LOCAL OPTIMIZATION IN COOPERATIVE AGENT NETWORKS

by

Jonathan P. Pearce

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

June 2007

Acknowledgments

I would like to thank my advisor, Milind Tambe, for his constant support and encouragement in all aspects of my research and career. I also thank my collaborator, Rajiv Maheswaran, for helping to guide me in my first two years at USC and for many helpful discussions and ideas about *k*-optimality. I thank Victor Lesser, Fernando Ordóñez, Sven Koenig, and Sarit Kraus, both for their insightful thoughts and discussions on the work in this thesis, and for their support in my career. I also would like to thank the late Jay Modi for helpful discussions and on this research and on navigating graduate school. Thanks also go to Gal Kaminka, Bhaskar Krishnamachari, Paul Scerri, and Makoto Yokoo for helpful insights on the material in this thesis. I also thank Amos Freedy and Gershon Weltman of Perceptronics Solutions, Inc. very much for their help in supporting this work and obtaining real-world data.

I also thank members and alumni of the TEAMCORE research group for their support and for being good friends: Emma Bowring, Tapana Gupta, Hyuckchul Jung, Janusz Marecki, Ranjit Nair, Praveen Paruchuri, Nathan Schurr, Zvi Topol, and Pradeep Varakantham. Additional thanks are due to Zvi Topol for his assistance in implementing the MGM-3 algorithm.

I thank my parents, brothers, and sister for supporting my graduate school adventure, and finally, I thank my wife Liang for all the help and support she has given me during this time.

Abstract

My research focuses on constructing and analyzing systems of intelligent, autonomous agents. These agents may include people, physical robots, or software programs acting as assistants, teammates, opponents, or trading partners. In a large class of multi-agent scenarios, the effect of local interactions between agents can be compactly represented as a network structure such as a distributed constraint optimization problem (DCOP) for cooperative domains. Collaboration between large groups of agents, given such a network, can be difficult to achieve; often agents can only manage to collaborate in smaller subgroups of a certain size, in order to find a workable solution in a timely manner. The goal of my thesis is to provide algorithms to enable networks of agents that are bounded in this way to quickly find high-quality solutions, as well as theoretical results to understand key properties of these solutions. Relevant domains for my work include personal assistant agents, sensor networks, and teams of autonomous robots.

In particular, this thesis considers the case in which agents optimize a DCOP by forming groups of one or more agents until no group of *k* or fewer agents can possibly improve the solution; we define this type of local optimum, and any algorithm guaranteed to reach such a local optimum, as *k-optimal*. In this document, I present four key contributions related to *k*-optimality. The first set of results are worst-case guarantees on the solution quality of *k*-optima in a DCOP.

These guarantees can help determine an appropriate *k*-optimal algorithm, or possibly an appropriate constraint graph structure, for agents to use in situations where the cost of coordination between agents must be weighed against the quality of the solution reached. The second set of results are upper bounds on the number of *k*-optima that can exist in a DCOP. Because each joint action consumes resources, knowing the maximal number of *k*-optimal joint actions that could exist for a given DCOP allows us to allocate sufficient resources for a given level of *k*, or, alternatively, choosing an appropriate level of *k*-optimality, given fixed resource. The third contribution is a set of 2-optimal and 3-optimal algorithms and an experimental analysis of the performance of 1-, 2-, and 3-optimal algorithms on several types of DCOPs. The final contribution of this thesis is a case study of the application of *k*-optimal DCOP algorithms and solutions to the problem of the formation of human teams spanning multiple organizations. Given a particular specification of a human team (such as a task force to respond to an emergency) and a pool of possible team members, a DCOP can be formulated to match this specification. A set of *k*-optimal solutions to the DCOP represents a set of diverse, locally optimal options from which a human commander can choose the team that will be used.

Contents

Acknowledgments **ii**

Abstract **iii**

List Of Figures **vii**

List Of Tables **ix**

List Of Algorithms **x**

1 Introduction and Related Work **1**
- 1.1 Introduction 1
- 1.2 Related Work 5
  - 1.2.1 Constraint Reasoning 5
  - 1.2.2 Distributed Markov Decision Processes 7
  - 1.2.3 Local Search and Game Theory 7
  - 1.2.4 Local Search in Combinatorial Optimization 8
  - 1.2.5 Multi-Linked Contracts between Self-Interested Agents 8
- 1.3 Guide 9

2 *k*-Optimality and Distributed Constraint Optimization **10**
- 2.1 The Distributed Constraint Optimization Problem (DCOP) 10
- 2.2 Properties of *k*-Optimal DCOP Solutions 14

