(G;Q)$. Therefore the set $\mathcal{I}(G) \subseteq \mathcal{H}(G;Q)$. The second inequality holds because the mean-field constraint is enforced in addition to edge consistency for the QP edges in Q. Therefore $\mathcal{H}(G;Q) \subseteq \mathcal{L}(G)$. □

The above proposition highlights that the set $\mathcal{H}(G;Q)$ provides a tighter approximation to the MAP than the local polytope. In fact, by controlling the set of QP edges Q, the set $\mathcal{H}(G;Q)$ provides a hierarchy of relaxations ranging from $\mathcal{I}(G)$ (when $Q = E$) to $\mathcal{L}(G)$ (when $|Q| = 0$). As the size of the set Q increases, the relaxation becomes tighter, but the non-convexity increases. However, since we have control over the degree of non-convexity, a judicious choice of the set Q will not only make the relaxation tighter, but also provide a good solution quality with respect to the optimization over the local polytope. Experimentally, we found this to be the case for almost all the instances.

Fig. 3.1(a) shows the high-level relationship among the marginal polytope and the inner and outer bounds and is based on [153]. Each extreme point of the marginal polytope

$\mathcal{M}(G)$ represents an integral assignment to all the variable of the MRF. They are shown as solid black circles in Fig. 3.1(a). Consequently, the marginal polytope can also be described as the convex hull of all the integral assignments to the MRF. The local polytope $\mathcal{L}(G)$ includes fractional assignments to the MRF variables such as $\mu_i(x_i) = 0.6$, in addition to all the integral assignments. If the LP solution occurs at one of these fractional vertices, it provides an upper bound on the MAP. Such fractional vertices are shown as gray circles in Fig. 3.1(a). The inner bound touches the marginal polytope at all its extreme points. However, it is non-convex and is shown as a curved bound in Fig. 3.1(a).

Fig. 3.1(b) shows the hybrid set $\mathcal{H}(G; Q)$. This set includes all the integral assignment to the MRF. In addition, it can include some fractional assignments too. Since, it incorporates some constraints of the inner bound, it is non-convex. However, the non-convexity in hybrid bound can be controlled precisely by judiciously choosing the set Q. Therefore, it can be made qualitatively less non-convex than the inner bound, thereby decreasing the chances of getting trapped in a poor local optimum. It is also more accurate than the outer bound $\mathcal{L}(G)$ as it does not include all the fractional vertices of the outer bound.

Fig. 3.1(c) shows a strategy to select the QP edges set Q. As the outer bound is tight for tree-structured graphs, we randomly choose a spanning tree of the graph with all its edges constituting the set L; the rest are the QP edges. We can also construct multiple such spanning trees independently with L being the union of their edges. Empirically, this method worked well in our experiments.

Proposition 3. *As the size of the QP edge set Q increases by 1, the number of parameters decreases by $O(k^2)$ and the number of constraints decreases by $O(2k)$.*

Proof. For the QP edges, it holds that $\mu_{ij}(x_i, x_j) = \mu_i(x_i)\mu_j(x_j)$. Instead of having a linear objective function $(\boldsymbol{\mu} \cdot \boldsymbol{\theta})$, we can substitute $\mu_{ij}(x_i, x_j)$ by $\mu_i(x_i)\mu_j(x_j)$ in the objective. Thus we no longer need to store the parameter $\mu_{ij}(x_i, x_j)$ nor the mean-field constraint explicitly for QP edges. Therefore, the total number of parameters is $O(k^2|L| + nk)$ and the total number of constraints is $O(2|L|k + n)$ where $L = E \backslash Q$. As the size of Q increases by 1, the size of the set L decreases by 1. This proves the proposition. □

The above result shows that unlike the previous cluster-based approaches [134, 8], where the number of parameters increases exponentially w.r.t. the cluster size to make the relaxation tighter, the number of parameters decreases in our hybrid inner-outer bound based approach as it becomes tighter. This can potentially make the optimization easier at the expense of losing some convexity.

To summarize, we have introduced a hybrid variational bound based on the combination of inner and outer bound on the marginal polytope. The hybrid bound provides a hierarchy of relaxations ranging from $\mathcal{I}(G)$ (when $Q = E$) to $\mathcal{L}(G)$ (when $|Q| = 0$). By controlling the set of QP edges Q, we can control the non-convexity, which is the main obstacle for optimization over the inner bound, and the accuracy, which is the main disadvantage for the outer bound.

3.4 DC Programming for Optimization Over Hybrid Bound

In this section, we describe how to formulate the optimization problem over the hybrid bound as a difference-of-convex functions (DC) program. The DC programming viewpoint is quite powerful as it allows us to solve both the QP formulation and the LP relaxation within the same framework. First, we explain the general DC optimization problem and briefly describe the concave-convex procedure (CCCP), introduced in [166] to solve a DC program. Consider the optimization problem:

$$\min\{g(x) : x \in \Omega\}$$

where $g(x) = u(x) - v(x)$ is an arbitrary function with u, v being real-valued, differentiable *convex* functions and Ω being a constraint set. CCCP was originally proposed for Ω that is described by linear equality constraints, but [136] showed the same idea extends to any constraint set including non-convex constraints. The CCCP method provides an iterative procedure that generates a sequence of points x^l by solving the following convex program:

$$x^{l+1} = \arg\min\{u(x) - x^T \nabla v(x^l) : x \in \Omega\} \tag{3.9}$$

Each iteration of CCCP decreases the objective function $g(x)$ for any Ω [136]. Furthermore, it is guaranteed to converge to a stationary point where the Karush-Kuhn-Tucker (KKT) conditions are satisfied when the constraint set Ω is convex [136]. Note that the objective $g(x)$ may be non-convex, which makes CCCP a general approach for non-linear optimization.

3.4.1 DC Formulation of MAP

We now describe how optimization over different variational formulations of MAP can be formulated as a DC program. As the hybrid bound subsumes both the inner and the outer bound, we describe our results in the context of hybrid bound. The optimization problem over the set $\mathcal{H}(G;Q)$ is described as minimizing the following function subject to constraints Ω of Eq. (3.8):

$$g(\boldsymbol{\mu};\theta,Q) = -\sum_{(i,j)\in Q}\sum_{x_i,x_j}\theta(x_i,x_j)\mu(x_i)\mu(x_j) - \sum_{(i,j)\in L}\sum_{x_i,x_j}\theta(x_i,x_j)\mu(x_i,x_j) \tag{3.10}$$

For the sake of readability, we drop the subscripts, using $\theta(x_i,x_j)$ for $\theta_{ij}(x_i,x_j)$, and $\mu(x_i,x_j)$ for $\mu_{ij}(x_i,x_j)$, as long as it is unambiguous. We also negate the MAP objective to get a minimization problem. The critical question in this reformulation into a DC program is how easy it is to perform the CCCP iteration in Eq. (3.9). Notice that the first term in $g(\boldsymbol{\mu};\theta,Q)$ is quadratic and neither concave nor convex. However, noting that all the marginals must be positive, a simple substitution $\mu(x_i) = e^{y(x_i)}$, where $y(x_i)$ is unconstrained, makes it convex because the function $e^{h(x)}$ is convex when $h(x)$ is affine. The CCCP procedure has also been applied to optimize the Bethe free energy of a MRF [165]. Motivated by the fact that negative entropy of an edge is convex and the constraints in the outer bound are the same as in the Bethe free energy, we further perform the following optimality preserving modifications:

- For each LP edge in L, add and subtract the negative entropy:
  $-H_{ij} = \sum_{x_i,x_j}\mu(x_i,x_j)\log\mu(x_i,x_j)$ to $g(\boldsymbol{\mu};\theta,Q)$.
- For each node i, add and subtract the term: $\sum_{x_i} e^{y(x_i)}$ to $g(\boldsymbol{\mu};\theta,Q)$.

The entropy term ensures that all the edge marginals μ_{ij} are positive and the exponential term $e^{y(x_i)}$ ensures that all the node marginals are positive as well as simplifies the CCCP

iteration. The objective $g(\boldsymbol{\mu}; \theta, Q)$ can now be written as the difference two functions u and v. The function $u(\boldsymbol{\mu}, \boldsymbol{y})$ is defined as:

$$-\sum_{(i,j)\in L}\sum_{x_i,x_j}\theta(x_i,x_j)\mu(x_i,x_j) - \sum_{(i,j)\in L} H_{ij} + \sum_{i,x_i} e^{y(x_i)} \tag{3.11}$$

The function $v(\boldsymbol{\mu}, \boldsymbol{y})$ is defined as:

$$\sum_{(i,j)\in Q}\sum_{x_i,x_j}\theta(x_i,x_j)e^{y(x_i)+y(x_j)} - \sum_{(i,j)\in L} H_{ij} + \sum_{i,x_i} e^{y(x_i)} \tag{3.12}$$

The modified constraint set Ω' is described by the following non-convex constraints:

$$\begin{aligned}
&\sum_{x_i,x_j}\mu(x_i,x_j) = 1 \;\forall (i,j)\in L;\\
&\sum_{\hat{x}_j}\mu_{ij}(x_i,\hat{x}_j) = e^{y(x_i)} \;\forall i \in V, \forall x_i, \forall Nb_l(i)
\end{aligned} \tag{3.13}$$

Notice that the constraints in Eq. (3.13) are different from those in Eq. (3.8) that describe $\mathcal{H}(G;Q)$. This is deliberate to simplify the CCCP iteration without changing the feasible parameter space. First, the factorization constraint $\mu(x_i,x_j) = \mu(x_i)\mu(x_j)$ for all $(i,j)\in Q$ is now explicit in the objective (3.10). Second, we impose a restriction that for each node i, there should be at least a single neighbor j such that edge (i,j) is an LP edge or $(i,j)\in L$. This restriction will not allow all the edges to become QP, i.e., $Q \neq E$. The reason for this restriction is that we no longer enforce the normalization constraint for any node explicitly. But the constraints $\sum_{x_j}\mu(x_i,x_j) = e^{y(x_i)}$ and $\sum_{x_i,x_j}\mu(x_i,x_j) = 1$ directly imply $\sum_{x_i} e^{y(x_i)} = 1$. We note that restricting the set Q in this way does not compromise generality—if we want to make the set $Q = E$, then for each MRF node i, we introduce a dummy node d_i with a single state and make an edge (i,d_i) with every potential being zero. Then we can add this edge to the LP edges set L. This will satisfy the previous requirement.

Also notice that by using the substitution $\mu(x_i) = e^{y(x_i)}$, we have made the constraint set non-convex. Let us denote the objective $g(\boldsymbol{\mu}; \theta, Q)$ (see Eq. (3.10)) with the

exp-transformation as $g(\boldsymbol{\mu}, \boldsymbol{y}; \theta, Q)$. We also refer as Ω to the counterpart of Ω' which uses the variables $\mu(x_i)$ rather than $e^{y(x_i)}$. We now study the properties of the stationary points of $g(\boldsymbol{\mu}; \theta, Q)$ and $g(\boldsymbol{\mu}, \boldsymbol{y}; \theta, Q)$ that will help answer questions about the convergence properties of CCCP.

Proposition 4. *Every stationary point of the problem* $\min g(\boldsymbol{\mu}, \boldsymbol{y}; \theta, Q)$ *subject to* Ω' *is also a stationary point of the problem* $\min g(\boldsymbol{\mu}; \theta, Q)$ *subject to* Ω.

The above proposition has interesting consequences. For the LP relaxation case when $|Q|=0$, the optimization of $g(\boldsymbol{\mu}; \theta, Q)$ subject to Ω is equivalent to solving the LP relaxation of MAP and there are no local optima. However, even for the case when $|Q| = 0$, the second optimization problem of $g(\boldsymbol{\mu}, \boldsymbol{y}; \theta, Q)$ over Ω' is a non-convex optimization problem as the equality constraints are not linear. However, we have shown that every stationary point of the second optimization problem is also a stationary point of the LP, which shows that there are no local optima in the second problem even though it is non-convex (only for the case $|Q|=0$). Thus, achieving the stationary point of the second optimization problem using CCCP will yield the global optima for the LP relaxation.

3.4.2 Solving the CCCP Iteration

We now describe how we can solve each iteration of CCCP, which itself is a convex optimization problem, in the context of MAP DC program. Each iteration of CCCP is given by:

$$\begin{aligned}\min_{\boldsymbol{\mu},\boldsymbol{y}} & -\sum_{(i,j)\in L}\sum_{x_i,x_j}\theta(x_i,x_j)\mu(x_i,x_j) - \sum_{(i,j)\in L} H_{ij} + \sum_{i,x_i} e^{y(x_i)} \\ & - \sum_{(i,j)\in L}\sum_{x_i,x_j}\mu(x_i,x_j)\nabla_{\mu(x_i,x_j)}v - \sum_{i\in V,x_i} y(x_i)\nabla_{y(x_i)}v\end{aligned}$$

subject to constraints Ω' of Eq. (3.13), which are non-convex. Thus it might appear that we may not be able to optimally solve the CCCP iteration. However, this is remedied by again substituting $\mu(x_i) = e^{y(x_i)}$ back to the above problem resulting in:

$$\min_{\boldsymbol{\mu},\boldsymbol{y}} - \sum_{(i,j)\in L} \sum_{x_i,x_j} \theta(x_i,x_j)\mu(x_i,x_j) - \sum_{(i,j)\in L} H_{ij} + \sum_{i,x_i} \mu(x_i)$$

$$- \sum_{(i,j)\in L} \sum_{x_i,x_j} \mu(x_i,x_j) \nabla_{\mu(x_i,x_j)} v - \sum_{i\in V, x_i} \nabla_{y(x_i)} v \ \log \mu(x_i) \tag{3.14}$$

$$subject\ to: \sum_{x_i,x_j} \mu(x_i,x_j) = 1 \ \forall (i,j) \in L;$$

$$\sum_{\hat{x}_j} \mu_{ij}(x_i,\hat{x}_j) = \mu(x_i) \ \forall i \in V, \forall x_i, \forall Nb_l(i) \tag{3.15}$$

Note that this re-substitution $\mu(x_i) = e^{y(x_i)}$ is in the context of CCCP iteration (3.9); it does not undo the substitution made in the context of the objective (3.10)—in (3.14) we still have the gradient $\nabla_{y(x_i)} v$ w.r.t. $y(x_i)$. Furthermore, the above optimization problem is a standard convex optimization problem with convex objective and linear equality constraints. Thus there are no local optima, and strong duality holds as we can show that Slater's conditions hold [22].

Proposition 5. *CCCP converges to a stationary point of the problem* $\min g(\boldsymbol{\mu}, \boldsymbol{y}; \theta, Q)$ *subject to* Ω'.

The above proposition extends the convergence properties of CCCP when applied in the context of non-convex constraints, which is the case for the MAP DC program. Previous results have analyzed the convergence properties of CCCP in the presence of convex constraints or DC constraints [136].

3.4.3 DC Programs and Proximal Minimization

In [120], a number of globally convergent message-passing algorithms are developed to solve the LP relaxation of MAP. Our DC programming based approach can also solve the LP relaxation optimally using message-passing as shown in the next section. In this section, we investigate the relationship between our approach and that of [120]. Interestingly, we can show that the proximal minimization algorithms based on Bregman divergence, on which the work of [120] is based, can be neatly described in the DC programming framework. The CCCP approach to solve such DC programs gives an identical optimization problem to the proximal minimization scheme. Therefore, the DC programming framework of MAP we

have developed is quite general and subsumes existing frameworks to solve the LP relaxation of the MAP.

We start by describing proximal proximal minimization schemes. Proximal minimization schemes solve a convex program $\min_{\mu\in\Omega} g(\mu)$ indirectly via a sequence of problems of the form [120]:

$$\mu^{n+1} = \arg\min_{\mu\in\Omega} \big\{ g(\mu) + \frac{1}{\omega^n} D_f(\mu||\mu^n) \big\} \tag{3.16}$$

where D_f is a generalized distance function and ω^n a positive weight. We focus on Bregman divergence based distance function defined as $D_f(\mu||\nu) = f(\mu) - f(\nu) - \langle \nabla f(\nu), \mu - \nu \rangle$, where f is a strictly convex, differentiable function (also called a Bregman function).

Proposition 6. *Each proximal iteration with Bregman divergence based distance function and a fixed weight ω is equivalent to the CCCP iteration for the DC program $\min_{\mu\in\Omega}\{u(\mu) - v(\mu)\}$ where $u(\mu) = g(\mu) + \frac{1}{\omega} f(\mu)$ and $v(\mu) = \frac{1}{\omega} f(\mu)$.*

Proof. Substituting the definition of Bregman function in Eq. (3.16), we get:

$$\mu^{n+1} = \arg\min_{\mu\in\Omega} \big\{ g(\mu) + \frac{1}{\omega} \big(f(\mu) - f(\mu^n) - \nabla f(\mu^n)\mu + \nabla f(\mu^n)\mu^n \big) \big\} \tag{3.17}$$

$$= \arg\min_{\mu\in\Omega} \big\{ g(\mu) + \frac{1}{\omega} \big(f(\mu) - \nabla f(\mu^n) \cdot \mu \big) \big\} \tag{3.18}$$

Consider the D.C. program $\min_{\mu\in\Omega}\{u(\mu) - v(\mu)\}$ equivalent to the original problem $\min_{\mu\in\Omega} g(\mu)$ with $u(\mu) = g(\mu) + \frac{1}{\omega} f(\mu)$ and $v(\mu) = \frac{1}{\omega} f(\mu)$. The CCCP iteration of Eq. (3.9) is given as: $\arg\min_{\mu\in\Omega}\{g(\mu) + \frac{1}{\omega} f(\mu) - \frac{1}{\omega}\nabla f(\mu^n) \cdot \mu\}$, which is equivalent to the proximal scheme of Eq. (3.18). □

Note that the DC program in the above proposition is equivalent to the original convex program $\min_{\mu\in\Omega} g(\mu)$. We also note that the weights ω^n can be adjusted for faster convergence in proximal schemes, but they can also be set to constant as the Bregman distance function D_f itself converges to zero as the algorithm approaches the optimum [120]. Furthermore, the DC program view can also simulate changing weight scenario as the DC decomposition $u(\mu) = g(\mu) + \frac{1}{\omega} f(\mu)$ and $v(\mu) = \frac{1}{\omega} f(\mu)$ is valid for any weight ω. In [120],

several globally convergent algorithms to solve the LP relaxation areproposed based on different choices of the Bregman function f such as entropic, quadratic, etc. Such algorithms can be interpreted as DC programs. Moreover, the proximal minimization schemes are only applicable to convex functions, while DC programs need not be convex and are therefore more general. Nonetheless, the connection we established is important. For example, CCCP is generally considered to have a first-order convergence rate [124]. However Bregman projections for certain choices of Bregman functions have fast superlinear convergence [120], implying a similar convergence rate for CCCP too in these cases. To recast the proximal iteration of Eq. (3.16) as Bregman projections, the Bregman functions are restricted to be of Legendre type in [120]. This prohibits them from using a tree-reweighted entropy-based Bregman function. However the DC view of the proximal approach does not pose this restriction and any Bregman function can be used.

3.5 Message-Passing for MAP DC Program

The CCCP iteration of Eq. (3.14) is a convex optimization problem. In theory, it can be solve using black-box convex optimization solvers. However, such solvers may not be able to scale well as they ignore the underlying graphical structure of this problem. Therefore, in this section, we develop a customized message-passing algorithm to solve it. We now describe how the CCCP iterations can be implemented using a message-passing paradigm. At a high level, CCCP updates typically require a double loop. The outer loop computes the gradient $\nabla v(\mu^l, y^l)$ and the inner loop solves the optimization problem of Eq. (3.14). Both steps can be implemented using message-passing on the MRF as shown below.

3.5.1 The Gradient of $v(\mu, y)$ and Outer Loop Message-Passing

The gradient w.r.t. $y(x_i)$ is given by:

$$\nabla_{y(x_i)} v(\mu, y) = e^{y(x_i)} + e^{y(x_i)} \sum_{j \in Nb_q(i)} e^{y(x_j)} \theta(x_i, x_j) \tag{3.19}$$

where $Nb(i)$ denotes the neighbors of a node i and $Nb_q(i)$ denotes the QP neighbors of a node i: $Nb_q(i) = \{j : j \in Nb(i) \wedge (i,j) \in Q\}$. The second term in the above equation can

be computed using message-passing on the graph. The other gradient, $\nabla_{\mu(x_i,x_j)} v(\mu, y) = 1 + \log \mu(x_i, x_j)$, does not require any message-passing.

3.5.2 CCCP Inner Loop and Message-Passing

We now describe how to perform the CCCP iteration of Eq. (3.14). Because the dual of this optimization problem is unconstrained and has simpler structure, we will perform block coordinate ascent in the dual. First we introduce the following Lagrange multipliers:

$$\lambda_{ij} : \sum_{x_i, x_j} \mu(x_i, x_j) = 1; \quad \lambda_{ji}(x_i) : \sum_{\hat{x}_j} \mu_{ij}(x_i, \hat{x}_j) = \mu(x_i)$$

The dual function is $q(\boldsymbol{\lambda}) = \inf_{\mu, y} L(\boldsymbol{\mu}, \boldsymbol{y}, \boldsymbol{\lambda})$, where $L(\cdot)$ denotes the Lagrangian. We can solve this optimization problem by solving for the KKT conditions, resulting in:

$$\begin{aligned} \mu(x_i) &= \frac{\nabla_{y(x_i)} v}{1 + \sum_{k \in Nb_l(i)} \lambda_{ki}(x_i)} \\ \mu(x_i, x_j) &= e^{\theta(x_i, x_j) + \nabla_{\mu(x_i, x_j)} v + \lambda_{ij}(x_j) + \lambda_{ji}(x_i) - \lambda_{ij} - 1} \end{aligned} \tag{3.20}$$

Substituting these back into the Lagrangian, we get the dual. The dual can be maximized iteratively by using the block coordinate ascent: hold all the λ's fixed except one and optimize the dual w.r.t. a single λ [16]. This iterative procedure is also guaranteed to converge to the optimum of the convex program of Eq. (3.14). We directly give the analytical update steps of the block coordinate ascent:

Proposition 7. *The Lagrange multipliers λ_{ij}, $\lambda_{ij}(x_j)$ can be derived by using an inner loop in the CCCP indexed by τ such that the multipliers for each edge $(i, j) \in L$ are updated once per inner loop. The update equations are as follows:*

$$\lambda_{ij}^{\tau+1}(x_j) = W\left(\frac{\nabla_{y(x_j)} v \; e^{\sum_{k \in Nb_l(j) \setminus i} \lambda_{kj}^{\tau}(x_j) + 1}}{\sum_{x_i} e^{\theta(x_i, x_j) + \nabla_{\mu(x_i, x_j)} v + \lambda_{ji}^{\tau}(x_i) - \lambda_{ij}^{\tau} - 1}} \right) - 1 - \sum_{k \in Nb_l(j) \setminus i} \lambda_{kj}^{\tau}(x_j) \tag{3.21}$$

$$e^{\lambda_{ij}^{\tau+1}} = \sum_{x_i, x_j} e^{\theta(x_i, x_j) + \nabla_{\mu(x_i, x_j)} v + \lambda_{ij}^{\tau}(x_j) + \lambda_{ji}^{\tau}(x_i) - 1} \tag{3.22}$$

where $W(\cdot)$ is the Lambert W-function.

Algorithm 1: Hybrid Bound Based Belief Propagation (HBP) for MAP

input: Graph $G = (V, E)$ and potentials θ_{ij} per edge

repeat

  Send message δ to $j \in Nb_q(i)$: $\delta_{i\to j}(x_j) = \sum_{x_i} \mu(x_i)\theta(x_i, x_j)$ for each agent $i \in V$

  $\nabla_{y(x_i)} v = \mu_i(x_i)\big(1 + \sum_{j\in Nb_q(i)} \delta_{j\to i}(x_i)\big)\ \forall i \in V$

  Initialize $\mu^0(x_i) = \nabla_{y(x_i)} v\ \forall i \in V$, $\mu^0(x_i, x_j) = \mu(x_i, x_j) e^{\theta(x_i,x_j)}\ \forall (i, j) \in L$

  repeat

    foreach *edge* $(i, j) \in L$ **do**

      $Normalize\big(\mu^\tau(x_i, x_j)\big)$

      $Z^\tau(x_j) = \nabla_{y(x_j)} \exp\big(\frac{\nabla_{y(x_j)} v}{\mu^\tau(x_j)}\big) / \sum_{x_i} \mu^\tau(x_i, x_j)$

      $\mu^{\tau+1}(x_i, x_j) = \mu^\tau(x_i, x_j) \exp\left(W(Z^\tau(x_j)) - \frac{\nabla_{y(x_j)} v}{\mu^\tau(x_j)}\right)$

      $\mu^{\tau+1}(x_j) = \nabla_{y(x_j)} v / W(Z^\tau(x_j))$

      Repeat steps 8 to 10 analogously for node x_i

  until *inner loop converges*

until *outer loop converges*

return: The decoded complete integral assignment

The Lambert W-function is the multi-valued inverse of the function $w \mapsto we^w$ and is useful in many engineering applications such as jet fuel planning and enzyme kinematics [29]. In our case, the argument of $W(\cdot)$ in Eq. (3.21) is always positive and the W-function is properly defined over this range. Furthermore, these dual updates can be reinterpreted in terms of the primal parameters, which alleviates the need to store the Lagrange multipliers explicitly and greatly simplifies the implementation. The proof is given in the supplementary material. The resulting message-passing algorithm called *Hybrid Bound Based Belief Propagation (HBP)* is detailed in Alg. 1. The superscripts on parameters (e.g. Z^τ) differentiate between the old parameters and the new updated ones. The function $Normalize(\cdot)$ normalizes a distribution by dividing each element by their sum. The inner loop convergence is detected by checking if the node and edge marginals sum up to 1, with some tolerance allowed (10^{-3}).

Complexity: From Proposition 3, the total number of parameters is $O(k^2|L| + nk)$, where k is the domain size. The total space complexity of Alg. 1, including the space for the δ messages, is $O(k^2|L| + nk + |Q|k)$. The time complexity is $O\big(Tk^2\big(|Q| + I|L|\big)\big)$ where T

denotes the total number of outer loop iterations and I denotes the total number of inner loop iterations.

3.5.3 Properties of HBP Algorithm

We now revisit some of the desirable properties for message-passing algorithms as highlighted in Section 2.4.

- **Convergence:** HBP algorithm increases the negated objective function of Eq. (3.10) monotonically. This objective function is bounded above by the optimum of the LP relaxation. Therefore, the HBP algorithm will converge eventually.

- **Anytime Behavior:** The HBP algorithm also has the anytime property w.r.t. the objective function of Eq. (3.10). This, in theory, does not guarantee that the extracted integral solution quality will also increase per iteration. However, in practice, we noticed that the increase in integral solution quality is highly correlated with the increase in the objective of HBP.

- **Error Bounded:** HBP algorithm can also provide bounds on the MAP solution quality. When there are no QP edges in the hybrid bound ($|Q| = 0$), then the hybrid bound is identical to the outer bound or the LP relaxation. Therefore, we can use the HBP algorithm to solve the LP relaxation too, which provides an upper bound on the MAP solution quality. Notice that due to Prop. (4) and Prop. (5), HBP will converge to an optimal solution of the LP relaxation as there are no local optima in the LP relaxation.

- **Adaptive:** HBP algorithm also has this property as the message-passing approach of Alg. 1 can easily incorporate new nodes in the graph. Due to iterative nature of the HBP algorithm, most of the messages from the previous iteration will be preserved, other than those which are related to the modified part of the problem. Thus, HBP provides a warm start for the modified problem by reusing messages from the unchanged part of the problem.

- **Privacy Preserving:** HBP algorithm has privacy in a *limited* sense. As the HBP algorithm is based on local message-passing, an agent i does not has any direct information about any agent k that is not an immediate neighbor of agent i. However, this does not formally guarantee that an the agent k can not deduce any information about agent i. Developing message-passing approaches which provide privacy guarantees in a formal sense is an ongoing research frontier in the multiagent systems community [40, 80, 81]. We plan to integrate such privacy preserving techniques in the HBP algorithm in the future.

3.6 Other Contributions

We have made other contributions for solving the MAP problem using message-passing. In [73], we developed a message-passing algorithm which solves the QP formulation. This message-passing approach was derived by representing the QP formulation as a DC program, similar to the approach described here. However, the QP formulation can suffer significantly from the problem of local optima. Therefore, we derived a message-passing approach for a convexified QP relaxation of the MAP problem. The resulting message-passing scheme can solve the convexified QP formulation optimally and significantly faster than CPLEX, which ignores the underlying graphically structure of the MRF. In [71], we showed a close relationship between the MAP problem and the problem of likelihood maximization. The MAP problem can be reformulated in a lossless manner to the problem of likelihood maximization in a mixture of simple Bayes nets. We then developed a message-passing approach for MAP by adapting the well know Expectation-Maximization (EM) approach for likelihood maximization.

Our experience in these previous approaches was that solving the QP formulation of MAP directly may lead the algorithm to get stuck in poor local optima for some instances. However, the QP formulation has the advantage that it is exact and results in a very compact program. This lead us to the present idea of combining the QP formulation with that of LP formulation to retain the good properties of both these formulations, namely accuracy and convexity.

3.7 Empirical Evaluation

We now describe empirical results demonstrating the effectiveness of the HBP algorithm. Our main goals are to show the effectiveness of the HBP and the hybrid bound on following metrics:

- **Solution quality:** This is an obvious criterion to compare HBP with other algorithms. We would like to show that the HBP algorithm is competitive with other state-of-the-art algorithms in the DCOP and machine learning literature on standard benchmarks.

- **Variational Bound Accuracy:** One of the main motivating factor for developing the hybrid bound was that it provides a better approximation to the marginal polytope than the LP relaxation. Even though the optimization over the hybrid bound in non-convex, we would also like to demonstrate that by the judicious choice of QP edges, we can avoid poor local optima. We show this by comparing the *fractional solution quality* of the LP relaxation and the hybrid bound. The term *fractional* quality denotes the value of the objective function the respective algorithm is optimizing—$\max_{\boldsymbol{\mu} \in \mathcal{L}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta}$ for the LP relaxation and $\max_{\boldsymbol{\mu} \in \mathcal{H}(G;Q)} \boldsymbol{\mu} \cdot \boldsymbol{\theta}$ for the hybrid bound. Note that, we approximate $\max_{\boldsymbol{\mu} \in \mathcal{H}(G;Q)} \boldsymbol{\mu} \cdot \boldsymbol{\theta}$ by using the final value given by HBP upon convergence, as the maximization cannot be performed exactly due to non-convexity.

- **Iterations:** In each iteration of the HBP, every agent sends and receives a message from all its neighbors. Therefore, the number of iterations required for convergence implicitly determine the execution time and the message overhead, which is an important metric in the multiagent setting.

- **Error Bound:** Finally, we would like to determine that how far the solution quality given by the HBP algorithm is from the optimal for each instance. As both the MAP and DCOP problems are NP-Hard, for many of the instances we use, we could not determine the exact optimal solution even by using centralized branch-and-bound search techniques. However, there are algorithms in the machine learning literature,

such as MPLP [134], that can provide an upper bound on the MAP solution quality. Therefore, we report the error bound w.r.t. the best upper bound reported by MPLP.

Next, we describe the algorithms which are used in experiments.

- **HBP:** It refers to our approach or the Hybrid Belief Propagation (Alg. 1). The HBP approach is parametrized by the set of QP edges Q. The strategy for computing this set is the following. As the LP relaxation is exact for tree structured graphs, we compute multiple spanning trees of the given problem instance. We then make the union of all edges in these spanning trees as LP edges and the rest of the edges constitute the QP edge set Q. We will denote the specific number of spanning trees along with the particular instance. We denote the fraction of the LP edges as the fraction of total edges for which the LP constraints are enforced ($=\frac{|L|}{|E|}$). The fraction of QP edges is given by $1-\frac{|L|}{|E|}$. This provides an insight about how much contribution was needed from the QP formulation to get a good solution.

- **HBP-LP:** It refers to our approach or the Hybrid Belief Propagation when it is applied to solve the LP relaxation. That is, in this case the QP edge set Q is empty. We tested this variant to show that HBP-LP converges to the optimal solution quickly and thus, can also provide quality bounds.

- **MS:** Max-Sum algorithm [137] is a state-of-the-art approximate algorithm to solve DCOPs. We obtained the code from the authors. This algorithm is a DCOP variant of the classical max-product algorithm, which solves the MAP estimation problem [107].

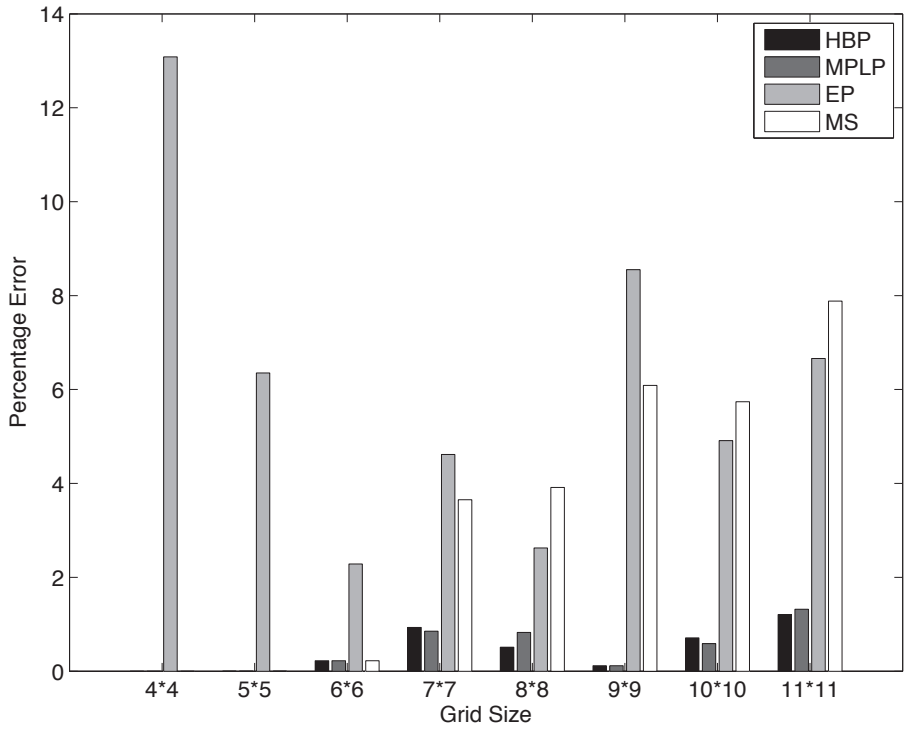
- **ADOPT:** Asynchronous Distributed Optimization (ADOPT) algorithm is a widely used complete DCOP algorithm based on asynchronous search. It is guaranteed to provide the optimal solution quality. However, it may have to exchange exponential number of messages in the number of agents before converging to the optimal solution. We used the latest variant of ADOPT called BnB-ADOPT [161]. The source code was provided by the authors.

- **MPLP:** Max-Product Linear Programming (MPLP) algorithm is a state-of-the-art algorithm to solve the MAP estimation problem in the machine learning literature [49].

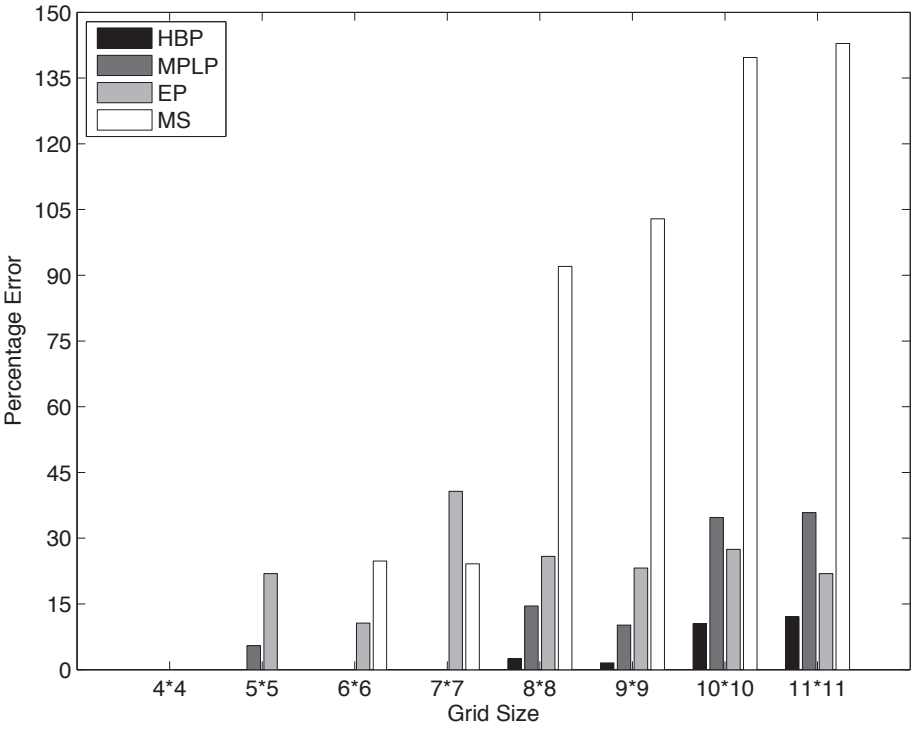
It is also based on iterative message-passing. We use the cluster-based variant of MPLP [134], that further tightens the LP relaxation by incorporating node clusters of size 3 in the LP relaxation. This algorithm works on the dual of the LP relaxation and decreases the upper bound on the MAP quality per iteration. We use the best upper bound computed by MPLP for computing all error bounds, unless stated otherwise.

- **EP:** Entropic Propagation (EP) is a globally convergent message-passing algorithm that solves the LP relaxation [120] in the machine learning literature. This algorithm works on the vanilla LP relaxation, unlike the previously described MPLP algorithm. We implemented this algorithm based on the publication [120]. We use EP to mainly compare its final solution quality against our approach HBP-LP.

We next describe the results of comparisons among different algorithms on synthetic and real-world benchmarks in the DCOP literature, operations research (OR) literature and machine learning literature.



(a) Integral solution quality comparisons for 1-step sensor scheduling domain



(b) Integral solution quality comparisons for 4-step sensor scheduling domain

Figure 3.2. Solution quality comparisons for sensor scheduling domain. The absence of a bar for a particular algorithm denote zero error implying the algorithm achieved the optimal solution.

3.7.1 DCOP Benchmarks

We experimented on the sensor network scheduling problems from [86, 161]. In this problem, sensors are arranged in the form of a $n \times n$ grid graph. Each node in this graph is a sensor agent. An agent can potentially scan all its neighboring grid cells. There is a target located in each grid cell. To track a target successfully, simultaneous scan by all four sensors surrounding the grid cell is required. There is a cost to detect each target, which is selected randomly between $[0, 100]$ per target. If the target is not detected, then the cost is 100. The goal is to schedule the scan action of sensors such that the total cost is minimized. We experimented with two problem settings—*1-step scheduling* and *4-step scheduling*. In the 1-step setting, there is only one time step. That is, each sensor can be detected at time step 0 or remain undetected. This results in a domain size of 2 for each variable. In the 4-step problems, there are four time steps and a target can be scanned during one of these time steps or remain undetected. This results in a domain size of 5 for each variable. The 4-step scheduling problems are naturally more complex than the 2-step problems.

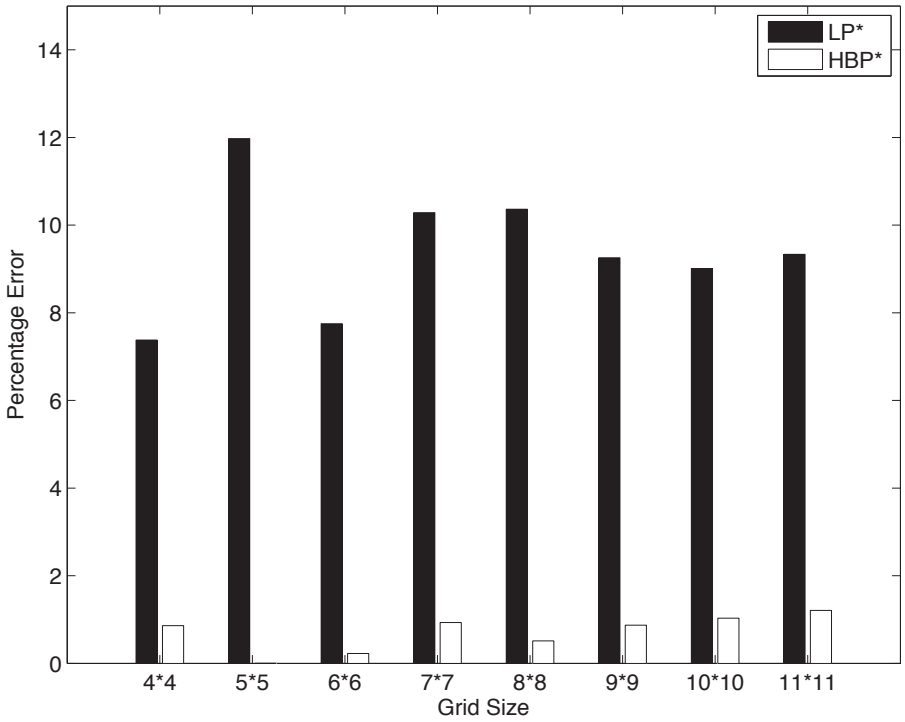
Solution quality: Figure 3.2(a) shows the solution quality comparisons for different algorithms and varying grid sizes for 1-step scheduling problems. On the y-axis, we report the error bound for each algorithm, which is computed by using the best upper bound provided by the MPLP algorithm. The results show that the HBP and the MPLP provide the best solution quality. Furthermore, the error bound is quite small ($< 2\%$), implying that both these algorithms provide near-optimal solution. The MS algorithm provides good solutions for small instances. However, for larger instances, it provides much worse solution quality than both HBP and MPLP. Furthermore, the LP relaxation based approach, EP, also provides worse solution quality than HBP. This proves our claim that the hybrid bound provides a more accurate approximation to the marginal polytope than the local polytope. The fraction of LP edges for the HBP approach was approximately 60%, corresponding to 2 spanning trees. This shows that a non-trivial contribution from the QP formulation is required to make the hybrid bound tighter than the local polytope. Furthermore, despite the non-convexity introduced by the QP constraints (for 40% of edges), our results show that the HBP algorithm does not get stuck in a poor local optima. In-fact, it achieves near-optimal solutions.