3 Lower Bounds on Solution Quality for *k*-optima **16**
- 3.1 Quality Guarantees on *k*-Optima 17
- 3.2 Graph-Based Quality Guarantees 24
- 3.3 Quality Guarantees for DCOPs with Hard Constraints 28
- 3.4 Domination Analysis of *k*-Optima 34
- 3.5 Experimental Results 38

4 Upper Bounds on the Number of *k*-Optima in a DCOP **45**
- 4.1 Upper Bounds on *k*-Optima 48
- 4.2 Graph-Based Analysis: *k*-Optima 54
- 4.3 Application to Nash Equilibria 59
- 4.4 Graph-Based Bounds 61

4.5 Experimental Results . . . 63

5 Algorithms **69**

5.1 1-Optimal Algorithms . . . 69

5.2 2-Optimal Algorithms . . . 74

5.3 3-Optimal Algorithms . . . 81

5.4 Experiments . . . 84

5.4.1 Medium-Sized DCOPs . . . 84

5.4.2 Large DCOPs . . . 93

6 *k*-Optimality and Team Formation **96**

6.1 Problem Formulation . . . 96

6.2 Sample Domain 1: Task force for Homeland Security Exercise . . . 97

6.2.1 Team Roles . . . 98

6.2.2 Data Sources . . . 98

6.2.3 DCOP Constraints . . . 99

6.2.3.1 Soft Unary Constraints . . . 99

6.2.3.2 Soft Binary Constraints . . . 101

6.2.3.3 Hard Constraints . . . 102

6.2.3.4 Weighting the Constraints . . . 103

6.2.4 Formation of *k*-Optimal Teams . . . 103

6.3 Sample Domain 2: Second Homeland Security Exercise . . . 103

7 Other Work **107**

7.1 Privacy in DCOPs . . . 107

7.2 Solving DCOPs Efficiently . . . 109

8 Related Work **110**

8.1 Constraint Reasoning . . . 110

8.2 Distributed Markov Decision Processes . . . 112

8.3 Local Search and Game Theory . . . 112

8.4 Local Search in Combinatorial Optimization . . . 113

9 Conclusions and Future Work **114**

9.1 Conclusions . . . 114

9.2 Future Work . . . 118

Reference List **122**

Curriculum Vitae **125**

List Of Figures

2.1 DCOP example 13

2.2 1-optima vs. assignment sets chosen using other metrics 15

3.1 Quality guarantees for k-optima with respect to the global optimum for DCOPs of various graph structures. 41

3.2 Quality guarantees for k-optima in DCOPs containing hard constraints. 42

3.3 Quality guarantees for k-optima in a multiply-constrained ring DCOP. 43

3.4 Quality guarantees for k-optima with respect to the space of dominated assignments for various graph structures. 44

4.1 Hypothetical example illustrating the advantages of tighter bounds 49

4.2 A visual representation of the effect of Proposition 8. 56

4.3 Exclusivity graphs for 1-optima for Example 1 with MIS shown in gray, (a) not using Proposition 8 and (b) using it. 58

4.4 Computation of β_{SRP} for Example 1 63

4.5 1-optima vs. assignment sets chosen using other metrics 64

4.6 β_{SRP} vs. β_{HSP} for DCOP graphs from Figure 2.2 65

4.7 Comparisons of β_{SRP} vs. β_{HSP} 67

4.8 Comparisons of $\beta_{SRP}, \beta_{HSP}, \beta_{FCLIQUE}$ 68

4.9 Improvement of β_{MH} on $\min\{\beta_H, \beta_S, \beta_P\}$ 68

5.1 Sample Trajectories of MGM and DSA for a High-Stakes Scenario 74

5.2 Comparison of the performance of MGM and DSA 87

5.3 Comparison of the performance of MGM and MGM-2 89

5.4 Comparison of the performance of DSA and SCA-2 90

5.5 Comparison of the performance of MGM, MGM-2, and MGM-3 91

5.6 Comparison of the performance of DSA, SCA-2, and SCA-3 92

5.7 Distributions of Solution Quality for X, X_E, X_{2E} and Cardinality of X_{2E} as a proportion of X_E 94

6.1 Tabulated results of Experience section of Check-In Form 100

6.2 2-optimal teams formed according to various criteria 104

6.3 2-optimal teams 106

9.1 A depiction of three key regions in the assignment space, given a k-optimum, that allowed for the discovery of various theoretical properties about k-optima 115

9.2 Graphical game example 119

List Of Tables

5.1 Results for MGM and MGM-3 for large DCOPs: Graph Coloring 95

5.2 Results for MGM and MGM-3 for large DCOPs: Random Rewards 95

List of Algorithms

1 Algorithm for Symmetric Region Packing (SRP) bound 62

2 DSA (myNeighbors, myValue) 71

3 MGM (myNeighbors, myValue) 71

4 MGM2 (myNeighbors, myConstraints, myValue) 79

5 MGM3 (myNeighbors, myConstraints, myValue) 82

Chapter 1

Introduction and Related Work

1.1 Introduction

In a large class of multi-agent scenarios, a set of agents chooses a joint action as a combination of individual actions. Often, the locality of agents' interactions means that the utility generated by each agent's action depends only on the actions of a subset of the other agents. In this case, the outcomes of possible joint actions can be compactly represented by graphical models, such as a distributed constraint optimization problem (DCOP)[Modi et al., 2005; Mailler and Lesser, 2004; Zhang et al., 2003], for cooperative domains, or by a graphical game [Kearns et al., 2001; Vickrey and Koller, 2002], for noncooperative domains. Each of these models can take the form of a graph in which each node is an agent and each edge denotes a subset of agents whose actions, when taken together, incur costs or rewards, either to the agent team (in DCOPs) or to individual agents (in graphical games). This thesis focuses primarily on the team setting, using DCOP, whose applications include multi-agent plan coordination [Cox et al., 2005], sensor networks [Modi et al., 2005], meeting scheduling [Petcu and Faltings, 2006] and RoboCup soccer [Vlassis et al., 2004], but also draws connections to noncooperative settings when applicable.

Traditionally, researchers have focused on obtaining a single, globally optimal solution to DCOPs, introducing complete algorithms such as Adopt [Modi et al., 2005], OptAPO [Mailler and Lesser, 2004], and DPOP [Petcu and Faltings, 2005]. However, because DCOP has been shown to be NP-hard[Modi et al., 2005], as the scale of these domains become large, current complete algorithms can incur large computation or communication costs. For example, a large-scale network of personal assistant agents might require global optimization over hundreds of agents and thousands of variables. However, incomplete algorithms in which agents form small groups and optimize within these groups can lead to a system that scales up easily and is more robust to dynamic environments. In existing incomplete algorithms, such as DSA [Fitzpatrick and Meertens, 2003] and DBA [Yokoo and Hirayama, 1996; Zhang et al., 2005a], agents are bounded in their ability to aggregate information about one another's constraints; in these algorithms, each individual agent optimizes based on its individual constraints, given the actions of all its neighbors, until a local optimum is reached where no single agent can improve the overall solution. Unfortunately, no guarantees on solution quality currently exist for these types of local optima. This thesis considers a generalization of this approach, in which agents optimize by forming groups of one or more agents until no group of k or fewer agents can possibly improve the solution; we define this type of local optimum as a *k-optimum*. According to this definition, for a DCOP with n agents, DSA and DBA are 1-optimal, while all complete algorithms are n-optimal. In this thesis I focus on k-optima and k-optimal algorithms for $1 < k < n$.

The k-optimality concept provides an algorithm-independent classification for local optima in a DCOP that allows for quality guarantees. In addition to the introduction of k-optimality itself, I present four sets of results about k-optimality. The first set of results are worst-case guarantees on the solution quality of k-optima in a DCOP. These guarantees can help determine an appropriate *k*-optimal algorithm, or possibly an appropriate constraint graph structure, for agents to use in situations where the cost of coordination between agents must be weighed against the quality of the solution reached. If increasing the value of *k* will provide a large increase in guaranteed solution quality, it may be worth the extra computation or communication required to reach a higher *k*-optimal solution. For example, consider a team of autonomous underwater vehicles (AUVs) [Zhang et al., 2005b] that must quickly choose a joint action in order to observe some transitory underwater phenomenon. The combination of individual actions by nearby AUVs may generate costs or rewards to the team, and the overall utility of the joint action is determined by the sum of these costs and rewards. If this problem were represented as a DCOP, nearby AUVs would share constraints in the graph, while far-away AUVs would not. However, the actual rewards on these constraints may not be known until the AUVs are deployed, and in addition, due to time constraints, an incomplete, *k*-optimal algorithm, rather than a complete algorithm, must be used to find a solution. In this case, worst-case quality guarantees for *k*-optimal solutions for a given *k*, that are independent of the actual costs and rewards in the DCOP, are useful to help decide which algorithm to use. Alternatively, the guarantees can help to choose between different AUV formations, i. e. different constraint graphs.