Figure 3.2(b) shows the solution quality comparisons for 4-step scheduling problems. These problems are much more complex than the 1-step scheduling problems. The results clearly show that HBP outperforms all other approaches by a wide margin. Even for the largest instance (11×11 grid), the HBP approach has an error of about 12%, whereas MPLP has an error of about 30% and MS has an error of about 135%. This shows that the HBP approach is highly competitive with previous state-of-the-art approaches in both machine learning and DCOP literature. Furthermore, the worst-case quality bounds indicate that the HBP algorithm provides near-optimal solutions (within 88% of the optimal) even for large instances.

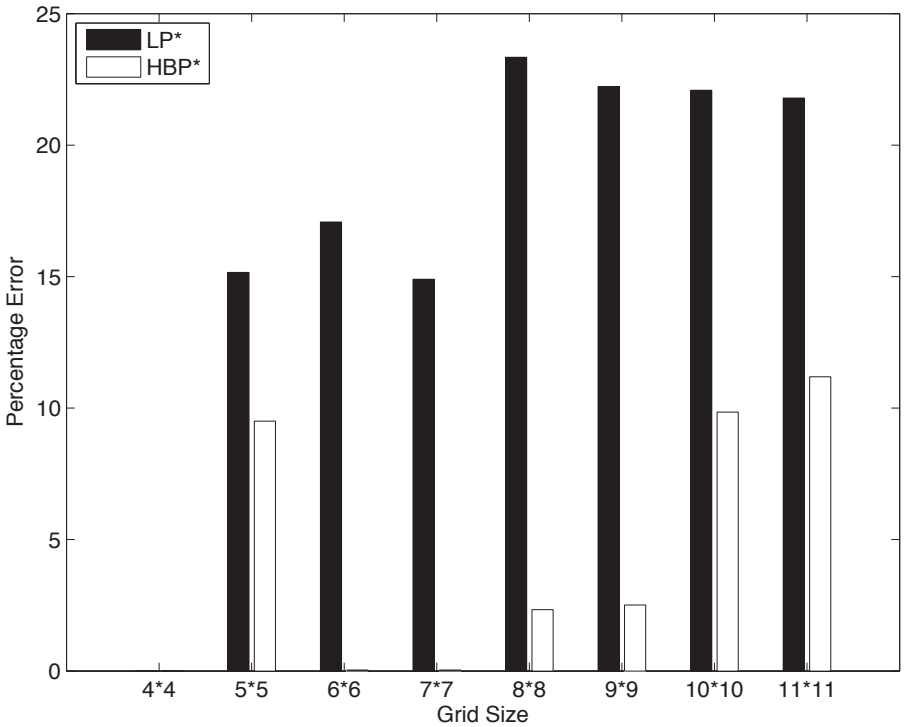
Bound accuracy: Figure 3.3 shows the fractional solution quality comparisons for the HBP algorithm, denoted as HBP*, and the LP relaxation, denoted as LP*. The y-axis denotes the percentage error using the upper bounds provided by MPLP. For both 1-step and 4-step problems, the HBP algorithm has significantly less error than the LP relaxation. This confirms empirically that the hybrid bound is much more accurate than the outer bound based LP relaxation. Furthermore, even though the hybrid bound is non-convex, a judicious choice of the QP edges helps it to avoid getting stuck in a poor local optima.

Time comparisons: Figure 3.4 shows the time comparisons between HBP and MS algorithm. Both algorithms were run for 2000 iterations. In addition, the HBP algorithm also used 20 inner loop iterations per outer loop iteration. The time shown in figure 3.4 denote the total running time. We can see that even for large benchmarks, the total running time of HBP is about 1 minute. The MS algorithm is faster than the HBP as MS does not has inner loops. However, the HBP algorithms makes up for it by providing much better solution quality. For the MPLP algorithm, we used a fixed time cutoff of 30 minutes. Even running it for 1 hour did not provide any significant benefit w.r.t. quality over shorter time durations.

Comparison with optimal approaches: Table 3.1 shows the comparison of HBP with optimal search based algorithms. We show results of comparisons with both decentralized search algorithm (BnB-ADOPT) and centralized search algorithm AND/OR branch-and-bound search (AOBB) [89] using the existential directional arc consistency (EDAC) heuristic [31]. The BnB-ADOPT is the state-of-the-art complete solver for DCOPs and



(a) Fractional solution quality comparisons for 1-step sensor scheduling domain



(b) Fractional solution quality comparisons for 4-step sensor scheduling domain

Figure 3.3. Fractional solution quality comparisons for sensor scheduling domain. The absence of a bar for a particular algorithm denote zero error implying the algorithm achieved the optimal solution.

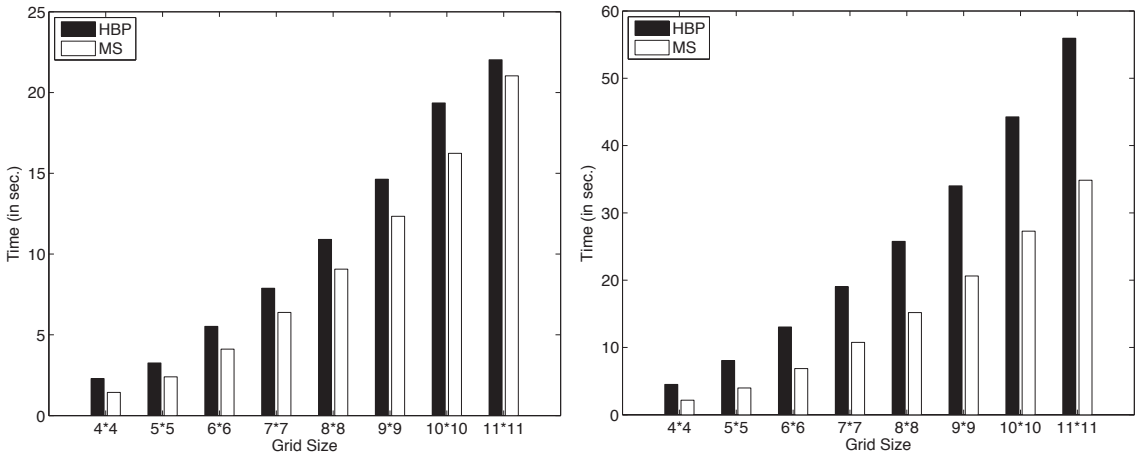


Figure 3.4. Time comparisons for sensor scheduling domain. a) 1-step sensor scheduling; b) 4-step sensor scheduling

the AOBB algorithm is the state-of-the-art solver for centralized constraint optimization. The 'Cycles' column for ADOPT indirectly indicates the message overhead as in each cycle, every agent sends a message to every other agent. The 'Time' column denotes the total running time for all the algorithms. The 'Nodes' column denote the total number of nodes expanded by the AOBB algorithm. The 'Iterations' column denote the total number of outer loop iterations for the HBP approach. The 'Error' column indicates the percentage error in the solution quality of the HBP algorithm against the optimal solution obtained by the AOBB.

As shown in Table 3.1, the HBP approach clearly outperforms the BnB-ADOPT algorithm, which does not terminate in the time limit of $16,000$ seconds except for the smallest of instances. The HBP algorithm obtains the optimal solution quality for every instance solved by the BnB-ADOPT algorithm. Furthermore, the HBP approach is much faster than the BnB-ADOPT for such instances.

The centralized search algorithm, AOBB, obtains the optimal solution for all of the 1-step scheduling problems. The running time of AOBB is faster than HBP. However, it should be noted that the AOBB approach cannot be used in a decentralized setting. The comparisons are used only to highlight the relative hardness of instances. The 1-step scheduling problems seem to be easier from the centralized search perspective. Moreover, in terms of solution quality, the HBP algorithm finds near-optimal solution for each of the

	ADOPT		AOBB+EDAC		HBP		
Instance	**Cycles**	**Time**	**Nodes**	**Time**	**Iterations**	**Time**	**Error**
4×4	54	1.2	12	0	2000	2.29	0
5×5	601	61.6	93	0	2000	3.25	0
6×6	11311	6085.7	170	0	2000	5.52	0
7×7	–	–	513	0.01	2000	7.88	0.076
8×8	–	–	1555	0.04	2000	10.90	0
9×9	–	–	10631	0.38	2000	14.63	0
10×10	–	–	119902	3.60	2000	19.35	0.122
11×11	–	–	2276099	95.63	2000	22.02	0.110
4×4	1192	63	54	0	2000	4.50	0
5×5	–	–	138	0	2000	8.06	0
6×6	–	–	1531	0.06	2000	13.03	0
7×7	–	–	6720	0.34	2000	19.02	0
8×8	–	–	1616973	117.56	2000	25.75	0.72
9×9	–	–	163536683	12621	2000	34.01	0.82
10×10	–	–	256307635	–	2000	44.26	–
11×11	–	–	252233565	–	2000	55.94	–

Table 3.1. Quality and time (in sec.) comparisons with BnB ADOPT and centralized branch-and-bound solver AOBB. A '–' indicates the algorithm did not terminate within 16000 sec. time limit. The top half of the rows indicate 1-step scheduling benchmarks and the bottom half denotes the 4-step scheduling benchmarks. Error is w.r.t. the optimal solution given by AOBB in percentage:= $\frac{(HBP-AOBB)*100}{AOBB}$

instances with an error of only 0.1%. This confirms that the HBP algorithm can obtain good solution quality on a number of benchmarks.

The 4-step scheduling problems are much harder even from the centralized search perspective, as shown in the bottom half of Table 3.1. AOBB algorithm is faster than HBP for medium size instances (up to 7×7 grid). However, for larger instances (8×8 to 11×11 grids), the HBP approach is much faster. The hardness of these larger instances can be judged from the fact that AOBB takes about 12621 seconds for 9×9 grid and does not terminate within the time limit of 16000 seconds for 10×10 and 11×11 grid problems. HBP algorithm on the other hand takes less than a minute even for these harder problem thanks to its scalable message-passing structure and provides near-optimal solution as shown in Figure 3.2(b). Thus, the HBP approach is also competitive with state-of-the-art centralized branch-and-bound solver.

Inst.	Opt.	HBP	$\lvert L\rvert$	HBP*	MPLP	MPLP*	HBP-LP	HBP-LP*	EP	EP*	T_{HBP}/r	T_{mplp}
100-1	7970	**7970**	0.553	7963.9	7594	8743.8	7634	10147.6	7634	10183.5	4.26/0	3.6
100-2	11036	**11036**	0.512	11035.9	10866	11341	10976	12231.1	11014	12228.41	33.4/1	10.1
100-3	12723	**12723**	0.514	12722.9	12723	12988.1	12723	13652.5	12723	13598.1	0.07/0	62.5
100-4	10368	**10368**	0.532	10367.7	10076	10728.1	10172	12044.9	10172	12059.2	2.40/0	4.4
100-5	9083	**9083**	0.562	9082.9	8685	9506.8	8914	10629.9	8914	10613.9	14.4/1	5.8
100-6	10210	**10210**	0.533	10099.9	9704	11553.7	10210	12937.6	10210	13045.3	6.9/1	3.5
100-7	10125	**10125**	0.554	10054.6	9952	10904.4	9977	11978.3	9977	11973.7	3.9/1	2.8
100-8	11435	**11435**	0.528	11434.6	11310	11793.8	11397	12615.4	11397	12436.1	4.6/0	3.3
100-9	11455	**11455**	0.533	11454.9	11361	11628.4	11368	12259.6	11348	12219.2	5.8/0	3.6
100-10	12565	**12565**	0.536	12564.6	12459	12831.1	12511	13676.8	12511	13617.9	40.3/0	6.1
250-1	45607	**45607**	0.477	45606.9	41862	54477.5	44872	77827.5	44956	77382.8	56.1/0	4498.1
250-2	44810	**44810**	0.484	44506.4	38659	55664.5	44234	77469.2	44234	77314.6	209.5/2	775.9
250-3	49037	**49037**	0.477	49011.6	44954	57118.4	48980	80278.5	48980	80015.4	3.98/1	822.1
250-4	41274	**41274**	0.474	41273.4	34670	51687.8	41124	74784.7	40637	74608.7	146.2/5	6164.9
250-5	47961	47939	0.520	47938.9	44461	56607.6	47843	79256.0	47843	78912.9	196.1/7	2708.8
250-6	41014	**41014**	0.475	39748.4	34092	53157.7	40225	77758.0	40225	77617.2	289.1/11	1271.5
250-7	46757	**46757**	0.477	46521.6	41974	56397.7	46449	79393.9	46431	79075.1	61.1/0	1992.5
250-8	35726	35336	0.489	35504.1	29312	49492.6	34656	72101.1	34656	71896.3	380.4/6	477.4
250-9	48916	**48916**	0.473	48907.6	44057	58305.3	48677	81115.7	48615	80684.5	353.5/0	3236.1
250-10	40442	40330	0.497	40032.6	32534	52525.6	40194	75142.7	40198	74783.1	116.8/7	908.1
1b.20	133	**133**	0.197	96.9	98	260.7	98	394.5	98	387.2	1.4/0	0.82
2b.30	121	**121**	0.121	86.9	67	345.7	65	505.8	65	504.2	2.2/0	0.08
3b.40	118	**118**	0.097	46.1	102	416.3	102	612.1	102	609.1	66.6/0	240.4
4b.50	129	**129**	0.079	23.1	85	548.3	54	799.8	54	797.5	3.7/0	702.7
5b.60	150	**150**	0.065	46.7	88	632.6	150	922.1	150	918.0	0.17/0	71.9
6b.70	146	133	0.057	14.8	63	758.1	62	1101.6	62	1099.6	49.2/0	714.16
7b.80	160	**160**	0.050	14.8	122	840.1	54	1220.1	54	1218.4	28.3/0	66.5
8b.90	145	**145**	0.040	13.8	58	1435.5	72	1389.4	72	1386.7	47.7/0	0.27
9b.100	137	135	0.032	12.9	58	1656.5	120	1599.6	120	1597.1	36.7/0	0.37
10b.125	154	**154**	0.044	10.8	–	–	104	1898.3	104	1895.6	84.9/0	–

Table 3.2. Quality and runtime comparisons for the Biq benchmarks. 'Inst.' is the instance name and 'Opt.' is the known optimal quality from [123]. '–' denotes failure of the algorithm to produce a solution.

3.7.2 Operations Research Benchmarks

We also tested the HBP approach on a number of benchmarks from the operations research literature. We use the Biq dataset or the binary integer quadratic (Biq) problems publicly available from the *Biq Mac* library [123, 122]. Our main goal in these experiments is twofold: a) show that the optimization over the hybrid bound $\mathcal{H}(G;Q)$ provides a tighter approximation of the MAP problem and a significantly better solution quality than LP relaxation over the outer bound; and b) show that CCCP converges to a global optimum of the LP relaxation over the outer bound by comparing its quality against EP, which optimizes the same objective function.

We use the convention that an asterisk (*) for a given algorithm denotes the best objective the algorithm achieved upon convergence (can be fractional); the name without it shows the decoded integral assignment. For example, MPLP* denotes the upper bound which MPLP minimizes, whereas MPLP denotes the integral quality. Similarly, $HBP^{\star}$ and $EP^{\star}$ denote the fractional solution quality.

Table 3.2 shows the results for the Biq benchmarks. These benchmarks are particularly interesting in that the LP relaxation on these problems is loose. Therefore, tightening the outer bound is particularly beneficial. The last two columns show time (in sec.), while the other columns show the best quality achieved for a given instance. For details on generating these Biq problems and formulating them as MRFs we refer to [123, 122]. We chose Biq problems with a varying number of nodes and edge densities. The instances '100-1' to '100-10' have 100-node graphs with edge density 0.1, and edge potentials in the range $[-200, 200]$ (referred to as 'bqp100-i' in [122]). The instances '250-1' to '250-10' have 250-node graphs with edge density 0.1 ('bqp250-i' in [122]). The instances $1b.n$ to $10b.n$ are complete graphs (density 1) with 'n' denoting the number of nodes ('gka$i$b.n' in [122]). We used the decoding scheme of [119] for both the HBP and EP algorithms.

Our hybrid bound based approach 'HBP' performs quite well in this dataset and achieves the optimal solution for 26 out of 30 instances. The column '$|L|$' shows the fraction of the LP edges ($|L|/|E|$) in the hybrid bound. We selected the LP edges using the strategy shown in Fig. 3.1(c): construct randomly a number of independent spanning tress and make L the union of their edges. For instances '100-1' to '250-10', we used 8 spanning trees and for

instances '1b.n' to '10b.n', only 2 trees were required. Despite such low fraction of LP edges, our approach returns optimal solutions indicating that a judicious choice of the LP edges can mitigate the adverse effect of non-convexity. This observation is further reinforced by noting the fractional objective under 'HBP*', which is quite close to the true optimum. On the other hand, the relaxed LP objective (under 'HBP-LP*' and 'EP*') seems to be loose, specially for '250-i' and 'gka' instances. The minor differences between 'HBP-LP*' and 'EP*' are mainly because the normalization constraints were enforced only to an accuracy of 10^{-3} for faster convergence.

The decoded solution quality for HBP-LP and EP is not as good as HBP, as the LP relaxation quite loose on these problems. MPLP provides tighter upper bounds than HBP-LP (under 'MPLP*'), but it fails to translate it into good solution quality as the upper bounds are still loose when compared to the true optimum. HBP algorithm, as noted earlier, provides the optimal solution for 26 out of 30 instances and a near-optimal solution for the rest of the instances. This again confirms that the HBP approach can significantly tighten the outer bound and provides good solution quality without getting stuck in a poor local optima.

The last two columns provide runtime comparisons between HBP and MPLP. The total outer loop iterations in HBP was fixed to 1100 and inner loop iterations to 60. MPLP was run for 2 hours. The time in these columns shows when the best integral solution quality was first achieved for each algorithm. In addition, for HBP, the performance was dependent on the choice of spanning trees. Therefore we reported the best of 12 runs. The r value in 'T_{ccqp}/r' denotes the best run. For most instances, HBP was quite robust. For the '250-i' dataset, a higher number of runs was required. These instances are also the most difficult of all the instances even for the optimal approach of [123], which required ≈ 4500 sec. at the minimum and up to 316000 sec. for some instances. HBP on the other hand terminates within ≈ 650 sec. for all the '250-i' instances. The complete set of results for max-sum in shown in table 3.3. MS converges and achieves good quality for the '100-i', '250-i' instances, but for 'gka' instances it performs poorly achieving a quality of 0 for most instances.

Instance	Optimal	MP
100-1	7970	7822
100-2	7970	11036
100-3	12723	12723
100-4	10368	10368
100-5	9083	9083
100-6	10210	10065
100-7	10125	10034
100-8	11435	11435
100-9	11455	11455
100-10	12565	12565
250-1	45607	45607
250-2	44810	44810
250-3	49037	49037
250-4	41274	41270
250-5	47961	47961
250-6	41014	41014
250-7	46757	46757
250-8	35726	34450
250-9	48916	48916
250-10	40442	40442
1b.20	133	0
2b.30	121	0
3b.40	118	0
4b.50	129	0
5b.60	150	0
6b.70	146	61
7b.80	160	0
8b.90	145	0
9b.100	137	0
10b.125	154	0

Table 3.3. Solution quality comparisons for max-sum (MS) on Biq benchmarks

3.7.3 Machine Learning Benchmarks

We also experimented on a number of synthetic and real-world benchmarks from the machine learning literature. For the synthetic benchmarks, we tested HBP on the $n \times n$ grid graphs with potentials functions generated using the Potts model [120] and the Ising model [132]. These models are commonly used in computer vision problems [44]. The total number of nodes in a grid graph are n^2 and total edges are $2n(n-1)$.

3.7.3.1 Synthetic Benchmarks

For both these models, the edge potentials were sampled from $\mathcal{U}[-\beta, \beta]$ for each edge independently, where β is the coupling strength parameter. The unary potentials were

sampled from $\mathcal{U}[-1, 1]$. The problems become harder with the increasing value of β. For these graphs, MPLP performs quite well in minimizing the upper bound. Therefore, we report normalized values for all the algorithms in Fig. 3.5(a–c), where the best MPLP upper bound (U.B.) is denoted as 1 and the quality obtained by rest of the algorithms are linearly scaled.

Fig. 3.5(a,b) show that HBP again achieves better solution quality than the LP relaxation (HBP-LP, EP). The decoded quality is within 98.5% of optimal for all the settings. For the Potts graphs, HBP used 2 spanning trees for the set L with the fraction of LP edges being $\approx .75$. The HBP fractional objective ('HBP*') is very close to the MPLP U.B., which further confirms that the outer bound is tightened significantly using QP edges. The plots for 'HBP-LP*' and 'EP*' almost overlap, which shows that HBP-LP converges to the same quality as EP for the LP relaxation.

For the Ising graphs (Fig. 3.5(c)), HBP provides better quality (within 97% of the optimal on average) than all the other algorithms including MPLP. Interestingly, the performance of CCQP improved with a lower fraction of LP edges ($\approx .45$) than in the Potts model. This can be explained by the relaxed LP objective ('HBP-LP*' or 'EP*'), which is relatively loose (≈ 1.25) as opposed to (≈ 1.06) in Potts. Therefore, a higher fraction of QP edges is required to tighten the outer bound. This can also help explain the deteriorated performance of MPLP, which may need even higher order clusters to further tighten the outer bound. MS provided the worse quality for all these problems and is not plotted for clarity sake. The normalized quality MS achieves for each coupling strength parameter setting is: (0.99, 0.93, 0.82, 0.77, 0.75) for the Potts model with domain size 4 and (0.99, 0.94, 0.89, 0.82, 0.77) with size 8. For the Ising graphs, it was (0.60, 0.63, 0.62, 0.61, 0.61). Thus, HBP algorithm provides consistent and much better performance than MS.

3.7.3.2 Real-World Protein Design Benchmarks

We also experimented on the real-world protein design problems (total of 97 instances), which are described in [159]. In these problems, given a desired backbone structure of the protein, the task is to find a sequence of amino-acids that is as stable as possible or has the lowest energy. This problem can be represented as finding the MAP configuration in

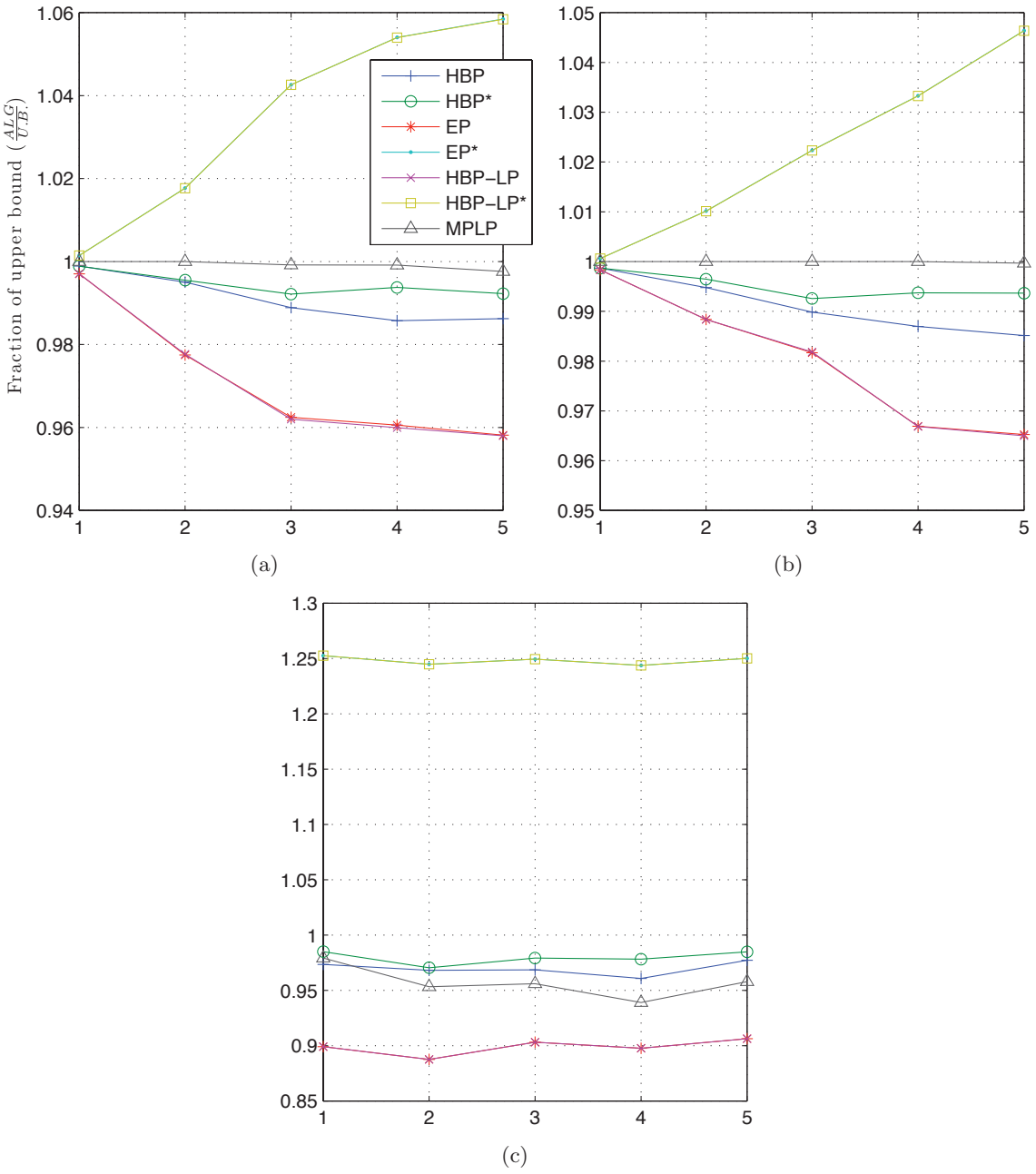


Figure 3.5. Quality comparison for 50×50 grids: a) Potts model with variable domain size 4 and b) with domain size 8; c) Ising model (domain size = 2). The x-axis shows the coupling parameter β and y-axis the normalized quality.

an MRF. These problems are particularly hard and dense with up to 170 variables, each having a large domain size of up to 150 values. In these algorithms, the LP relaxation is quite tight. However, even the LP relaxation becomes so large that it can not be solved using the standard solvers such as CPLEX. For some of the largest problems, there are about 27 million LP variables and 0.5 million constraints. Thus, message-passing algorithms, such as HBP and MPLP, are crucial to solve the LP relaxation for such large benchmarks. The main goal in this set of experiments is to determine the convergence rate and scalability of the HBP-LP algorithm to solve the LP relaxation. The optimal solution for these benchmarks are obtained using the cluster based MPLP algorithm [49, 134].

Figure 3.6 shows the solution quality obtained by HBP-LP for the largest of 25 instances. We can see that for almost all of the instances, the HBP-LP algorithm obtains near-optimal solution quality, close to 94% of the optimal. This confirms that the LP relaxation is able to provide good solution quality for these instances. Figure 3.7 shows the detailed results for 6 of the largest proteins. The x-axis denotes the iteration number for the HBP-LP algorithm (outer loop) and the y-axis denotes the corresponding solution quality. The legend 'LP' denotes the fraction solution quality and 'Integral' denotes the solution quality of the integral solution. 'Optimal' denotes the optimal solution obtained using the MPLP, which tightens the LP relaxation using 3-node clusters.

As expected, the fraction solution quality ('LP') increases monotonically with each iteration and provides the upper bound on the optimal solution. Furthermore, the HBP-LP algorithm converges well within 1500 iterations. This shows that the HBP algorithm has fast convergence rate even for such large instances. The final gap between the 'LP' quality and the optimum is relatively small. This shows that the LP relaxation is quite tight in these problems. Therefore, attempting to tighten it using the hybrid bound did not yield significant improvement in the solution quality.

3.8 Discussion

We have analyzed the problem of single-step multiagent decision making using the framework of MAP estimation in graphical models. We investigated this problem through the lens of mathematical optimization. We showed how existing variational formulations of the

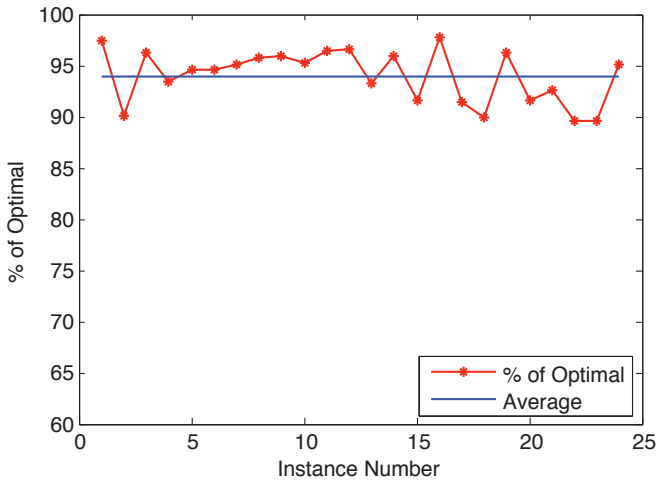


Figure 3.6. Protein design

MAP problem can be unified under a hybrid variational framework. This hybrid framework addresses the main problems of existing frameworks, namely, the non-convexity of the QP formulation and inaccuracy of the LP formulation. It retains the main advantages of these frameworks such as the accuracy of the QP formulation and the convexity of the LP formulation, while minimizing their disadvantages. We also develop a message-passing algorithm that can solve the optimization problem over the hybrid bound using message-passing. The resulting algorithm is highly scalable, increases the quality monotonically and is guaranteed to converge, unlike other approximate algorithms such as max-sum.

We experimented on a number of benchmarks in the DCOP, OR and machine learning literature. Our approach provides near-optimal solutions for a variety of benchmarks even when other approximate algorithms such as max-sum incur an error of more than 100%. Our approach is also more than an order-of-magnitude faster then complete DCOP algorithm BnB-ADOPT, which could only scale to small instances. Our approach also provides optimal solutions on a number of benchmarks from the OR literature, where standard machine learning approaches such as MPLP fail to do so. Overall, our hybrid bound based belief propagation algorithm proved to be a robust and scalable approach that provides high quality solutions across a number of benchmarks in diverse domains.

The main publications that describe the contributions of this chapter are the following:

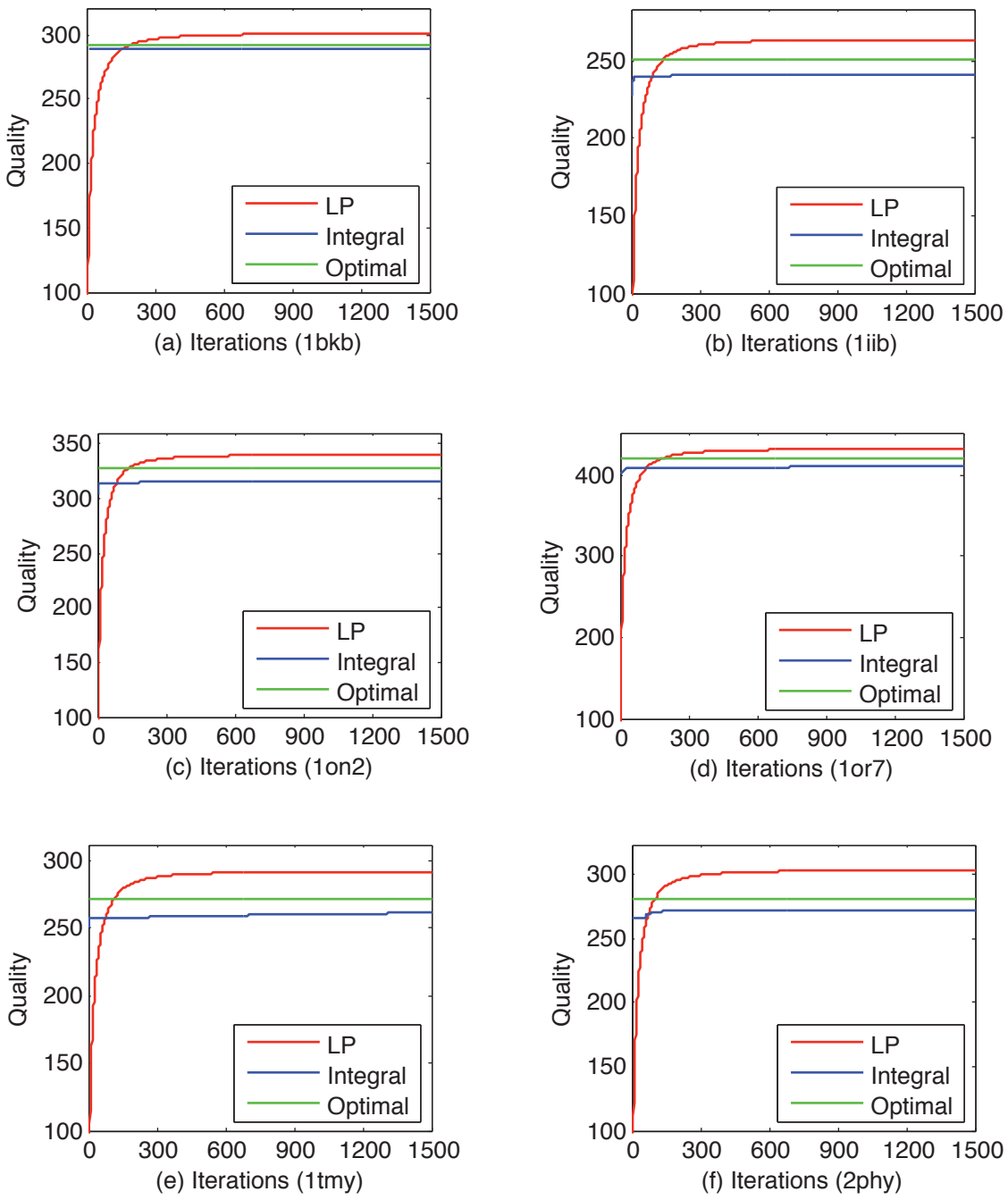


Figure 3.7. Protein design

- The variational framework based on the hybrid bound is described by the following publication. This publication also develops the message-passing algorithm to solve the optimization problem over the hybrid bound.

  A. Kumar, S. Zilberstein and M. Toussaint. Message-Passing Algorithms for MAP Estimation Using DC Programming. In *Proc. of the international conference on Artificial Intelligence and Statistics (AISTATS)*, pages 656–664, 2012.

- A message-passing algorithm that is based on the relationship between the quadratic programming formulation of MAP estimation and likelihood maximization is described by the following publication:

  A. Kumar and S. Zilberstein. MAP Estimation for Graphical Models by Likelihood Maximization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1180–1188.

- The relationship between convex and non-convex QP formulations of MAP and DC programming is described in the following publication:

  A. Kumar and S. Zilberstein. Message-Passing Algorithms for Quadratic Programming Formulations of MAP Estimation. In *Proc. of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 428–435, 2011.

- A formal mapping between DCOPs and MAP estimation along with an overview of MAP estimation algorithms that can be easily adapted to the DCOP setting is described in the following publication:

  A. Kumar, W. Yeoh and S. Zilberstein. On Message-Passing, MAP Estimation in Graphical Models and DCOPs. In *Proc. of the IJCAI Workshop on Distributed Constraint Reasoning (DCR)*, pages 57–70, 2011.

- A message-passing algorithm for decentralized management of smart power grids using the DCOP formulation is presented in the following publication:

A. Kumar, B. Faltings and A. Petcu. Distributed Constraint Optimization with Structured Resource Constraints. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 923–930, 2009.

- A technique based on constrained decision diagrams [27] to reduce the size of messages exchanged in a dynamic programming based DCOP algorithm called DPOP [108] is described in the following publication:

  A. Kumar, A. Petcu and B. Faltings. H-DPOP: Using Hard Constraints for Search Space Pruning in DCOP. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 325–330, 2008.

Sequential Decision Making

CHAPTER 4

THE DECENTRALIZED POMDP MODEL

The decentralized POMDP (Dec-POMPD) model has emerged in recent years as a unified framework for sequential decision making in multiagent systems. This chapter describes the basic characteristics of the model. We begin with a formal definition followed by a description of how agents' policies are represented. The decentralized nature of the planning process is highlighted along the way. This chapter also sets the foundation for the development of approximate algorithms in later chapters by describing how to represent policies using bounded memory for both finite and infinite horizon planning problems. We also discuss the objective function associated with different types of policy representations. Finally, we discuss some application domains for decentralized decision making.

4.1 The Dec-POMDP Model

The Dec-POMDP model can be defined with a tuple $\langle I, S, \{A^i\}, P, R, \{Y^i\}, O, T\rangle$, where:

- I denotes a finite set of n agents
- S denotes a finite set of states with designated initial state distribution η_0
- A^i denotes a finite set of actions for each agent i
- P denotes state transition probabilities: $P(s'|s, \vec{a})$, the probability of transitioning from state s to s' when the joint-action $\vec{a}$ is taken by the agents
- R denotes the reward function: $R(s, \vec{a})$ is the immediate reward for being in state s and joint-action taken as $\vec{a}$
- Y^i denotes a finite set of observations for each agent i

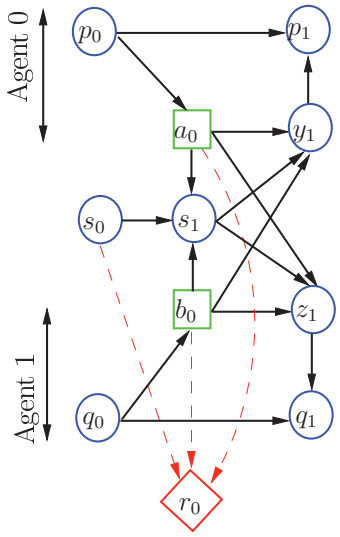


Figure 4.1. A 2-time slice dynamic decision network (DDN) representation of a 2-agent Dec-POMDP. All nodes are random variables. Square nodes represent decisions; diamond nodes represent the reward; p and q represent the states of policy for agent 0 and 1 respectively; subscripts denote time.

- O denotes the observation probabilities: $O(\vec{y}|s', \vec{a})$ is the probability of receiving the joint-observation $\vec{y}$ when the last joint-action taken was $\vec{a}$ that resulted in the environment state being s'
- T denotes the plan horizon or the number of time steps after which the problem terminates

Solving a Dec-POMDP means finding the joint-policy $\boldsymbol{\theta}$ that maximizes the total expected reward:

$$\mathbb{E}\Big[\sum_{t=1}^{T} R\big(s_t, \vec{a}_t; \boldsymbol{\theta}\big)\Big] \tag{4.1}$$

where $\boldsymbol{\theta}$ denotes the join-policy and subscript t denotes the dependence on time. In a Dec-POMDP, agents are acting under uncertainty not only about the underlying environment state but also about each other. Although the joint-observation received by agents may be correlated, each agent only observes its own component of the joint-observation. This makes the coordination problem particularly challenging.

When there are 2-agents in a given multiagent system, we adopt a simplified notation as follows. The action set of agent 1 is denoted by $a \in A$ and agent 2 by $b \in B$. The state transition probability $P(s'|s,a,b)$ depends upon the actions of both the agents. Upon taking the joint action $\langle a,b \rangle$ in state s, agents receive the joint reward $R(s,a,b)$. Y is the finite set of observations for agent 1 and Z for agent 2. $O(yz \mid s,a,b)$ denotes the probability $P(y,z|s,a,b)$ of agent 1 observing $y \in Y$ and agent 2 observing $z \in Z$ when the joint action $\langle a,b \rangle$ was taken and resulted in state s. For infinite-horizon Dec-POMDPs, a reward discounting factor $\gamma < 1$ is used. Figure 4.1 shows a DDN representation of a 2-agent Dec-POMDP. We use the convention that subscripts denote time and superscripts, if any, identify agents. Different policy representations for agents are described next.

4.2 Policy representation

An *information set* φ_t for an agent i is a sequence of actions performed by the agent and the observations received for t time steps: $\left(a_1^i \cdot y_2^i \cdot a_2^i \cdot y_3^i \cdots y_t^i\right)$ [7]. The set of all possible information sets of length less than or equal to $T-1$ for an agent i is denoted by Φ^i.

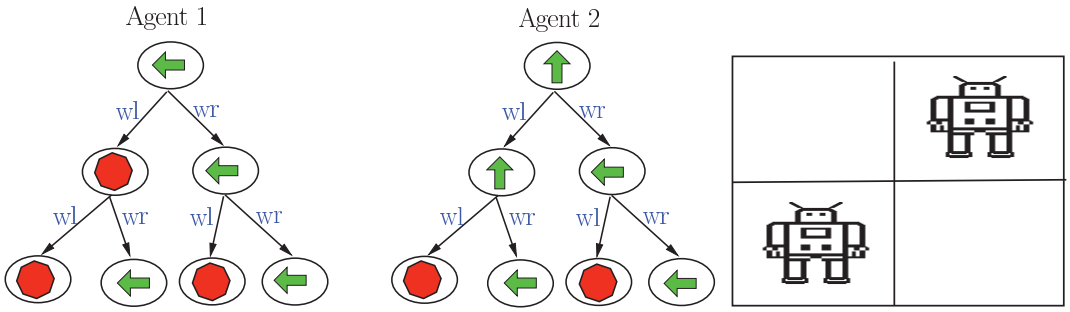
Definition 8. *An individual policy θ^i for an agent i is defined as a mapping from information sets to actions:*

$$\theta^i : \Phi^i \rightarrow A^i$$

A joint-policy $\boldsymbol{\theta} = \langle \theta^1, \theta^2, \ldots, \theta^n \rangle$ is a tuple representing the individual policy for each agent. The size of the set Φ^i is exponential in the length T of the plan. Therefore, the total number of individual policies for an agent i is *doubly-exponential* in the length of the plan. This makes decentralized planning significantly more challenging than the centralized case. The following result establishes the formal complexity:

Theorem 1. *The complexity of solving optimally 2-agent finite-horizon Dec-POMDP is NEXP-Complete [13].*

Despite this negative worst-case result, several approximate algorithms have been developed. Since even representing a policy takes an exponential amount of space in the plan length, we next introduce the notion of finite-horizon policy trees, which are the backbone of most approximate algorithms.