The second set of results are upper bounds on the number of *k*-optima that can exist in a DCOP. These bounds are necessitated by two key features of the typical domains where a *k*-optimal set is applicable. First, each *k*-optimum in the set consumes some resources that must be allocated in advance. Such resource consumption arises because: (i) a team actually executes each *k*-optimum in the set, or (ii) the *k*-optimal set is presented to a human user (or another agent) as a list of options to choose from, requiring time. In each case, resources are consumed based on the *k*-optimal set size. Second, while the existence of the constraints between agents is known *a priori*, the actual rewards and costs on the constraints depend on conditions that are not known until runtime, and so resources must be allocated before the rewards and costs are known and before the agents generate the *k*-optimal set. To see this, consider another domain involving a team of disaster rescue agents that must generate a set of *k*-optimal joint actions in order to present a set of diverse options to a human commander, where each option represents the best joint action within a neighborhood of similar joint actions. The commander will choose one joint action for the team to actually execute. Constraints exist between agents whose actions must be coordinated (i.e., members of subteams) but their costs and rewards depend on conditions on the ground that are unknown until the time when the agents must be deployed. Here, the resource is the time available to the commander to make the decision. Presenting too many options will cause the commander to run out of time before considering them all, and presenting too few may cause high-quality options to be omitted; in either case, the commander's decision could be impaired. Because each joint action consumes resources, knowing the maximal number of *k*-optimal joint actions that could exist for a given DCOP allows us to allocate sufficient resources for a given level of *k*.

The third contribution is a set of 2- and 3-optimal algorithms and an experimental analysis of the performance of 1-, 2- and 3-optimal algorithms on several types of DCOPs. Although we now have theoretical lower bounds on solution quality of *k*-optima, experimental results are useful to understand average-case performance on common DCOP problems.

The final contribution of this thesis is a case study of the application of *k*-optimal DCOP algorithms and solutions to the problem of the formation of human teams spanning multiple organizations. Given a particular specification of a human team (such as a task force to respond to an emergency) and a pool of possible team members, a DCOP can be formulated to match this specification. A set of *k*-optimal solutions to the DCOP represents a set of diverse, locally optimal options from which a human commander can choose the team that will be used. This thesis will show how DCOP can be mapped to real data from different domains and how *k*-optimality can be used in order to generate these options quickly and intuitively.

1.2 Related Work

Although *k*-optimality is a fundamentally new concept in distributed constraint reasoning, we can draw parallels to related work.

1.2.1 Constraint Reasoning

An active area of recent research has been in the developtment of complete "*n*-optimal" algorithms, including Adopt [Modi et al., 2005], OptAPO [Mailler and Lesser, 2004] and DPOP [Petcu and Faltings, 2005].

The Asynchronous Distributed Optimization (Adopt) algorithm [Modi et al., 2005] is the first known algorithm for DCOP; it is an asychnronous algorithm in which a Depth-First Search (DFS) tree is extracted from the DCOP graph, and small messages are passed up and down the tree. The runtime of Adopt was significantly improved using preprocessing techniques in [Maheswaran et al., 2004b] and [Ali et al., 2005]. A second algorithm for DCOP, Optimal Asynchronous Partial Overlay (OptAPO) [Mailler and Lesser, 2004], requires no DFS tree. Instead agents partially centralize the problem by forming subgroups; one agent in the subgroup would receive the constraints on the other agents and would solve the subproblem for that subgroup; in most cases complete centralization was not necessary to reach an optimal solution. A third algorithm, DPOP

[Petcu and Faltings, 2005], is a dynamic programming-based algorithm in which a DFS tree is used, as in Adopt. In DPOP, each agent passes just one message up the tree. The agent considers every possible combination of values that could be chosen by the higher-priority agents that have constraints with this agent, and finds the best value it can choose, for each of these combinations; the value and reward for every combination is included in this message. Multiply-constrained DCOPs, for which the bounds on solution quality for *k*-optima in this thesis can also be applied, were introduced in [Bowring et al., 2006], along with a modification of Adopt to solve these problems optimally.

Previous work in incomplete, 1-optimal algorithms for DCOP was described in detail in Chapter 5. The aim of this thesis is to consider the large space in between these two classes of algorithms and solutions.

This thesis also provides a theoretical complement to the experimental analysis of local minima (1-optima) and landscapes in centralized constraint satisfaction problems (CSPs) [Yokoo, 1997] as well as incomplete DCOP algorithms [Zhang et al., 2003]. While this thesis is primarily concerned with the effects of varying *k* and graph structure, the cited works provide insight into the effects of the choice between *k*-optimal algorithms of the same *k*-level on solution quality and convergence time. Note that *k*-optimality can also apply to centralized constraint reasoning as a measure of the relative quality and diversity of local optima. However, examining properties of solutions that arise from coordinated value changes of small groups of variables is especially useful in distributed settings, given the computational and communication expense of large-scale coordination.

Finally, despite the seeming similarity of *k*-optimality to *k*-consistency [Freuder, 1978] in centralized constraint satisfaction, the two concepts are entirely different, as *k*-consistency refers to reducing the domains of subsets of variables to maintain internal consistency in a satisfaction framework while *k*-optimality refers to comparing fixed solutions where subsets of variables optimize with respect to an external context.

1.2.2 Distributed Markov Decision Processes

It should also be noted that the bounds on solution quality of *k*-optima in this thesis also apply to networked distributed partially-observable Markov decision processes as defined in [Nair et al., 2005]. These structures can be viewed as DCOPs where each agent's domain represents its entire policy space, and rewards arise from the combinations of policies chosen by subsets of agents. Since the solution quality bounds do not depend on domain size, they can be directly applied to *k*-optimal ND-POMDP policies; in fact, the locally optimal algorithm presented in [Nair et al., 2005] is a 1-optimal algorithm.

1.2.3 Local Search and Game Theory

My research on upper bounds on the number of *k*-optima in DCOPs is related to work in landscape analysis in local search and evolutionary computing. In particular, [Caruana and Mullin, 1999] and [Whitley et al., 1998] have provided techniques for estimating numbers of local optima in these problems. In contrast, my work provides worst-case bounds on the number of *k*-optima in a DCOP, rather than an estimate based on sampling the solution space, and also exploits the structure of the agent network (constraint graph) to obtain these bounds, which was not done in previous work.

Given that counting the number of Nash equilibria in a game with known payoffs is #$\mathcal{P}$-hard [Conitzer and Sandholm, 2003], upper bounds on this number have indeed been investigated for particular types of games [McLennan and Park, 1999; Keiding, 1995]. Additionally, graph structure is utilized in several different algorithms to expedite finding Nash equilibria for a given graphical game with known payoffs [Kearns et al., 2001; Vickrey and Koller, 2002; Blum et al., 2003; Littman et al., 2002]. However, using graph structure to finding bounds on pure-strategy Nash equilibria over all possible games on a given graph (i.e., reward-independent bounds) remained an open problem; this thesis contains the first known upper bounds on Nash equilibria in such games.

1.2.4 Local Search in Combinatorial Optimization

The definition of *k*-optimality in DCOPs is analogous to the *k-opt* family of algorithms for the canonical Traveling Salesman Problem (TSP) [Lin, 1965]. The objective of the TSP is, given a complete graph in which each edge is associated with a weight, to choose a set of edges that forms a cycle (route) over all nodes in the graph that has the lowest total weight. In TSP, a route is considered *k*-optimal if it cannot be improved by replacing any *k* or fewer edges with new edges in the route graph. In contrast, in this thesis, *k*-optimality refers to a DCOP assignment that cannot be improved if *k* or fewer agents change their values. A quality guarantee on 2-optimal TSP tours has been given in [Lin, 1965].

1.2.5 Multi-Linked Contracts between Self-Interested Agents

We note that the definition of *k*-optimality in DCOPs is also analogous (but different) to *k*-optimality as given in [Sandholm, 1996]. In this work, agents make contracts with each other to optimally allocate tasks. A *k*-optimal assignment is one in which no cluster of *k* tasks can be beneficially transfered between any two agents.

1.3 Guide

The rest of this document is arranged as follows: Chapter 2 gives the background and motivation for the work and introduces the concept of *k*-optimality. Chapter 3 includes the lower bounds on solution quality for *k*-optima, and Chapter 4 includes the upper bounds on the number of *k*-optima in a DCOP. Chapter 5 includes new 2-optimal DCOP algorithms and an experimental analysis of these algorithms, as well as existing 1-optimal algorithms, in several domains. Other work that I have done in DCOPs and game theory, not directly related to *k*-optimality, is briefly detailed in Chapter 7. Finally, conclusions and areas for future work are given in Chapter 9.

Chapter 2

k-Optimality and Distributed Constraint Optimization

This chapter formally introduces the concept of k-optimality and provides background on the distributed constraint optimization formalism.