(a) Individual policy for agent 1 and 2 (b) Meeting in a grid domain

Figure 4.2. A 2-step policy tree representation for 'Meeting in a grid' problem

4.2.1 Finite-Horizon Policy Trees

Most of the existing approximate as well as optimal Dec-POMDP algorithms for finite-horizon planning use an explicit tree structured representation of the local policy for each agent. A node in such a policy tree denotes the action to be executed and edges correspond to the observations. Edges connect to subtrees, which are executed by the agent when the corresponding observation is received. If p denotes a policy tree, then a_p denotes the action specified at the root of the tree p. The edges correspond to observations $y^i \in Y^i$ and connect to the respective subtree p_{y^i}.

Figure 4.2 shows an example of policy trees for a toy meeting in a grid domain. In this problem, two agents want to meet at a particular cell in a grid, say the top left corner. However, they do not observe their current location in the grid and can only observe if the grid edge (or the wall) is to their left or right. Their actions also have stochastic outcomes—they can fail with certain probability. Figure 4.2(a) shows a 2-step policy tree for each agent. The actions are move left, right, top, down and stop. They are denoted symbolically in the policy tree. The observations are *wall left* (wl) and *wall right* (wr). They are shown on the edges of the tree.

One computational disadvantage with representing such policy trees is that their size will become exponential with the number of levels in the tree. To deal with such a blowup, a common approach is to share sub-trees at each level [127], as shown in Figure 4.3. In such an approximate representation, at each level there are `maxTree` number of nodes. For example, in Figure 4.3, the parameter `maxTree` = 5. Edges in such a representation connect

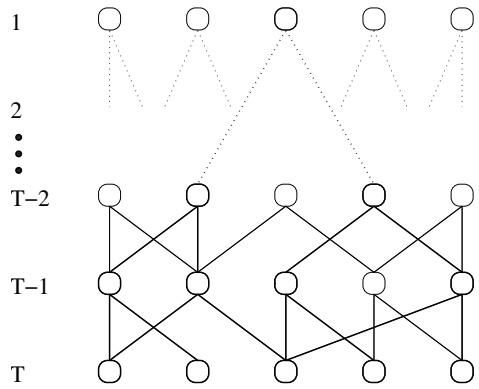


Figure 4.3. Representation of an exponential sized policy using linear space by re-using policy trees [127]

to sub-trees, which can be shared among multiple higher level trees. Using such a representation guarantees that the space requirements are linear in the parameter `maxTree`. This representation may not be sufficient to capture the optimal Dec-POMDP policy, however, it often provides accurate approximation. As the parameter `maxTree` increases, the accuracy of this representation also increases.

4.2.1.1 Objective Function

For two agents, the joint-value of the trees p of agent 1 and q of agent 2 in the starting state s can be computed recursively as follows:

$$V(p,q,s) = R(s,a_p,a_q) + \sum_{y,z}\sum_{s' \in S} O(yz \mid s', a_p, a_q) P(s'|s,a_p,a_q) V(p_y, q_z, s') \tag{4.2}$$

where p_y denotes the root of the subtree that results from a deterministic transition from the current node p upon receiving the observation y; p_z denotes the same for agent 2. The value for the initial belief point η_0 can be defined as

$$V(p,q,\eta_0) = \sum_{s \in S} \eta_0(s) \cdot V(p,q,s) \tag{4.3}$$

The goal for finite-horizon Dec-POMDP algorithms is to find the best structure for the policy trees such that the value $V(p,q,\eta_0)$ is maximized.

4.2.2 Infinite-Horizon Finite-State Controllers

In the case of infinite-horizon Dec-POMDPs, agents operate continuously with the immediate reward R being discounted by a factor $\gamma < 1$. Representing agents' local policies using an explicit tree structured representation is not feasible in this case. Therefore, agents' policies are represented as *cyclic* finite-state controllers (FSC). Such FSC based representation generalizes the idea of approximate policy trees as shown in Figure 4.3 for the infinite-horizon case.

We represent each agent's policy as a bounded, *finite state controller* (FSC). This approach has been used successfully for both POMDPs [114] and Dec-POMDPs [6]. In this case, each agent has a finite internal memory state, $q^i \in Q^i$, which summarizes the *crucial* information obtained from past observations to support efficient action selection. The size of the set Q^i determines the expressiveness of the FSC based policy. For single agent POMDPs, FSCs are beneficial due to their compactness compared to the full belief state. In Dec-POMDPs, it is the only approach to tackle effectively both finite and infinite horizon problems. There are other approaches based on nested agent beliefs such as the I-POMDP framework [50], however we do not address them in this work.

The FSC of the ith agent is parameterized by $\theta^i = (\pi^i, \lambda^i, \nu^i)$. An agent chooses actions depending on its internal state, q: $P(a|q;\pi) = \pi(a,q)$. The internal state is updated with each new observation, by the node transition function: $P(q'|q,y;\lambda) = \lambda(q',q,y)$. Finally, ν is the initial node distribution $P(q_0)$ for each agent. Figure 4.4 shows the structure of such controllers for two agents. Both the action selection parameter π and the node transition parameter λ could be deterministic or stochastic. In theory, higher values can be obtained with stochastic controllers. However, optimizing stochastic controllers involves continuous variables and achieving global optimality can be challenging. On the other hand, for deterministic controllers, the search space is discrete and one can use branch-and-bound procedure to find optimal deterministic controllers [2]. However, the search space can be quite large due to deterministic controller optimization problem being NP-Hard [93].

Furthermore, we also note that the tree based policy representation for finite-horizon Dec-POMDPs in Sec. 4.2.1 can be viewed as an *acyclic* finite-state controller. Thus, despite the apparent differences between the two policy representations for finite and infinite

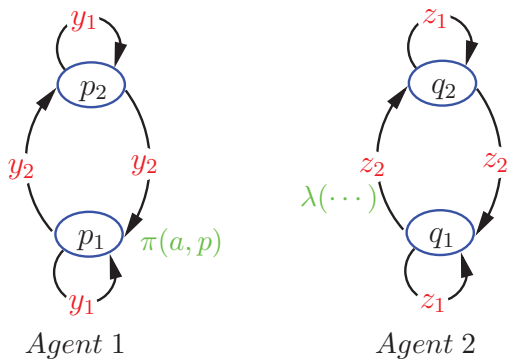


Figure 4.4. Representation of a 2-agent infinite-horizon joint-policy using finite-state controllers. Each node is a memory state. Edges represent the node transition function. Agent 1 has two observations y_1 and y_2. Agent 2 also has two observations z_1 and z_2.

horizon Dec-POMDPs, they actually belong to the same class of finite-state controllers. An implication of this fact is that most of the algorithms we develop in this thesis for infinite-horizon Dec-POMDPs can be applied to finite-horizon Dec-POMDPs with minimal change.

4.2.2.1 Objective Function

As stated earlier, we denote the controller nodes for agent 1 by p and agent 2 by q. The value for starting the controllers in nodes $\langle p, q \rangle$ at state s is given by:

$$V(p,q,s) = \sum_{a,b} \pi(a,p)\pi(b,q)\Big[R(s,a,b) + \gamma \sum_{s'} P(s' \mid s,a,b) \sum_{y,z} O(yz \mid s',a,b) \sum_{p'q'} \lambda(p',p,y)\lambda(q',q,z)V(p',q',s')\Big] \tag{4.4}$$

The goal is to set the parameters $\langle \pi, \lambda, \nu \rangle$ of the agents' controllers (of some given size) that maximize the expected discounted reward for the initial belief η_0:

$$V(\eta_0) = \sum_{p,q,s} \nu(p)\nu(q)\eta_0(s)V(p,q,s)$$

4.3 Applications

Historically, decentralized decision making has been studied extensively in operations research and control theory [117, 104, 145, 105, 144] and is still being actively pursued [28].

A classical problem in control theory has been that of *decentralized detection* by a team of sensors [144, 145], a problem whose sequential variant can be modeled as a Dec-POMDP, as shown in later chapters of the thesis. In this problem, there are multiple hypotheses on the state of the environment and each sensor receives some relevant information. In a centralized scheme, each sensor will transmit all its information to a fusion center, which then makes a decision about one of the hypotheses. However, in the decentralized variant, each sensor sends a *summary* of its observations taking values over a finite set of alphabets. In another variant of the problem, the fusion center is absent and sensor actions can be viewed as local decisions. In such a decentralized scheme, apart from the hypothesis testing, a new problem arises: how should a sensor decide which summary to send? This setting provides a definitive advantage when sensors are geographically distributed and operate in a communication limited environment. Decentralized detection has many application areas. For example, it can be used for detecting an intruder in a secure area. It can also be used in failure detection, where different sensors monitor multiple components of an equipment and must transmit small messages to a central monitoring system which then makes a decision about handling various types of failures. A further motivating application arises in the context of human-computer interaction. In this setting, multiple sensors receive detailed information about a complex system and given the limited attention of a human supervisor, must transmit a small summary based on their local observations.

Decentralized POMDPs generalize the one step decentralized decision problem to a sequential setting. The modeling advantage of Dec-POMDPs for various practical applications is that they allow multiple agents to control the underlying MDP in a decentralized manner under partial observability. Consider, for example, a decentralized target tracking problem [82, 83], where coordination among sensors is required to track mobile targets. The decentralized detection model discussed earlier, can model the single step decision problem where each hypothesis corresponds to possible locations of the targets. Noisy sensors receive observation about the targets and must decide which region to scan based on this local information such that the joint reward is maximized. When the targets are mobile, the prior distribution over the hypothesis varies with time and can be modeled using Dec-POMDPs.

Other motivating applications include multi-robot control such as coordinating the operation of planetary exploration rovers [12, 169]. In this application, rovers' tasks include exploring multiple sites on the planet, some of which can be visited by multiple rovers. Rovers periodically transmit the information collected to the ground based control or fusion center. As the communication is costly and resource constrained, rovers must coordinate the exploration effort in order to maximize the value of information. Another application, robotic team tag, is introduced by Montemerlo *et al.* [38], where a team of robots try to herd and capture a moving target. Another relevant application is the firefighting domain introduced by Oliehoek *et al.* [100], where a team of agents try to extinguish fire at a set of locations having only local observability of fire locations. Decentralized control has also been explored in developing broadcast channel protocols [102, 13]. In this case, two or more agents control a network of communication channels and try to maximize the overall throughput by avoiding message collisions. Agents only have local and partial observability of collisions.

As highlighted earlier, the modeling advantage of Dec-POMDPs comes with a price, namely their high computational complexity. Bernstein *et al.* [13] have already shown that optimally solving finite-horizon Dec-POMDPs is NEXP-complete. Optimal algorithms can only solve small problems involving two agents and small horizons [54, 140, 100], which is expected in light of the complexity result. Therefore, the objective of our work is to develop efficient approximate algorithms that can scale well w.r.t. different problem parameters. In the next few chapters, we describe a number of different techniques that can be used to accomplish this task.

4.4 Discussion

In this chapter, we formally introduced the Dec-POMDP model. We described how agents' individual policies can be represented as node-limited trees for the finite-horizon planning or as finite-state controllers for the infinite-horizon case. Different objective functions associated with such policy representations were discussed. We also discussed several potential applications of decentralized decision making and their historical connections to the field of OR and control theory. The need to develop efficient approximate algorithms

was highlighted as optimal planning even for a 2-agent Dec-POMDP has already been shown to be NEXP-Hard.

CHAPTER 5

EFFICIENT DYNAMIC PROGRAMMING FOR FINITE-HORIZON SEQUENTIAL DECISION MAKING

In this chapter, we examine the problem of *finite-horizon* multiagent planning, where agents operate over a finite number of time steps without any discounting of the reward they receive. Computing optimal policies under this framework even for *two* agents is NEXP-Hard [13]. Therefore, approximation algorithms are commonly used. Our main contributions lie in the analysis of the complexity and the development of optimal algorithms for the decentralized backup problem, which is a computational bottleneck in several dynamic programming approximate algorithms. We show that the decentralized backup problem is NP-Hard. Despite this negative result, we present an efficient and scalable optimal algorithm by exploiting the structure of the backup problem using *constraint networks*—a subclass of graphical models. We reformulate the backup problem as a *weighted constraint satisfaction* (WCSP) problem. We analyze several different multiagent settings, which include two agent systems and larger multiagent systems, and present different WCSP formulations suited for their particular characteristics.

Our results show that bringing the perspective of constraint optimization helps solve the backup problem more than an order-of-magnitude faster than the state-of-the-art solver PBIP [36]. We also investigate the backup problem in restricted sub-classes of the Dec-POMDP model, such as the ND-POMDP model, that can capture larger multiagent systems. We again show how to solve this problem efficiently using WCSP solvers. This approach provides magnitudes of speedup in the policy computation and generates better quality solution for all test instances than previous best approaches [146, 88, 70]. Our approach is one of the first that can scale well to problems with a few dozens of agents.

5.1 Approximate Dynamic Programming for Dec-POMDPs

The intractability of optimal single agent POMDP algorithms can be attributed to planning for the complete belief space. A similar barrier exists for planning in the multiagent setting. The idea of planning for a finite set of belief points has lead to several successful point-based POMDP algorithms in recent years such as PBVI [112] and Perseus [135] among others. All these algorithms compute a set of reachable belief points using a heuristic such as the underlying MDP policy or using randomized trajectories [135]. The policy is computed over these beliefs by performing a sequence of point-based backups starting from the last time step. Before we describe how such backups are performed in the context of Dec-POMDPs, we make some observations about the belief space in Dec-POMDPs.

In a single agent POMDP, all the relevant information from the previous actions and observations can be encoded succinctly as a belief over the environment states. This belief about the environment state can also be shown to be a sufficient statistic for a POMDP [113]. If η_t represents the belief of the agent at time step t and the agent takes an action a_t and receives an observation y_{t+1}, then the next belief can be computed using Bayes rule as follows:

$$\eta_{t+1}(s') = \alpha \sum_{s \in S} \eta_t(s) P(s' \mid s, a_t) O(y_{t+1} \mid s', a_t) \tag{5.1}$$

where α is a normalization constant.

Interestingly, in a multiagent setting, such maintaining of the underlying belief about the environment state is not possible. The reason is that agents perceive only their local observations. They do not observe the actions or local observations of other agents directly. This makes computing and updating beliefs during *execution* time infeasible. However, during the planning phase when a central planner is computing the policies for the agents, it is indeed possible to simulate belief updating. We call such a belief state the *centralized* belief state. In a multiagent setting, a belief state during the planning stage will always imply centralized belief state. This is indeed an approximation as such centralized beliefs cannot be maintained by agents during execution time. However, experimentally, it has

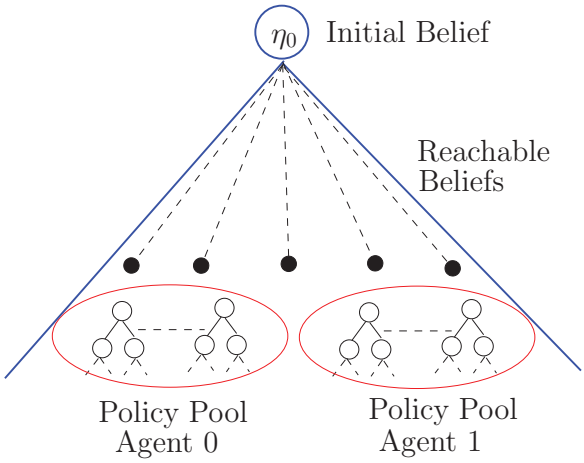


Figure 5.1. Bottom up dynamic programming framework for 2-agent Dec-POMDPs

been shown to work well for a variety of problems [127, 126, 23]. Such beliefs can be analogously updated during the planning phase as follows:

$$\eta_{t+1}(s') = \alpha \sum_{s \in S} \eta_t(s) P(s' \mid s, a, b) O(y_{t+1} z_{t+1} \mid s', a_t, b_t) \tag{5.2}$$

We now provide a brief overview of point-based dynamic programming algorithms for Dec-POMDPs. Such approaches include the following steps:

1. Compute a belief selection heuristic
2. Select a number of belief points for each time step $t = 0$ to T
3. Construct the policy tree bottom up for each agent (see Figure 4.3 for a policy tree example) by performing point-based backups starting from the last time step.

Figure 5.1 shows a high level outline for point-based dynamic programming algorithms. The belief selection heuristic usually involves solving the underlying MDP or the POMDP for the given Dec-POMDP problem. A number of belief points can be sampled for each time step by simulating plan execution using the belief selection heuristic. The critical step, which is the bottleneck of dynamic programming based algorithms, is that of performing decentralized backup in the step 3. We show that performing this backup is indeed NP-Hard even for 2-agents. Despite this negative result, in the next few sections, we present efficient

optimal algorithms for the decentralized backup operation by leveraging techniques from constraint optimization.

5.2 Related Work

As the modeling advantage of Dec-POMDPs comes with the price of NEXP-Hard complexity, most of the recent research has focused on approximate algorithms, of which point-based approaches have shown great promise in scaling up and finding good quality policies [127]. As in single agent POMDP point-based approaches [135, 112], the key idea is to compute a set of reachable beliefs using a heuristic and compute the value function by performing a sequence of point-based backups. However, the backup technique differs fundamentally in the multi-agent setting. In POMDPs, backups can be performed efficiently. However, due to the decentralized nature of a Dec-POMDP policy, naively performing such backups requires *exponential* effort in the number of observations [127].

In the past, there has been substantial research on solving this problem more efficiently because point-based backups constitute the core of any point-based algorithm. For example, the IMBDP algorithm [126] avoids exponential blowup by limiting the full backup to a fixed number of observations which are most likely to occur at the given belief point. In the MBDP-OC algorithm [23], an information-theoretic criterion is used to merge observations into sets, while minimizing the total loss in solution quality. Dibangoye *et al.* [36] take a different approach to solving this problem optimally by using a branch-and-bound search. While in the worst case, the complexity remains the same, their approach is significantly faster than previous approaches in practice. Amato *at al.* [5] further improve the scalability of the previous algorithm by limiting the possible next step sub-policies using state reachability analysis. There has been some work on this problem in the context of collaborative Bayesian games and a search algorithm has been developed to solve such games [101]. It can be shown that such Bayesian games are equivalent to the decentralized backup problem. Linear programming based approximate algorithms for solving the backup problem have been developed in [158].

All the above approaches are limited to two agent systems. Scaling even such approximate schemes to larger multiagent systems without additional assumptions about the struc-

ture of interaction among agents is infeasible due to exponential computational requirements in the number of agents. Therefore, an emerging approach to improve scalability w.r.t. the number of agents has been to consider restricted forms of interaction that arise frequently in practice [9, 99]. In particular, ND-POMDP [99] is one such general model which is inspired by a realistic sensor network coordination problem [83]. The key assumption in ND-POMDP is that of transition and observation independence and the *locality of interaction*, which help construct efficient algorithms that exploit the independence among the agents.

A rich portfolio of algorithms has been developed for solving ND-POMDPs, featuring locally optimal policy search [99], approximation schemes [146, 88] and a globally optimal algorithm [99]. Most of these algorithms are based on policy search. This thesis presents the first bottom up dynamic programming (DP) algorithm for ND-POMDPs. The advantage of the DP approach lies in its ability to focus planning on the reachable part of the belief space using a forward heuristic search. As also highlighted earlier, such approximate DP algorithms–referred to as *point-based dynamic programming* techniques–have shown great success in solving POMDPs. The algorithm we develop, *Constraint-Based Dynamic Programing* (CBDP), to solve the ND-POMDP model shares its motivation with such point-based approaches.

Next, we discuss the decentralized backup operation, the core of dynamic programming approaches for Dec-POMDPs and establish its computational complexity.

5.3 Decentralized Backup—Two Agents

We first consider the decentralized backup operation for the two agents case. We denote using p and q, the t-step policies for agent 1 and 2 respectively. Recall from Section 4.2.1.1 that the joint-value of the policy tree p of agent 1 and q of agent 2 in the starting state s can be computed recursively as follows:

$$V(p, q, s) = R(s, a_p, a_q) + \sum_{y,z} \sum_{s' \in S} O(yz \mid s', a_p, a_q) P(s'|s, a_p, a_q) V(p_y, q_z, s') \tag{5.3}$$

Therefore, the value for the belief point η_t can be defined as:

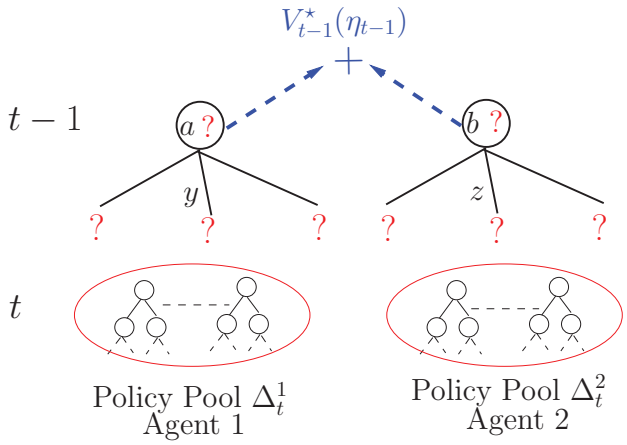


Figure 5.2. Schematic representation of decentralized backup operation for 2-agents. Optimization variables are denoted using the placeholder '?'.

$$V(p, q, \eta_t) = \sum_{s \in S} \eta_t(s) \cdot V(p, q, s) \tag{5.4}$$

During the bottom up dynamic programming step, a fixed number of policies are retained for each time step per agent. Let us denote the policy pool for agent 1 at time step t by Δ^1_t and for agent 2 by Δ^2_t. In the decentralized backup step, we are interested in building the best $(t+1)$-step policy for a given belief η_{t-1} and a pool of t-step policies Δ^1_t and Δ^2_t for agent 1 and 2 respectively. Figure 5.2 shows a schematic representation of this decentralized backup step.

Definition 9. *Let δ_a denote a decision rule $\delta_a : Y \to \Delta^1_t$ for each action a of agent 1 and δ_b denote the same for agent 2, $\delta_b : Z \to \Delta^2_t$. The decentralized backup step consists of finding the best root action a^*, b^* and the best policy mapping δ_{a^*}, δ_{b^*} for agents 1 and 2 that maximize the following objective function:*

$$\begin{aligned} V^{\star}_{t-1}(\eta_{t-1}) \quad = \max_{a,b,\delta_a,\delta_b} \Big[& \textstyle\sum_{s \in S} R(s,a,b)\eta_{t-1}(s) + \sum_{yz} \sum_{s,s'} O(yz \mid s', a, b) \\ & P(s'|s,a,b) V\big(\delta_a(y), \delta_z(z), s'\big) \eta_{t-1}(s) \Big] \end{aligned} \tag{5.5}$$

Intuitively, the solution of the above optimization problem constructs the optimal $(t+1)$-step policy tree for each agent for the belief η_{t-1}. The actions a^* and b^* replace the placeholder '?' at the root of the policy trees in Figure 5.2 and the policy mapping δ_{a^*}, δ_{b^*}

specify the best next-step policy that will be attached to the appropriate edge. The above optimization problem can be derived by noting that the value of the best $(t+1)$-step policy for belief η_{t-1} can be written as follows:

$$V^{\star}_{t-1}(\eta_{t-1}) = \max_{a,b,\delta_a,\delta_b} \Bigg[\sum_{s \in S} R(s,a,b)\eta_{t-1}(s) + \sum_{y,z} P(yz|a,b,\eta_{t-1}) V\big(\delta_a(y), \delta_b(z), \tau(\eta_{t-1},a,b,y,z)\big) \Bigg] \tag{5.6}$$

where $\tau(\eta_{t-1}, a, b, y, z)$ is the updated belief after agents take the joint action $\langle a, b \rangle$ and receive observations $\langle y, z \rangle$ with current belief being η_{t-1}. It can be calculated in a straightforward manner using Bayes rule as shown in Eq. (5.2). Upon substituting the value of $V(\delta_a(y), \delta_b(z), \tau(\cdot))$ by using Eq. (5.4), we get the optimization problem in definition 9.

The decentralized backup optimization problem does not admit efficient solutions and, as we show next, it is NP-Complete to find the optimal solution. Intuitively, the reason is that solving the above equation requires *optimization over functions* δ for each agent and the space of all possible mappings δ is exponential in the number of observations.

5.3.1 Complexity of Decentralized Backup

Before establishing the complexity, we introduce the NP-Complete team decision problem (TDP) [145, 28]. In TDP, there are two decision makers, each of which observes some local component of the system state. That is, if $(o_1, o_2) \in O_1 \times O_2$ is a system state, then agent 1 observes only o_1 and agent 2 observes o_2. The goal of the agents is to choose an action $(u_1, u_2) \in U_1 \times U_2$ which maximizes the joint reward $r(o_1, o_2, u_1, u_2)$ based only upon their local observation. Stated formally, the goal is to find decision rules $\gamma_1 : O_1 \to U_1$ and $\gamma_2 : O_2 \to U_2$ that maximize the expected reward, assuming each state is equally likely. That is,

$$\max_{\gamma_1,\gamma_2} \sum_{o_1,o_2} r(o_1, o_2, \gamma_1(o_1), \gamma_2(o_2))$$

Theorem 2. *Solving the decentralized backup problem optimally is NP-Complete.*

Proof. We reduce the TDP to a two agent Dec-POMDP $\mathcal{M}$ where taking one step backup is equivalent to solving the TDP. It is clear that the backup problem is in NP, as given a policy mapping δ_i, we can evaluate the R.H.S. of Eq. (5.5) in polynomial time.

There are $1 + |O_1||O_2|$ states in $\mathcal{M}$. Let s be the initial state s.t. $\eta_0(s) = 1$. The rest of the system states are denoted by $s_{o_1 o_2} \forall (o_1, o_2) \in O_1 \times O_2$. The observation sets of the agents are: $Y = O_1$ and $Z = O_2$. The action space is defined as: $A = U_1$ and $B = U_2$. The state s transitions with equal probability to each state $s_{o_1 o_2}$ regardless of the action taken and $P(s|s) = 0$. The observation probabilities are also independent of the action taken and observations identify the resulting state deterministically. That is, $O(yz \mid s_{o_1 o_2}) = 1$ if $o_1 = y$ and $o_2 = z$, else it is zero. The reward for any joint action in the initial state s is 0 and the horizon for the problem is 2. The rest of the reward function will be defined shortly. First, note that the backup Eq. 5.5 can be written as:

$$V_t(\eta_0) \quad = \quad \max_{\delta_1, \delta_2} \left[\sum_{y,z} \sum_{s_{o_1 o_2}} P(s_{o_1 o_2}|s) O(yz \mid s_{o_1 o_2}) V(\delta_1(y), \delta_2(z), s_{o_1 o_2}) \right]$$

In the above equation, we used the fact that $R(s, \cdot) = 0$ by the problem construction and that any action can be taken at the initial belief η_0 as state transition and observation probabilities are independent of actions. As the problem horizon is 2, $V(\cdot, \cdot, \cdot)$ is simply the immediate reward R. Since observations identify the resulting state deterministically and $P(s_{o_1 o_2}|s) = 1/|Y||Z|$, the equation can be further simplified as follows:

$$V_t(\eta_0) = \frac{1}{|Y||Z|} \max_{\delta_1, \delta_2} \left[\sum_{o_1, o_2} R(s_{o_1 o_2}, \delta_1(o_1), \delta_2(o_2)) \right] \tag{5.7}$$

Now, we set the rest of the reward function such that

$$R(s_{o_1 o_2}, a, b) = r(y_{o_1}, z_{o_2}, u_a, u_b) \forall s_{o_1 o_2} \in O_1 \times O_2$$

where y_{o_1}, z_{o_2}, and u_a and u_b are the counterparts of the Dec-POMDP in the given TDP instance. Clearly, the reduction from TDP to the Dec-POMDP has polynomial complexity.

Finally, we can show that there exists a solution to the TDP problem providing total reward W if and only if there exists a solution to the backup problem achieving a reward of

$W/(|Y||Z|)$. This is straightforward to see as the functions γ_i from the TDP instance map directly to δ_i and vice-versa. The reward function for the Dec-POMDP is the same as the reward function for the TDP. □

Next, we develop new algorithms for solving the decentralized backup problem efficiently and in a principled way.

5.3.2 Optimal Backup Approach Using WCSPs

The optimal algorithm for the decentralized backup leverages recent advances in the weighted constraint satisfaction (WCSP) literature by reformulating the backup problem as finding the least cost solution of a WCSP instance. We briefly introduce the weighted constraint satisfaction problem below; further details can be found in [31].

A WCSP is defined by the tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, k \rangle$ where:

- $\mathcal{X} = \{X_1, \ldots, X_n\}$ is a set of variables.
- $\mathcal{D}$ is a set of domains D_i for each variable X_i. D_i is discrete and denotes all possible assignments to the variable x_i.
- $\mathcal{C}$ is a set of constraints.
- Each constraint C_S is defined over a subset of variables $S \subseteq \mathcal{X}$ and maps the tuples corresponding to assignments on S to a real valued cost. For example, $C_{ij} : D_i \times D_j \rightarrow [0 \ldots k]$, where C_{ij} is a constraint over variables X_i and X_j and k denotes the upper bound on the cost of any assignment tuple. When a constraint assigns a cost k to any assignment, it implies that the assignment is forbidden, else it is allowed with the corresponding cost.

The goal is to find the complete assignment X to variables such that the global cost is minimized. $X[S]$ represents the projection of tuple X over variables in S. The total cost is

$$\sum_{C_S \in \mathcal{C}} C_S(X[S]).$$

We now describe the reformulation of the backup problem as a WCSP instance. First, we note that for each joint action $\langle a, b\rangle$, the optimality equation (Eq. 5.5) can be written as follows:

$$V_{t-1}^{ab}(\eta_{t-1}) = \sum_{s\in S} R(s,a,b)\eta_{t-1}(s) + \max_{\delta_a,\delta_b}\Bigg[\sum_{yz}\sum_{s,s'} O(yz \mid s',a,b) P(s'|s,a,b)V\big(\delta_a(y),\delta_b(z),s'\big)\eta_{t-1}(s)\Bigg] \tag{5.8}$$

If we solve the above equation optimally for every joint-action, then finding $V^{\star}_{t-1}(\eta_{t-1})$ is easy as it requires iterating over all the joint actions, which has polynomial complexity for two agents. In the above equation, only the second summation depends on the decision rule δ_i. Therefore, the optimization problem becomes:

$$\max_{\delta_a,\delta_b}\Bigg[\sum_{yz}\sum_{s,s'} O(yz \mid s',a,b)P(s'|s,a,b)V\big(\delta_a(y),\delta_b(z),s'\big)\eta_{t-1}(s)\Bigg] \tag{5.9}$$

We seek to optimize the above equation by reformulating it as a WCSP for each joint action $\langle a, b\rangle$. The WCSP parameters are detailed below.

Definition 10. *The WCSP for the decentralized backup problem* (5.9) *is defined by the tuple* $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$—

- $\mathcal{X}$ ***Variables:*** *One variable is created for each observation of each agent. For example, if* $y_i \in Y$ *is a possible observation for agent* 1*, then variable* X_{y_i} *is created.*
- $\mathcal{D}$ ***Domain:*** *The domain of all the variables corresponding to an agent is the set of all next step sub-policies available for that agent. For example, if* $\mathcal{P}$ *denotes the set of sub-policies for agent* 1*, then* $D_{y_i} = \{p \mid p \in \mathcal{P}\}\ \forall y_i \in Y$*.*
- $\mathcal{C}$ ***Constraints:*** *A constraint is created for every pair of observations from agent* 1 *and* 2*. That is, we create a binary constraint* $C_{i,j}$ *between variables* X_{y_i} *and* X_{z_j} *for every pair of observations* $y_i \in Y$ *and* $z_j \in Z$*.*

C ***Valuations:*** *The valuation for each constraint $C_{i,j} \in \mathcal{C}$ is defined as in Eq.* (5.9).

$$C_{i,j}(p,q) \quad = \quad \alpha - \sum_{s,s'} O(y_i z_j \mid s', a, b) P(s'|s,a,b) V\big(p,q,s'\big) \eta_{t-1}(s)$$

where α is a large positive constant which is used to transform the maximization objective of Eq. (5.9) *to minimizing the WCSP cost. Intuitively, the second part of the above equation (summation over states) represents the value accrued when agent* 1*, upon receiving observation y_i, follows the sub-policy p and agent* 2*, upon receiving observation z_j, follows the sub-policy q.*

We can visualize this WCSP by using its primal graph. The primal graph of a WCSP with binary constraints is a graph whose nodes are the variables of the problem, and each edge connects a pair of variables that occur together in the scope of a constraint function. Figure 5.3 shows the primal graph of the WCSP for a backup instance when each of the two agents has three observations. Each edge represents a constraint.

Theorem 3. *Minimizing the objective function of the WCSP created in definition 10, $\sum_{C_{i,j}} C_{i,j}$, is equivalent to solving the backup problem* (5.9).

Proof. It is easy to see that minimizing the objective function of the WCSP, $\sum_{C_{i,j}} C_{i,j}$, is equivalent to maximizing Eq. (5.9). The complete assignment X represents the decision rules δ_i for each agent and the optimal assignment solves Eq. (5.9). The global cost of assignment X is given by:

$$\sum_{C_{i,j} \in \mathcal{C}} C_{i,j} = |Y||Z|\alpha - \sum_{y_i, z_j} \sum_{s,s'} O(y_i z_j | s', a, b) P(s'|s,a,b) V(X[X_{y_i}], X[X_{z_j}], s') \eta_{t-1}(s) \quad (5.10)$$

Therefore, if we minimize $\sum_{C_{i,j}} C_{i,j}$, then the second part of the above equation gets maximized, which solves the backup problem. The first part of the above equation, $|Y||Z|\alpha$, is a constant w.r.t. the backup problem. □

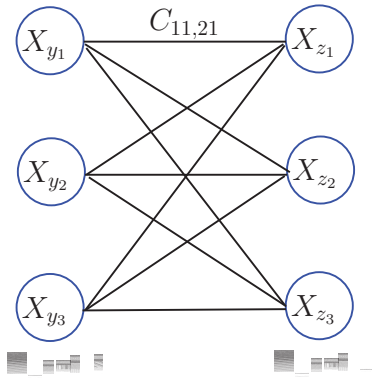


Figure 5.3. Primal graph of a WCSP for the backup problem. Each agent has three observations

5.3.2.1 Solving the Backup WCSP

Unfortunately, optimally solving a WCSP is NP-Hard. However, as constraint reasoning has numerous practical applications, many algorithms exist which can solve them efficiently either using dynamic programming or search techniques. It has been shown that if the primal graph of a WCSP has bounded tree-width, then the WCSP can be solved efficiently using dynamic programming with the bucket elimination algorithm [33]. However, the WCSP instance for the backup problem is a complete bipartite graph as shown in Figure 5.3. For such graphs, the tree-width is $O\big(\max(|Y|, |Z|)\big)$ regardless of the variable ordering used. Since the bucket elimination algorithm has complexity exponential in the tree-width, it cannot scale well with the number observations. Therefore, we used a search-based solver, AOBB [89], which uses a heuristic function to prune a large part of the search space. The heuristic function provides a lower bound for the WCSP at each step of the search. We used the state-of-the-art heuristic *existential directional arc consistency* (EDAC) [31]. We tried other heuristics too, such as the mini buckets [89], however EDAC outperformed the others by a significant margin, sometimes expanding an order of magnitude less nodes. Next, we explain key differences between our approach and the previous optimal approach PBIP [36].

5.3.2.2 Comparison with PBIP

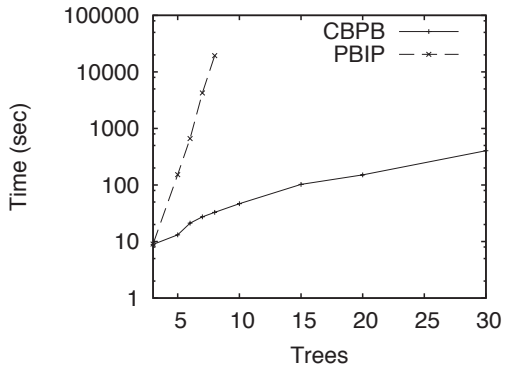
While point-based incremental pruning (PBIP) is also a search algorithm [36], our approach is fundamentally different. The key differences lie in how the search process is

structured and how the heuristic function is computed. In PBIP, a node represents a partial joint-policy tree, where the sub-policies that agents will follow are only specified for *some* joint observations and not for others. Expanding a fringe node requires selecting an unspecified joint observation and generating all possible successors by attaching all possible subtrees to this joint-observation, which number $\mathsf{MaxTree}^2$, assuming that the size of the policy pool for each agent is $\mathsf{MaxTree}$. This is one aspect where our formulation differs from PBIP. In our approach, the search *does not* take place over joint observations, but over individual observations of an agent (see Figure 5.3). Consequently, the number of successors of a node is only $\mathsf{MaxTree}$. The depth of the search tree in our case is $2|Y|$, whereas in PBIP it is $|Y|$. However, the worst-case complexity remains the same because the branching factor in our case is $\mathsf{MaxTree}$ and in PBIP, it is $\mathsf{MaxTree}^2$. By searching over the space of joint observations, PBIP loses the structure present in the backup problem whereas our approach explicitly represents this structure using a constraint graph (Figure 5.3). The heuristic function we use, EDAC, explicitly utilizes this constraint graph and produces much tighter bounds than PBIP.

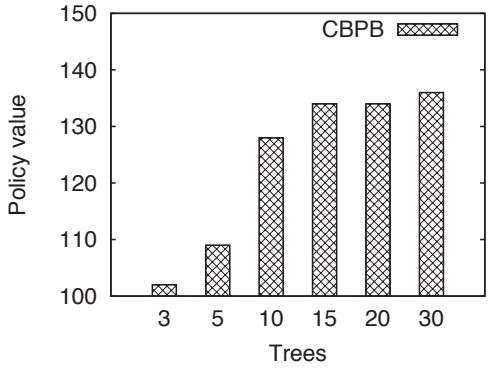
5.3.3 Experiments

We incorporated our backup algorithm into the point-based solver MBDP [127], which provides the foundation for most approximate solvers for Dec-POMDPs. We compared our WCSP-based optimal algorithm named *Constraint based point backup* (CBPB) with the best existing approach PBIP [36], which also solves the backup problem optimally and is built upon MBDP. We used the latest version of PBIP that uses incremental policy generation [5] to further enhance its performance. Experiments were performed on a Linux machine with 2GB RAM and 2.6GHz CPU. We used two of the largest Dec-POMDP benchmarks: the box pushing problem [126] and the stochastic Mars rover [5]. Each data point is an average over 10 runs.

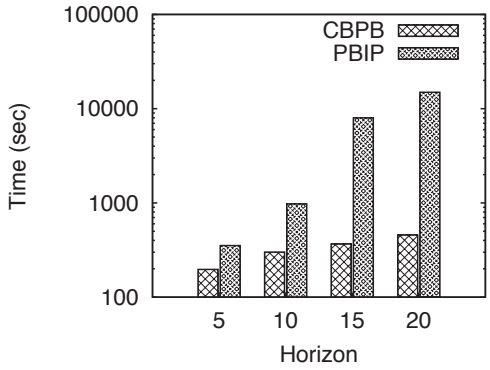
The aim of our experiments is threefold. First, we demonstrate the computational advantages of CBPB over the PBIP algorithm in terms of execution time and the ability to increase the number of belief points – CBPB provides more than 2 orders of magnitude of speedup for some settings and is about an order of magnitude faster on average, and can



(a) Box pushing: Time comparisons



(b) Box pushing: Solution quality



(c) Mars rover: Time comparisons

Figure 5.4. Comparison of our approach CBPB with PBIP

increase the number of belief points (the MaxTree parameter) by more than a factor of 3. As we compute one joint-policy for each belief point, the size of the policy pool for each agent is also MaxTree. Second, we show the scalability of CBPB by explicitly comparing the actual time required for performing all the backups versus the total execution time. We show that as we increase the MaxTree parameter, the bottleneck becomes evaluating the joint-policies. CBPB can still solve the larger search problem without considerable overhead expanding sub-hundred nodes on average.

Figure 5.4(a) shows a comparison of CBPB with PBIP on the Box pushing domain ($|S| = 100,\ |A| = 4,\ |Y| = 5$) with varying MaxTree values for horizon 10. Clearly, CBPB provides significant savings over PBIP, whose time requirements become excessive very quickly upon increasing the number of belief points. For MaxTree $= 8$, CBPB is about 500 times faster than PBIP, which takes over $19,000$ sec, whereas CBPB takes only 32

sec. Furthermore, CBPB can scale up the number of belief points to 30 whereas PBIP cannot scale above 8 belief points. Figure 5.4(b) shows how solution quality is impacted by increasing the number of belief points. Overall, when MaxTree is increased from 3 to 30, solution quality does increase significantly from 102 to 135. However, the rate of increase with belief points is slow, which indicates that the number of beliefs has to be increased further to gain higher solution quality. This observation also favors CBPB, which can potentially increase this parameter further as highlighted in the next part of the experiments.

Figure 5.4(c) shows time comparisons with PBIP on one of the largest DEC-POMDP domains – stochastic Mars rover ($|S| = 256$, $|A| = 6$, $|Y| = 8$) with varying horizon and MaxTree = 3. Again, we observe that as the horizon is increased from 5 to 20, the speedup provided by CBPB increases significantly. At horizon 20, CBPB is about 33 times faster than PBIP. For Mars rover too, CBPB can increase the belief points from 3 to 10 as shown in Table 5.1. The runtime of CBPB does increases, but we will show later that this increase is largely due to the overhead of evaluating joint-policies in MBDP. The actual search time remains a tiny fraction of the total time. Solution quality increases as well by increasing MaxTree. For horizon 20, with MaxTree = 3 the best policy value is 37.8; with MaxTree = 10, it increases to 43.6. For this domain, we could not increase MaxTree further, as simply storing and evaluating all joint policies would exceed the system RAM of 2GB.