2.1 The Distributed Constraint Optimization Problem (DCOP)

A Distributed Constraint Optimization Problem (DCOP) consists of a set of variables, each assigned to an agent which must assign a value to the variable; these values correspond to individual actions that can be taken by the agents. Constraints exist between subsets of these variables that determine costs and rewards to the agent team based on the combinations of values chosen by their respective agents. Because we assume that each agent controls a single variable, we will use the terms "agent" and "variable" interchangeably.

Formally, a DCOP is a set of variables (one per agent) $N := \{1, \ldots, n\}$ and a set of domains $\mathcal{A} := \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, where the i^{th} variable takes value $a_i \in \mathcal{A}_i$. We denote the assignment of a subgroup of agents $S \subset \mathcal{I}$ by $a_S := \times_{i \in S} a_i \in \mathcal{A}_S$ where $\mathcal{A}_S := \bigtimes_{i \in S} \mathcal{A}_i$ and the assignment of the multi-agent team by $a = [a_1 \cdots a_n]$. Valued constraints exist on various minimal subsets $S \subset N$ of these variables. By minimality, we mean that the reward component R_S cannot be decomposed further through addition. Mathematically: $\forall S \in \theta, R_S(a_S) \neq R_{S_1}(a_{S_1}) + R_{S_2}(a_{S_2})$ for any $R_{S_1}(\cdot) : \mathcal{A}_{S_1} \rightarrow \mathbb{R}, R_{S_2}(\cdot) : \mathcal{A}_{S_2} \rightarrow \mathbb{R}, S_1, S_2 \subset \mathcal{N}$ such that $S_1 \cup S_2 = S, S_1, S_2 \neq \emptyset$. It is important to express the constraints minimally to accurately represent dependencies among agents.

A constraint on S is expressed as a reward function $R_S(a_S)$. This function represents the reward to the team generated by the constraint on S when the agents take assignment a; costs are expressed as negative rewards. θ is the set of all such subsets S on which a constraint exists.For convenience, we will refer to these subsets S as "constraints" and the functions $R_S(\cdot)$ as "constraint reward functions." The solution quality for a particular complete assignment a is the sum of the rewards for that assignment from all constraints in the DCOP: $R(a) = \sum_{S\in\theta} R_S(a) = \sum_{S\in\theta} R_S(a_S)$.

Several algorithms exist for solving DCOPs: complete algorithms, which are guaranteed to reach a globally optimal solution, and incomplete algorithms, which reach a local optimum, and do not provide guarantees on solution quality. In this thesis, we provide k-optimality as an algorithm-independent classification of local optima, and show how their solution quality can be guaranteed.

Before formally introducing the concept of k-optimality, we must define the following terms. For two assignments, a and $\tilde{a}$, the *deviating group* is

$$D(a, \tilde{a}) := \{i \in \mathcal{I} : a_i \neq \tilde{a}_i\},$$

i. e., the set of agents whose actions in assignment $\tilde{a}$ differ from their actions in a. The *distance* is

$$d(a, \tilde{a}) := |D(a, \tilde{a})|$$

where $|\cdot|$ denotes the cardinality of the set. The *relative reward* of an assignment a with respect to another assignment $\tilde{a}$ is

$$\Delta(a, \tilde{a}) := R(a) - R(\tilde{a}) = \sum_{S \in \theta : S \cap D(a, \tilde{a}) \neq \emptyset} \left[R_S(a_S) - R_S(\tilde{a}_S) \right].$$

In this summation, only the rewards on constraints incident on deviating agents are considered, since the other rewards remain the same.

We classify an assignment a as a *k-optimal assignment* or *k-optimum* if

$$\Delta(a, \tilde{a}) \geq 0 \; \forall \tilde{a} \quad \text{such that} \quad d(a, \tilde{a}) \leq k.$$

That is, a has a higher or equal reward to any assignment a distance of k or less from a. Equivalently, if the set of agents have reached a k-optimum, then no subgroup of cardinality $\leq k$ can improve the overall reward by choosing different actions; every such subgroup is acting optimally with respect to its context.

If no ties occur between the rewards of DCOP assignments that are a distance of k or less apart, then a collection of k-optima must be mutually separated by a distance $\geq k + 1$ as they each have the highest reward within a radius of k. Thus, a higher k-optimality of assignments

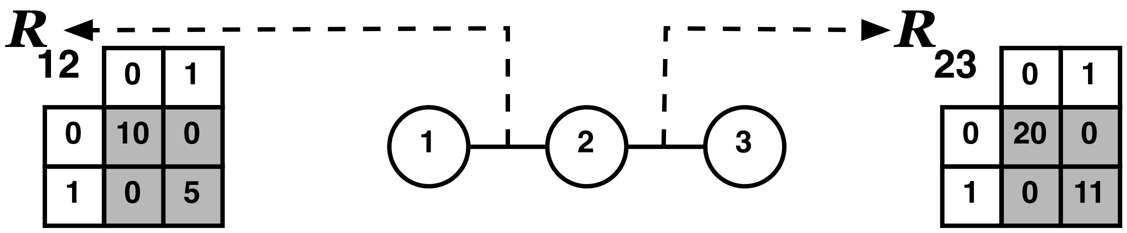


Figure 2.1: DCOP example

in a collection of assignments, or *assignment set*, implies a greater level of relative reward and diversity. Let

$$A_q(n,k) = \{a \in \mathcal{A} : \Delta(a,\tilde{a}) > 0 \ \forall \tilde{a} \quad \text{such that} \quad d(a,\tilde{a}) \leq k\}$$

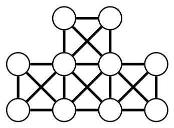
be the set of all k-optima for a team of n agents with domains of cardinality q. It is straightforward to show $A_q(n,k+1) \subseteq A_q(n,k)$.

Example 1 *Figure 2.1 is a binary DCOP in which agents choose actions from* $\{0,1\}$*, with rewards shown for the two constraints (minimal subgroups)* $S_{1,2} = \{1,2\}$ *and* $S_{2,3} = \{2,3\}$*. The assignment* $a = [1\ 1\ 1]$ *(with a total reward of 16) is* 1*-optimal because any single agent that deviates reduces the team reward. For example, if agent 1 changes its value from 1 to 0, the reward on* $S_{1,2}$ *decreases from 5 to 0. If agent 2 changes its value from 1 to 0, the rewards on* $S_{1,2}$ *and* $S_{2,3}$ *decrease from 5 to 0 and from 11 to 0, respectively. If agent 3 changes its value from 1 to 0, the reward on* $S_{2,3}$ *decreases from 11 to 0. However,* $[1\ 1\ 1]$ *is not* 2*-optimal because if the group* $\{2,3\}$ *deviated, making the assignment* $\tilde{a} = [1\ 0\ 0]$*, team reward would increase from 16 to 20. The globally optimal solution,* $a^* = [0\ 0\ 0]$*, with a total reward of 30, is k-optimal for all* $k \in \{1,2,3\}$.□

2.2 Properties of *k*-Optimal DCOP Solutions

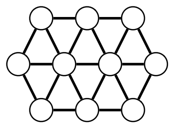
We now show, in an experiment, the advantages of *k*-optimal assignment sets as capturing both diversity and high reward compared with assignment sets chosen by other metrics. Diversity is important in domains where many *k*-optimal assignments are presented as choices to a human, so that all the choices are not essentially the same with very minor discrepancies. The lower half of Figure 2.2(a) shows a DCOP graph representing a team of 10 patrol robots, each of whom must choose one of two routes to patrol in its region. The nodes are agents and the edges represent binary constraints between agents assigned to overlapping regions. The actions (i. e., the chosen routes) of these agents combine to produce a cost or reward to the team. For each of 20 runs, the edges were initialized with rewards from a uniform random distribution. The set of all 1-optima was enumerated. Then, for the same DCOP, we used two other metrics to produce equal-sized sets of assignments. For one metric, the assignments with highest reward were included in the set, and for the next metric, assignments were included in the set by the following method, which selects assignments purely based on diversity (expressed as distance). We repeatedly cycled through all possible assignments in lexicographic order, and included an assignment in the set if the distance between it and every assignment already in the set was not less than a specified distance; in this case 2. The average reward and the diversity (expressed as the minimum distance between any pair of assignments in the set) for the sets chosen using each of the three metrics over all 20 runs is shown in the upper half of Figure 2.2(a). While the sets of 1-optima come close to the reward level of the sets chosen purely according to reward, they are clearly more diverse (T-tests for this claim showed a significance within .0001%). If a minimum distance of 2 is required in order to guarantee diversity, then using reward alone as a metric is insufficient; in fact the

	avg. reward	avg. min. distance
1-optima	.850	2.25
reward only	.950	1.21
dist. of 2	.037	2.00



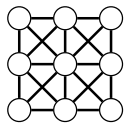
(a)

	avg. reward	avg. min. distance
1-optima	.809	2.39
reward only	.930	1.00
dist. of 2	-.101	2.00