For the next set of experiments, we emphasize the scalability of CBPB by explicitly comparing the actual time required for performing all the backups versus the total execution time (all time units in sec). Table 5.2 shows the total search time (time required by the WCSP solver) over all point-based backups and the total execution time of CBPB for box pushing for horizon 10. Clearly, even with increased MaxTree, the total search time remains a small fraction of the total time. This is because the average number of nodes expanded by CBPB for each backup (shown in the last column) remains below 100 for each setting. Table 5.3 shows similar results for the Mars rover domain with MaxTree=10. The overall execution time for CBPB is relatively high compared to box pushing as the rover domain's state and observation space is much larger than box pushing and evaluating joint-policies is much more expensive. But, notice that performing backups requires only a tiny fraction of the total time. Nodes expanded per backup instance also remain low. These results are

MaxTree \ Horizon	5	10	15	20
3	197.2	301.8	363.4	457
10	276.6	1250.3	2000.6	2729

Table 5.1. Mars Rover: Execution time

MaxTree	Search Time	Total Time	Nodes Exp.
5	2.4 (18.3%)	13.1	27.6
10	6.1 (13.1%)	46.7	34.8
15	15.9 (15.5%)	102.5	39.5
20	24.6 (16.3%)	151.1	80.5
30	60 (14.8%)	404.2	82.1

Table 5.2. Box pushing: Search time Vs. Total time

a further testimony to the accuracy of the EDAC heuristic and show that if the memory limitations of MBDP are overcome (using more efficient data structures), then CBPB can scale to large numbers of belief points.

To summarize, we have shown that performing decentralized backup in a 2-agent setting is NP-Hard. Despite this negative result, we presented an efficient and scalable technique by leveraging the connection of decentralized backup with WCSPs. Empirically, we show that our approach called CBPB is more than an order-of-magnitude faster than the previous state-of-the-art algorithm for solving the backup problem. In the next section, we address challenges that arise when there are more than two agents in the context of decentralized backup.

5.4 Decentralized Backup—Multiple Agents

The decentralized backup problem presents several new challenges in the context of multiple ($\gg 2$) agents. Let us assume that the size of available next-step policy pool for each agent is $O(\mathsf{MaxTree})$, where $\mathsf{MaxTree}$ denotes the number of sampled beliefs per time step. The two main challenges in the context of large multiagent systems are as following:

1. Assume for simplicity that the number of observations for each agent is small enough such that the space of all possible policy mappings, δ_{a^i}, for each agent i is manageable. That is, the total number of backed up policies for the time step $(t-1)$,

Horizon	Search Time	Total Time	Nodes Exp.
5	8.1 (2.9%)	276.6	37.4
10	22 (1.8%)	1250.3	55.3
15	46.2 (2.3%)	2000.6	91.4
20	68.6(2.5%)	2729	94.5

Table 5.3. Mars Rover: Search time Vs. Total time

$O(|A^i|\mathsf{MaxTree}^{|Y^i|})$, is relatively small. This is possible in scenarios when the number of observations per agent, Y^i, is small. To find the best joint-policy for a given belief, the search space is still exponential in the number of agents N, which is a bottleneck for DP based algorithms. The combinatorial explosion is w.r.t. the number of agents.

2. In scenarios when the number of observations per agent are large, then enumerating the space of all possible policy mappings, δ_{a^i}, is not feasible. In this case, the combinatorial explosion is w.r.t. both the number of agents and the size of the observation space of each agent.

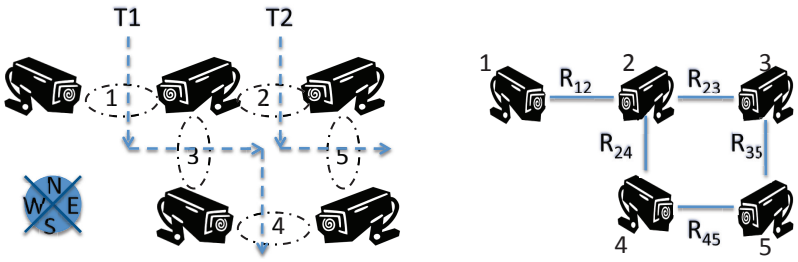
In this chapter, we present a constraint-based formulations for both these cases, which can be solved using WCSP algorithms. In order to provide a mapping between the decentralized backup and a WCSP, we need additional assumptions over the structure of the joint-policy value. These additional conditions are indeed satisfied in several existing subclasses of Dec-POMDPs such as ND-POMDPs [99] and TD-POMDPs [156]. We briefly introduce the concept of *value factorization*, which will be discussed in more detail in Section 7.2. Let us assume that the state-space S is factored s.t. $S{=}(s^1 \times \ldots \times S^M)$, which is true in several multiagent planning models such as ND-POMDPs [99] and TD-POMDPs [156].

Definition 11. *A **value factor** f defines a subset $A_f \subseteq \{1,..,N\}$ of agents and a subset $S_f \subseteq \{1,..,M\}$ of state variables.*

Definition 12. *A multiagent planning problem satisfies **value factorization** if the joint-policy value for a belief η can be decomposed into a sum over value factors:*

$$V\big(\boldsymbol{\theta}^{1:N}, \eta\big) = \sum_{f \in F} \sum_{s^f \in S^f} \eta(s^f) V_f\big(\theta^f, s^f\big) \ , \tag{5.11}$$

where F is a set of value factors, $\theta^f \equiv \theta^{Ag_f}$ is the joint-policy of agents of factor f, and $s^f \equiv s^{S_f}$ is the collection of state variables of this factor and $\eta(s^f)$ represents the belief over states in factor f.



(a) Camera sensor network (b) Interaction graph

Figure 5.5. Targets T1 and T2 follow dotted trajectories.

Next, we present an example domain where the value factorization property is satisfied.

Example 1. *The value factorization property holds in a sensor network based domain, where the goal for the agents is to track targets, as shown in Figure 5.5. This example includes a sensor network with* 5 *camera sensors (or agents). They can scan in four directions. Their task is to continuously track two targets, T1 and T2, which move along their dotted trajectories in a stochastic manner.*

To track a target and earn the associated reward, sensors must coordinate to simultaneously scan the location where the target is present. If sensors scan in an uncoordinated manner or scan a location where no target is present, a penalty is given. Furthermore, sensors are noisy, allowing for false positives and negatives in the observations sensors receive after each scan action. Each target has its own independent Markovian transition function. There is no central controller that knows what each sensor is observing. Sensors must act based upon their individual observations.

5.4.1 WCSP Formulation—Case 1

In this section, we present the WCSP formulation of the decentralized backup problem for the case when the observation space of each agent is small. Based on the value factorization property, we can visualize the dependencies among agents using an *interaction hypergraph*. In this hypergraph, there is a node for each of the N agents. There is an (hyper)edge for each value factor f that connects all the agents that are involved in that value factor.

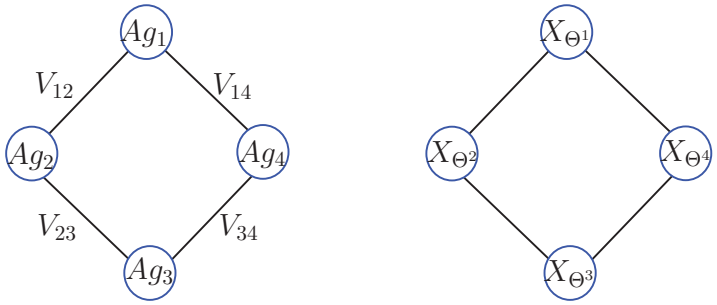


Figure 5.6. An example of a pairwise interaction graph (left) and its corresponding WCSP primal graph (right)

Example 2. *Consider the decomposition of the joint-value function for* 4 *agents as follows:*

$$V\big(\boldsymbol{\theta}^{1:4}, \eta\big) = V_{12}(\theta^1, \theta^2, \cdot) + V_{23}(\theta^2, \theta^3, \cdot) + V_{34}(\theta^3, \theta^4, \cdot) + V_{14}(\theta^1, \theta^4, \cdot)$$

The interaction graph for this instance is shown in Figure 5.6. Nodes represent agents and edges represent value function components.

Let Θ^i denote the set of $(t+1)$-step backed up policies for each agent i. Based on joint value factorization property of Eq. (5.11), we can write the best value for the belief η_{t-1} as:

$$V^{\star}_{t-1}(\eta_{t-1}) = \max_{\theta^1 \in \Theta^1, \ldots, \theta^N \in \Theta^N} \sum_{f \in F} \sum_{s^f \in S^f} \eta(s^f) V_f\big(\theta^f, s^f\big) \tag{5.12}$$

A naive algorithm to solve the above equation that enumerates over all θ^is will require exponential effort in the number of agents. However, we can leverage the joint-value factorization property to develop a WCSP based formulation that can be solved much more efficiently, as also shown in Section 5.3.2. In the WCSP formulation, there is a policy variable X_{Θ^i} for each agent i. The domain $D(X_{\Theta^i}) = \{\theta^i \mid \theta^i \in \Theta^i\}$ is the set of all backed up policies of agent i. Variables X_{Θ^i} and X_{Θ^j} are connected by an edge if their corresponding agents i and j share a value factor f. The structure of this graph is similar to the interaction graph, except that the agents are replaced by their corresponding policy variables. A constraint is defined for each edge. The valuation for the edge associated with value factor f is:

$$C_f(\theta^f) = \sum_{s^f \in S^f} \eta(s^f) V_f\big(\theta^f, s^f\big)$$

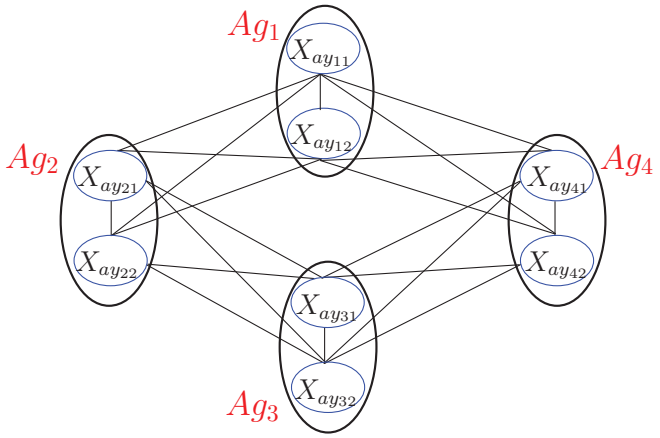


Figure 5.7. WCSP formulation (Case 2) for 4 agents with the interaction structure of Figure 5.6(a). Each agent has 2 observations.

For example, consider the interaction structure of Figure 5.6(a). A constraint is defined for each of the 4 links and the resulting WCSP's primal graph is shown in Figure 5.6(b).

The optimization problem is to find the complete assignment $\theta^\star$ to the variables which maximizes the sum $\sum_f C_f$, which is also the best joint-policy for the particular belief. Using the bucket elimination algorithm, this problem can be solved in $O\big(N(|A|\mathsf{MaxTree}^{|Y|})^d\big)$ time and space where d is the induced tree-width of the interaction graph, $|Y|$ denotes the size of the largest observation set for any agent and N denotes the number of agents. Thus, we have reduced the complexity from being exponential in the number of agents to being exponential in the induced width, which typically will be much smaller and linear in the number of agents. We can also use a branch-and-bound algorithm such as AOBB [89] that can prove to be more efficient when the induced tree-width is high.

5.4.2 WCSP Formulation—Case 2

When the observation space of agents is large, then the previous approach of searching within the space of backed up policies is infeasible, as number of backed up policies, $O\big(|A|\mathsf{MaxTree}^{|Y|}\big)$, is prohibitively large. Therefore, we need a more sophisticated approach that can implicitly search within the space of backed up policies. To address this, we extend the approach of Section 5.3.2. The main reason that the previous approach of Section 5.3.2 cannot be applied directly to large multiagent systems is that the previous approach solves

one WCSP for each joint-action. The number of joint-actions for large multiagent systems is exponential in the number of agents. Therefore, a straightforward adoption of the previous approach is not computationally tractable. We next a present a new wcsp formulation to handle these issues.

Often, the joint-value factorization property of Eq. (5.11) leads to factorization of the value function for a particular belief η_{t-1}, joint-action $\vec{a}$ and policy mappings $\vec{\delta}$ as follows:

$$V^{\star}(\eta_{t-1}) = \max_{\vec{a},\vec{\delta}} \sum_{f \in F} \Big[\sum_{s \in s^f} R(s^f, a^f)\eta_{t-1}(s^f) + \sum_{y^f} P(y^f \mid a^f, \eta_{t-1}) V_f\big(\delta_{a^f}(y^f), \tau(\eta_{t-1}, a^f, y^f)\big) \Big] \tag{5.13}$$

One subclass of Dec-POMDPs where this result holds is the ND-POMDP model [70]. We can reformulate the above optimization problem as the following WCSP:

$\mathcal{X}$ **Variables:** We create one variable for each observation of each agent. For example, if $y_{1i} \in Y$ is a possible observation for agent 1, then variable $X_{ay_{1i}}$ is created. Figure 5.7 shows an example with four agents, each having two observations.

$\mathcal{D}$ **Domain:** The domain of all the variables corresponding to an agent is the cross-product of available actions to the agent and the set of next step sub-policies available for that agent. For example, if Δ_t denotes the set of sub-policies for agent 1, then $D_{ay_{1i}} = \{\langle a^1, p\rangle \in A^1 \times \Delta_t\}\ \forall y_{1i} \in Y^1$.

$\mathcal{C}$ **Constraints:** Two types of constraints are created—inter-agent constraints and intra-agent constraints. Intra-agent constraints ensure that the agent always takes the same action. If an agent has $|Y|$ observations, then $|Y-1|$ intra-agent constraints are created that link each variable of the agent in a linear chain. For example, a constraint is created among variables $X_{ay_{11}}$ and $X_{ay_{12}}$ of agent 1 in Figure 5.7.

Inter-agent constraints are created for every pair of observations between neighboring agents in the interaction graph. Figure 5.7 shows such constraints.

C **Valuations:** The valuation for intra-agent agent constraints is set as:

$$C(\langle a, p\rangle, \langle a', p'\rangle) = \begin{cases} 0 & \text{if } a = a' \ \forall p, p' \\ \infty & \text{if } a \neq a' \ \forall p, p' \end{cases} \tag{5.14}$$

Intuitively, they encode the fact that an agent should always perform a single action.

The valuation for inter-agent constraints denote the value accrued when agents follow a particular joint-policy upon receiving a particular observation. For example, the constraint valuation among variables $X_{ay_{11}}$ and $X_{ay_{22}}$ for the example in Figure 5.7 is shown as follows:

$$C(X_{ay_{11}} = \langle a^1, p^1\rangle, X_{ay_{22}} = \langle a^2, p^2\rangle) = \alpha - \sum_{s \in s^{12}} R(s^{12}, a^1, a^2)\eta_{t-1}(s^{12}) - P(y_{11}, y_{22} \mid a^1, a^2, \eta_{t-1}) V_{12}\big(p^1, p^2, \tau(\eta_{t-1}, a^1, a^2, y_{11}, y_{22})\big) \tag{5.15}$$

where α is a large positive constant which is used to transform the maximization to minimizing the WCSP cost. To avoid double counting, we add the immediate reward component, $\sum_{s \in s^{12}} R(s^{12}, a^1, a^2)\eta_{t-1}(s^{12})$, to only one inter-agent constraint between agent 1 and 2.

It can be easily verified that the solution of the above WCSP solves the optimization problem of Eq. (5.13). We can solve the above WCSP using branch-and-bound search similar to the one used in Section 5.3.2.1. Using such state-of-the-art WCSP solvers will significantly reduce the computational complexity of an exhaustive search based algorithm, which is exponential in the number of agents and number of observations.

In the next section, we present experimental results confirming that the decentralized backup for multiple agents can be solved efficiently using the WCSP based techniques we have developed.

5.4.3 Experiments

We implemented the WCSP based backup approach of Section 5.4.1 for the ND-POMDP model [99]. The resulting algorithm is called constraint-based dynamic programming (CBDP).

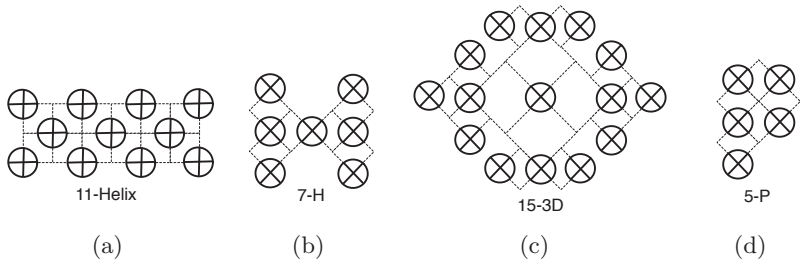
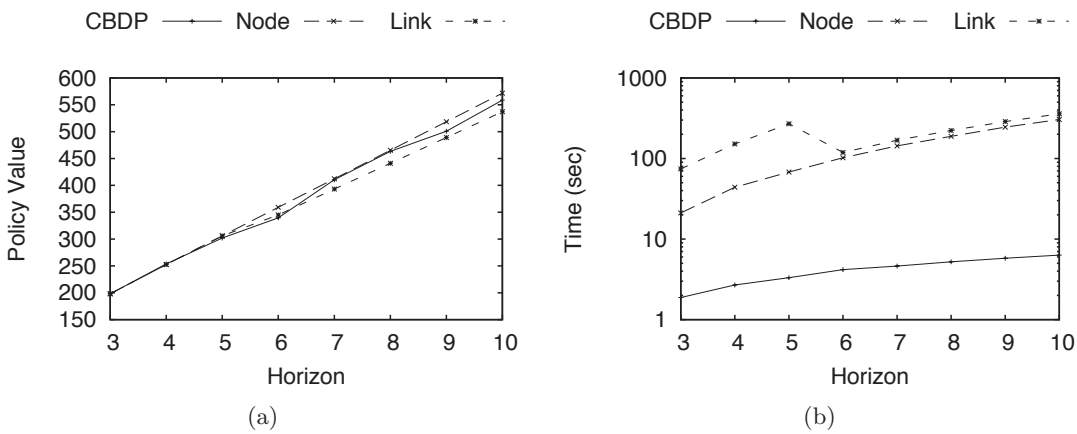


Figure 5.8. Sensor network configurations

All test examples are an instance of the target tracking problem introduced in Example 1. The sensor network configurations are shown in Figure 5.8 and are taken from previous work by Nair et al. [99] and Marecki et al. [88]. Each node represents a sensor agent. Each square shaped region denotes a location where target can be present. To successfully track a target, two adjacent sensors must scan the location simultaneously. In these examples, the number of observations per agent was 2, which made the approach of Section 5.4.1 feasible. The approach of Section 5.4.2 was not implemented.

We compare CBDP with FANS [88]–the most efficient of the existing algorithms including SPIDER [146] and LID-JESP [99]. We conducted experiments on the sensor network configurations introduced in [88], shown in Figure 5.8. All experiments were conducted on a Dual core machine with 1GB RAM, 2.4GHz CPU. The main purpose of the experiments is to show the scalability of CBDP, which has linear time and space complexity w.r.t. the horizon. We use a publicly available constraint solver implementing the bucket elimination algorithm [108]. The MaxTree parameter that governs the size of policy pool per horizon was set to 5 as it provided good tradeoff between speed and solution quality across a range of tried settings.

The experiments are divided into two sets. In the first set, we use relatively smaller domains (5-P and 7-H) because they are the only domains for which FANS can scale up to significant horizons. In the second set of experiments, we use the remaining larger domains (11-helix, 15-3D, 15-mod). For these domains, FANS cannot scale well (for 15-3D it fails to solve problems with horizons greater than 5).



(a) (b)

Figure 5.9. Comparisons of a) the solution quality and b) the execution time of our approach CBDP and different versions of FANS (Node, Link) on the domain 5P

Figure 5.9(a) shows a comparison of the solution quality of CBDP and two different versions of FANS, Node ($k = 0.5$) and Link, on the 5-P domain. FANS has multiple heuristics which tradeoff solution quality for speed. It has been suggested in [88] that for smaller domains, the Link and Node heuristics provide the best tradeoff. Hence, we chose these two for comparisons. All three algorithms achieve almost the same solution quality for all horizons. However, the key difference is in the runtime (Figure 5.9(b)). CBDP provides significant speedup over both the Node and Link heuristics. For horizon 10, CBDP is about 2 orders of magnitude faster than FANS with either heuristic. Another notable observation is that the increase in runtime for CBDP with the horizon is nearly linear, which can also be shown analytically [70].

Figure 5.10(a) shows a comparison of the solution quality on the 7-H domain. We again compare with FANS using the Node ($k = 0.5$) and Link heuristics. On this domain, due to increase in the number of agents, FANS can provide solutions up to horizon 7. With the increase in the domain size, FANS is no longer as competitive in the solution quality as it was in the 5-P domain. CBDP provides much better solution quality for all horizons. The runtime graph (Figure 5.10(b)) shows again that CBDP is much faster and that more scalable. It is 5 times faster than the Link heuristic, and orders of magnitude faster than the Node heuristic. The average time required by CBDP per horizon is 710ms which implies CBDP can solve a 100 horizon instance within 1.5 minutes due to its linear complexity as

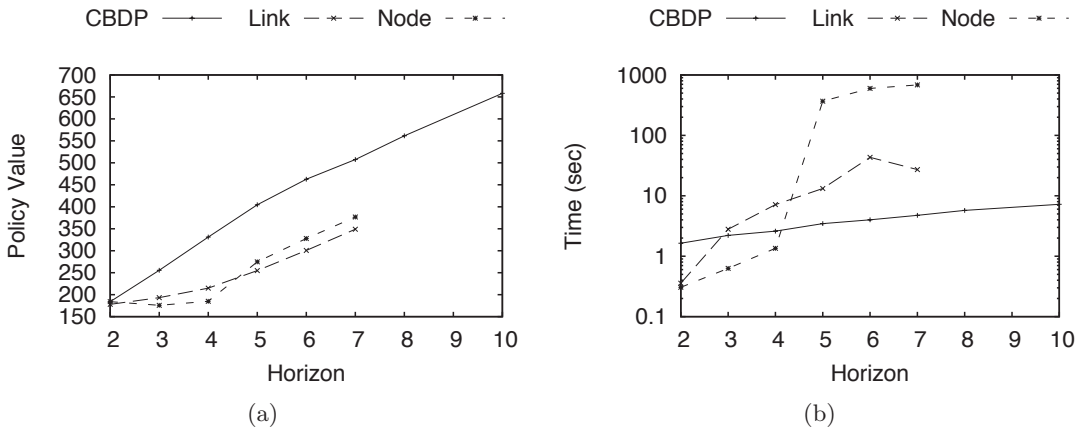


Figure 5.10. Comparisons of a) the solution quality and b) the execution time of CBDP and different versions of FANS (Node, Link) on the domain 7-H

opposed to the Node heuristic which requires 685 seconds to solve a horizon 7 instance. This further illustrates the scalability of CBDP.

Figure 5.11 shows the next set of experiments in which we use larger domains (11-helix, 15-3D, 15-mod). To be consistent with the results of Marecki *et al.* [88], we set the horizon to 3 because for 15-3D, FANS scales up to horizon 4 only. For these larger domains, a Greedy heuristic has been proposed [88] and it provides best tradeoff than other heuristics. Hence, we used this heuristic along with the Node ($k = 0.75$) heuristic for comparisons. Figure 5.11(a) shows a comparison of the solution quality. For 11-helix, CBDP provides much better solution quality than either the Greedy or Node heuristic. Similar trends are observed for the 15-3D and 15-mod domains. CBDP provides better solution quality for them as well. The runtime comparison (Figure 5.11(b)) again shows the stark difference between CBDP and FANS. For 15-mod, CBDP takes 0.6 secs while FANS with the Greedy heuristic takes over 2,000 secs. One can easily see the magnitude of speedup CBDP provides over FANS.

The results of the last set of experiments are shown in Figure 5.12. These experiments show the solution quality and execution time statistics for CBDP for different horizons on the problem 15-3D, which is the most complex problem instance for both CBDP and FANS. FANS could scale up to only horizon 5 on this problem. In contrast, CBDP can easily scale up to horizon 100. The execution time curve in this experiment clearly supports the linear

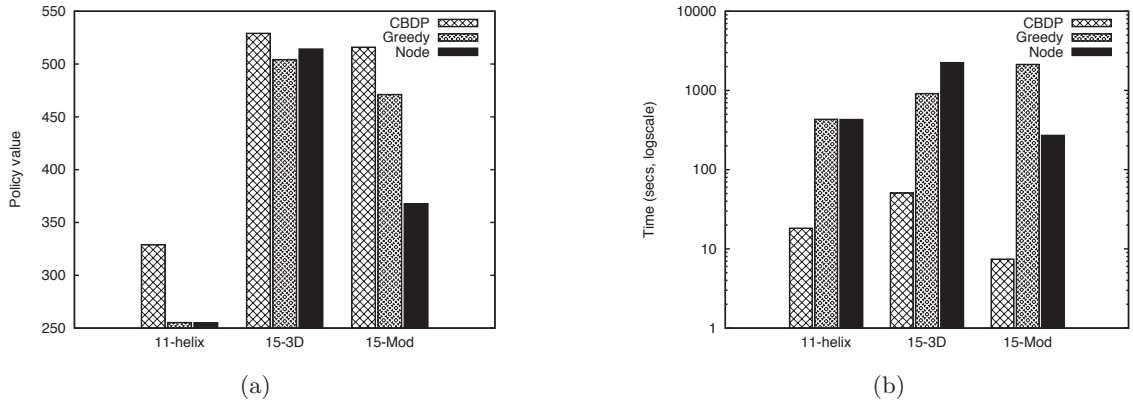


Figure 5.11. Comparisons of a) the solution quality and b) the execution time between our approach CBDP and FANS for 11-helix, 15-3D and 15-mod with T = 3

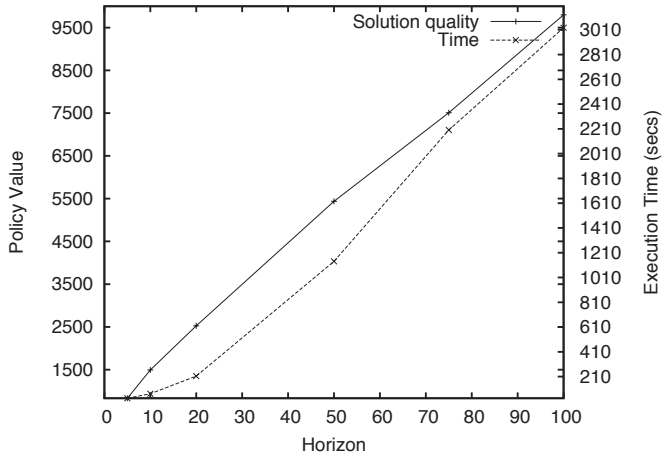


Figure 5.12. Solution Quality and Execution time comparisons of CBDP and FANS for a range of horizons for 15-3D.

time complexity of CBDP. The solution quality too increases at a nearly constant rate with the horizon, which means that CBDP is able to maintain good solution quality for large horizons.

These experiments consistently show that CBDP provides several orders-of-magnitude of speedup over different versions of FANS, while providing better solution quality for all the instances examined. CBDP further has a linear time and space complexity w.r.t. the horizon, which makes it possible to apply it to problems with much larger horizons than was previously possible.

5.5 Discussion

In this chapter, we addressed the decentralized backup problem that forms the core of several approximate dynamic programming algorithms to solve finite-horizon Dec-POMDPs. Despite our analysis showing that this problem is NP-Hard, we developed a number of optimal approaches based on the connection of the backup problem with weighted constraint satisfaction problem. This insight allowed us to use highly efficient branch-and-bound and dynamic programming based solvers in the WCSP literature to solve the decentralized backup problem. We also analyzed a number of different multiagent contexts and proposed different WCSP formulations suited to those contexts. Our results consistently illustrated that using the WCSP-based insight, our approach significantly outperforms the existing state-of-the-art algorithms. In terms of scalability, our approach was able to successfully scale to large plan horizons for larger multiagent systems (up to 15-agents, horizon 100), which had not been possible until now.

The main publications that describe the contributions of this chapter are the following:

- Complexity results and WCSP based algorithms for the decentralized backup problem are described in the following publication:

  A. Kumar and S. Zilberstein. Point-Based Backup for Decentralized POMDPs: Complexity and New Algorithms. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1315–1322, 2010.

- A bottom up dynamic programming algorithm for the ND-POMDP model that exploits the value factorization property for efficiently performing decentralized backup in large multiagent systems is described in the following publication:

  A. Kumar and S. Zilberstein. Constraint-Based Dynamic Programming for Decentralized POMDPs with Structured Interactions. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 561-568, 2009.

The WCSP formulation for the backup problem developed in this chapter has been found to be effective in other algorithms as described in following publication:

- J. Dibangoye, C. Amato and A. Doniec. Scaling Up Decentralized MDPs Through Heuristic Search. In *Proc. of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012.

CHAPTER 6

PROBABILISTIC INFERENCE FOR INFINITE-HORIZON SEQUENTIAL DECISION MAKING

In this chapter, we shift our attention to infinite-horizon sequential decision making problems. While finite-horizon Dec-POMDPs have enjoyed significant success, progress remains slow for the infinite-horizon case mainly due to the inherent complexity of optimizing stochastic controllers representing agent policies. We present a promising new class of algorithms for the infinite-horizon case by bringing in the perspective of graphical models and probabilistic inference to multiagent planning. We show how the policy optimization problem can be recast as inference in a mixture of dynamic Bayesian networks (DBNs). An attractive feature of this approach is development of new insights that connect inference techniques for DBNs with the multiagent planning, especially for supporting richer representations such as factored or continuous states and actions, which so far have been out of the scope of existing algorithms.

Such connections between multiagent planning and machine learning show significant promise by opening the door to the application of a rich set of techniques developed in the ML literature to multiagent planning. We make this connection precise by developing the well known Expectation Maximization (EM) algorithm to optimize the joint policy of agents represented as DBNs. Experiments on benchmark domains show that EM compares favorably against the state-of-the-art solvers.

6.1 Related Work

As shown in the previous chapter, a number of point-based approximate algorithms have been developed for solving finite-horizon Dec-POMDPs [72, 36, 127]. However, unlike their point-based counterparts in POMDPs ([112, 128]), they cannot be easily adopted for the infinite-horizon case due to a variety of reasons. For example, POMDP algorithms represent

Variables: $\epsilon, x(a), x(a, y, p')\ \forall p', a, y$

Maximize: ϵ (6.1)

Subject to:

$$V(p,q,s) + \epsilon \leq \sum_{a,b} \pi(b,q) \Big[x(a) R(s,a,b) + \gamma \sum_{s',y,z,p',q'} x(a,y,p') \lambda(q',q,z) P(s' \mid s,a,b) O(yz \mid s',a,b) V(p',q',s') \Big] \ \forall q, s \quad (6.2)$$

$$\sum_a x(a) = 1,\ \sum_{p'} x(a,y,p') = x(a)\ \forall a, y \quad (6.3)$$

$$x(a) \geq 0\ \forall a,\ x(a,y,p') \geq 0\ \forall a, y, p' \quad (6.4)$$

Table 6.1. The linear program used by the DEC-BPI algorithm to find new parameters for a node p of agent 1 [15]. Variable $x(a)$ represents the action selection parameter $P(a \mid p)$; variable $x(a, y, p')$ represents controller node transition parameter $P(a, p' \mid y, p)$.

the policy compactly as α-vectors, whereas most Dec-POMDP algorithms explicitly store the policy as a mapping from observation sequences to actions, making them unsuitable for the infinite-horizon case. In POMDPs, the Bellman equation forms the basis of most point-based solvers, but as Bernstein *et. al.* [15] highlight, no tractable Bellman equation exists for Dec-POMDPs.

To alleviate such problems, most infinite-horizon algorithms represent agent policies as finite-state controllers [6, 15]. So far, very few algorithms have shown promise for effectively solving infinite-horizon Dec-POMDPs, which include decentralized bounded policy iteration (DEC-BPI) [15] and a non-linear programming based approach (NLP) [6]. The DEC-BPI algorithm uses a linear programming formulation to improve the parameters of a node, one at a time. That is, it fixes the parameters of all the nodes of every agent's controllers, except a single node of a particular agent. Then, it uses a linear program, as shown in table 6.1, to find a better estimate of the parameters of that particular node, say node p of agent 1. This LP guarantees that the policy value is increased for every belief state. One drawback of this scheme is that it is not designed to optimize the value of a given belief state. Therefore, to produce a good policy, DEC-BPI may need a large number of controller nodes, which may make the LP formulation infeasible.

Variables: $x(p,a), x(q,b), x(p,y,p'), x(q,z,q'), V(p,q,s)\ \forall p,q,a,b,s,y,z$

Maximize: $$\sum_s \eta_o(s) z(p_o, q_o, s) \tag{6.5}$$

Subject to:

$$V(p,q,s) = \sum_{a,b} \Bigg(x(p,a)x(q,b) \Bigg[R(s,a,b) + \gamma \sum_{s'} P(s' \mid s,a) \sum_{y,z} O(yz \mid s',a,b) \sum_{p',q'} x(p,y,p')x(q,z,q')V(p',q',s') \Bigg] \Bigg) \forall p,q,s \tag{6.6}$$

$$\sum_a x(p,a) = 1\ \forall p, \quad \sum_b x(q,b) = 1\ \forall q \tag{6.7}$$

$$\sum_{p'} x(p,y,p') = 1\ \forall p,y \quad \sum_{q'} x(q,z,q') = 1\ \forall q,z \tag{6.8}$$

$$x(p,a) \geq 0,\ x(q,b) \geq 0,\ x(p,a,y,p') \geq 0,\ x(q,b,z,q') \geq 0 \tag{6.9}$$

Table 6.2. Nonlinear programming based formulation for optimizing a 2-agent, infinite-horizon Dec-POMDP policy [6]. Variable $x(p,a)$ represents the action selection parameter $P(a \mid p)$ for agent 1; $x(p,y,p')$ represents the controller node transition parameter $P(p' \mid p,y)$ for agent 1. Variable $V(p,q,s)$ represents the value function for the controller state (p,q) and environment state s. Other variables are defined analogously for agent 2.

Table 6.2 represents the nonlinear programming formulation for optimizing a 2-agent Dec-POMDP policy. In contrast to the DEC-BPI algorithm, the NLP formulation is focused for the initial belief state η_o. This formulation has a linear objective function. However, the Bellman constraints of Eq. (6.6) are nonlinear and non-convex in all the variables $x(\cdot)$ and $V(\cdot)$. This can cause the NLP solver to get stuck in a local optimum.

Another significant drawback of both DEC-BPI and NLP approach is the range of problems that can be handled. For example, solving Dec-POMDPs with continuous state or action spaces is not supported by either of these approaches. Scaling up to structured representations such as factored or hierarchical state-space is difficult due to convergence issues in DEC-BPI and a potential increase in the number of non-linear constraints in the NLP solver. Furthermore, none of the above approaches have been shown to work for more than 2 agents, a significant bottleneck for solving practical problems. Recently, another approach to solve infinite-horizon Dec-POMDPs has been proposed [37]. This approach is competitive with the NLP approach, but it is not clear how it can be extended to more than two agents. In contrast to such approaches, this chapter provides a framework that

can exploit the underlying structure present in the planning model by reasoning explicitly with a graphical representation of the problem using dynamic Bayesian networks (DBNs). Furthermore, the inference algorithm is based on well known techniques in the machine learning literature, which opens the door to the application of other advanced machine learning techniques to efficiently solve the multiagent planning problem.

6.2 Policy Optimization for Infinite-Horizon Dec-POMDPs

For notational simplicity, we use a modified model definition for a Dec-POMDP than shown in Section 4.1. The modified 2-time slice dynamic decision network for a 2-agent Dec-POMDP is shown in Figure 6.1. As before, the set S denotes the set of environment states, with a given initial state distribution $\eta_0(s)$. The action set of agent 1 is denoted by $a \in A$ and agent 2 by $b \in B$. The state transition probability $P(s'|s,a,b)$ depends upon the actions of both the agents. Upon taking the joint action $\langle a,b \rangle$ in state s, agents receive the joint reward $R(s,a,b)$. Y is the finite set of observations for agent 1 and Z for agent 2. $O(yz \mid s,a,b)$ denotes the probability $P(y,z|s,a,b)$ of agent 1 observing $y \in Y$ and agent 2 observing $z \in Z$ when the joint action $\langle a,b \rangle$ was taken and resulted in state s. We are concerned with solving infinite-horizon Dec-POMDPs with a discount factor $\gamma < 1$.

As highlighted in Section 4.2.2, the policy of each agent is represented as a finite-state controller (FSC). The FSC for an agent is specified by the parameters $\theta = \langle \mathcal{P}, \pi, \lambda, \nu \rangle$. The set $\mathcal{P}$ is a set of memory nodes or controller nodes for the agent. The stochastic decision rule $\pi : \mathcal{P} \rightarrow \Delta A$ represents the actions selection model or the probability $\pi_{ap} = P(a|p)$; $\lambda : \mathcal{P} \times Y \rightarrow \Delta \mathcal{P}$ represents the node transition model or the probability $\lambda_{p'py} = P(p'|p,y)$; $\nu : \mathcal{P} \rightarrow \Delta \mathcal{P}$ represents the initial node distribution $\nu_p = P(p)$. We adopt the convention that nodes of agent 1's controller are denoted by p and agent 2's by q. Other problem parameters such as observation function $O(y,z|s,a,b)$ are represented using subscripts as O_{yzsab}. The value for starting the controllers in nodes $\langle p,q \rangle$ at state s is given by:

$$V(p,q,s) = \sum_{a \in A, b \in B} \pi_{ap} \pi_{bq} \Big[R_{sab} + \gamma \sum_{s'} P_{s'sab} \sum_{y,z} O_{yzs'ab} \sum_{p',q'} \lambda_{p'py} \lambda_{q'qz} V(p',q',s') \Big]$$

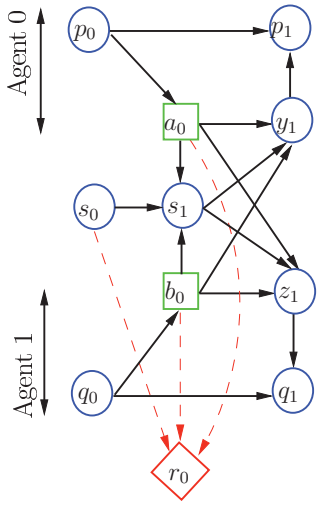


Figure 6.1. A two time slice dynamic decision network (DDN) for a two-agent Dec-POMDP

The goal is to set the parameters $\langle \pi, \lambda, \nu \rangle$ of the agents' controllers (of some given size) that maximize the expected discounted reward for the initial belief η_0:

$$V(\eta_0) = \sum_{p,q,s} \nu_p \nu_q \eta_0(s) V(p, q, s)$$

6.3 Dec-POMDPs as Mixture of DBNs

In this section, we describe how Dec-POMDPs can be reformulated as a mixture of DBNs, such that maximizing the reward likelihood (to be defined later) in this framework is equivalent to optimizing the joint-policy. Our approach is based on the framework proposed in Toussaint et al. [142] and Toussaint and Storkey [141] to solve Markovian planning problems using probabilistic inference. In this chapter, we develop the *planning-by-inference* strategy for 2-agent Dec-POMDPs. In contrast, the previous approach of Toussaint et al. [142] and Toussaint and Storkey [141] focused on single agent MDPs and POMDPs. In the next chapter, we develop new mixture models that allow us to extend the planning-by-inference strategy to multiple agents, a significant generalization of the single agent case. First we briefly describe the intuition behind this reformulation (for details please refer to [142]) and then we describe in detail the modifications required for Dec-POMDPs.

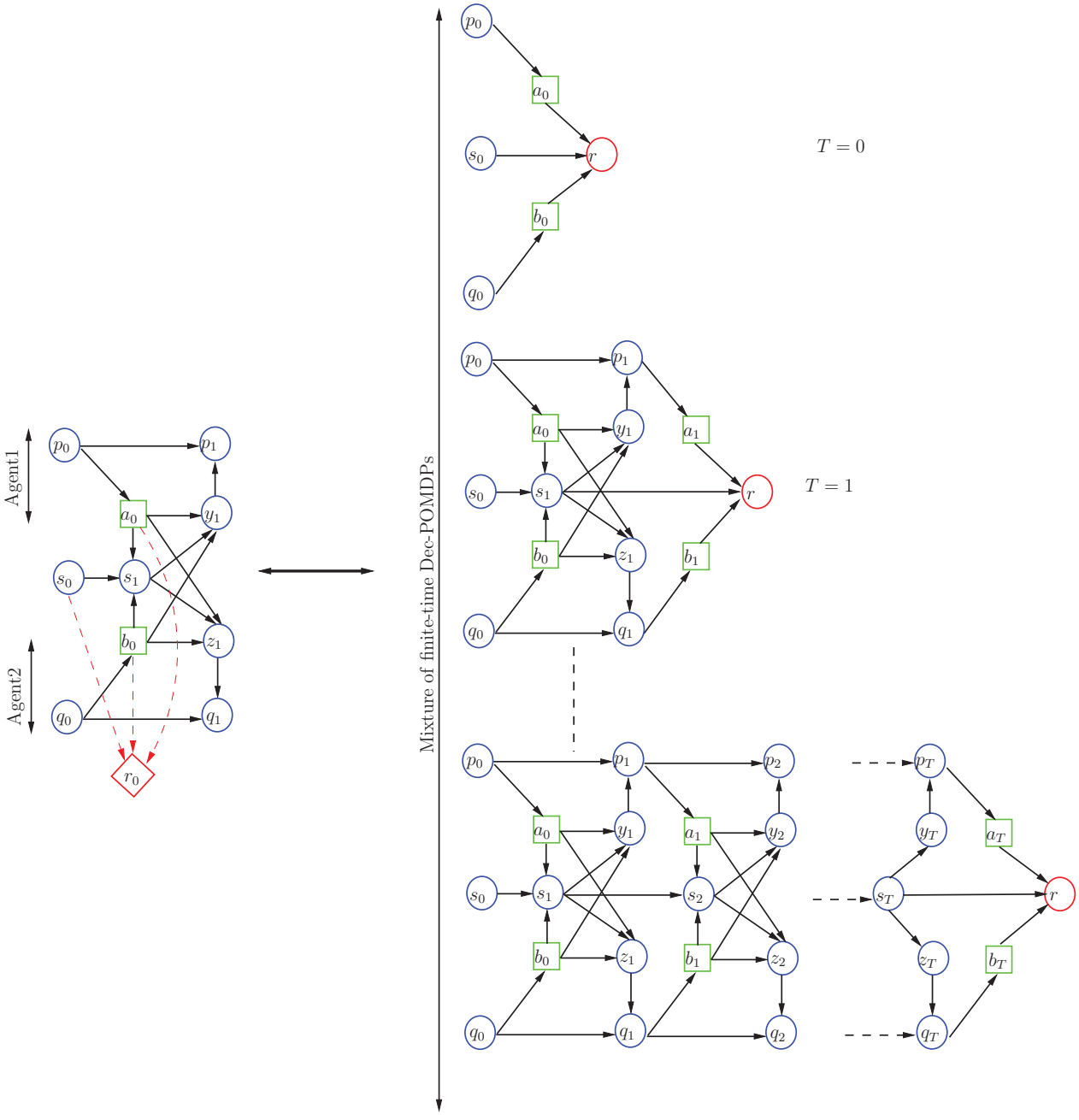


Figure 6.2. The time dependent DBN mixture (right) corresponding to a 2-agent Dec-POMDP (left). The first DBN component in the mixture corresponds to the reward for time step 1, second DBN corresponds to the reward for time step 2. The last DBN in the mixture shows the general structure of a T-step DBN.

A Dec-POMDP can be described using a single DBN where the reward is emitted at each time step, such as the unrolled DBN corresponding to the one shown in Figure 6.1. However in our approach, it is described by an infinite mixture of a special type of DBNs where reward is emitted *only at the end*. For example, the first DBN in the DBN mixture model shown in Figure 6.2 describes the DBN for time $T = 0$. The key intuition is that for the reward emitted at any time step T, we have a separate DBN with the general structure as shown in the last T-step DBN shown in Figure 6.2. Such a DBN shows how the reward obtained at time step T depends on policy parameters and the underlying Dec-POMDP model.

Furthermore, to simulate the discounting of rewards, probability of the time variable T is set as $P(T = t) = \gamma^t(1 - \gamma)$. This ensures that the time prior is normalized or $\sum_{t=0}^{\infty} P(T = t) = 1$. In addition, the random variable r shown in the DBN mixture of Fig. 6.2 is a binary variable with its conditional distribution (for any time T) described using the normalized immediate reward as:

$$\hat{R}_{sab} = P(r = 1|s_T = s, a_T = a, b_T = b) = (R_{sab} - R_{min})/(R_{max} - R_{min}).$$

The parameter R_{max} is the maximum reward for any state action pair in the given Dec-POMDP instance and R_{min} denotes the minimum reward. This scaling of the reward is the key to transforming the optimization problem from the realm of planning to likelihood maximization as stated below. Let $\boldsymbol{\theta}$ denotes the joint parameters $\langle \pi, \lambda, \nu \rangle$ for each agent's controller.

Theorem 4. *By choosing the condition probability of binary rewards r such that $\hat{R}_{sab} \propto R_{sab}$ and introducing the discounting time prior $P(T) = \gamma^T(1 - \gamma)$, maximizing the likelihood $L^{\boldsymbol{\theta}} = P(r = 1; \boldsymbol{\theta})$ in the mixture of DBNs is equivalent to optimizing the Dec-POMDP policy. Furthermore, the joint-policy value relates linearly to the likelihood as:*

$$V^{\boldsymbol{\theta}} = (R_{max} - R_{min})L^{\boldsymbol{\theta}}/(1 - \gamma) + \sum_T \gamma^T R_{min}$$

The above result shows that the policy optimization problem can be reformulated as a parameter learning problem in a suitable DBN mixture. This immediately suggests using machine learning approaches to maximize the likelihood. Furthermore, this reformulation is lossless. We note that we never explicitly create a mixture of DBNs. All computations on this DBN mixture can be implemented as message-passing over the original Dec-POMDP DBN. The only additional computational requirement is that of scaling the rewards using R_{max} and R_{min}, which can be done easily in linear time.

We next present the Expectation-Maximization (EM), a well known technique to maximize the likelihood..

6.4 The Expectation-Maxmization Algorithm

This section describes the EM algorithm [35] for maximizing the reward likelihood in the mixture of DBNs representing a Dec-POMDPs. In the corresponding DBNs, only the binary reward is treated as observed ($r = 1$); all other variables are latent. While maximizing the likelihood, EM yields the Dec-POMDP joint-policy parameters θ. EM also possesses the desirable anytime characteristic as the likelihood (and the policy value which is proportional to the likelihood) is guaranteed to increase per iteration until convergence. We note that EM is not guaranteed to converge to the global optima. However, in the experiments we show that EM almost always achieves similar values as the state-of-the-art NLP based solver [6] and much better than DEC-BPI [15]. The main advantage of using EM lies in its ability to easily generalize to much richer representations than currently possible for Dec-POMDPs such as factored or hierarchical controllers, continuous state and action spaces.

Another important advantage is the ability to generalize the solver to larger multi-agent systems with more than 2 agents by exploiting the relative independence among agents, as we will show in the next chapter. The E step we derive next is generic as any probabilistic inference technique can be used. In the current formulation, we develop techniques for the exact E-step. However, sampling based techniques can also be used, which for example will be useful for handling problems with continuous state, action spaces [57, 56]. Further, EM has been applied for planning in settings where the underlying planning model is not available. Instead, a black-box simulator is used to generate state-action trajectory samples

to approximate the E-step [150]. These characteristics of the EM make it suitable for reinforcement learning for Dec-POMDPs.

6.4.1 E-step

In the E-step, for the fixed parameter θ, forward messages α and backward messages β are propagated. First, we define the following Markovian transitions on the (p, q, s) state in a T-step DBN of Figure 6.2. These transitions are independent of the time t due to the stationary joint policy. We also adopt the convention that for any random variable v, v' refers to the next time slice and $\bar{v}$ refers to the previous time slice. For any group of variables $\mathbf{v}$, $P_t(\mathbf{v}, \mathbf{v}')$ refers to $P(\mathbf{v}_t = \mathbf{v}, \mathbf{v}_{t+1} = \mathbf{v}')$.

$$P(p', q', s'|p, q, s) = \sum_{aby'z'} \lambda_{p'py'} \lambda_{q'qz'} O_{y'z'abs'} \pi_{ap} \pi_{bq} P_{s'sab} \tag{6.10}$$

α_t is defined as $P_t(p, q, s; \theta)$. It might appear that we need to propagate α messages for each DBN in the DBN mixture separately, but as pointed out in [142], only one sweep is required as the head of the DBN is shared among all the mixture components. That is, α_2 is the same for all the T-step DBNs with $T \geq 2$. We will omit using θ as long as it is unambiguous.

$$\alpha_0(p, q, s) = \nu_p \nu_q \eta_0(s) \tag{6.11}$$

$$\alpha_t(p', q', s') = \sum_{p,q,s} P(p', q', s'|p, q, s) \alpha_{t-1}(p, q, s) \tag{6.12}$$

β messages are propagated backwards and are defined as $P_t(r = 1|p, q, s)$. However, this particular definition would require separate inference for each DBN as for T and T' step DBN, β_t will be different due to difference in the time-to-go ($T-t$ and $T'-t$). To circumvent this problem, β messages are indexed backward in time as $\beta_\tau(p, q, s) = P_{T-\tau}(r = 1|p, q, s)$ using the index τ such that $\tau = 0$ denotes the time slice $t = T$. Hence we get:

$$\beta_0(p,q,s) = \sum_{ab} \hat{R}_{sab}\pi_{ap}\pi_{bq} \tag{6.13}$$

$$\beta_\tau(p,q,s) = \sum_{p',q',s'} \beta_{\tau-1}(p',q',s')P(p',q',s'|p,q,s) \tag{6.14}$$

Based on the α and β messages we also calculate two more quantities:

$$\hat{\alpha}(p,q,s) = \sum_t P(T=t)\alpha(p,q,s) \tag{6.15}$$

$$\hat{\beta}(p,q,s) = \sum_t P(T=t)\beta(p,q,s) \tag{6.16}$$

These are used in the M-step. The cut-off time for message propagation can either be fixed a priori or be more flexible based on the likelihood accumulation. If α messages are propagated for t-steps and β-messages for τ steps, then the likelihood for $T = t+\tau$ is given as:

$$L^\theta_{t+\tau} = P(r{=}1|T = t+\tau;\theta) = \sum_{p,q,s} \alpha_t(p,q,s)\beta_\tau(p,q,s) \tag{6.17}$$

If both α and β messages are propagated for k steps and $L^\theta_{2k} \ll \sum_{T=0}^{2k-1} \gamma^T L^\theta_T$, then the message propagation can be stopped.

6.4.1.1 Complexity

Calculating the Markov transitions on the (p,q,s) chain has complexity $O(|\mathcal{P}|^4|S|^2|A|^2|Y|^2)$, where $|\mathcal{P}|$ is the maximum number of nodes for a controller. The message propagation has complexity $O(T_{max}|\mathcal{P}|^4|S|^2)$. Techniques to effectively reduce this complexity without sacrificing accuracy will be discussed later.

6.4.2 M-step

In the DBNs of Fig. 6.2 every variable is hidden except the binary reward variable r. After each M-step, EM provides better estimates of these variables, improving the likelihood L^θ and hence the policy value. For details of EM, we refer to [35]. The parameters to estimate are $\langle \pi, \lambda, \nu \rangle$ for each agent. For a particular DBN for time T, let $\tilde{L} = (P, Q, A, B, S)$

denote the latent variables, where each variable denotes a sequence of length T. That is, $P = p_{0:T}$. EM maximizes the following expected complete log-likelihood for the Dec-POMDP DBN mixture. Let $\boldsymbol{\theta}$ denotes the previous iteration's parameters and $\boldsymbol{\theta}^\star$ denotes new parameters.

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_T \sum_{\tilde{L}} P(r{=}1, \tilde{L}, T; \boldsymbol{\theta}) \log P(r{=}1, \tilde{L}, T; \boldsymbol{\theta}^\star) \tag{6.18}$$

In the rest of the section, all the derivations refer to the general T-step DBN structure as shown in Fig. 6.2. The joint probability of all the variables is:

$$P(r=1, \tilde{L}, T; \theta) = P(T)\big[\hat{R}_{sab}\big]_{t=T} \prod_{t=1}^{T} \big[\pi_{ap}\pi_{bq} P_{s\bar{s}\bar{a}\bar{b}} O_{yzs\bar{a}\bar{b}} \lambda_{p\bar{p}y} \lambda_{q\bar{q}z}\big]_t \big[\pi_{ap}\pi_{bq}\nu_p\nu_q\eta_0(s)\big]_{t=0} \tag{6.19}$$

where brackets indicate the time slices, i.e., $\big[\hat{R}_{sab}\big]_{t=T} = \hat{R}(s_T, a_T, b_T)$. Taking the log, we get:

$$\log P(r=1, \tilde{L}, T) = \ldots + \sum_{t=0}^{T} \log \pi_{a_t p_t} + \sum_{t=0}^{T} \log \pi_{b_t q_t} + \sum_{t=1}^{T} \log \lambda_{p_t p_{t-1} y_t} + \sum_{t=1}^{T} \log \lambda_{q_t q_{t-1} z_t} + \log \nu_{p_0} + \log \nu_{q_0} \tag{6.20}$$

where the missing terms represents the quantities independent of $\boldsymbol{\theta}$. As all the policy parameters $\langle \pi, \lambda, \nu \rangle$ get separated out for each agent in the log above, we first derive the action updates for an agent by substituting Eq. 6.20 in $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$.

6.4.2.1 Action updates

The update for action parameters $\pi^\star_{ap}$ for agent 1 can be derived by simplifying $Q(\theta, \theta^\star)$ as follows:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \sum_{a,p} \big[P(r{=}1, a, p|T; \theta)\big]_t \log \pi^\star_{ap} \tag{6.21}$$

By breaking the above summation between $t = T$ and $t = 0$ to $T - 1$, we get

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{T=0}^{\infty} P(T) \sum_{apqbs} \hat{R}_{sab} \pi_{ap} \pi_{bq} \alpha_T(p, q, s) \log \pi^\star_{ap} + \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T-1} \sum_{app'q's'} \beta_{T-t-1}(p', q', s') P_t(a, p, p', q', s') \log \pi^\star_{ap} \tag{6.22}$$

In the above equation, we marginalized the last time slice over the variables (q, b, s). For the intermediate time slice t, we condition upon the variables (p', q', s') in the next time slice $t + 1$. We now use the definition of $\hat{\alpha}$ and move the summation over time T inside for the last time slice and further marginalize over the remaining variables (q, s) in the intermediate slice t:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{a,p,q,b,s} \hat{R}_{sab} \pi_{ap} \pi_{bq} \hat{\alpha}(p, q, s) \log \pi^\star_{ap} + \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T-1} \sum_{ap} \log \pi^\star_{ap} \sum_{p'q's'sq} \beta_{T-t-1}(p', q', s') \pi_{ap} P(p', q', s' | a, p, q, s) \alpha_t(p, q, s) \tag{6.23}$$

Upon further marginalizing over the joint observations $y'z'$ and simplifying we get:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{ap} \pi_{ap} \log \pi^\star_{ap} \sum_{qs} \Bigg[\sum_{b} \hat{R}_{sab} \pi_{bq} \hat{\alpha}(p, q, s) + \sum_{p'q's'y'z'} \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T-1} \beta_{T-t-1}(p', q', s') P(s' | a, q, s) \lambda_{p'py'} \lambda_{q'qz'} P(y'z' | a, q, s') \alpha_t(p, q, s) \Bigg] \tag{6.24}$$

We resolve the above time summation, as in [142], based on the fact that:

$$\sum_{T=0}^{\infty} \sum_{t=0}^{T-1} f(T - t - 1) g(t) = \sum_{t=0}^{\infty} \sum_{T=t+1}^{\infty} f(T - t - 1) g(t)$$

and then setting $\tau = T - t - 1$ to get $\sum_{t=0}^{\infty} g(t) \sum_{\tau=0}^{\infty} f(\tau)$.

Finally we get:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{ap} \pi_{ap} \log \pi^\star_{ap} \sum_{qs} \hat{\alpha}(p, q, s) \Bigg[\sum_{b} \hat{R}_{sab} \pi_{bq} + \frac{\gamma}{1 - \gamma} \sum_{p'q's'y'z'} \hat{\beta}(p', q', s') \lambda_{p'py'} \lambda_{q'qz'} P(s' | a, q, s) P(y'z' | a, q, s') \Bigg] \tag{6.25}$$

The product $P(s'|a,q,s)P(y'z'|a,q,s')$ can be further simplified by marginalizing out over actions b of agent 2 as follows:

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\star}) = \sum_{ap} \pi_{ap} \log \pi^{\star}_{ap} \sum_{qs} \hat{\alpha}(p,q,s) \Big[\sum_{b} \hat{R}_{sab} \pi_{bq} + \frac{\gamma}{1-\gamma} \sum_{p'q's'y'z'} \hat{\beta}(p',q',s') \lambda_{p'py'} \lambda_{q'qz'} \sum_{b} O_{y'z's'ab} \pi_{bq} P_{s'sab} \Big]$$

The above expression is maximized by setting the parameter $\pi^{\star}_{ap}$ to be:

$$\pi^{\star}_{ap} = \frac{\pi_{ap}}{C_p} \sum_{qs} \hat{\alpha}(p,q,s) \Big[\sum_{b} \hat{R}_{sab} \pi_{bq} + \frac{\gamma}{1-\gamma} \sum_{p'q's'y'z'} \hat{\beta}(p',q',s') \lambda_{p'py'} \lambda_{q'qz'} \sum_{b} O_{y'z's'ab} \pi_{bq} P_{s'sab} \Big] \tag{6.26}$$

where C_p is a normalization constant. The action parameters $\pi^{\star}_{bq}$ of the other agent can be found similarly by the analogue of the previous equation.

6.4.2.2 Controller node transition updates

The update for controller node transition parameters $\lambda_{p\bar{p}y}$ for agent 1 can be found by maximizing $Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\star})$ w.r.t $\lambda^{\star}_{p\bar{p}y}$ as follows.

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\star}) = \sum_{T=0}^{\infty} P(T) \sum_{t=1}^{T} \sum_{p\bar{p}y} \big[P(r=1,p,\bar{p},y|T;\theta) \big]_t \log \lambda^{\star}_{p\bar{p}y} \tag{6.27}$$

By marginalizing over the variables (q,s) for the current time slice t, we get

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\star}) = \sum_{T=0}^{\infty} P(T) \sum_{t=1}^{T} \sum_{p\bar{p}ysq} \log \lambda^{\star}_{p\bar{p}y} \beta_{T-t}(p,q,s) P_t(p,\bar{p},y,s,q|T;\theta) \tag{6.28}$$

By further marginalizing over the variables $(\bar{s},\bar{q})$ for the previous time slice of t and over the observations z of the other agent, we get

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\star}) = \sum_{p\bar{p}y} \lambda_{p\bar{p}y} \log \lambda^{\star}_{p\bar{p}y} \sum_{T=0}^{\infty} P(T) \sum_{t=1}^{T} \sum_{sq\bar{s}\bar{q}z} \beta_{T-t}(p,q,s) \lambda_{q\bar{q}z} P(yz|\bar{p},\bar{q},s) P(s|\bar{p},\bar{q},\bar{s}) \alpha_{t-1}(\bar{p},\bar{q},\bar{s}) \tag{6.29}$$

The above equation can be further simplified by marginalizing the product $P(yz|\bar{p},\bar{q},s)P(s|\bar{p},\bar{q},\bar{s})$ over actions a and b of both the agents as follows:

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\star}) = \sum_{p\bar{p}y} \lambda_{p\bar{p}y} \log \lambda^{\star}_{p\bar{p}y} \sum_{T=0}^{\infty} P(T) \sum_{t=1}^{T} \sum_{sq\bar{s}\bar{q}z} \beta_{T-t}(p,q,s)\lambda_{q\bar{q}z}\alpha_{t-1}(\bar{p},\bar{q},\bar{s}) \sum_{ab} O_{yzsab} P_{s\bar{s}ab} \pi_{a\bar{p}} \pi_{b\bar{q}} \tag{6.30}$$

Upon resolving the time summation as before and maximizing $Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\star})$ w.r.t. $\lambda^{\star}_{p\bar{p}y}$, we get the final M-step estimate:

$$\lambda^{\star}_{p\bar{p}y} = \frac{\lambda_{p\bar{p}y}}{C_{\bar{p}y}} \sum_{sq\bar{s}\bar{q}z} \hat{\alpha}(\bar{p},\bar{q},\bar{s})\hat{\beta}(p,q,s)\lambda_{q\bar{q}z} \sum_{ab} O_{yzsab} P_{s\bar{s}ab} \pi_{a\bar{p}} \pi_{b\bar{q}} \tag{6.31}$$

The parameters $\lambda^{\star}_{q\bar{q}z}$ for the other agent can be found in an analogous way.

6.4.2.3 Initial node distribution

The initial node distribution ν for controller nodes of agent 1 and 2 can be updated as follows. We do not show the complete derivation as it is similar to that of the other parameters.

$$\nu^{\star}_{p} = \frac{\nu_p}{C_p} \sum_{qs} \hat{\beta}(p,q,s)\nu_q P_s \eta_0(s) \tag{6.32}$$

6.4.2.4 Complexity and implementation issues

The complexity of updating all action parameters is $O(|\mathcal{P}|^4|S|^2|A||Y|^2)$. Updating node transitions requires $O(|\mathcal{P}|^4|S|^2|Y|^2 + |\mathcal{P}|^2|S|^2|Y|^2|A|^2)$. This is relatively high when compared to a single agent POMDP updates requiring $O(|\mathcal{P}|^2|S|^2|A||Y|)$ mainly due to the scale of the interactions present in Dec-POMDPs.

In our experimental settings, we observed that having a relatively small sized controller ($N \leq 5$) suffices to yield good quality solutions. The main contributor to the complexity is the factor S^2 as we experimented with large domains having nearly 250 states. The good news is that the structure of the E and M-step equations provides a way to effectively reduce this complexity by significant factor without sacrificing accuracy. For a given state

s, joint action $\langle a, b\rangle$ and joint observation $\langle y, z\rangle$, the possible next states can be calculated as follows: $succ(s, a, b, y, z) = \{s' | P(s'|s, a, b)O(y, z|s', a, b) > 0\}$. For most of the problems, the size of this set is typically a constant $k < 10$. Such simple reachability analysis and other techniques could speed up the EM algorithm by more than an order of magnitude for large problems. The effective complexity reduces to $O(|\mathcal{P}|^4|S||A||Y|^2k)$ for the action updates and $O(|\mathcal{P}|^4|S||Y|^2k + |\mathcal{P}|^2|S||Y|^2|A|^2k)$ for node transitions. Other enhancements of the EM implementation are discussed in Section 6.6.

6.4.3 The Intuition Behind EM Update Strategy

In this section, we provide some insights about the update strategies of EM. We further analyze the expected log likelihood function for the action update as shown in Section 6.4.2.1 as:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \sum_{a,p} \big[P(r{=}1, a, p|T; \theta)\big]_t \log \pi^\star_{ap} \tag{6.33}$$

$$= \sum_{a,p} \log \pi^\star_{ap} \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \big[P(r{=}1, a, p|T; \theta)\big]_t \tag{6.34}$$

$$= \sum_{a,p} \log \pi^\star_{ap} \mathbb{E}_{\boldsymbol{\theta}}\big[r = 1, a, p\big] \tag{6.35}$$

In the above expression, the expectation $\mathbb{E}_{\boldsymbol{\theta}}\big[r = 1, a, p\big]$ is the total expected reward when the controller is in state p and action a is taken according to the previous iteration's policy parameter $\boldsymbol{\theta}$. When we maximize the above expression for $\pi^\star_{ap}$ subject to normalization constraints $\sum_a \pi^\star_{ap} = 1$, we get the update as:

$$\pi^\star_{ap} = \frac{\mathbb{E}_{\boldsymbol{\theta}}\big[r = 1, a, p\big]}{C_p} \tag{6.36}$$

where C_p is a normalization constant. The detailed expressions in Section 6.4.2.1 are essentially computing this expectation using forward-backward message passing. To understand Eq. (6.36) more clearly, consider the following iterative algorithm for optimizing the controller. First, we fix every controller node's parameters for every agent except for the parameters for a single controller node for a particular agent. Now, we deterministically

try every action setting for the particular node and greedily set the action parameter for the node to be the action that results in maximum joint value. Clearly, this strategy will monotonically improve the policy value until it reaches a local optima. Such a strategy has been used in other decision making models such as influence diagrams and is referred to as *Single Policy Update* (SPU) algorithm [79].

The updates of the EM algorithm are essentially a *soft* version of the above greedy and deterministic rule. To understand this, let $a^\star$ denote the action with maximum expectation:

$$a^\star = \arg\max_{a \in A} \mathbb{E}_{\boldsymbol{\theta}}\big[r = 1, a, p\big] \tag{6.37}$$

Now consider applying the update rule of Eq. (6.36) infinitely many times without re-computing the E-step. Clearly in the limit, we will have $\pi^\star_{a^\star p} = 1$ and the rest of action parameters will be zero. This is essentially the SPU algorithm.

The above connection also provides insight as to why the soft-max approach of EM may be a better strategy than the greedy deterministic update rule. First, the greedy update rule computes deterministic controllers for both the agents. It has been already shown that stochastic controllers can achieve better solution quality than deterministic controllers [114]. EM updates can provide stochastic controllers, which is an advantage. Second, it has been already known in the graphical models community that the greedy update rule, also referred to as *Hard-Assignment EM* [63, Chapter 19] and the EM algorithm optimize different objectives. They can in general lead to different solutions, for example, in situations when stochastic controllers are preferred to deterministic ones. The hard-assignment EM traverses a combinatorial path and needs to fix all the parameters except one. The soft-asignment EM, on the other hand, can simultaneously change the parameters of multiple nodes. Thus, the moves in the parameter space of the soft-assignment EM are more sophisticated and in general, infeasible for the hard-assignment EM, which cannot simultaneously change multiple parameters. Thus, soft-assignment EM can converge to a better policy. Therefore, using the soft-max based update strategy of the EM algorithm can be more advantageous than the greedy deterministic rule.

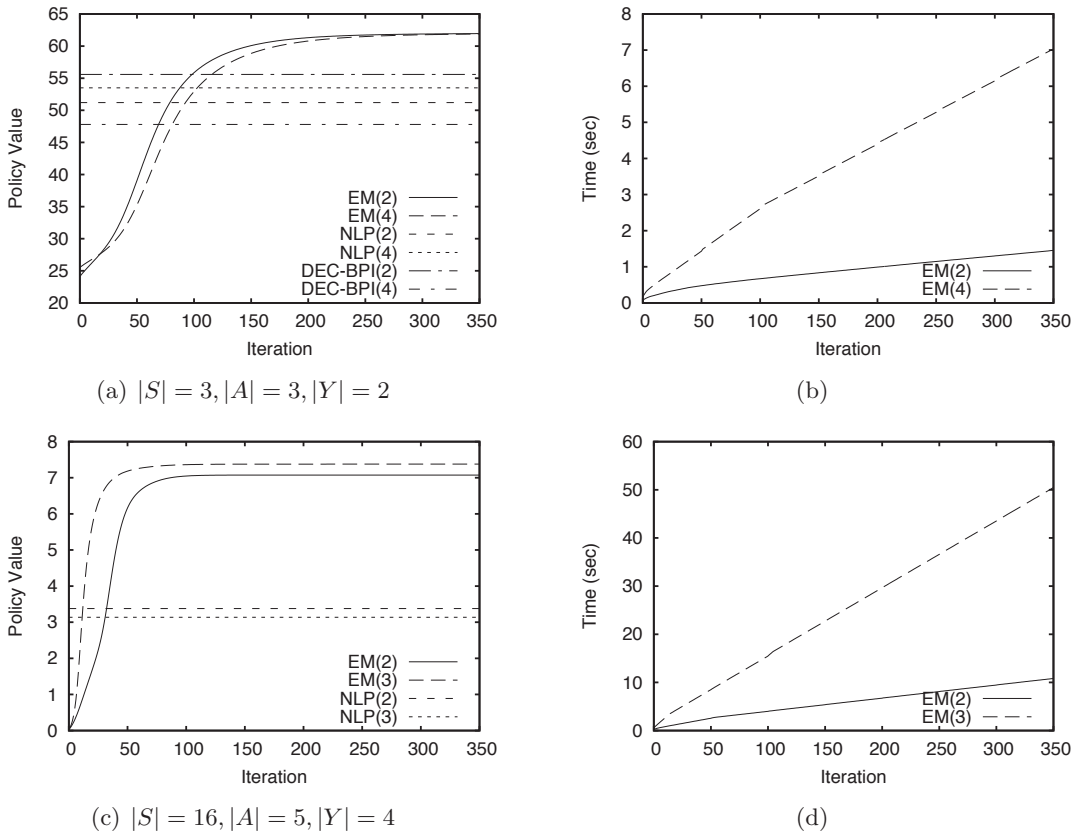
Size	DEC-BPI	NLP	EM	DEC-BPI	EM
1	4.687	9.1	9.05	< 1s	< 1s
2	4.068	9.1	9.05	< 1s	< 1s
3	8.637	9.1	9.05	2s	1.7s
4	7.857	9.1	9.05	5s	4.62s

Table 6.3. Broadcast channel: Policy value, execution time

6.5 Experiments

We experimented with several standard 2-agent Dec-POMDP benchmarks with discounting factor $\gamma = 0.9$. Complete details of these problems can be found in [6, 15]. We compare our approach with the decentralized bounded policy iteration (DEC-BPI) algorithm [15] and a non-linear, non-convex optimization solver (NLP) [6]. The DEC-BPI algorithm iteratively improves the parameters of a node using a linear program while keeping the other nodes' parameters fixed. The NLP approach, which has been the state-of-the-art for infinite-horizon Dec-POMDPs, recasts the policy optimization problem as a non-linear program and uses an off-the-shelf solver, Snopt [48], to obtain a solution. We implemented the EM algorithm in JAVA. All our experiments were on a Mac with 4GB RAM and 2.4GHz CPU. Each data point is an average of 10 runs with random initial controller parameters. In terms of solution quality, EM is always better than DEC-BPI and it achieves similar or higher solution quality than NLP. We note that the NLP solver [48] is an optimized package and therefore for larger problems is currently faster than the EM approach. For the EM algorithm, we did not implement optimizations such as parallel execution using multithreading, that can decrease the runtime significantly.

Table 6.3 shows results for the *broadcast channel* problem, which has 4 states, 2 actions per agent and 5 observations. This is a networking problem where agents must decide whether or not to send a message on a shared channel and must avoid collision to get a reward. We tested with different controller sizes. On this problem, all the algorithms compare reasonably well, with EM being better than DEC-BPI and very close in value to NLP. The time for NLP is also $\approx 1s$.

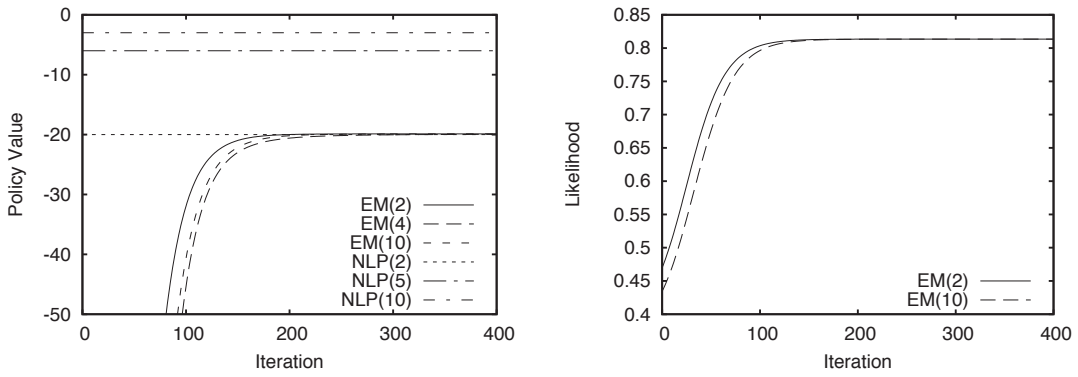


(a) $|S| = 3, |A| = 3, |Y| = 2$ (b)

(c) $|S| = 16, |A| = 5, |Y| = 4$ (d)

Figure 6.3. Solution quality and runtime for 'recycling robots' (a) & (b) and 'meeting on a grid' (c) & (d)

Figure 6.3(a) compares the solution quality of the EM approach against DEC-BPI and NLP for varying controller sizes on the *recycling robots* problem. In this problem, two robots have the task of picking up cans in an office building. They can search for a small can, a big can or recharge the battery. The large item is only retrievable by the joint action of the two robots. Their goal is to coordinate their actions to maximize the joint reward. EM(2) and NLP(2) show the results with controller size 2 for both agents in Figure 6.3(a). For this problem, EM works much better than both DEC-BPI and the NLP approach. EM achieves a value of ≈ 62 for all controller sizes, providing nearly 12% improvement over DEC-BPI (= 55) and 20% improvement over NLP (= 51). Figure 6.3(b) shows the time comparisons for EM with different controller sizes. Both the NLP and DEC-BPI take nearly $1s$ to converge. EM with controller size 2 has comparable performance, but as expected, EM with 4-node controllers takes longer as the complexity of EM is proportional to $O(|\mathcal{P}|^4)$.

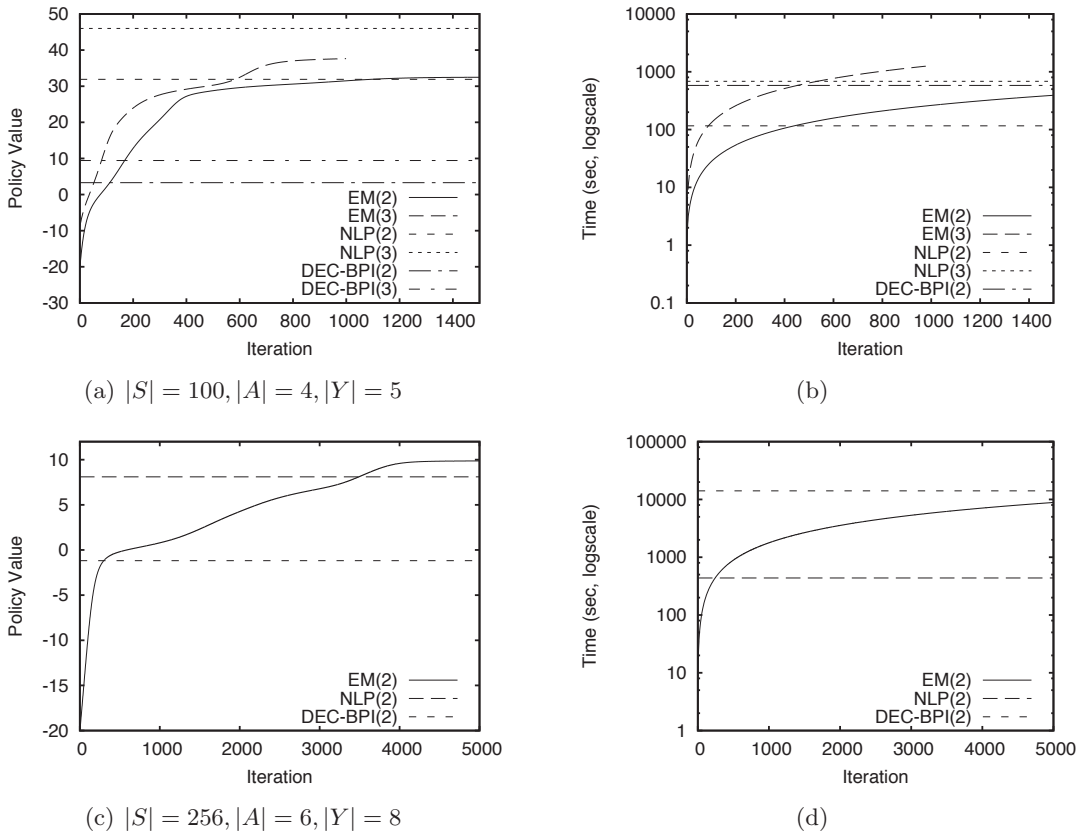


(a) $|S| = 2, |A| = 3, |Y| = 2$ (b)

Figure 6.4. Solution quality (a) and likelihood (b) for 'multiagent tiger'

Figure 6.3(c) compares the solution quality of EM on the *meeting on a grid* problem. In this problem, agents start diagonally across in a 2×2 grid and their goal is to take actions such that they meet each other (i.e., share the same square) as much as possible. As the figure shows, EM provides much better solution quality than the NLP approach. EM achieves a value of ≈ 7, which nearly *doubles* the solution quality achieved by NLP ($= 3.3$). DEC-BPI results are not plotted as it performs much worse and achieves a solution quality of 0, essentially unable to improve the policy at all even for large controllers. Both DEC-BPI and NLP take around $1s$ to converge. Figure 6.3(d) shows the time comparison for EM versions. EM with 2-node controllers is very fast and takes $< 1s$ to converge (50 iterations). Again, because of EM's quartic complexity in the controller size $|\mathcal{P}|$, the time required for larger controllers is higher. Also note that in both the cases, EM could run with much larger controller sizes (≈ 10), but the increase in size did not provide tangible improvement in solution quality.

Figure 6.4 shows the results for the *multi-agent tiger* problem, involving two doors with a tiger behind one door and a treasure behind the other. Agents should coordinate to open the door leading to the treasure [6]. Figure 6.4(a) shows the quality comparisons. EM does not perform well in this case; even after increasing the controller size, it achieves a value of -19. NLP works better with large controller sizes. However, this experiment presents an interesting insight into the workings of EM as related to the scaling of the rewards. Recalling the relation between the likelihood and the policy value from Theorem 4, the



(a) $|S| = 100, |A| = 4, |Y| = 5$ (b)

(c) $|S| = 256, |A| = 6, |Y| = 8$ (d)

Figure 6.5. Solution quality and runtime for box pushing (a) & (b) and Mars rovers (c) & (d)

equation for this problem is: $V^\theta = 1210L^\theta - 1004.5$. For EM to achieve the same solution as the best NLP setting ($= -3$), the likelihood should be .827. Figure 6.4(b) shows that the likelihood EM converges to is .813. Therefore, from EM's perspective, it is finding a really good solution. Thus, the scaling of rewards has a significant impact (in this case, adverse) on the policy value. This is a potential drawback of the EM approach, which applies to other Markovian planning problems when using the technique of [142]. Incidently, DEC-BPI performs much worse on this problem and gets a quality of -77.

Figure 6.5 shows the results for the two largest Dec-POMDP domains—*Box pushing* and *Mars rovers*. In the box pushing domain, agents need to coordinate and push boxes into a goal area. In the Mars rovers domain, agents need to coordinate their actions to perform experiments at multiple sites. Figure 6.5(a) shows that EM performs much better than DEC-BPI for every controller size. For controller size 2, EM achieves better quality

than NLP with comparable runtime (Figure 6.5(b), 500 iterations). However, for the larger controller size (= 3), it achieves slightly lower quality than NLP. For the largest Mars rovers domain (Figure 6.5(c)), EM achieves better solution quality (= 9.9) than NLP (= 8.1). However, EM also takes many more iterations to converge than for previous problems and hence, requires more time than NLP. EM is also much better than DEC-BPI, which achieves a quality of -1.18 and takes even longer to converge (Figure 6.5(d)).

6.6 Discussion

We presented a new approach to solve DEC-POMDPs using inference in mixtures of DBNs. Even a simple implementation of the approach provides good results. Extensive experiments show that EM is always better than DEC-BPI and compares favorably with the state-of-the-art NLP solver. The experiments also highlight one potential drawback of the EM approach: slow convergence rate for large problems. We can address the runtime issue by parallelizing the algorithm, as all the steps of EM can be easily performed in parallel. For example, α and β can be propagated in parallel. Even updating each node's parameters can be done in parallel for each iteration. Furthermore, the structure of EM's update equations is amenable to Google's Map-Reduce paradigm [32], allowing each parameter to be computed by a cluster of machines in parallel using Map-Reduce. Such scalable techniques will certainly make our approach many times faster than the current *serial* implementation. An interesting future direction of research is to investigate how a different scaling of rewards will affect the convergence properties of EM.

The main benefit of the EM approach is that it opens up the possibility of using powerful probabilistic inference techniques to solve decentralized planning problems. Using a graphical DBN structure, EM can easily generalize to richer representations such as factored or hierarchical controllers, or continuous state and action spaces. Unlike the existing techniques, EM can easily extend to larger multi-agent systems with more than 2 agents, as will be shown in the next chapter. The ND-POMDP model [99] is a class of Dec-POMDPs specifically designed to support large multi-agent systems. It makes some restrictive yet realistic assumptions such as locality of interaction among agents, and transition and observation independence. EM can naturally exploit such independence structure in the DBN

and scale to larger multi-agent systems, something that current infinite-horizon algorithms fail to achieve.

The main publications that describe the contributions of this chapter are the following:

- The mapping between planning for decentralized POMDPs, probabilistic inference and the EM algorithm is developed in the following publication:

  A. Kumar and S. Zilberstein. Anytime Planning for Decentralized POMDPs using Expectation Maximization. In *Proc. of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 294–301, 2010.

An extension of the inference-based approach to general influence diagrams was developed in the following publication:

- X. Wu, A. Kumar and S. Zilberstein. Influence Diagrams With Memory States: Representation and Algorithms. In *Proc. of the International Conference on Algorithmic Decision Theory (ADT)* , pages 306–319, 2011.

CHAPTER 7

ACHIEVING SCALABILITY FOR RESTRICTED MODELS

In previous chapters, we have discussed and developed approximate algorithms that can efficiently solve 2-agent Dec-POMDPs. However, scaling even such approximate algorithms for more than 2-agents is a non-trivial task. In fact, most of the 2-agent approximate algorithms we have developed, such as the EM approach of chapter 6, have exponential complexity in the number of agents. Therefore, we ask the following question: *Is there a general characterization of the interaction among agents that when present in a multiagent planning model leads to a relatively scalable approximate algorithm?*

In this chapter, we answer this question affirmatively and identify certain mild conditions that are sufficient to make multiagent planning amenable to a scalable approximation w.r.t. the number of agents. This is achieved by constructing a graphical model that exploits such restricted interactions among agents. We again illustrate a close relationship with machine learning by showing that likelihood maximization in such a graphical model is equivalent to policy optimization. Using the Expectation-Maximization framework for likelihood maximization, we show that the necessary inference can be decomposed into processes that often involve a small subset of agents, thereby facilitating scalability. We derive a global update rule that combines these local inferences to monotonically increase the overall solution quality. Furthermore, our approach is easily parallelizable and takes the form of message-passing among agents, ideally suited for large multiagent systems.

Experiments on a large multiagent planning benchmark, with state and action spaces up to $O(5^{22})$, $O(3^{20})$ respectively, confirm that our approach is scalable w.r.t. the number of agents and can solve planning problems for up to 20 agents, which cannot be handled by previous best approaches. For smaller multiagent systems, our approach provides better solution quality and is about an order-of-magnitude faster than the previous best nonlinear programming approach.

7.1 Related Work

Many recent attempts to increase the scalability of planners w.r.t. the number of agents impose restrictions on the allowed interactions among the agents such as transition independence [12, 99], weak-coupling among agents that is limited to certain states [147], and directional transition dependence [156]. One of the earliest models that demonstrated scalability by limiting interactions among agents was TI-Dec-MDP or transition-independent Dec-MDP [10]. Agents in these models can fully observe their local state. The dependence among agents was only through the joint-reward function. The complexity of this model was shown to be NP-Complete [51]. A detailed complexity analysis of different types of interactions among agents is provided in [51]. An extension of TI-Dec-MDP was proposed in [11], which introduced structured *transition dependencies*.

The TI-Dec-MDP model was extended to handle the partial observability about environment states in [99], resulting in a model called network-distributed POMDP or ND-POMDP. It was shown that the joint-value factorizes among smaller sub-groups of agents based on immediate reward decomposability. Such factorization was exploited to develop a number of algorithms that scale relatively well w.r.t. the number of agents [99, 146, 88]. Another restricted class of models is the transition-decoupled POMDP or TD-POMDP [156]. This model explicitly differentiated among an agent's private state that can only be affected by the agent's local action and the shared states, that can be affected by other agents. However, the representation power of TD-POMDP is mostly limited to non-concurrent interactions, in which agents are structured in a *parent-child* relationship. An agent can only affect the state transitions of its children, not vice-versa. Structured interactions were also exploited for deciding how to communicate among agents [96, 97].

A majority of the above models try to identify restrictions on agent interactions that can enable scalable planning. With the exception of the work of Witwicki and Durfee [157], which was conducted concurrently with us, there has not been much work towards a general characterization of conditions under which multiagent planning can be made scalable. Witwicki and Durfee [157] provide a characterization of weak-coupling among agents similar to our work. However, a significant difference is that, in our work we propose a concrete algorithm that can efficiently exploit such weak-coupling among agents to enable

scalability to large multiagent systems. The algorithm proposed by Witwicki and Durfee [157] can only solve very small multiagent problems. In their formulation, the planning problem is mapped to a constraint optimization problem, where variables denote agents. The domain of a variable is the set of *all possible policies* for the agent. This is highly inefficient as the policy space for an agent is extremely large and in general, limiting the scalability of the approach. In contrast, the approach we present requires probabilistic inference in small sub-groups of agents. Such probabilistic inference, in general, has the computational requirement equivalent to evaluating the policies. Therefore, our approach presents a concrete framework that can scale up to large multiagent systems by exploiting weak-coupling among agents.

7.2 Value Factorization Framework for Multiagent Planning

As before, we represent each agent's policy as a bounded, *finite state controller* (FSC). This approach has been used successfully for both POMDPs [114] and Dec-POMDPs [6]. Let us assume that the state space S is factored s.t. $S = (S^1 \times ... \times S^M)$, an assumption that holds in several multiagent planning models such as ND-POMDPs [99] and TD-POMDPs [156]. Without making further (conditional independence) assumptions on the problem structure, a general Dec-POMDP requires exact inference in the full corresponding (finite-time) DBNs, which would be exponential in the number of state variables and agents. Our approach relies on a general simplifying property of agent interaction, which we later show to be consistent with many of the existing multiagent planning models.

Definition 13. *A **value factor** f defines a subset $A_f \subseteq \{1, .., N\}$ of agents and a subset $S_f \subseteq \{1, .., M\}$ of state variables.*

Definition 14. *A multiagent planning problem satisfies **value factorization** if the joint-policy value function can be decomposed into a sum over value factors:*

$$V(\boldsymbol{\theta}, s) = \sum_{f \in F} V_f(\theta^f, s^f) \ , \tag{7.1}$$

where F is a set of value factors, $\theta^f \equiv \theta^{A_f}$ is the collection of parameters of the agents of factor f, and $s^f \equiv s^{S_f}$ is the collection of state variables of this factor.

Even when the value factorization property holds, planning in such models is still highly coupled because factors may overlap. That is, an agent can appear in multiple factors as

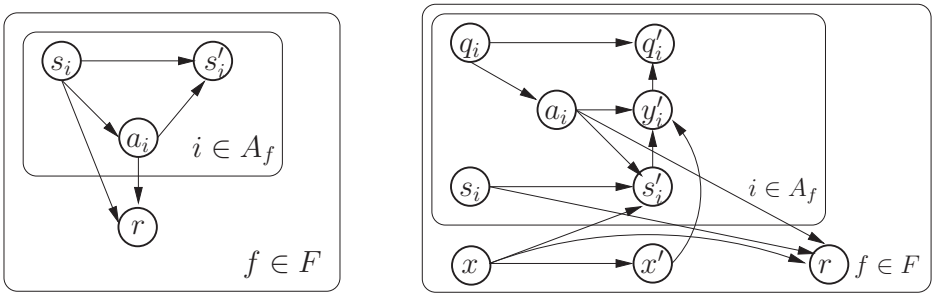


Figure 7.1. Plate notation for (a) Dec-MDPs; (b) ND-POMDPs

can state variables. Therefore, a value factor cannot be optimized *independently*. But, as we show later, it leads to an efficient Expectation Maximization algorithm. Such additive value functions have also been used to solve large factored MDPs [64]. We require that each value factor V_f can be evaluated using the DBN mixture based approach of Sec. 6.3. However, this is not a restriction, as the DBN-based approach can model arbitrary Markovian planning problems.

Theorem 5. *The worst case complexity of optimally solving a multiagent planning problem satisfying the value factorization property is NEXP-Hard.*

The proof of the above theorem is straightforward—any two agent finite-horizon Dec-POMDP is NEXP-Complete and also satisfies the value factorization property as there is only a single factor involving two agents. In fact, in the previous chapter we precisely addressed this issue using the EM framework (see Section 6.4). Next, we investigate when this property holds and when it is computationally advantageous, and establish the following result.

Theorem 6. *The value factorization property holds in transition independent Dec-MDPs [12], Network-Distributed POMDPs [99] and Transition-Decoupled POMDPs [156]. Each value factor in these models also allows for efficient probabilistic inference in the EM framework.*

The joint value is shown to be factorized based on the immediate-reward factorization in Dec-MDPs [12] and ND-POMDPs [99]. Figure 7.1 shows the plate notation for our value factor representation for both of these models. The outer plate shows a factor f and the inner plate depicts the interaction among agent parameters which include state,

action and observation variables. We provide the steps of the EM algorithm for these two models in Sec. 7.2.2 and show that the inferences required by the E-step decompose along the immediate-reward factorization, which involves fewer number of agents and is thus scalable.

Our approach can also model Transition-Decoupled POMDPs (TD-POMDPs) [156]. In this case, agents have local parameters (factored local state and rewards). However, certain features of the local state can depend on other agents' actions. This dependency among agents is described using an *influence* directed acyclic graph (DAG) where each node is an agent. A parent-child relationship in this DAG implies that the child's local states are controllable by the parent, but not vice-versa. An important characteristic is that given its parents' policies, an agent can compute its value function. In our representation, this translates into a value factor for *each agent* i, which consists of i and all its *ancestors* in the DAG. The joint-value decomposition along value factors is straightforward. It might appear that the DBN corresponding to the value factor of a leaf node will become very large leading to prohibitive inference. However, this is not so, as a summary of an agent's influence on its immediate children can be compactly represented as an influence DBN [156]. Intuitively, the inference queries EM requires can be performed by updating such influence DBNs for each edge in this DAG in a top-down fashion, avoiding exponential complexity in the number of agents.

We note that the value factorization property of Eq. (7.1) is trivially satisfied when all the agent and state variables are included in a single factor. Obviously, the computational advantages of our approach are limited to settings where each factor is *sparse*, involving much fewer agents than the entire team. This allows for efficient inference in the respective DBNs (inference can still be efficient for special cases such as TD-POMDPs that have larger factors). In the general case, the additive value function may include components depending on all states and agent parameters. This is analogous to the factored HMMs [47] where, conditioned on the observations, all Markov chains become coupled and the exact E-step of EM becomes infeasible. While this is beyond the scope of this paper, a promising approach for the general case is using variational methods to approximate the posterior $P(s_{1:T}^{1:M} \mid \hat{r}=1)$ (minimizing the KL-divergence between the factored representation and the

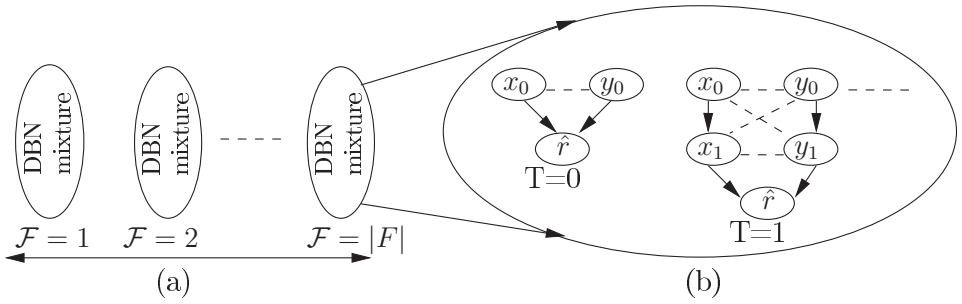


Figure 7.2. (a) Value factor mixture; (b) Zoomed-in view of each mixture component (x, y are generic placeholders for random variables).

true posterior) [47]. Given such an approximate posterior, the M-step updates we derive next can be used to realize an approximate EM scheme.

Theorem 7. *In models satisfying the value factorization property, the inferences in the Expectation-Maximization framework can be performed independently for each value factor f without imposing* ***any restrictions*** *on the interaction among the agents associated with each value factor.*

A constructive proof of this result is provided below. This result highlights the generality and scalability of our approach, which —unlike previous works—does not require any further independence assumptions.

7.2.1 DBN mixture for value factors

Figure 7.2 shows a new problem independent DBN mixture, also called *value factor* mixture, which models Eq. 7.1. It consists of two mixture variable: $\mathcal{F}$ and T. $\mathcal{F}$ ranges from 1 to $|F|$, the total number of value factors. Intuitively, $\mathcal{F} = i$ denotes the time dependent DBN mixture for value factor i. A zoomed-in view of the this DBN mixture is provided in Figure 7.2(b). The mixture variable $\mathcal{F}$ has a fixed, uniform distribution ($= 1/|F|$). The distribution of the variable T is set as in Theorem 4.

This model relies on the fact that in our representation, each value factor can be represented and evaluated using a time dependent DBN mixture of Figure 7.2(b) and the binary reward variable r, as also shown in the Section 6.3. This DBN mixture for a particular value factor f will contain all the variables for agents involved in factor f: actions, con-

troller nodes and observations, and the state variables S_f. The valuation $V_f(\theta^f)$ can be calculated by finding the likelihood $L(r;\theta^f) = P(r=1;\theta^f)$ of observing the binary reward variable as $r=1$.

$$V_f(\theta^f) = kL(r;\theta^f) + k_f \tag{7.2}$$

where k and k_f are constants, with k being the same for all value factors. This can be easily ensured by making all the original rewards positive by adding a suitable constant. Next we state one of our main results. We are also assuming that the policy is being optimized for an initial belief η_0. We will also use a notational convenience that $\sum_{\mathcal{F}=1}^{|F|}$ is equivalent to $\sum_{f\in F}$.

Theorem 8. *Maximizing the likelihood $L(r;\boldsymbol{\theta})$ of observing $r=1$ in the value factor mixture (Figure 7.2(a)) is equivalent to optimizing the global policy $\boldsymbol{\theta}$.*

Proof. The overall likelihood is given by:

$$L(r;\boldsymbol{\theta}) = P(r=1;\boldsymbol{\theta}) = \sum_{f\in F} \frac{1}{|F|} L(r;\theta^f) \tag{7.3}$$

The theorem follows by substituting the value of each $L(r;\theta^f)$ in the previous equation from Eq. (7.2) and the joint-policy value decomposition assumption of Eq. (7.1). □

7.2.2 The Expectation-Maximization Algorithm

We now derive the EM algorithm for maximizing the likelihood $L(r;\boldsymbol{\theta})$ in the value factor mixture. Only $r=1$ is observed data, the rest of the variables are latent. We first handle the *infinite-horizon* case, assuming a stationary policy. Later, we address finite-horizon problems. Note that our derivations differ markedly from [141] as they focus on a single-agent problem or from the EM approach for 2-agent Dec-POMDPs (see Section 6.4). Our focus is on scalability w.r.t. the number of agents and generality.

An assignment to the mixture variables T and $\mathcal{F}=f$ denotes a T-step DBN for the value factor f. For example, assume that the time dependent DBN mixture in Figure 7.2(b) is

for value factor f. Then $\mathcal{F}=f$ and $T=1$ denote the 1-step DBN (the second DBN) in Figure 7.2(b). Let z_t denote all the hidden variables for a single time slice t in this DBN. Let $Z_f = z_{0...T}$ denote a complete assignment to all the variables from slice 0 to T. We assume for simplicity that each value factor f involves k agents. The full joint for the mixture is:

$$P(r=1, Z_f, T, \mathcal{F}=f) = P(T)P(\mathcal{F}=f)\big[\hat{R}_{s^f a^f}\big]_{t=T} \prod_{i=1}^{k}\prod_{t=0}^{T}\big[\pi_{f_i}(a,q)\big]_t \prod_{i=1}^{k}\prod_{t=1}^{T}\big[\lambda_{f_i}(q,\bar{q},y)\big]_t \prod_{i=1}^{k}[\nu_{f_i}(q)]_0 P(Z_f \backslash (A_f, Q_f)|T, \mathcal{F}=f) \tag{7.4}$$

where the subscript f_i denotes the respective parameters for agent i involved in factor f. The square brackets denote dependence upon time: $[\pi_{f_i}(a,q)]_t = \pi_{f_i}(a_t=a, q_t=q)$. We also use $[P(v,\bar{v})]_t$ to denote $P(v_t=v, v_{t-1}=\bar{v})$. $Z_f\backslash(A_f,Q_f)$ denotes all the variables in this DBN except the action and controller nodes of all the agents. Importantly, the structure of this equation is model independent as by the conditional independence of *policy parameters* ($\pi(a,q)=P(a|q)$), the first part of the equation (the product terms) can always be written this way. Since EM maximizes the expected log-likelihood, we take the log of the above to get:

$$\log P(r=1, Z_f, T, \mathcal{F}=f) = \sum_{i=1}^{k}\sum_{t=0}^{T}\big[\log \pi_{f_i}(a,q)\big]_t + \sum_{i=1}^{k}\sum_{t=1}^{T}\big[\log \lambda_{f_i}(q,\bar{q},y)\big]_t + \sum_{i=1}^{k}\big[\log \nu_{f_i}(q)\big]_0 + \langle \text{other terms}\rangle \tag{7.5}$$

where $\langle$other terms$\rangle$ denote terms independent of policy parameters $\boldsymbol{\theta}$. EM maximizes the following expected log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{f\in\mathcal{F}}\sum_{T=0}^{\infty}\sum_{Z_f} P(r=1, Z_f, T, \mathcal{F}=f; \theta^f)\log P(r=1, Z_f, T, \mathcal{F}=f; \theta^{f^\star}) \tag{7.6}$$

where $\boldsymbol{\theta}$ denotes the previous iteration's joint-policy and $\boldsymbol{\theta}^\star$ is the current iteration's policy to be determined by maximization. The structure of the log term (Eq. (7.5)) is particularly

advantageous as it allows us to perform maximization for each parameter of each agent *independently*. This does not imply complete problem decoupling as all the parameters still depend on the *previous iteration's* parameters for all other agents. We first derive the update for the action parameter of an agent j. $P(\mathcal{F})$ can be ignored as it is a constant. $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$ for action updates is given by:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{f \in F} \sum_{T} P(T) \sum_{Z_f} P(r=1, Z_f | T, f; \theta^f) \sum_{t=0}^{T} \left[\log \pi_j^\star(a,q) \right]_t \tag{7.7}$$

To maximize this expression, we only need to consider the set of factors f that include agent j, denoted by $F(j)$. And because the policy is stationary, $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$ can be simplified:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{f \in F(j), T} P(T) \sum_{t=0}^{T} \sum_{(a,q)_j} P(r=1, (a_t, q_t) = (a,q)_j | T, f; \theta^f) \log \pi_j^\star(a,q) \tag{7.8}$$

Let $\mu_j(a, q; \theta^f)$ be defined for each agent j and each factor $f \in F(j)$ as follows:

$$\mu_j(a, q; \theta^f) = \sum_{T} P(T) \sum_{t=0}^{T} P(r=1, (a_t, q_t) = (a,q)_j | T, f; \theta^f) \tag{7.9}$$

Then, the expression for $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$ can be further simplified:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{(a,q)_j} \log \pi_j^\star(a,q) \sum_{f \in \mathcal{F}(j)} \mu_j(a, q; \theta^f) \tag{7.10}$$

This expression can be easily maximized by using the Lagrange multiplier for the constraint $\sum_a \pi_j(a,q) = 1$. The final update for agent j's action parameters is:

$$\pi_j^\star(a,q) = \frac{1}{C_q} \sum_{f \in \mathcal{F}(j)} \mu_j(a, q; \theta^f) \tag{7.11}$$

C_q is a normalization constant. The update for controller transition parameter $\lambda_j(q, \bar{q}, y)$ can be found analogously by defining a function $\mu_j(q, \bar{q}, y; \theta^f)$ along the lines of Eq. (7.9):

$$\mu_j(q, \bar{q}, y; \theta^f) = \sum_{T} P(T) \sum_{t=0}^{T} P(r=1, (q_t, q_{t-1}, y_t) = (q, \bar{q}, y)_j | T, f; \theta^f) \tag{7.12}$$

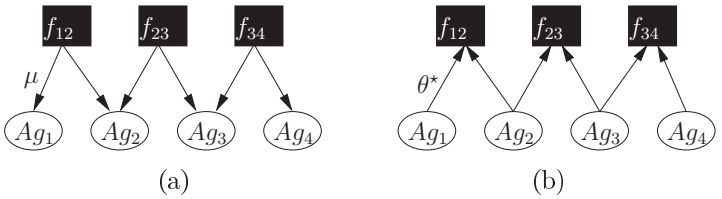


Figure 7.3. Message passing on the value factor graph: (a) shows the message direction for E-step; (b) shows the M-step.

The parameter ν_j can be updated similarly. Therefore the *E-step* in the EM algorithm involves computing the function μ_j for each agent j and each factor $f \in F(j)$. Eq. (7.11) specifies the *M-step*. Algorithm 2 describes a high level view of the EM algorithm for multiagent systems.

7.2.3 Finite-Horizon Planning

EM can be easily adapted to the finite-horizon case with planning horizon H and no discounting (hence $P(T)=1/H$ is uniform). Each agent has a non-stationary policy represented as an acyclic FSM. That is, each controller variable q^t represents possible controller states at time t. The only difference lies in the definition of the μ functions:

$$\mu_j(a, q^t; \theta^f) = \sum_{T=t}^{H} P(r \ = 1, (a_t, q_t) = (a, q^t)_j | T, f; \theta^f) \tag{7.13}$$

Notice that computing the μ functions is simpler than in the infinite-horizon case (Eq. 7.9). The above inference requires *separate* computation in each DBN of length $T=t$ until $T=H$. The global update rule (Eq. 7.11) remains the same.

7.2.4 Scalability and Message-Passing Implementation

The description of the μ functions highlights the high scalability of EM. Even though the planning problem is highly coupled with each agent, state variables allowed to participate in multiple value factors (see Eq. 7.1), yet updating policy parameters requires *separate* local inference in each factor to compute the μ function. The global update rule (Eq. 7.11) combines such local inferences to provide a monotonic increase in overall solution quality. Each local inference is model dependent and can be computed using standard probabilistic

Algorithm 2: Message-Passing For Value Factorization based Multiagent Planning

input: Value Factor Graph $G = (V, E)$

Initialize policy parameters of each agent i randomly: $\theta^i \leftarrow randInit()$

while $iter \leq MAX\_ITER$ **do**

  foreach *Agent* $i = 1$ *to* N **do**

    Send parameters θ^i to each value factor $f \in \mathcal{F}(i)$

  foreach *Value factor* $f \in \mathcal{F}$ **do**

    foreach *Agent* $i \in f$ **do**

      Compute $\mu_i(a^i, q^i; \theta^f)\ \forall a \in A^i, q^i \in Q^i$ (See Eq. (7.9))

      Compute $\mu_i(q^i, \bar{q}^i, y^i; \theta^f)\ \forall q^i, \bar{q}^i \in Q^i, y^i \in Y^i$ (See Eq. (7.12))

      Compute $\mu_i(q^i; \theta^f)\ \forall q^i \in Q^i$

      Send message $\langle \mu_i(a^i, q^i; \theta^f) \rangle$ to agent i

      Send message $\langle \mu_i(q^i, \bar{q}^i, y^i; \theta^f) \rangle$ to agent i

      Send message $\langle \mu_i(q^i; \theta^f) \rangle$ to agent i

  foreach *Agent* $i = 1$ *to* N **do**

    Receive all messages from all factors $f \in \mathcal{F}(i)$

    Set $\pi(a^i, q^i) \leftarrow \frac{1}{C_q} \sum_{f \in \mathcal{F}(i)} \mu_i(a^i, q^i; \theta^f)\ \forall a \in A^i, q^i \in Q^i$

    Set $\lambda(q^i, \bar{q}^i, y^i) \leftarrow \frac{1}{C_{\bar{q}^i, y^i}} \sum_{f \in \mathcal{F}(i)} \mu_i(q^i, \bar{q}^i, y^i; \theta^f)\ \forall q^i, \bar{q}^i \in Q^i, y^i \in Y^i$

    Set $\nu(q^i) \leftarrow \frac{1}{C} \sum_{f \in \mathcal{F}(i)} \mu_i(q^i; \theta^f)\ \forall q^i \in Q^i$

    Set $\theta^i \leftarrow \langle \pi, \lambda, \nu \rangle$

  $iter \leftarrow iter + 1$

return: The last iteration's policy parameters for each agent

techniques. As our problem setting includes *sparse* factors, such local inference will be computationally much simpler than performing it on the complete planning model.

Furthermore, the E and M-steps can be implemented using a *parallel*, *distributed* message-passing on a bipartite *value-factor graph* $G = (U, V, E)$. The set U contains a node u_j for each agent j. The set V contains a node v_f for each factor $f \in F$. An edge $e = (u_j, v_f)$ is created if agent j is involved in factor f. Figure 7.3 shows such a graph with three value factors in the black squares (the set V) and 4 agents (the set U).

During the E-step, each factor node v_f computes and sends the message $\mu_{f \to j} = \mu_j(\bullet; ; \theta^f)$ to each node u_j that is connected to v_f. Figure 7.3(a) shows this step. An agent node u_j upon receiving all the μ messages from each factor node connected to it, updates its parameters as in Eq. (7.11) and sends the updated policy parameters $\theta^\star$ back to each factor node it is connected to (see Figure 7.3(b)). This procedure repeats until convergence. Algorithm 2 describes such message passing procedure for multiagent systems. Based on this message-passing interpretation of EM, we can state the following result:

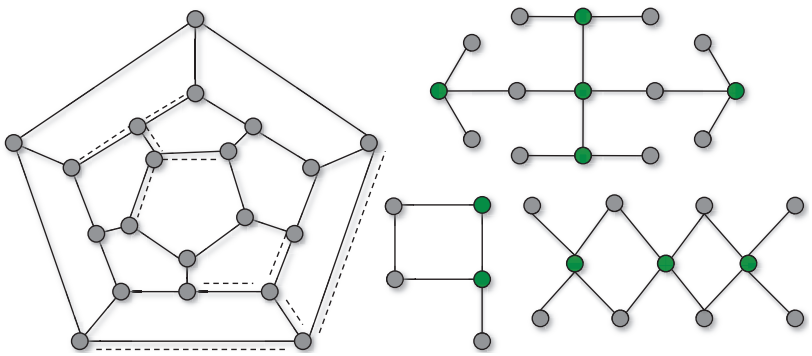


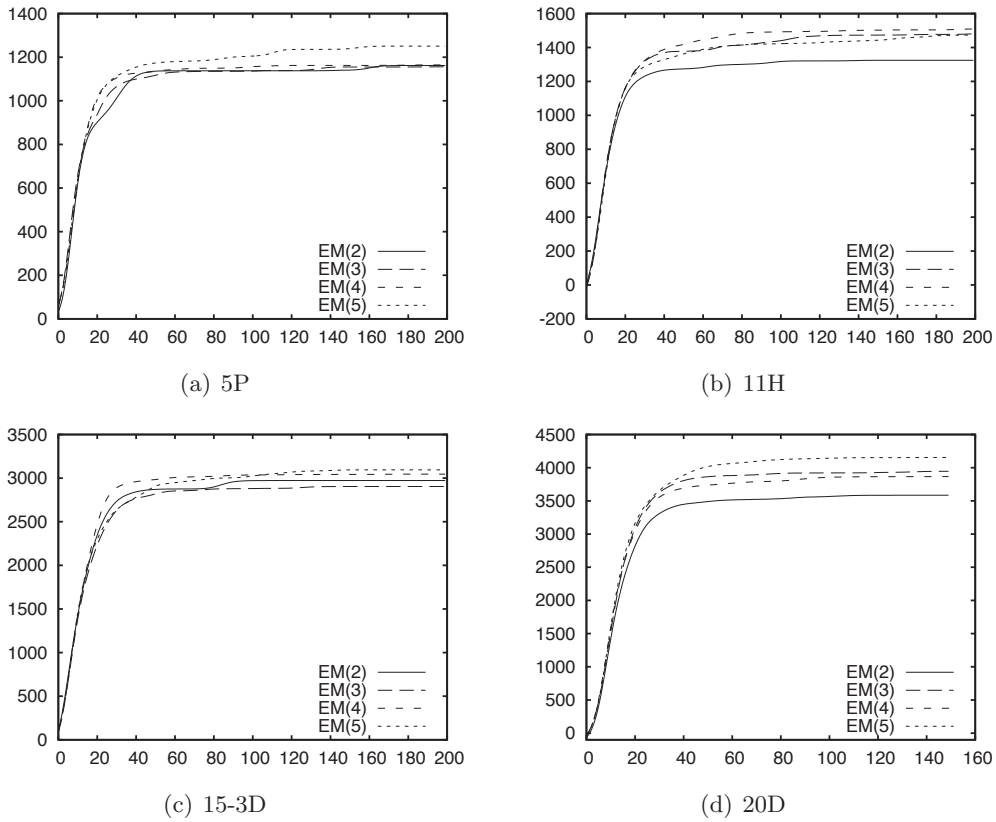
Figure 7.4. Benchmarks 20D (left), 15-3D, 5P and 11H (right)

Theorem 9. *The EM algorithm has linear complexity in the number of edges in the value factor graph and exponential complexity with respect to the maximum number of agents involved in a single value factor.*

As stated earlier, when the value factors are sparse, EM algorithm will provide significant computational savings over an approach that is oblivious to the underlying interaction among agents. Another significant advantage of the EM algorithm is that all messages can be computed in parallel by each factor node. Our experiments, using an 8-core CPU resulted in near linear speedup over a sequential version. These characteristics of the EM algorithm significantly enhance its scalability for large, but sparse planning problems.

7.2.5 Nature of Local Optima

Variants of the Dec-POMDP model are often solved by fixing the policies of a group of agents and then finding the best response policy of the agent that interacts with the group [98, 156]. EM offers significant advantages over such strategy. While both find local optima, EM is more stringent and satisfies the Karush-Kuhn-Tucker conditions [17], which is the norm in nonlinear optimization. The best-response strategy provides no such guarantees. Furthermore, in Dec-POMDPs computing the best response is *exponential* in the plan horizon [98]. EM does not suffer this drawback as it directly performs inferences on the DBN.



(a) 5P (b) 11H

(c) 15-3D (d) 20D

Figure 7.5. Solution quality achieved by EM (y-axis denotes quality and x-axis denotes the iteration number).

7.3 Experiments

We experimented on a target tracking application in sensor networks modeled as an ND-POMDP [99]. Fig. 7.4 shows four sensor topologies: 5P, 11H and 15-3D from [88] and the largest 20D from [69].

We describe a variant of this application next, originally introduced in [99]. Each node in these graphs is a sensor agent and edges are locations where targets can move. To track a target and gain reward ($=+80$), two sensors must scan the target location simultaneously, otherwise a penalty ($=-1$) is given. All targets have independent, stochastic trajectories and their all possible locations form the *external* state-space. Sensors have an *internal state*, an indication of the battery level of the sensor. Each scan action depletes the battery. In addition to scanning, sensors have two additional actions *sensor off* and *recharge*. The *sensor off* action allows sensors to conserve energy by remaining idle. When the battery is

Instance\Size	2-Node	3-Node	4-Node	5-Node
5P	.232	1.07	3.22	7.74
11H	1.29	6.07	18.90	45.23
15-3D	1.17	5.39	16.69	40.47
20D	5.03	22.01	67.85	171.26

Table 7.1. Time in seconds per iteration of EM

completely depleted, a sensor must perform *recharge* action, which has a cost ($=-1$). Each sensor has three observation: *target present*, *target absent* and *idle*. False positives/negatives are allowed for the first two observations. At runtime, sensors operating in a *decentralized* manner without a central controller.

Note that our formulation of sensor network application is much richer and challenging than the previously used benchmarks [88]. Earlier benchmarks did not include any internal states, sensors were assumed to have an unbounded battery life. In the current formulation, planning is much more complex as sensors must reason not only about scanning, but also about conserving their energy, as they have a limited battery. EM was implemented in JAVA. All our experiments were done on an 8-core Mac with 2GB RAM. Our implementation used multithreading to parallelize EM and utilized all 8-cores. Each plot is the best of 10 runs. The discount factor γ was 0.95. To speed up EM's convergence, we used a greedy variant of the M-step [143].

The 5P domain has 2 targets, 11H has 3 targets, 15-3D has 5 targets, and 20D has 6 targets. These problems have very large state-spaces: 6×5^5 for 5P, 64×5^{11} for 11H, 144×5^{15} for 15-3D and 2700×5^{20} for 20D. Evidently, EM efficiently exploits the factored representation of the state and action spaces and is highly scalable with *linear complexity* w.r.t. the number of edges in the graph.

Figure 7.5 shows the solution quality EM achieves for each of these benchmarks with different controller sizes. A notable observation from these graphs is that EM converges quickly, taking fewer than 200 iterations even for such large multiagent planning problems. The solution quality, as expected, increases monotonically with each iteration highlighting the anytime property of EM. In general, the solution quality increased with the number of

Instance/Value	EM	U.B.	Random
5P	1250.89 (44.3%)	2820	61.23
11H	1509.27 (35.6%)	4230	8.41
15-3D	3094.05 (43.8%)	7050	104.2
20D	4154.04 (49.1%)	8460	−31.67

Table 7.2. Quality comparisons with a loose upper bound and random controllers for all instances

	Internal State = 2		Internal State = 3	
N	EM	NLP	EM	NLP
2	670.8/3.8	79/5.4	972.5/8.9	905.7/17.8
3	670.8/13.02	140.4/14.5	1053.16/35.8	887.2/139
4	710.4/35.8	140.4/139.4	1062.4/107.4	1024.8/1078.1

Table 7.3. Solution quality/time comparison of EM (100 iterations) with NLP for the 5P domain, N denotes controller size, Time in seconds

controller nodes. For example, for 20D, a 2-node controller achieves a quality of 3585.06 and a 5 node controller achieves 4154.04. However, for 5P and 15-3D, we did not observe a significant increase in quality by increasing controller size, possibly due to the relative simplicity of these configurations.

Table 7.1 shows the runtime per iteration of EM for different instances and varying controller sizes. Encouragingly, the runtime is fairly small—particularly for smaller controller sizes—even for large problems such as 20D. To further decrease the runtime for larger controllers, we plan to use Monte-Carlo sampling techniques in the future.

Table 7.3 shows the solution quality comparison of EM with random controllers and a loose upper bound. The upper bound was computed by assuming that each target is detected at every time step including the battery recharge cost. Against random controllers, EM always achieves much better solution quality. When compared against the upper bound, we can see that EM performs quite well. Despite being a very loose bound, EM still achieves a quality within 49.1% of this bound for the largest 20D domain—a solid performance.

Previously, no algorithm could solve infinite-horizon ND-POMDPs (>2-agents). To further assess EM's performance, we developed a nonlinear programming (NLP) formulation of the problem and used a state-of-the-art NLP solver called Snopt [48]. Snopt could only

Version \ FSC Size	Handcrafted	2 (EM)	3 (EM)	4 (EM)
No penalty	13.92	13.95	15.48	15.7
Penalty (−.25)	-3.36	5.27	5.27	5.27

Table 7.4. Quality of handcrafted controllers vs. EM (11H)

solve the smallest 5P domain and could not scale beyond controller size 4 and internal state 3 as it ran out of memory (=2GB). Table 7.3 shows the solution quality and time comparisons. For internal state size of 2, Snopt gets stuck in a poor local optimum compared to EM. It provides more competitive solutions for internal state 3, but EM is still better in solution quality. Furthermore, the runtime of Snopt degrades quickly with the increase in nonlinear constraints. This makes Snopt about an *order-of-magnitude* slower for controller size 4 and internal state 3. These results further highlight the scalability of EM, which could scale up to controller size 10 and internal state 5 within 2GB RAM and $\approx$ 4 hours for 100 iterations.

Table 7.4 shows a comparison of EM against handcrafted controllers designed to take into account the target trajectories and partial observation in the 11H benchmark (Figure 7.4). To simplify the problem for the handcrafted solution, all penalties were zero and the reward for detecting a target was 1. This allowed continuous scan by sensors without worrying about miscoordination penalty. The first row in this table shows this no penalty case. We see that EM is competitive with the handcrafted controller. The second row shows the results when there was a cost to charge batteries. In this case, sensors need to decide when to become *idle* to conserve power. The handcrafted controller cannot learn this behavior and hence EM produces much better quality in this case.

Finally, Table 7.5 highlights the significant opportunities that EM provides for parallel computation. We consistently obtained almost linear speedup when using multithreading on an 8-core CPU (total possible parallel threads in the largest domain 20D is 60). By using a massively parallel platform such a Google's MapReduce,we could easily solve much larger team decision problems than currently possible.

Version \ FSC Size	2	3	4	5
Serial	41.05	177.54	543.52	1308.20
Parallel	5.03	22.01	67.85	171.26

Table 7.5. Serial vs. parallel execution times per EM iteration in 20D.

7.4 Discussion

Despite the rapid progress in multiagent planning, the scalability of the prevailing formal models has been limited. We developed a new approach to multiagent planning by identifying the general property of *value factorization* that facilitates the development of a scalable approximate algorithm using probabilistic inference. We show that several existing classes of Dec-POMDPs satisfy this property. In contrast to previous approaches, our framework does not impose any further restrictions on agent interaction beyond this property, thus providing a general solution for structured multiagent planning.

The key result that supports the scalability of our approach is that, within the EM framework, the inference process can be decomposed into separate components that are much smaller than the complete model, thus avoiding an exponential complexity. Additionally, the EM algorithm allows for distributed planning using message-passing along the edges of the value-factor graph, and is amenable to parallelization. It also provides significant advantages over existing locally optimal approaches for Dec-POMDPs, delivering more rigorous guarantees on the solution. Results on large sensor network problems confirm the scalability of our approach. Empirically, our approach, which has linear complexity in the number of edges in the agent interaction graph could scale up to 20 agents, whereas the previous best approach based on nonlinear programming could only scale up to 5 agents due to increase in the number of nonlinear constraints.

Our approach offers a form of policy iteration using finite-state controllers to represent policies; extending it to the value iteration is an important future work. We currently assume that the structure of each value factor remains static. A significant contribution would be to allow agents to change this structure via their actions and still provide the current guarantees. Overall, this new approach promises to significantly enhance the scalability and practical applicability of decentralized POMDPs.

The main publication that describes the contributions of this chapter is the following:

- A. Kumar, S. Zilberstein and M. Toussaint. Scalable Multiagent Planning using Probabilistic Inference. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2140–2146, 2011.

Bounded Optimality for Sequential Decision Making

CHAPTER 8

QUALITY BOUNDED SOLUTIONS FOR SEQUENTIAL DECISION MAKING

In previous chapters, we presented a number of approximate algorithms for multiagent sequential decision making within the Dec-POMDP framework. These algorithms had polynomial runtime complexity, however, they did not provide quality bounds or any guarantee of optimality. Developing algorithms that can optimally solve multiagent decision-theoretic planning problems within the Dec-POMDP framework seems inherently inefficient due to the complexity being NEXP-Hard. In this chapter, we address an alternative question— *If agents' policies are restricted to a particular sub-class of all the possible policies, then how can we find the optimal policy within this sub-class?*

We develop mixed integer linear programming (MILP) formulation for optimizing agents' policies represented as finite-state controllers (FSC). Once we have a MILP representation of the planning problem, then we can use off-the-shelf and highly efficient MILP solvers such as CPLEX to solve the MILP to optimality or get non-trivial upper bounds. Previously, several attempts have been made to formulate both the single agent and multiagent planning problems under uncertainty using mathematical programming [114, 24, 3, 4, 15, 6]. However, most of these approaches either result in a non-convex program, which suffers from the problem of local optima or they approximate the policy optimization using a convex program, which does not guarantee accuracy. Our approach is the first to formulate the policy optimization problem as a compact mixed integer program for both Dec-POMDPs and single agent POMDPs. A crucial aspect of our approach is representing the multiagent planning problem using an alternate graphical model that highlights the main features necessary for the MILP formulation. In our case, we represent the multiagent planning problem using a cross-product MDP defined over the environment states and the finite-state con-

troller states representing agents' policies. This cross-product MDP provides the necessary constraints that enable the formulation of the MILP.

Such connections to the MILP formulation are significant, as mixed integer programming is one of the most heavily researched area in the mathematical optimization community unlike other non-convex or non-linear formulations. There are several techniques that can provide an approximate solution to large mixed integer programs, while also providing worst case quality guarantees. We show how to apply one such technique called Lagrangian relaxation [16] to settings when a large number of agents ($\gg 2$) are involved in the planning problem. These techniques can be easily implemented in a message-passing manner, making them particularly suitable for large multiagent systems.

8.1 Related Work

There have been several attempts to optimize the policy represented as a finite-state controller for both single agent POMDPs and Dec-POMDPs. Most of the techniques to optimize agents' policies under the Dec-POMDP model borrow heavily from algorithms for the POMDP model. Therefore, we first review approaches to optimize an agent's policy under the POMDP model.

A technique to optimize fixed-size controllers based on linear programming was presented in [114]. However, this technique did not provide a focused way to find the optimal policy for the given initial belief. It modifies the finite-state controller such that the value increases for the complete belief space. As such, it could get stuck in a local optimum and may need a large number of controller nodes to provide a good policy. A non-convex quadratic programming formulation was developed in [3]. As this approach involved solving a non-convex optimization problem, it could also get stuck in a local optimum. A number of techniques based on non-linear programming to optimize hierarchical controllers for the POMDP model were presented in [24]. As the proposed non-linear programs were non-convex, optimality could not be guaranteed. An interesting approach to optimize deterministic finite-state controllers was presented in [94]. This approach could provide the optimal FSC. However, the technique was based on a customized branch-and-bound search, rather than a mathematical programming formulation. As we show empirically, using highly efficient industrial-strength

solvers such as CPLEX is likely to outperform a customized search algorithm. Furthermore, modeling the policy optimization problem explicitly as a mathematical program provides deeper insights into the structure of the problem. It also opens the door to the application of approximate techniques such as Lagrangian relaxation that can solve large mixed integer programs efficiently while also providing quality guarantees.

One of the first technique to optimize agents' policies under the Dec-POMDP model was proposed in [14]. It was an extension of the LP based approach of [114]. However, this technique could also get stuck in a local optimum and does not target the given initial belief state. A more efficient non-linear programming (NLP) based formulation was presented in [6]. This approach resulted in a non-convex program and thus, no optimality guarantee or quality bounds could be provided. The inference based approach which we developed in Chapters 6 and 7, provides good solutions, but cannot provide any quality guarantees, as the EM algorithm can also get stuck in local optima. In another line of work, Aras and Dutech [7] presented a number of MILP formulations to optimize *finite-horizon* Dec-POMDP policies. Their approach was based on the sequence form representation of a Dec-POMDP policy. This approach resulted in large mixed integer programs as the number of sequences or agent histories increases exponentially with the plan horizon. Furthermore, this approach is not suitable for infinite-horizon Dec-POMDPs as the size of an agent's history is unbounded in the infinite-horizon case. A number of MILP formulations were presented in [96, 97] for a sub-class of Dec-MDPs. These formulations also used the sequence form and thus, are not suitable for infinite-horizon Dec-POMDPs where policies are represented as cyclic finite-state controllers.

8.2 LP Formulation of MDPs

We first describe the linear programming formulation for an infinite-horizon MDP as our approach to optimize Dec-POMDP controllers is based on this LP formulation. The two time slice DBN for an MDP is shown in Figure 8.1. The variable s denotes the underlying state, which is observable to the agent. The variable a denotes the action. The probability $P(\cdot|s, a)$ denotes the transition function, $R(\cdot, \cdot)$ denotes the reward function; η_0 denotes the

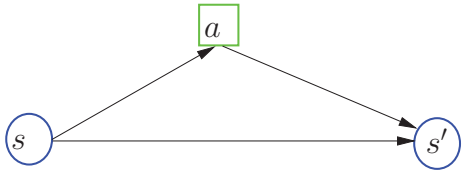


Figure 8.1. Two time slice DBN for single agent MDP

initial belief over the state. The linear program for finding the optimal MDP policy is given as:

$$\max \sum_{s} \sum_{a} R(s,a)x(s,a) \tag{8.1}$$

$$\sum_{a} x(j,a) - \sum_{s} \sum_{a} \gamma\, P(j|s,a)x(s,a) = \eta_0(j)\ \forall j \in S \tag{8.2}$$

where the parameter $x(s,a)$, intuitively, denotes the total discounted amount of time the environment state was s and action a was taken. Therefore, the total reward corresponding to being in state s and taking action a is $R(s,a)x(s,a)$. This describes the objective function of the LP formulation. The constraints are analogous to the flow conservation principle.

Definition 15. *The parameter $x(s,a)$ is defined as follows [116]:*

$$x(s,a) = \sum_{j \in S} \eta_0(j) \sum_{t=1}^{\infty} \gamma^{t-1} P(s_t = s, a_t = a \mid s_1 = j) \tag{8.3}$$

The MDP policy can be extracted from the $x(\cdot)$ parameters as follows:

$$P(a \mid s) = \frac{x(s,a)}{\sum_{a' \in A} x(s,a')} \tag{8.4}$$

8.3 Cross-Product MDP For a 2-Agent Dec-POMDP

In this section, our goal is to develop a mathematical programming formulation analogous to the LP formulation for an MDP to optimize *deterministic* finite-state controllers for a infinite-horizon 2-agent Dec-POMDP. Even though the formulation is developed for infinite-horizon planning, it can be extended in a straightforward manner to handle finite-horizon planing as the policy trees used in finite-horizon planning (see Section 4.2.1) can be viewed as acyclic finite-state controllers.

We use the same model definition for a 2-agent infinite-horizon Dec-POMDP as in Section 6.2 and the corresponding 2 time slice DBN shown in Figure 6.1. The set S denotes the set of environment states, with a given initial state distribution $\eta_0(s)$. The action set of agent 1 is denoted by $a \in A$ and agent 2 by $b \in B$. The state transition probability $P(s'|s,a,b)$ depends upon the actions of both the agents. Upon taking the joint action $\langle a,b \rangle$ in state s, agents receive the joint reward $R(s,a,b)$. Y is the finite set of observations for agent 1 and Z for agent 2. $O(yz\,|\,s,a,b)$ denotes the probability $P(y,z|s,a,b)$ of agent 1 observing $y \in Y$ and agent 2 observing $z \in Z$ when the joint action $\langle a,b \rangle$ was taken and resulted in state s. We are concerned with solving infinite-horizon Dec-POMDPs with a discount factor $\gamma < 1$. We adopt the convention that nodes of agent 1's controller are denoted by p and agent 2's by q. The controller parameters $\langle \pi, \lambda, \nu \rangle$ are the same as described in Section 6.2.

Our approach is to interpret a 2-agent Dec-POMDP as an *single agent MDP* defined over the underlying environment state and the states of finite-state controllers of both the agents. We then introduce constraints in the LP formulation of this *cross-product* MDP that enable the resulting policy to be executed in a decentralized and partially observable setting. The state space, action space and transition function of this cross-product MDP are defined below.

State space: Figure 8.2 shows the 2-time slice DBN for the underlying cross-product MDP. The state of this MDP is the tuple (p,q,s). The state space is the cross-product of the joint-controller state and the environment state.

Action space: The action space has two components. First, the actions a and b of both the agents are included in the action space. In addition, the controller transition parameters $\lambda(\cdot)$ for both the agents are also included in the action space. This is required as our goal in solving this cross-product MDP is to not only find the action mapping for each state (p,q,s), but is also to find out which node the control should transfer to for each possible observation received. This is achieved by creating a new random variable n_{y_i} for each observation y_i of agent 1 and variable n_{z_i} for each observation z_i of agent 2. The domain of each random variable n_{y_i} is the set of all controller nodes to which the control can transfer

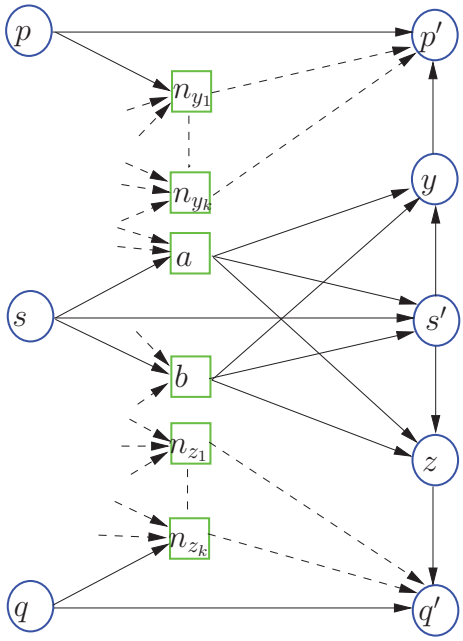


Figure 8.2. Cross-product MDP corresponding to a 2 agent Dec-POMDP

upon receiving observation y_i. The same is true for all variables n_{z_i} of agent 2. Therefore the joint-action for the underlying cross-product MDP is

$$(a, b, n_{y_1}, \ldots, n_{y_k}, n_{z_1}, \ldots, n_{z_k}).$$

To simplify the notation, we denote the tuple $(n_{y_1}, \ldots, n_{y_k})$ by $\langle n_y \rangle$ and the tuple $(n_{z_1}, \ldots, n_{z_k})$ by $\langle n_z \rangle$.

Transition function: The transition function for this cross-product MDP is given by:

$$P(p', q', s' \,|\, p, q, s, a, b, \langle n_y \rangle, \langle n_z \rangle) = \sum_{y_r \in Y, z_l \in Z} P(p' \,|\, p, \langle n_y \rangle, y_r) P(q' \,|\, q, \langle n_z \rangle, z_l) O(y_r z_l \,|\, s', a, b) P(s' \,|\, s, a, b) \tag{8.5}$$

where distributions $P(p' \,|\, p, \langle n_y \rangle, y_r)$ and $P(q' \mid q, \langle n_z \rangle, z_l)$ are defined as:

$$P(p' \,|\, p, \langle n_y \rangle, y_r) = \begin{cases} 1 & \text{if } n_{y_r} = p' \\ 0 & \text{otherwise} \end{cases} \tag{8.6}$$

$$P(q' \,|\, q, \langle n_z \rangle, z_l) = \begin{cases} 1 & \text{if } n_{z_l} = q' \\ 0 & \text{otherwise} \end{cases} \tag{8.7}$$

Intuitively, the above distributions encode the fact that a complete assignment to variables $\langle n_y \rangle$ and $\langle n_z \rangle$ defines a deterministic controller transition function.

Similar to the single agent MDP linear program, we create parameters

$$x(p, q, s, a, b, n_{y_1}, \ldots, n_{y_k}, n_{z_1}, \ldots, n_{z_k})$$

by noting that the underlying state in the cross-product MDP is (p, q, s) and the action is $(a, b, n_{y_1}, \ldots, n_{y_k}, n_{z_1}, \ldots, n_{z_k})$. For notational simplicity, we represent this distribution compactly as $x(p, q, s, a, b, \langle n_y \rangle, \langle n_z \rangle)$. Notice that the total number of these variables is exponential in the size of observation space of each agent. Fortunately, we can exploit the independence structure present in the DBN of Figure 8.2 to reason with marginalized version of these variables. Details are presented in later sections.

8.4 Mixed-Integer Linear Program for Dec-POMDPs

8.4.1 Objective

The objective function of the cross-product MDP can be written as:

$$\max \sum_{p,q,s} \sum_{a,b} R(s, a, b) x(p, q, s, a, b) \tag{8.8}$$

Intuitively, the parameter $x(p, q, s, a, b)$ arises as a result of marginalizing out $\big(\langle n_y \rangle, \langle n_z \rangle\big)$ from $x(p, q, s, a, b, \langle n_y \rangle, \langle n_z \rangle)$. A precise definition relating the two will be presented later.

8.4.2 Variables and Constraints

We first describe the variables in the MILP formulation of table 8.1.

- $x(p, q, s, a, b)$ represents the total discounted amount of time when the controller of agent 1 is in state p, controller of agent 2 is in state q, the underlying state is s, agent 1 performed action a and agent 2 performed action b. The precise definition is presented in the later sections.

- $x(p, q, s, a, b, n_y = p', n_z = q')$ represents the total discounted amount of time when the controller of agent 1 is in state p, controller of agent 2 is in state q, the underlying state is s, agent 1 performed action a, the controller transition rule for agent 1 is to

transition to node p' if observation y is received, agent 2 performed action b and the the controller transition rule for agent 2 is to transition of node q' if observation z is received.

- $x(p, a)$ represents the total discounted amount of time when the controller of agent 1 is in state p and action taken is a. Variable $x(q, b)$ can be described analogously.
- $x(p)$ represents the total discounted amount of time when the controller of agent 1 is in state p. Variable $x(q)$ can be defined analogously.
- $x(p, p'_y)$ represents the total discounted amount of time when the controller of agent 1 is in state p and the controller transition rule for agent 1 is to transition to node p' if observation y is received. Variable $x(q, q'_z)$ can be described analogously.
- $x(a|p)$ represents the action selection parameter $\pi(\cdot)$ for agent 1. That is, if $x(a|p) = 1$, then action a is taken when the controller of agent 1 is in state p. It is an integer variables, which implies the resulting controller is deterministic. Variable $x(b|q)$ can be defined analogously.
- $x(p'|p, y)$ represents the controller node transition parameter $\lambda(\cdot)$ for agent 1. That is, if $x(p'|p, y) = 1$, then the control transfers to the node p' if the observation y is received and the current controller node is p. Variable $x(q'|q, z)$ can be defined analogously.

We now write the equivalent of the MDP flow constraint in Eq. (8.2).

$$\sum_{a,b} x(p', q', s', a, b) = \eta_0(p', q', s') + \gamma \sum_{p,q,s} \sum_{a,b,y,z} O(yz \mid s', a, b) P(s' \mid s, a, b) x(p, q, s, a, b, n_y = p', n_z = q') \quad (8.9)$$

To make sure that variables $x(p', q', s', a, b)$ and $x(p, q, s, a, b, n_y = p', n_z = q')$ are consistent with each other, we introduce the following constraints:

$$x(p,q,s,a,b) = \sum_{p',q',s',y,z} O(yz \mid s',a,b)P(s' \mid s,a,b)x(p,q,s,a,b,p'_y,q'_z) \tag{8.10}$$

$$x(p,q,s,a,b) = \sum_{y,z,p',q'} x(p,q,s,a,b,p'_y,q'_z) \tag{8.11}$$

where we have represented the assignment $n_y = p'$ as p'_y and the assignment $n_z = q'$ as q'_z. An explanation of the rest of the constraints is provided in the next section.

8.4.3 Correctness of the MILP for 2-Agent Dec-POMDP

The main idea behind the mixed-integer program of Table 8.1 is that one can interpret optimizing the Dec-POMDP policy as optimizing an MDP policy. The MDP in this regard is the cross-product MDP as shown in Figure 8.2. Therefore, we first write the constraints for the LP formulation of this cross-product MDP. From the known results about MDPs [116, Section 6.9], we can define the following:

Definition 16. *The parameter $x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle)$ is defined as:*

$$\sum_{j\in S,k\in\mathcal{P},l\in\mathcal{Q}} \eta_0(j,k,l)\sum_{t=1}^{\infty}\gamma^{t-1}P_t(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle \mid s_1=j,p_1=k,q_1=l) \tag{8.31}$$

where we use the notation that $P(u_t = u, v_t = v|\cdot)$ can be concisely represented as $P_t(u,v|\cdot)$, where u_t and v_t denote random variables for the time slice t and u, v are a particular assignment to theses random variables; $\cdot$ represents conditioning on the appropriate variables. Subscripts denote time (for example s_1 indicates the state random variable for time step 1), unless stated otherwise. The set of all controller nodes for agent 1 is denoted by $\mathcal{P}$ and for agent 2 by $\mathcal{Q}$.

The main issue while reasoning with variables $x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle)$ is that there are exponentially many such variables in the number of observations of both the agents as total assignments to the random vector $\langle n_y\rangle$, $\langle n_z\rangle$ are exponential in the size of the observation space. Fortunately, by exploiting the conditional independencies in the DBN of the cross-product MDP of Figure 8.2, we can substantially reduce the number of such variables to only *polynomial* in all model parameter. To achieve this, we define the parameter $x(p,q,s,a,b)$ as follows:

Variables: $x(p,q,s,a,b),\ x(p,q,s,a,b,p'_y,q'_z), x(p,a), x(q,b), x(p), x(q),$
$x(p,p'_y), x(q,q'_z), x(a|p), x(b|q), x(p'|p,y), x(q'|q,z)\ \forall\ p,q,s,a,b,y,z,p',q'$

Maximize: $$\sum_{p,q,s}\sum_{a,b} R(s,a,b)x(p,q,s,a,b) \tag{8.12}$$

Subject to:

$$\sum_{a,b} x(p',q',s',a,b) = \eta_0(p',q',s') + \gamma \sum_{p,q,s}\sum_{a,b,y,z} O(yz \mid s',a,b)P(s' \mid s,a,b)$$
$$x(p,q,s,a,b,n_y = p', n_z = q') \qquad \forall (p',q',s') \tag{8.13}$$

$$x(p,q,s,a,b) = \sum_{p',q',s',y,z} O(yz \mid s',a,b)P(s' \mid s,a,b)x(p,q,s,a,b,p'_y,q'_z) \quad \forall (p,q,s,a,b) \tag{8.14}$$

$$x(p,q,s,a,b) = \sum_{y,z,p',q'} x(p,q,s,a,b,p'_y,q'_z) \qquad \forall (p,q,s,a,b) \tag{8.15}$$

$$x(p,a) = \sum_{q,s,b} x(p,q,s,a,b) \qquad \forall (p,a) \tag{8.16}$$

$$x(q,b) = \sum_{p,s,a} x(p,q,s,a,b) \qquad \forall (q,b) \tag{8.17}$$

$$x(p) = \sum_{a} x(p,a) \qquad \forall p \tag{8.18}$$

$$x(q) = \sum_{b} x(q,b) \qquad \forall q \tag{8.19}$$

$$x(p,p'_y) = \sum_{q,s,a,b,q'} x(p,q,s,a,b,p'_y,q'_z) \qquad \forall (p,y,z,p') \tag{8.20}$$

$$x(q,q'_z) = \sum_{p,s,a,b,p'} x(p,q,s,a,b,p'_y,q'_z) \qquad \forall (q,y,z,q') \tag{8.21}$$

$$x(p) - x(p,a) \leq \frac{1 - x(a|p)}{1-\gamma} \qquad \forall (p,a) \tag{8.22}$$

$$x(q) - x(q,b) \leq \frac{1 - x(b|q)}{1-\gamma} \qquad \forall (q,b) \tag{8.23}$$

$$x(p) - x(p,p'_y) \leq \frac{1 - x(p'|p,y)}{1-\gamma} \qquad \forall (p,y,p') \tag{8.24}$$

$$x(q) - x(q,q'_z) \leq \frac{1 - x(q'|q,z)}{1-\gamma} \qquad \forall (q,z,q') \tag{8.25}$$

$$\sum_{a} x(a|p) = 1 \qquad \forall p \tag{8.26}$$

$$\sum_{b} x(b|q) = 1 \qquad \forall p \tag{8.27}$$

$$\sum_{p'} x(p'|p,y) = 1 \qquad \forall (p,y) \tag{8.28}$$

$$\sum_{q'} x(q'|q,z) = 1 \qquad \forall (q,z) \tag{8.29}$$

$$x(a|p) \in \{0,1\}, x(b|q) \in \{0,1\}, x(p'|p,y) \in \{0,1\}, x(q'|q,z) \in \{0,1\} \tag{8.30}$$

Table 8.1. Mixed-integer program for optimizing a 2-agent Dec-POMDP policy. The 0-1 binary variables are shown in Eq. (8.30). The rest are continuous, positive variables: $x(\cdot) \geq 0$.

Definition 17. *The variable $x(p, q, s, a, b)$ is defined as:*

$$x(p,q,s,a,b) = \sum_{\langle n_y\rangle,\langle n_z\rangle} x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) \tag{8.32}$$

$$= \sum_{j\in S,k\in\mathcal{P},l\in\mathcal{Q}} \eta_0(j,k,l)\sum_{t=1}^{\infty}\gamma^{t-1}P_t\big(p,q,s,a,b \mid s_1=j,p_1=k,q_1=l\big) \tag{8.33}$$

The above variable denotes the marginalization over exponentially sized random vector $\langle n_y\rangle$ and $\langle n_z\rangle$. Intuitively, it represents the total discounted amount of time when the controller of agent 1 is in state p, controller of agent 2 is in state q, the underlying state is s, agent 1 performed action a and agent 2 performed action b. It is straightforward to derive the above result from definition 16 as follows:

$$\begin{aligned}
x(p,q,s,a,b) &= \sum_{\langle n_y\rangle,\langle n_z\rangle} x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle)\\
&= \sum_{\langle n_y\rangle,\langle n_z\rangle}\sum_{j\in S,k\in\mathcal{P},l\in\mathcal{Q}} \eta_0(j,k,l)\sum_{t=1}^{\infty}\gamma^{t-1}P_t\big(s,p,q,a,b,\langle n_y\rangle,\langle n_z\rangle \mid s_1=j,p_1=k,q_1=l\big)\\
&= \sum_{j\in S,k\in\mathcal{P},l\in\mathcal{Q}} \eta_0(j,k,l)\sum_{t=1}^{\infty}\gamma^{t-1}\sum_{\langle n_y\rangle,\langle n_z\rangle} P_t\big(s,p,q,a,b,\langle n_y\rangle,\langle n_z\rangle \mid s_1=j,p_1=k,q_1=l\big)\\
&= \sum_{j\in S,k\in\mathcal{P},l\in\mathcal{Q}} \eta_0(j,k,l)\sum_{t=1}^{\infty}\gamma^{t-1}P_t\big(s,p,q,a,b \mid s_1=j,p_1=k,q_1=l\big)
\end{aligned}$$

Let us consider a particular observation y_r of agent 1 and z_l of agent 2; a particular controller node p' of agent 1 and q' of agent 2. Let us define the variable $x(p,q,s,a,b,n_{y_r}=p',n_{z_l}=q')$ as follows:

Definition 18. *The variable $x(p,q,s,a,b,n_{y_r}=p',n_{z_l}=q')$ is defined as follows:*

$$\begin{aligned}
x(p,q,s,a,b,n_{y_r}=p',n_{z_l}=q') &= \sum_{\langle n_y\rangle\setminus p'_{y_r},\langle n_z\rangle\setminus q'_{z_l}} x(p,q,s,a,b,p'_{y_r},q'_{z_l},\langle n_y\rangle\setminus p'_{y_r},\langle n_z\rangle\setminus q'_{z_l})\\
&= \sum_{j\in S,k\in\mathcal{P},l\in\mathcal{Q}} \eta_0(j,k,l)\sum_{t=1}^{\infty}\gamma^{t-1}P_t\big(p,q,s,a,b,p'_{y_r},q'_{z_l} \mid s_1=j,p_1=k,q_1=l\big)
\end{aligned}$$

where we use the notational simplification that the assignment $n_{y_r}=p'$ is represented as p'_{y_r}. The same is true for q'_{z_l}. The above definition can be easily derived by using the expression of the variable $x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle)$ from definition 16. Intuitively, the variable $x(p,q,s,a,b,n_{y_r}=p',n_{z_l}=q')$ represents the total discounted amount of time when the controller of agent 1 is in state p, controller of agent 2 is in state q, the underlying state is s, agent 1 performed action a, the controller transition rule for agent 1 is to transition to node p' if observation y_r is received when the current controller node is p, agent 2 performed action b and the the controller transition rule for agent 2 is to transition of node q' if observation z_l is received when the current controller node is q.

8.4.3.1 Justification For MILP Constraint (8.14) and (8.15)

We now investigate the relationship between variables $x(p, q, s, a, b, p'_{y_r}, q'_{z_l})$ and the variable $x(p, q, s, a, b, \langle n_y \rangle, \langle n_z \rangle)$. Although, these two variables are related to each other using the joint frequency variable $x(p, q, s, a, b, \langle n_y \rangle, \langle n_z \rangle)$, the key idea is to avoid using explicitly the joint frequency variables as they are exponential in number. Therefore, we next investigate an explicit constraint between the variable $x(p, q, s, a, b, p'_{y_r}, q'_{z_l})$ and the variable $x(p, q, s, a, b, \langle n_y \rangle, \langle n_z \rangle)$. This relationship will become a constraint in the MILP formulation as shown in Eq. (8.14). Using definition 17, the variable $x(p, q, s, a, b)$ is defined as:

$$
\begin{aligned}
&= \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} P_t\big(p, q, s, a, b \mid s_1 = j, p_1 = k, q_1 = l\big) \\
&= \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} \sum_{p', q', s', y_r, z_l, \langle n_y \rangle, \langle n_z \rangle} P_t\big(p, q, s, a, b, p', q', s', y_r, z_l, \langle n_y \rangle, \langle n_z \rangle \mid s_1 = j, p_1 = k, q_1 = l\big) \\
&= \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} \sum_{p', q', s', y_r, z_l, \langle n_y \rangle, \langle n_z \rangle} P\big(p' | p, y_r, \langle n_y \rangle\big) P\big(q' | q, z_l, \langle n_z \rangle\big) \\
&\qquad\qquad P_t\big(p, q, s, a, b, s', y_r, z_l, \langle n_y \rangle, \langle n_z \rangle \mid s_1 = j, p_1 = k, q_1 = l\big)
\end{aligned}
$$

where we have used the conditional independencies present in the DBN of Figure 8.2. Notice that the probability $P\big(p' | p, y_r, \langle n_y \rangle\big)$ is 1 only if $n_{y_r} = p'$; 0 otherwise. Using this fact, we can further simplify the above equation as:

$$
\begin{aligned}
&= \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} \sum_{p', q', s', y_r, z_l} \\
&\qquad \sum_{\langle n_y \rangle \setminus p'_{y_r}, \langle n_z \rangle \setminus q'_{z_l}} P_t\big(p, q, s, a, b, s', y_r, z_l, p'_{y_r}, q'_{z_l}, \langle n_y \rangle \setminus p'_{y_r}, \langle n_z \rangle \setminus q'_{z_l} \mid s_1 = j, p_1 = k, q_1 = l\big)
\end{aligned}
$$

We can marginalize the expression involving P_t over $\langle n_y \rangle \setminus p'_{y_r}$ and $\langle n_z \rangle \setminus q'_{z_l}$ to get:

$$
\begin{aligned}
&= \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} \sum_{p', q', s', y_r, z_l} P_t\big(p, q, s, a, b, s', y_r, z_l, p'_{y_r}, q'_{z_l} \mid s_1 = j, p_1 = k, q_1 = l\big) \\
&= \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} \sum_{p', q', s', y_r, z_l} O(y_r z_l \mid s', a, b) P(s' \mid s, a, b) P_t\big(p, q, s, a, b, p'_{y_r}, q'_{z_l} \mid s_1 = j, p_1 = k, q_1 = l\big)
\end{aligned}
$$

We can take the inner summation over states and observations outside to get:

$$
= \sum_{p', q', s', y_r, z_l} O(y_r z_l \mid s', a, b) P(s' \mid s, a, b) \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} P_t\big(p, q, s, a, b, p'_{y_r}, q'_{z_l} \mid s_1 = j, p_1 = k, q_1 = l\big)
$$

Now using the definition 18, we get the following constraint:

$$x(p,q,s,a,b) = \sum_{p',q',s',y_r,z_l} O(y_r z_l \,|\, s',a,b) P(s' \,|\, s,a,b) x(p,q,s,a,b,p'_{y_r},q'_{z_l}) \tag{8.34}$$

This is the constraint that specifies the relationship between the variable $x(p,q,s,a,b)$ and the variable $x(p,q,s,a,b,p'_{y_r},q'_{z_l})$, and is represented in the MILP constraint (8.14).

The MIP constraint (8.15) can be understood as marginalizing out the action variables p'_y and q'_z from the joint-distribution $x(p,q,s,a,b,p'_y,q'_z)$ and can be easily derived by applying the marginalization operation to the definition 18 to get the equivalent expression for $x(p,q,s,a,b)$ in definition 17.

8.4.3.2 Justification For MILP Constraint (8.13)

Intuitively, the MILP constraint (8.13) represents the flow constraint for the cross-product MDP. Using variables $x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle)$, this constraint can be written as:

$$\sum_{a,b,\langle n_y\rangle,\langle n_z\rangle} x(p',q',s',a,b,\langle n_y\rangle,\langle n_z\rangle) = \eta_0(p',q',s') + \gamma \sum_{p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle} P(p',q',s'|p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) \\ x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) \tag{8.35}$$

Using definition 17, the LHS of the above equation is equal to $\sum_{a,b} x(p',q',s',a,b)$. This is identical to the LHS of the MILP constraint (8.13). We now focus on the second term in the RHS. We have:

$$= \sum_{p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle} P(p',q',s'|p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) \tag{8.36}$$

$$= \sum_{p,q,s,a,b,y_r,z_l,\langle n_y\rangle,\langle n_z\rangle} P(p',q',s',y_r,z_l|p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) \tag{8.37}$$

$$= \sum_{p,q,s,a,b,y_r,z_l,\langle n_y\rangle,\langle n_z\rangle} P(p'|p,y_r,\langle n_y\rangle) P(q'|q,z_l,\langle n_z\rangle) P(s',y_r,z_l|p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) x(p,q,s,a,b,\langle n_y\rangle,\langle n_z\rangle) \tag{8.38}$$

From the DBN of Figure 8.2, we note that the variables (s',y_r,z_l) are independent of the variables $(p,q,\langle n_y\rangle,\langle n_z\rangle)$ given variables (s,a,b). Also, the distributions $P(p'|p,y_r,\langle n_y\rangle)$ and $P(q'|q,z_l,\langle n_z\rangle)$ are deterministic and are 1 only when $n_{y_r} = p'$ and $n_{z_l} = q'$. Therefore, we can simplify the above equation as:

$$= \sum_{p,q,s,a,b,y_r,z_l} P(s',y_r,z_l|s,a,b) \sum_{\langle n_y\rangle\setminus p'_{y_r},\langle n_z\rangle\setminus q'_{z_l}} x(p,q,s,a,b,p'_{y_r},q'_{z_l},\langle n_y\rangle\setminus p'_{y_r},\langle n_z\rangle\setminus q'_{z_l}) \tag{8.39}$$

By using the expression for the variable $x(p,q,s,a,b,n_{y_r} = p',n_{z_l} = q')$ from definition 18, we have:

$$= \sum_{p,q,s,a,b,y_r,z_l} O(y_r z_l \mid s', a, b) P(s' \mid a, b) x(p, q, s, a, b, p'_{y_r}, q'_{z_l}) \tag{8.40}$$

Substituting the above simplification into the RHS of the flow constraint of Eq. (8.35), we get the MILP constraint (8.13).

8.4.3.3 Decentralization and Partial Observability Constraints

So far we have discussed how the constraints in the LP formulation of an MDP can be represented for the cross-product MDP without using exponentially many variables in the number of observations. In this section, we discuss the additional constraints that make the resulting policy of the cross-product MDP to be realizable in a partially observable multiagent setting. To see why this may not be trivially true, consider the variables $x(p, q, s, a, b)$. These variables imply that the selection of the joint-action (a, b) depends on the underlying state (p, q, s). However, no agent observes completely this joint-state during the execution time. Therefore, without additional constraints, the policy of the cross-product MDP is not useful for the multiagent case. The following results identify additional constraints that make the policy of the cross-product MDP valid for partially observable multiagent settings.

Theorem 10. *The MILP constraints* (8.22) *and* (8.23) *along with the integrality constraints of Eq.* (8.30) *guarantee that the action selection parameter of the cross-product MDP policy is executable in a decentralized and partially observable environment.*

Proof. We prove the result for agent 1, it follows automatically for the agent 2. We know from MILP constraints that:

$$x(p) = \sum_a x(p, a) \tag{8.41}$$

$$x(p) \geq x(p, a) \; \forall \; a \tag{8.42}$$

As the variable $x(a|p)$ is an integer, we consider a particular action a such that $x(a|p) = 1$. The rest of the action selection parameters will be zero due to the constraint that $\sum_a x(a|p) = 1$. Substituting the value $x(a|p) = 1$ to the constraint (8.22), we have:

$$x(p) - x(p, a) \leq 0 \tag{8.43}$$

$$x(p) \leq x(p, a) \tag{8.44}$$

From inequalities (8.42) and (8.44), we infer that:

$$x(p) = x(p, a) \tag{8.45}$$

That is, whenever the controller of the agent 1 is in state p, it always executes the action a corresponding to the variable $x(a|p) = 1$. This is exactly how a policy in a decentralized and partially observable environment should look like. The action selection depends only on the controller state of the particular agent and not on the underlying state or the controller state of other agents.

When the variable $x(a|p) = 0$, that is, when action a is not performed at controller state p, then we have:

$$x(p) - x(p, a) \leq \frac{1}{1-\gamma} \tag{8.46}$$

This inequality also holds as the maximum value the variable $x(p, a)$ can get is $\frac{1}{1-\gamma}$. The following equation based on definition 17 and constraint (8.18) explains it:

$$x(p) = \sum_{q,s,a,b} \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} P_t\big(p, q, s, a, b \mid s_1 = j, p_1 = k, q_1 = l\big) \tag{8.47}$$

$$= \sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j,k,l) \sum_{t=1}^{\infty} \gamma^{t-1} P_t\big(p \mid s_1 = j, p_1 = k, q_1 = l\big) \tag{8.48}$$

$$\leq \gamma^0 + \gamma^1 + \gamma^2 + \ldots = \frac{1}{1-\gamma} \tag{8.49}$$

Using the above results, we conclude that constraints (8.22) and (8.23) along with integrality constraints ensure that the action selection part of the cross-product MDP policy is executable in a decentralized and partially observable setting. □

Theorem 11. *The MILP constraints* (8.24) *and* (8.25) *along with the integrality constraints of Eq.* (8.30) *guarantee that the controller node transition parameter of the cross-product MDP policy is executable in decentralized and partially observable environment.*

Proof. We prove the result for agent 1, it follows automatically for the agent 2. We can infer from the MILP constraint (8.20) that:

$$x(p) = \sum_{p'} x(p, p'_y) \; \forall \; y \tag{8.50}$$

$$x(p) \geq x(p, p'_y) \; \forall \; p'_y \tag{8.51}$$

As the variable $x(p'|p, y)$ is an integer, we consider a particular node p' such that $x(p'|p, y) = 1$. The rest of the node transition parameters will be zero due to the constraint that $\sum_{p'} x(p'|p, y) = 1$. Substituting the value $x(p'|p, y) = 1$ to the constraint (8.24), we have:

$$x(p) - x(p, p'_y) \leq 0 \tag{8.52}$$

$$x(p) \leq x(p, p'_y) \tag{8.53}$$

From inequalities (8.51) and (8.53), we infer that:

$$x(p) = x(p, p'_y) \tag{8.54}$$

The above equality implies that whenever the controller of agent 1 is in state p, then the controller node transition rule upon receiving the observation y is to transition deterministically to the node p'. Thus, the controller transition parameters of an agent do not depend on the underlying environment state s or the controller state q of the other agent. This satisfies the requirement of a decentralizable policy in a partially observable environment.

When the variable $x(p'|p, y) = 0$, then we have:

$$x(p) - x(p, p'_y) \leq \frac{1}{1 - \gamma} \tag{8.55}$$

The above inequality is always satisfied as we know that $x(p) \leq \frac{1}{1-\gamma}$. Therefore, the theorem holds. □

8.4.4 Complexity

The complexity of the MILP in table 8.1 is governed by the number of variables and constraints. The number of continuous variables are dominated by the number of variables $x(p, q, s, a, b, p'_y, q'_z)$. The total number of such variables is $(k^4|S||A|^2|Y|^2)$, assuming that both the agents have the same number of actions and observations; k denotes the controller size of an agent. The number of these continuous variables can become large for larger benchmarks. However, they are still polynomial in all problem parameters. Furthermore, the actual number of variables can be significantly reduced by exploiting the structure

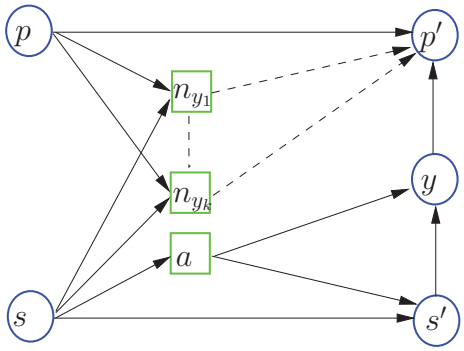


Figure 8.3. Cross-product MDP corresponding to a single agent POMDP

present in the conditional probability tables of the given instance, such as nearly deterministic transition and observation functions. Moreover, the complexity of solving the MILP mainly depends on number of integer variables on which the branching is done in the search tree.

The number of integer variables are $2 \times (k|A| + k^2|Y|)$. They increase linearly w.r.t. the action and observation space, and quadratically w.r.t to the size of the controller. Fortunately, unlike continuous variables, which are defined for the joint parameters of both the agents, the number of integer variables are defined *independently* for each agent's FSC. This is a significant advantage of our MILP formulation, as otherwise a large number of integer variables would have had a significantly negative impact on the scalability of solving the MILP.

The total number of constraints are $O(k^2|S||A|^2+k^2|Y|+k|A|)$. Importantly, the number of constraints increase linearly w.r.t. the state and observation space and quadratically w.r.t. the action space. Furthermore, the number of these constraints can be reduced significantly by considering the structure present in a particular problem instance. The number of constraints which involve integer variables is quite small: $O(4k|A| + 2k^2|Y| + 2k|Y|)$.

8.5 Mixed-Integer Linear Program for POMDPs

As there is no existing mixed integer programming formulation for single agent POMDPs, table 8.2 presents a MILP formulation for POMDPs. The constraints in this formulation are derived by ignoring the second agent's variables from the MILP formulation of Dec-

Variables: $x(p,s,a),\ x(p,s,a,p'_y), x(p,a), x(p), x(p,p'_y), x(a|p), x(p'|p,y)\ \forall\ p,s,a,y,p'$

Maximize: $$\sum_{p,s}\sum_{a} R(s,a)x(p,s,a) \tag{8.56}$$

Subject to:

$$\sum_{a} x(p',s',a) = \eta_0(p',s') + \gamma \sum_{p,s}\sum_{a,y} O(y \mid s',a)P(s' \mid s,a)x(p,s,a,n_y = p') \quad \forall (p,s) \tag{8.57}$$

$$x(p,s,a) = \sum_{p',s',y} O(y \mid s',a)P(s' \mid s,a)x(p,s,a,p'_y) \quad \forall (p,s,a) \tag{8.58}$$

$$x(p,s,a) = \sum_{p',y} x(p,s,a,p'_y) \quad \forall (p,s,a) \tag{8.59}$$

$$x(p,a) = \sum_{s} x(p,s,a) \quad \forall (p,a) \tag{8.60}$$

$$x(p) = \sum_{a} x(p,a) \quad \forall p \tag{8.61}$$

$$x(p,p'_y) = \sum_{s,a} x(p,s,a,p'_y) \quad \forall (p,y,p') \tag{8.62}$$

$$x(p) - x(p,a) \leq \frac{1 - x(a|p)}{1-\gamma} \quad \forall (p,a) \tag{8.63}$$

$$x(p) - x(p,p'_y) \leq \frac{1 - x(p'|p,y)}{1-\gamma} \quad \forall (p,y,p') \tag{8.64}$$

$$\sum_{a} x(a|p) = 1 \quad \forall p \tag{8.65}$$

$$\sum_{p'} x(p'|p,y) = 1 \quad \forall (p,y) \tag{8.66}$$

$$x(a|p) \in \{0,1\}, x(p'|p,y) \in \{0,1\} \tag{8.67}$$

Table 8.2. Mixed-integer program for optimizing a single agent POMDP policy. The 0-1 binary variables are shown in Eq. (8.67). The rest are continuous, positive variables: $x(\cdot) \geq 0$.

POMDPs. As expected, the size of the POMDP MILP is smaller than that of Dec-POMDP formulation. The number of integer variables in this formulation are $O(k^2|Y|)$, which are about half as many variables in a Dec-POMDP. This suggests that solving POMDP MILP may be more scalable than the Dec-POMDP MILP.

8.6 Extension to Multiple Agents—Lagrangian Relaxation

In this section, we present an extension of the MILP formulation for 2-agent Dec-POMDPs to handle multiple agents under the value factorization framework we introduced earlier—see definition 14 and Eq. (5.13). According to this property, the joint value can be factorized as:

$$V(\boldsymbol{\theta}, s) = \sum_{f \in F} V_f(\theta^f, s^f) \tag{8.68}$$

where f denotes subgroups of agents that interact together. Often, we can also decompose the value for the initial belief η_0 as in the ND-POMDP model [70]:

$$V(\boldsymbol{\theta}, \eta_0) = \sum_{f \in F} \sum_{s^f \in S^f} \eta_0(s^f) V_f(\theta^f, s^f) \tag{8.69}$$

Ideally, to find the best joint-policy for a particular belief η_0, if each component of the joint value function could be optimized *independently* of each other, it would lead to significant computational savings and increase scalability. This is because in many planning problems the subgroups of agents that interact together is much smaller than the total number of agents.

However, this does not appear to be the case as each agent can be involved in multiple value factors. Therefore, if we optimize the policy for each value factor independently, then an agent can be assigned different policies in different value factors. Clearly, this leads to inconsistency. However, we can take an optimization based principled approach to retain the advantage of optimizing the policy for each value factor independently and to also minimize inconsistency between different policies computed for a single agent. A technique called Lagrangian relaxation allows us to achieve such a decoupling [17].

Optimizing the joint-policy for all the agents gives rise to the following optimization problem:

$$\max_{\boldsymbol{\theta} \in \Theta} \sum_{f \in F} \sum_{s^f \in S^f} \eta_0(s^f) V_f(\theta^f, s^f) \tag{8.70}$$

where Θ is the joint-parameter space. Let us denote an agent by index i. If an agent is involved in value factor f, then we denote its policy by θ_i^f. The idea behind consistency is that if an agent is involved in two value factors f_1 and f_2, then it should be assigned the same policy. That is $\theta_i^{f_1} = \theta_i^{f_2}$. We explicitly write this constraint in the optimization problem as:

$$\min_{\theta^f \in \Theta^f \ \forall f \in F,\ k_i \ \forall i} -\sum_{f \in F} \sum_{s^f \in S^f} \eta_0(s^f) V_f(\theta^f, s^f) \tag{8.71}$$

$$\text{Subject to:} \quad \theta_i^f = k_i \ \forall i,\ \forall\ f \in F \tag{8.72}$$

Notice that we changed the sign of the optimization problem to make it a minimization problem. In the above optimization problem, we let k_i represents some arbitrary policy for agent i. The above optimization problem is maintaining consistency among different copies of the policy for an agent i by forcing all of them to be equal to k_i. The above optimization problem is still hard to solve due to the coupling constraint (8.72). We relax the above constraint by writing the Lagrangian function as:

$$L(\langle \theta^f \rangle, \boldsymbol{\lambda}) = -\sum_{f \in F} \sum_{s^f \in S^f} \eta_0(s^f) V_f(\theta^f, s^f) + \sum_i \sum_{f \in F_i} \lambda_i^f (\theta_i^f - k_i) \tag{8.73}$$

where F_i denotes all the value factors f such that agent i is involved those value factors. The variable λ_i^f is a dual variable introduced for each constraint in Eq. (8.72). Using the above Lagrangian function, we can write the dual of optimization problem (8.71) as:

$$q(\boldsymbol{\lambda}) = \min_{\theta^f \in \Theta^f \ \forall f \in F,\ k_i \ \forall i} -\sum_{f \in F} \sum_{s^f \in S^f} \eta_0(s^f) V_f(\theta^f, s^f) + \sum_i \sum_{f \in F_i} \lambda_i^f (\theta_i^f - k_i) \tag{8.74}$$

$$= \min_{\theta^f \in \Theta^f \ \forall f \in F,\ k_i \forall i} \sum_{f \in F} \Big\{ -\sum_{s^f \in S^f} \eta_0(s^f) V_f(\theta^f, s^f) + \sum_i \lambda_i^f \theta_i^f \Big\} - \sum_i k_i \sum_{f \in F_i} \lambda_i^f \tag{8.75}$$

The dual function has the constraint that $q(\boldsymbol{\lambda}) \geq -\infty$ for all the $\boldsymbol{\lambda}$ in its domain. In the above optimization problem, notice that the variable k_i is unconstrained. Therefore, minimizing over it can make the dual unbounded from below. To prevent this, we introduce the following set of constraints:

$$\Lambda_i = \big\{ \{\lambda_i^f\}_{f \in F_i} \mid \sum_{f \in F_i} \lambda_i^f = 0 \big\} \ \forall i \tag{8.76}$$

By using the above condition, the expression for the dual simplifies to:

$$q(\boldsymbol{\lambda}) = \sum_{f \in F} \left\{ \min_{\theta^f \in \Theta^f \ \forall f \in F} - \sum_{s^f \in S^f} \eta_0(s^f) V_f(\theta^f, s^f) + \sum_i \lambda_i^f \theta_i^f \right\} \tag{8.77}$$

Notice that, in the above optimization problem, the dual function can be evaluated by solving the inner optimization problem for each value factor *independently*. Furthermore, the dual solution provides a quality bound for our original optimization problem (8.71).

Theorem 12. *For every value of dual variables $\boldsymbol{\lambda}$, the dual solution satisfies the inequality*

$$q(\boldsymbol{\lambda}) \leq V^\star$$

where $v^\star$ is the optimal solution to the joint-policy optimization problem (8.71)*.*

Using the above result, we know that the dual solution always provides a quality bound for the optimal solution. Based on this result, our new optimization problem is to find dual variables $\boldsymbol{\lambda}$ such that the dual function $q(\boldsymbol{\lambda})$ is maximized:

$$\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} q(\boldsymbol{\lambda}) \tag{8.78}$$

where $\boldsymbol{\Lambda}$ is the intersection of all the constraints Λ_i for each agent i. There are standard techniques to solve the dual maximization problem such as dual sub-gradient optimization [17]. In such a sub-gradient based technique, we can calculate a sub-gradient $\nabla(\boldsymbol{\lambda})$ of the function $q(\boldsymbol{\lambda})$ given a particular value of $\boldsymbol{\lambda}$. Then we change the dual variables by taking a step in the direction of the sub-gradient:

$$\boldsymbol{\lambda}' \rightarrow \left[\boldsymbol{\lambda} + \alpha \nabla(\boldsymbol{\lambda})\right]_{\boldsymbol{\Lambda}} \tag{8.79}$$

where $[\cdot]_{\boldsymbol{\Lambda}}$ denotes the projection onto the constraint set $\boldsymbol{\Lambda}$; α denotes the step parameter; $\boldsymbol{\lambda}$ is the previous iteration's estimate of the dual variable and $\boldsymbol{\lambda}'$ denotes the new estimate of the dual variable. The sub-gradient itself is readily available from the solution of each inner minimization problem in (8.77). Furthermore, the dual is a concave function. Therefore, there are *no local optima* in the dual and under some regularity assumption on the set size α, the sub-gradient technique will converge to the optimal dual solution. However, there might be duality gap, which implies that the dual optimal solution quality may not be the same as the primal optimal solution quality.

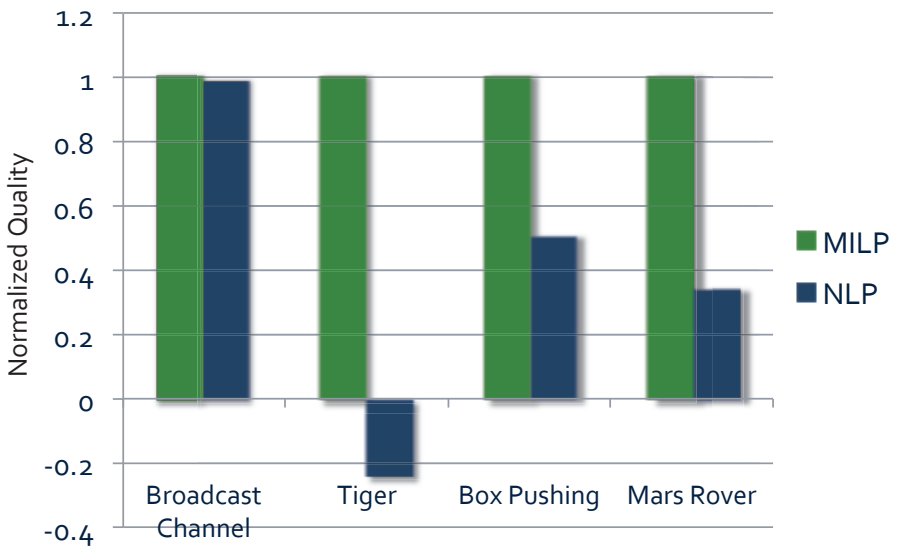


Figure 8.4. Quality comparison between the Dec-POMDP MILP solved using CPLEX and the NLP-based solver. Solution quality (y-axis) is normalized, with 1 representing the optimal solution.

Definition 19. *The duality gap can be measured by using the difference between the dual optimal and primal optimal solution*

$$Duality\ Gap\ = V^{\star} - q(\lambda^{\star}),$$

where $\lambda^{\star}$ is the optimal dual solution and $V^{\star}$ is the optimal primal solution quality.

Often in real-world problems the duality gap is small [17]. Furthermore, one can always estimate an upper bound on the duality gap by measuring $V - q(\boldsymbol{\lambda})$, where V is our current primal solution and $\boldsymbol{\lambda}$ denotes the current iteration's estimate of the dual variables. Therefore, during the execution of the sub-gradient technique, we always know how far the current solution is from the optimum. A significant advantage of such a dual decomposition based technique is its ability to scale well to large multiagent systems by optimizing each value factor independently and at same time encouraging consistency among different value factors using dual variables.

Problem	|S|	|A|	|Y|
Broadcast Channel	4	2	5
Multiagent Tiger	2	3	2
Box Pushing	100	4	5
Mars Rover	256	6	8

Table 8.3. Problem size for different instances; $|S|$ represents the state-space, $|A|$ represents the action-space per agent and $|Y|$ represents the observation-space per agent.

8.7 Experiments

We experimented with several standard 2-agent Dec-POMDP benchmarks with discounting factor $\gamma = 0.9$. We compare the MILP-based approach with the non-linear, non-convex optimization solver (NLP) [6], which is one of the state-of-the-art solver for solving 2-agent infinite-horizon Dec-POMDPs. More details about the NLP formulation can be found in Section 6.2. We used CPLEX 12.0 to solve the MILP (8.1) for the given Dec-POMDP instance. We compare the resulting solution quality and the execution time of the CPLEX with the NLP solver. We also report if the CPLEX was able to find the optimal solution for the given Dec-POMDP instance.

Figure 8.4 shows the solution quality comparison on a number of Dec-POMDP benchmarks between the CPLEX and the NLP solver. The size of these problems are shown in Table 8.3. As it is clear from this figure, CPLEX is able to provide high quality solutions for the MILP, significantly better than the NLP solver for all the problems. One key reason is that the NLP solver is solving a non-convex problem and therefore, is susceptible to local optima. The CPLEX, which employs a branch-and-bound procedure, can provide the optimal solution given sufficient time and space, otherwise it can be terminated early, still providing a non-trivial upper bound.

Table 8.4 shows the runtime comparison between CPLEX and the NLP solver for a number of problems. The runtime for the NLP solver are taken from [2, 6]. We used the same setting for the NLP formulation as reported in previously published works. For the MILP formulation, we chose to optimize a reactive controller [84]. A reactive controller represents a controller design in which each agent memorizes the last observation received and takes the next action based on the last observation. We included an additional start

Problem	MILP			NLP	
	Time	Optimal	# FSC Nodes	Time	# FSC Nodes
Broadcast Channel	1	yes	4	1.1	4
Multiagent Tiger	1.1	yes	7	6137.0	19
Box Pushing	1.9	yes	6	1824.0	4
Mars Rover	13.5	yes	9	43.0	2

Table 8.4. Runtime comparison between CPLEX for the Dec-POMDP MILP and the NLP solver.

node for the reactive controller. The advantage of such reactive controllers is that the controller structure is fixed. The search space only includes the mapping of actions to the controller nodes. Finding optimal reactive controllers is itself a non-trivial problem, being NP-Hard even for a single-agent POMDP [84]. However, the MILP formulation for a Dec-POMDP provides high quality information to CPLEX to guide its heuristic search. This results in CPLEX being orders-of-magnitude faster than the NLP solver as shown in Table 8.4. Furthermore, CPLEX was able to find optimal reactive controller for each agent and for each problem. The solution quality achieved by these reactive controller is significantly better than the NLP solver as shown in Figure 8.4. The 'Mars Rover' problem, which is one of the largest 2-agent Dec-POMDP problem can be solved optimally by CPLEX within a few seconds. This clearly highlights the power of using mixed-integer programming and the industrial strength MILP solvers such as CPLEX.

Our future work includes taking the optimization based formulations to the next step, namely solving factored multiagent planning models by exploring the domain structure using the Lagrangian relaxation based approach of Section 8.6.

8.8 Discussion

In this chapter, we addressed the question of how to achieve bounded optimality for sequential decision making under partial observability. We presented mixed integer programming based formulation to optimize agents' policies represented as finite-state controllers under the Dec-POMDP framework. The connections to mixed integer programming lays the foundation to providing quality bounds for sequential multiagent decision making problems and incorporating advanced optimization based techniques such as Lagrangian relaxation

to handle large multiagent systems. Progress in this direction has been slow so far due to the inherent difficulty of the highly non-linear nature of planning in partially observable and multiagent settings.

Empirically, using the MILP formulation developed in our work, CPLEX could solve standard Dec-POMDP benchmarks to optimality orders-of-magnitude faster than existing approaches. The solution quality achieved by CPLEX was also significantly better than previous approaches, that could get stuck in a local optimum. These encouraging results provide an initial glimpse towards significant advances that can be made possible by bringing optimization based techniques to multiagent planning.

Our work is the first to present a mixed integer programming formulation for both single agent POMDPs and Dec-POMDPs. Using mixed integer programs is advantageous than using other non-linear formulations as there are already highly efficient solvers such as CPLEX available for solving them. Such optimization based techniques hold a significant potential to address quality bounded planning in large multiagent systems. Our work in this chapter only starts to explore the great promise of a number of other possibilities for bringing the perspective of optimization techniques into multiagent systems

The main publication which illustrates the application of Lagrangian relaxation technique similar to the one presented in Section 8.6 to large mixed integer programs in the context of planning for spatial conservation is described is the following:

- A. Kumar, X. Wu and S. Zilberstein. Lagrangian Relaxation Techniques for Scalable Spatial Conservation Planning. In Proc. of the international conference on Artificial Intelligence (AAAI), pages 309–315, 2012.

CHAPTER 9

SUMMARY

In this thesis, we addressed the question of how can multiple agents coordinate to achieve a common goal within a decision-theoretic framework. This thesis developed several novel techniques to enable scalable and quality-bounded decision making in multiagent systems. The key insight was to exploit the domain structure in a multiagent system in an algorithmic framework based on synthesis of techniques from artificial intelligence, machine learning and operations research. Despite the proven hardness results about multiagent decision-theoretic planning, the techniques developed in this thesis were successful in pushing the envelope of scalability by opening the door to the application of optimization and ML based approaches to multiagent systems.

We used two rich frameworks to model multiagent decision making problems—*distributed constraint optimization* and *decentralized partially observable MDPs*. Reasoning and planning within both these frameworks is challenging—NP-Hard for DCOPs and NEXP-Hard for Dec-POMDPs. Therefore, techniques developed in this thesis leveraged domain structure by analyzing and exploiting structured interactions and independencies present in a large multiagent system. Such structure, often represented within the broad framework of graphical models such as Markov random field and dynamic Bayesian network, was then embedded in an optimization framework that formed the basis for developing various efficient and scalable algorithms.

Often, the optimization problem, even after exploiting such structured interactions, remains challenging. We then used the *variational principle* from machine learning, in which the optimization was carried out over a representative subset of the complete parameter space, to develop efficient approximate algorithms. For example, we developed the hybrid bound for DCOPs in Chapter 3 and used finite-memory policies for agents within the Dec-POMDP framework, as shown in Chapter 4. Algorithmically, we developed a

synthesis of different techniques from multiple sub-areas of AI and operations research to address the scalability of algorithms for decision-theoretic reasoning and planning. We drew upon various mathematical optimization frameworks such as convex optimization, difference-of-convex functions programming and linear programming as a toolbox for developing rigorously grounded approximate algorithms. We also provided optimality preserving transformations which transform a multiagent planning problem to that of parameter learning problem that can be handled by using a number of learning techniques from the machine learning literature.

Next, we highlight our specific contributions for single-step decision making problems and sequential decision making problems in multiagent systems.

9.1 Single-Step Decision Making

We used the framework of distributed constraint optimization (DCOP) [95] to model the single-step multiagent decision making problems. We addressed a broader version of this problem by showing that the problem of finding the maximum a posteriori (MAP) assignment in Markov random fields [153] is equivalent to solving the DCOP under certain conditions in Chapter 2. Our main contribution is the development of a new variational framework, called the *hybrid bound*, for DCOP and the MAP problem, and a scalable message-passing algorithm to solve the optimization problem over the hybrid bound.

The two existing variational frameworks for the DCOP and MAP are based on linear programming and quadratic programming. The main advantage of the LP formulation is that it is convex and computationally tractable, thus there are no local optima. However, this formulation is *inexact*, implying that the optimum of the LP does not solve the MAP problem optimally. The QP formulation, on the other hand, is *exact*. However, it is non-convex. Thus, globally optimizing the QP is intractable. Our new variational framework combined the benefits of both the QP and the LP formulations, while minimizing effects of their undesired properties. We showed analytically as well as empirically that the proposed hybrid formulation provides a tighter approximation to the MAP problem than the LP based formulation, which translates into a significantly better solution quality. By judiciously choosing how to introduce the QP constraints in the hybrid formulation, we can control the

non-convexity of the hybrid formulation, which otherwise could lead to poor local optima. Thus, optimization over the hybrid formulation can avoid getting stuck in such poor local optima, which is the main obstacle for the QP formulation.

We also developed a message-passing algorithm called *Hybrid Belief Propagation* (HBP) that solves the optimization problem over the hybrid bound. This message-passing approach was developed by exploiting the connection between the MAP problem and the difference-of-convex function programming. This approach is also guaranteed to converge unlike other approaches such as max-sum [107, 41]. Such message-passing approach is also ideal for large multiagent systems as it only requires exchanging local, fixed-size messages among neighboring agents.

Empirically, we tested the HBP algorithm on a number of synthetic and real-world benchmarks from multiagent systems, operations research and machine learning literature. The DCOP benchmarks were based on a sensor scheduling domain. Our results showed that the HBP algorithm provided near-optimal solutions much faster than the state-of-the-art DCOP algorithms such as ADOPT and max-sum. The HBP algorithm was more than an order-of-magnitude faster than ADOPT. The max-sum algorithm, being also based on message-passing was faster than ADOPT, but it provided much worse solution quality than HBP. The error bound for some instances for the max-sum algorithm was as high as $\approx 135\%$, whereas it was $\approx 15\%$ for the HBP approach. Results on a number of other benchmarks from the OR and ML literature showed that the HBP approach provided significantly better solution quality than other state-of-the-art algorithms in the machine learning literature based on LP relaxation, such as MPLP [134].

9.2 Sequential Decision Making

Our main contributions for multiagent sequential decision making problems are along three directions:

- Development of efficient dynamic programming algorithms for *finite-horizon* decision making.

- Development of probabilistic inference based algorithms for *infinite-horizon* decision making.

- Development of mathematical programming based techniques for *quality bounded solutions* for both finite and infinite horizon decision making.

For *finite-horizon* decision making, we focused on the bottleneck dynamic programming step of several approximate point-based algorithms, also known as the *backup* problem. We investigated the computational characteristics of the backup problem for 2-agents and showed that it is NP-Hard. Despite this negative result, we presented an efficient and scalable optimal algorithm. We showed how this problem can be mapped to a weighted constraint satisfaction problem (WCSP) and can be solved using state-of-the-art WCSP solvers. Our results showed that bringing the perspective of constraint optimization helped solve the backup problem more than an order-of-magnitude faster than state-of-the-art solver PBIP [36]. We also investigated the backup problem in other restricted sub-classes of Dec-POMDPs, such as ND-POMDPs, that can model larger multiagent systems. We again showed how to solve this problem efficiently using WCSP solvers. This approach provided magnitudes of speedup in the policy computation and generated better quality solution for all test instances than earlier approaches [146, 88, 70].

For *infinite-horizon* decision making, our main contribution was the development of probabilistic inference based approaches to optimize agents' policies represented as finite-state controllers and an analysis of general conditions that can make such inference-based approach scalable to larger ($\gg 2$) multiagent systems. We presented a promising new class of algorithms for the infinite-horizon case for 2-agent Dec-POMDPs, which recasts the optimization problem as inference in a mixture of dynamic Bayesian networks (DBNs). An attractive feature of this approach is the development of new insights that connect inference techniques for DBNs with the multiagent planning, especially for supporting richer representations such as factored or continuous states and actions, which so far have been outside of the scope of existing algorithms. We performed inference in such DBNs by adapting the well known Expectation Maximization (EM) algorithm from the machine

learning literature to multiagent systems. Experiments on benchmark domains showed that EM compared favorably against the state-of-the-art solvers.

We also provided a general characterization of the interaction among agents that when present in a multiagent planning model leads to a relatively scalable probabilistic inference based approximate algorithm. We identified such conditions based on the joint-value factorization property, that enhances algorithmic scalability w.r.t. the number of agents. We also developed a scalable, message-passing algorithm that can solve such planning problems using the EM framework. The EM framework is particularly suited for developing algorithms for such restricted models as the EM approach works directly on the DBN representation of the planning problem. Such graphical representation facilitates significant exploitation of the independencies present in the planning model than the existing nonlinear programming approach. Empirical results on a large multiagent benchmarks with up to 20 agents and state and action spaces as large as $O(3^{20})$ showed that our approach scaled well. The nonlinear programming approach, on the other hand, could only solve the smallest instance with 5 agents due to large increase in the number of nonlinear constraints.

Finally, we developed a novel framework based on mathematical programming that can achieve *bounded optimality* for multiagent sequential decision making problems. The main drawback of previous approaches is their inability to provide any quality bound for an approximate solution. Our approach, that formulates the problem of optimizing finite-state controllers as a mixed integer program (MILP), remedies this drawback. Once we have a MILP representation of the planning problem, we can then use off-the-shelf and highly efficient MILP solvers such as CPLEX to solve the MILP to optimality or provide non-trivial upper bounds. Previously, several attempts have been made to formulate both the single agent and multiagent planning problems under uncertainty using mathematical programming [114, 24, 3, 4, 15, 6]. However, most of these approaches either resulted in a non-convex program, which suffers from the problem of local optima or they approximate the policy optimization using a convex program, which does not guarantee accuracy. We resolved these problems by interpreting the *multiagent* planning problem under *partial observability* as a *single agent* planning problem under *full observability*. We then incorporated a small number of constraints into the MILP which guarantee that the resulting policy will work

for the multiagent setting under partial observability. Empirically, we showed that our MILP based approach worked quite well, often providing optimal controller-based policies within a few seconds even for large 2-agent Dec-POMDP benchmarks that were previously considered intractable.

Such connections to the MILP formulation also provides the groundwork for applying advanced mathematical programming techniques to planning. Mixed integer programming is one of the heavily researched area in the mathematical optimization community unlike non-convex programming. There are several techniques that can approximate the solution to large mixed integer programs while also providing quality bounds. We investigated one such technique, Lagrangian relaxation, in the context of multiagent planning models with value-factorization property. Our work in this context only starts to explore the great promise of a number of other possibilities for bringing the perspective of optimization techniques into multiagent systems

9.3 Future Directions

The thesis explores several new directions in multi-agent reasoning and planning that merit significant further examination and analysis. We highlight several questions that are of particular interest.

In the context of single-step decision making problems and the variational framework of the hybrid bound, one open question is how to determine the edges for which the QP constraint needs to be enforced. In the current work, we provided a heuristic that worked by identifying a number of spanning trees of the graphical model. However, a better future approach would be to analyze the solution from the previous iteration and identify certain consistency properties that are violated by such solution. One such class of consistency properties is based on *cycle inequalities* that any valid solution must satisfy [131, 129]. A better approach to identify the edges for which the QP constraint should be enforced is to identify edges that contribute most to the violation of such valid cycle inequalities. Such a technique has significant potential to minimize the adverse effect of local optima and can tighten the LP relaxation to provide better quality solutions.

Extending LP and QP based variational frameworks to resource constrained DCOPs, also known as RC-DCOPs [20], is another important future research direction. Multiagent coordination problems that involves resource constraints, such as line capacity constraints in smart power grids [74], present a significant challenge for DCOP algorithms. The traditional way to handle such resource constrained problems is to create *virtual agents* corresponding to each resource. During the search procedure, these virtual agents assign infinite cost to any assignment that violates resource constraints. However, such search procedure leads to highly connected and dense graphs, which are harder to solve. A better approach would be to solve the LP relaxation of such resource constrained problems using message-passing. In the MAP estimation community, resource constraints are typically not used. Such constraints are the hallmark of realistic multiagent coordination problems. Therefore, developing message-passing algorithms for the LP relaxation of such problems would be a novel contribution.

In the context of inference based approaches for sequential decision making, one future direction is the analysis of local optima in the EM algorithm when applied to planning. Approximate algorithms can get stuck into a local optimum. However, if we can analyze the properties of such local optima and devise a way to escape them, then this would address one of the main drawbacks of such approximate algorithms. There has been work on analyzing the properties of the EM algorithm for single agent POMDPs [115]. Extending such local optima analysis to the EM algorithm in the multiagent setting remains an important future work.

Another future direction within the context of planning-by-inference strategy is the incorporation of sampling based techniques for multiagent planning. So far, most of the work in the Dec-POMDP community addresses discrete problems. There is an acute shortage of techniques that can handle problems with continuous model parameters such as continuous state or action spaces. Inference based approaches are a great candidate to fill this void. In the EM algorithm, the main challenge is the E-step, which requires performing a probabilistic inference query in a mixture of DBNs. Incorporating sampling based approaches, such as Monte-Carlo Markov chain (MCMC) [63], can provide the much needed ability to per-

form inference in continuous domains and thus, significantly enrich the class of multiagent planning problems that can be handled presently.

Another promising direction is based on the connection between multiagent planning and likelihood maximization that we established in this thesis. There are sampling based approaches, such as MCMC-MLE [46], for maximum likelihood estimation. Investigating such approaches in the context of multiagent planning is a potentially fruitful research direction. One advantage while working with such sampling based approaches is that they often have good convergence guarantees, such as convergence to global optima given enough samples. Having such properties in sampling based multiagent planning algorithms would prove to be very useful.

The optimization based approaches that we developed in Chapter 8 open up a number of possibilities for bringing advanced optimization based techniques to multiagent planning. There exist several approaches that can approximately solve large mixed integer programs such as Lagrangian relaxation and branch-and-price. Adapting such approaches within multiagent systems has great promise to increase the scalability of existing approaches. Once we have a precise mathematical programming formulation of a Dec-POMDP, then we can use techniques based on the variational framework that analyze the given problem through the lens of optimization. One can often find suitable simplifying approximations to complicating constraints, which lead to a more tractable problem. This technique has significant promise to handle large, weakly-coupled multiagent systems, such as those that satisfy the value-factorization property.

To summarize, this thesis has explored several new directions in multi-agent reasoning and planning including connections with machine learning and optimization. In the current era of modern computing with the proliferation of smart sensing devices, embedded within the so-called *Internet-of-things*, making effective long-term decisions is crucial. Algorithmic advances in multiagent decision making are going to be the key to usher in the age of such autonomous agents. This thesis has established several new insights that provide access to a number of tools from the wide literature of machine learning and optimization. Exploring deeper connections along this direction will go a long way towards addressing the challenges of multiagent decision making and create a significant practical impact.

BIBLIOGRAPHY

[1] S. Aji, G. Horn, R. McEliece, and M. Xu. Iterative min-sum decoding of tail-biting codes. In *Information Theory Workshop, 1998*, pages 68–69, 1998.

[2] C. Amato and S. Zilberstein. Whats worth memorizing: Attribute-based planning for Dec-POMDPs. In *ICAPS Multiagent Planning Workshop*, 2008.

[3] C. Amato, D. S. Bernstein, and S. Zilberstein. Solving POMDPs using quadratically constrained linear programs. In *International Joint Conference on Artificial Intelligence*, pages 2418–2424, 2007.

[4] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *International Conference on Uncertainty in Artificial Intelligence*, pages 1–8, 2007.

[5] C. Amato, J. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *International Conference on Automated Planning and Scheduling*, pages 2–9, 2009.

[6] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3): 293–320, 2010.

[7] R. Aras and A. Dutech. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research*, 37:329–396, 2010.

[8] D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm. In *International Conference on Artificial Intelligence and Statistics*, pages 146–154, 2011.

[9] R. Becker. *Exploiting Structure in Decentralized Markov Decision Processes*. PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 2006.

[10] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized Markov decision processes. In *International Conference on Autonomous Agents and Multi Agent Systems*, pages 41–48, 2003.

[11] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov decision processes with event-driven interactions. In *International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 302–309, 2004.

[12] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[13] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27:819–840, 2002.

[14] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *International Joint Conference on Artificial Intelligence*, pages 1287–1292, 2005.

[15] D. S. Bernstein, C. Amato, E. A. Hansen, and S. Zilberstein. Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research*, 34:89–132, 2009.

[16] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Cambridge, MA, USA, 1999.

[17] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[18] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B*, 36(2):192–236, 1974.

[19] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[20] E. Bowring, M. Tambe, and M. Yokoo. Multiply-constrained distributed constraint optimization. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1413–1420, 2006.

[21] E. Bowring, J. Pearce, C. Portway, M. Jain, and M. Tambe. On k-optimal distributed constraint optimization algorithms: New bounds and algorithms. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 607–614, 2008.

[22] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[23] A. Carlin and S. Zilberstein. Value-based observation compression for DEC-POMDPs. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 501–508, 2008.

[24] L. Charlin, P. Poupart, and R. Shioda. Automated hierarchy discovery for planning in partially observable environments. In *Advances in Neural Information processing Systems*, pages 225–232, 2007.

[25] A. Chechetka and K. Sycara. An any-space algorithm for distributed constraint optimization. In *AAAI Spring Symposium on Distributed Plan and Schedule Management*, pages 33–40, 2006.

[26] A. Chechetka and K. Sycara. No-commitment branch and bound search for distributed constraint optimization. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1427–1429, 2006.

[27] K. C. K. Cheng and R. H. C. Yap. Constrained decision diagrams. In *AAAI Conference on Artificial Intelligence*, pages 366–371, 2005.

[28] R. Cogill and S. Lall. An approximation algorithm for the discrete team decision problem. *SIAM Journal on Control and Optimization*, 45(4):1359–1368, 2006.

[29] R. M. Corless, G. H. Gonnet, D. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. In *Advances in Computational Mathematics*, pages 329–359, 1996.

[30] T. Cour and J. Shi. Solving Markov random fields with spectral relaxation. *International Conference on Artificial Intelligence and Statistics*, pages 75–82, 2007.

[31] S. de Givry, F. Heras, M. Zytnicki, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *International Joint Conference on Artificial Intelligence*, pages 84–89, 2005.

[32] J. Dean and S. Ghemawat. MapReduce: A flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[33] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[34] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical society, Series B*, 39(1):1–38, 1977.

[36] J. S. Dibangoye, A.-I. Mouaddib, and B. Chaib-draa. Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 569–576, 2009.

[37] J. S. Dibangoye, F. A.-I. Mouaddib, and B. Chaib-draa. Toward error-bounded algorithms for infinite-horizon DEC-POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 947–954, 2011.

[38] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 136–143, 2004.

[39] B. Faltings, D. C. Parkes, A. Petcu, and J. Shneidman. Optimizing streaming applications with self-interested users using MDPOP. In *International Workshop on Computational Social Choice*, Amsterdam, The Netherlands, 2006.

[40] B. Faltings, T. Léauté, and A. Petcu. Privacy guarantees through distributed constraint satisfaction. In *International Conference on Intelligent Agent Technologies*, pages 350–358, 2008.

[41] A. Farinelli, A. Rogers, A. Petcu, and N. Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 639–646, 2008.

[42] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *International Conference on Autonomous Agents and Multiagent System*, pages 639–646, 2008.

[43] J. Feldman, M. Wainwright, and D. Karger. Using linear programming to decode binary linear codes. *IEEE Transactions on Information Theory*, 51(3):954 – 972, 2005.

[44] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[45] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In V. Lesser, C. Ortiz, and M. Tambe, editors, *Distributed Sensor Networks: A Multiagent Perspective*, pages 257–295. Kluwer, 2003.

[46] C. Geyer. Markov chain monte carlo maximum likelihood. In *Computer Science and Statistics: Symposium on the Interface*, pages 156–163, 1991.

[47] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. In *Advances in Neural Information Processing Systems*, volume 8, pages 472–478, 1995.

[48] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[49] A. Globerson and T. Jaakkola. Fixing Max-Product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information processing Systems*, pages 553–560, 2007.

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

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

[52] R. Greenstadt. An overview of privacy improvements to k-optimal DCOP algorithms. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1279–1280, 2009.

[53] T. Grinshpoun and A. Meisels. Completeness and performance of the apo algorithm. *Journal of Artificial Intelligence Research*, 33:223–258, 2008.

[54] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *National Conference on Artificial Intelligence*, pages 709–715, 2004.

[55] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *International Conference on Constraint Programming*, pages 222–236, 1997.

[56] M. Hoffman, N. de Freitas, A. Doucet, and J. Peters. An expectation maximization algorithm for continuous Markov decision processes with arbitrary rewards. In *International Conference on Artificial Intelligence and Statistics*, pages 232–239, 2009.

[57] M. Hoffman, H. Kueck, N. de Freitas, and A. Doucet. New inference strategies for solving Markov decision processes using reversible jump MCMC. In *International Conference on Uncertainty in Artificial Intelligence*, 2009.

[58] M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the real world: exploring unknown reward matrices with applications to mobile sensor networks. In *International Joint Conference on Artificial Intelligence*, pages 181–186, 2009.

[59] V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *International Conference on Machine Learning*, pages 503–510, 2010.

[60] R. Junges and A. L. C. Bazzan. Evaluating the performance of dcop algorithms in a real world, dynamic problem. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 599–606, 2008.

[61] C. Kiekintveld, Z. Yin, A. Kumar, and M. Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 133–140, 2010.

[62] C. L. Kingsford, B. Chazelle, and M. Singh. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21(7):1028–1039, 2005.

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

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

[65] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1568–1583, 2006.

[66] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:65–81, 2004.

[67] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *International Conference on Computer Vision*, pages 1–8, 2007.

[68] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531 –552, 2011.

[69] A. Kumar and S. Zilberstein. Event-detecting multi-agent MDPs: Complexity and constant-factor approximation. In *International Joint Conference on Artificial Intelligence*, pages 201–207, 2009.

[70] A. Kumar and S. Zilberstein. Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 561–568, 2009.

[71] A. Kumar and S. Zilberstein. MAP estimation for graphical models by likelihood maximization. In *Advances in Neural Information Processing Systems*, pages 1180–1188, 2010.

[72] A. Kumar and S. Zilberstein. Point based backup for decentralized POMDPs: Complexity and new algorithms. In *International Conference on Autonomous Agents and Multiagent Systems*, 2010.

[73] A. Kumar and S. Zilberstein. Message passing algorithms for quadratic programming formulation of MAP estimation. In *Conference on Uncertainty in Artificial Intelligence*, pages 428–435, 2011.

[74] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimization with structured resource constraints. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 923–930, 2009.

[75] A. Kumar, W. Yeoh, and S. Zilberstein. On message-passing MAP estimation in graphical models and DCOPs. In *IJCAI Workshop on Distributed Constraint Reasoning (DCR-11)*, 2011.

[76] A. Kumar, S. Zilberstein, and M. Toussaint. Scalable multiagent planning using probabilistic inference. In *International Joint Conference on Artificial Intelligence*, pages 2140–2146, 2011.

[77] A. Kumar, S. Zilberstein, and M. Toussaint. Message-passing algorithms for MAP estimation using DC programming. In *International Conference on Artificial Intelligence and Statistics*, 2012.

[78] M. P. Kumar, V. Kolmogorov, and H. S. P. Torr. An analysis of convex relaxations for MAP estimation of discrete MRFs. *Journal of Machine Learning Research*, 10:71–106, 2009.

[79] S. L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251, 2001.

[80] T. Léauté and B. Faltings. Privacy-preserving multi-agent constraint satisfaction. In *IEEE International Conference on Privacy, Security, Risk and Trust*, pages 17–25, 2009.

[81] T. Léauté and B. Faltings. Coordinating logistics operations with privacy guarantees. In *International Joint Conference on Artificial Intelligence*, pages 2482–2487, 2011.

[82] V. Lesser. Experiences buliding a distributed sensor network. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 1–6, 2003.

[83] V. Lesser, M. Tambe, and C. L. Ortiz, editors. *Distributed Sensor Networks: A Multiagent Perspective*. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[84] M. Littman. Memoryless policies: theoretical limitations and practical results. In *Conference on Simulation of Adaptive Behavior*, 1994.

[85] R. Maheswaran, J. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical game-based approach. In *International Conference on Parallel and Distributed Computing Systems*, pages 432–439, 2004.

[86] R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 310–317, 2004.

[87] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445, 2004.

[88] J. Marecki, T. Gupta, P. Varakantham, M. Tambe, and M. Yokoo. Not all agents are equal: Scaling up distributed POMDPs for agent networks. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 485–492, 2008.

[89] R. Marinescu and R. Dechter. AND/OR branch-and-bound search for graphical models. In *International Joint Conference on Artificial Intelligence*, pages 224–229, 2005.

[90] A. F. T. Martins, P. M. Q. Aguiar, M. A. T. Figueiredo, N. A. Smith, and E. P. Xing. An augmented lagrangian approach to constrained MAP inference. In *International Conference on Machine Learning*, pages 169–176, 2011.

[91] D. J. Mclaughlin, V. Ch, K. Droegemeier, S. Frasier, J. Kurose, F. Junyent, B. Philips, R. Cruz-pol, and J. Colom. Distributed collaborative adaptive sensing (DCAS) for improved detection, understanding, and predicting of atmospheric hazards. In *American Meteorological Society Annual Meeting*, 2005.

[92] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *International Conference on Computer Vision*, pages 428–435, 2005.

[93] N. Meuleau, K. eung Kim, L. P. Kaelbling, and A. R. Cassandra. Solving POMDPs by searching the space of finite policies. In *International Conference on Uncertainty in Artificial Intelligence*, pages 417–426, 1999.

[94] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra. Solving POMDPs by searching the space of finite policies. In *International Conference on Uncertainty in Artificial Intelligence*, pages 417–426, 1999.

[95] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 161:149–180, 2006.

[96] H. Mostafa and V. R. Lesser. Offline planning for communication by exploiting structured interactions in decentralized MDPs. In *International Conference on Intelligent Agent Technology*, pages 193–200, 2009.

[97] H. Mostafa and V. R. Lesser. Compact mathematical programs for DEC-MDPs with structured agent interactions. In *International Conference on Uncertainty in Artificial Intelligence*, pages 523–530, 2011.

[98] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, S. Marsella, R. Nair, and M. Tambe. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *International Joint Conference on Artificial Intelligence*, pages 705–711, 2003.

[99] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *National Conference on Artificial Intelligence*, pages 133–139, 2005.

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

[101] F. A. Oliehoek, M. T. J. Spaan, J. S. Dibangoye, and C. Amato. Heuristic search for identical payoff Bayesian games. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1115–1122, 2010.

[102] J. Ooi and G. Wornell. Decentralized control of a multiple access broadcast channel: Performance bounds. In *IEEE Conference on Decision and Control*, pages 293–298, 1996.

[103] B. Ottens and B. Faltings. Coordinating agent plans through distributed constraint optimization. In *Multi Agent Planning Workshop - ICAPS*, 2008.

[104] C. H. Papadimitriou and J. N. Tsitsiklis. On the complexity of designing distributed protocols. *Information and Control*, 53:211–218, 1982.

[105] C. H. Papadimitriou and J. N. Tsitsiklis. Intractable problems in control theory. *SIAM Journal on Control and Optimization*, 24(4):639–654, 1986.

[106] J. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *International Joint Conference on Artificial Intelligence*, pages 1446–1451, 2007.

[107] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers Inc., 1988.

[108] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence*, pages 266–271, 2005.

[109] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence*, pages 266–271, 2005.

[110] A. Petcu and B. Faltings. Superstabilizing, fault-containing multiagent combinatorial optimization. In *National Conference on Artificial Intelligence*, pages 449–454, 2005.

[111] A. Petcu and B. Faltings. Optimal solution stability in dynamic, distributed constraint optimization. In *International Conference on Intelligent Agent Technology*, pages 321–327, 2007.

[112] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[113] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, Department of Computer Science, University of Toronto, 2005.

[114] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems*, 2003.

[115] P. Poupart, T. Lang, and M. Toussaint. Analyzing and escaping local optima in planning as inference for partially observable domains. In *European Conference on Machine Learning*, pages 613–628, 2011.

[116] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[117] R. Radner. Team decision problems. *Annals of Mathematical Statistics*, 33:211–218, 1962.

[118] S. Ramchurn, P. Vytelingum, A. Rogers, and N. Jennings. Agent-based homeostatic control for green energy. *ACM Transactions on Intelligent Systems and Technology*, 2(4), 2011.

[119] P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and Markov random field MAP estimation. In *International Conference on Machine Learning*, pages 737–744, 2006.

[120] P. Ravikumar, A. Agarwal, and M. J. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research*, pages 1043–1080, 2010.

[121] S. J. Reddi, S. Sarawagi, and S. Vishwanathan. MAP estimation in binary MRFs via bipartite multi-cuts. In *Advances in Neural Information Processing Systems*, pages 955–963, 2010.

[122] F. Rendl, G. Rinaldi, and A. Wiegele. Biq Mac solver - binary quadratic and Max cut solver, 2010. URL `http://biqmac.uni-klu.ac.at/`.

[123] F. Rendl, G. Rinaldi, and A. Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010.

[124] R. Salakhutdinov, S. Roweis, and Z. Ghahramani. On the convergence of bound optimization algorithms. In *International Conference on Uncertainty in Artificial Intelligence*, pages 509–516, 2003.

[125] N. Schurr, S. Okamoto, R. Maheswaran, P. Scerri, and M. Tambe. Evolution of a teamwork model. *Cognitive Modeling and Multi-Agent Interactions*, pages 307–327, 2005.

[126] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *International Conference on Uncertainty in Artificial Intelligence*, 2007.

[127] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *International Joint Conference on Artificial Intelligences*, pages 2009–2015, 2007.

[128] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *International Conference on Uncertainty in Artificial Intelligence*, pages 520–527, 2004.

[129] D. Sontag. Cutting plane algorithms for variational inference in graphical models. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2007.

[130] D. Sontag. *Approximate Inference in Graphical Models using LP Relaxations*. PhD thesis, Massachusetts Institute of Technology, 2010.

[131] D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems*, 2007.

[132] D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems*, pages 1393–1400, 2008.

[133] D. Sontag, A. Globerson, and T. Jaakkola. Clusters and coarse partitions in LP relaxations. In *Advances in Neural Information Processing Systems*, pages 1537–1544, 2008.

[134] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *International Conference on Uncertainty in Artificial Intelligence*, pages 503–510, 2008.

[135] M. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[136] B. Sriperumbudur and G. Lanckriet. On the convergence of the concave-convex procedure. In *Advances in Neural Information Processing Systems*, pages 1759–1767, 2009.

[137] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *International Joint Conference on Artificial Intelligence*, pages 299–304, 2009.

[138] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of continuously valued control parameters using the max-sum algorithm. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 610–608, 2009.

[139] R. Stranders, F. D. Fave, A. Rogers, and N. R. Jennings. A decentralised coordination algorithm for mobile sensors. In *AAAI Conference on Artificial Intelligence*, pages 874–880, 2010.

[140] D. Szer and F. Charpillet. MAA*: a heuristic search algorithm for solving decentralized POMDPs. In *International Conference on Uncertainty in Artificial Intelligence*, pages 576–583, 2005.

[141] M. Toussaint and A. J. Storkey. Probabilistic inference for solving discrete and continuous state Markov decision processes. In *International Conference on Machine Learning*, pages 945–952, 2006.

[142] M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report EDIINF-RR-0934, University of Edinburgh, School of Informatics, 2006.

[143] M. Toussaint, L. Charlin, and P. Poupart. Hierarchical POMDP controller optimization by likelihood maximization. In *International Conference on Uncertainty in Artificial Intelligence*, pages 562–570, 2008.

[144] J. N. Tsitsiklis. Decentralized detection. *Advances in Signal Processing*, 2:297–344, 1993.

[145] J. N. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, 30(2):440–446, 1985.

[146] P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, 2007.

[147] P. Varakantham, J. young Kwak, M. E. Taylor, J. Marecki, P. Scerri, and M. Tambe. Exploiting coordination locales in distributed POMDPs via social model shaping. In *International Conference on Automated Planning and Scheduling*, 2009.

[148] M. Vinyals, M. Pujol, J. Rodriguez-Aguilarhas, and J. Cerquides. Divide-and-coordinate: DCOPs by agreement. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 149–156, 2010.

[149] M. Vinyals, E. Shieh, J. Cerquides, J. Rodríguez-Aguilar, Z. Yin, M. Tambe, and E. Bowring. Quality guarantees for region optimal DCOP algorithms. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2011.

[150] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis. Learning model-free robot control by a monte carlo EM algorithm. *Autonomous Robots*, 27(2):123–130, 2009.

[151] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. *IEEE Transactions on Information Theory*, 51:3697–3717, 2002.

[152] M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14:143–166, 2004.

[153] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

[154] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

[155] T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[156] S. J. Witwicki and E. H. Durfee. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *International Conference on Automated Planning and Scheduling*, pages 185–192, 2010.

[157] S. J. Witwicki and E. H. Durfee. Towards a unifying characterization for quantifying weak coupling in Dec-POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 29–36, 2011.

[158] F. Wu, S. Zilberstein, and X. Chen. Point-based policy generation for decentralized POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1307–1314, 2010.

[159] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – An empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

[160] W. Yeoh, P. Varakantham, and S. Koenig. Caching schemes for dcop search algorithms. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 609–616, 2009.

[161] W. Yeoh, A. Felner, and S. Koenig. Bnb-adopt: An asynchronous branch-and-bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 38:85–133, 2010.

[162] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *International Conference on Multiagent Systems*, pages 401–408, 1996.

[163] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *International Conference on Multiagent Systems*, pages 401–408, 1996.

[164] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10:673–685, 1998.

[165] A. L. Yuille. CCCP algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14:1691–1722, 2002.

[166] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15: 915–936, 2003.

[167] C. Zhang, V. Lesser, and P. Shenoy. A multi-agent learning approach to online distributed resource allocation. In *International Joint Conference on Artificial Intelligence*, pages 361–366, 2009.

[168] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. An analysis and application of distributed constraint satisfaction and optimization algorithms in sensor networks. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 185–192, 2003.

[169] S. Zilberstein, R. Washington, D. S. Bernstein, and A.-I. Mouaddib. Decision-theoretic control of planetary rovers. In *International Seminar on Advances in Plan-Based Control of Robotic Agents*, pages 270–289, 2002.

[170] R. Zivan. Anytime local search for distributed constraint optimization. In *AAAI Conference on Artificial Intelligence*, pages 393–398, 2008.

[171] R. Zivan, R. Glinton, and K. Sycara. Distributed constraint optimization for large teams of mobile sensing agents. In *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 347–354, 2009.