(b)

	avg. reward	avg. min. distance
1-optima	.832	2.63
reward only	.911	1.21
dist. of 2	.033	2.00



(c)

Figure 2.2: 1-optima vs. assignment sets chosen using other metrics

assignment sets generated using that metric had an average minimum distance of 1.21, compared with 2.25 for 1-optimal assignment sets (which guarantee a minimum distance of $k + 1 = 2$). The 1-optimal assignment set also provides significantly higher average reward than the sets chosen to maintain a given minimum distance, which had an average reward of 0.037 (T-test significance within .0001%.). Similar results with equal significance were observed for the 10-agent graph in Figure 2.2(b) and the nine-agent graph in Figure 2.2(c). Note also that this experiment used $k = 1$, the lowest possible k. Increasing k would, by definition, increase the diversity of the k-optimal assignment set as well as the neighborhood size for which each assignment is optimal.

In addition to categorizing local optima in a DCOP, k-optimality provides a natural classification for DCOP algorithms. Many known algorithms are guaranteed to converge to k-optima for $k = 1$, including DBA Zhang et al. [2003], DSA Fitzpatrick and Meertens [2003], and coordinate ascent Vlassis et al. [2004] for $k = 1$. Complete algorithms such as Modi et al. [2005], OptAPO Mailler and Lesser [2004] and DPOP Petcu and Faltings [2005] are k-optimal for $k = n$. New 2 and 3-optimal algorithms will be presented in Chapter 5.

Chapter 3

Lower Bounds on Solution Quality for *k*-optima

In this chapter we introduce the first known guaranteed lower bounds on the solution quality of *k*-optimal DCOP assignments. These guarantees can help determine an appropriate *k*-optimal algorithm, or possibly an appropriate constraint graph structure, for agents to use in situations where the cost of coordination between agents must be weighed against the quality of the solution reached. If increasing the value of *k* will provide a large increase in guaranteed solution quality, it may be worth the extra computation or communication required to reach a higher *k*-optimal solution. For example, consider a team of autonomous underwater vehicles (AUVs) [Zhang et al., 2005b] that must quickly choose a joint action in order to observe some transitory underwater phenomenon. The combination of individual actions by nearby AUVs may generate costs or rewards to the team, and the overall utility of the joint action is determined by their sum. If this problem were represented as a DCOP, nearby AUVs would share constraints in the graph, while far-away AUVs would not. However, the actual rewards on these constraints may not be known until the AUVs are deployed, and in addition, due to time constraints, an incomplete, *k*-optimal algorithm, rather than a complete algorithm, must be used to find a solution. In this case, worst-case quality guarantees for *k*-optimal solutions for a given *k*, that are independent of the actual costs and rewards in the DCOP, are useful to help decide which algorithm to use. Alternatively, the guarantees can help to choose between different AUV formations, i. e. different constraint graphs.

We present two distinct types of guarantees for *k*-optima. The first, in Sections 3.1 and 3.2, is a lower bound on the quality of any *k*-optimum, expressed as a fraction of the quality of the optimal solution. The second, in Section 3.4, is a lower bound on the proportion of all DCOP assignments that a *k*-optimum must dominate in terms of quality. This type is useful in approximating the difficulty of finding a better solution than a given *k*-optimum. For both, we provide general bounds that apply to all constraint graph structures, as well as tighter bounds made possible if the graph is known in advance.

3.1 Quality Guarantees on *k*-Optima

This section provides reward-independent guarantees on solution quality for any *k*-optimal DCOP assignment. If we must choose a *k*-optimal algorithm for agents to use, it is useful to see how much reward will be gained or lost in the worst case by choosing a higher or lower value for *k*. We assume the actual costs and rewards on the DCOP are not known *a priori* (otherwise the DCOP could be solved centrally ahead of time; or all *k*-optima could be found by brute force, with the lowest-quality *k*-optimum providing an exact guarantee for a particular problem instance). We provide a guarantee for a *k*-optimal solution as a fraction of the reward of the optimal solution, assuming that all rewards in the DCOP are non-negative (the reward structure of any DCOP can be normalized to one with all non-negative rewards as long as no infinitely large costs exist).

Proposition 1 *For any DCOP of n agents, with maximum constraint arity of m, where all constraint rewards are non-negative, and where a^* is the globally optimal solution, then, for any k-optimal assignment, a, where $R(a) < R(a^*)$ and $m \leq k < n$,*

$$R(a) \geq \frac{\binom{n-m}{k-m}}{\binom{n}{k} - \binom{n-m}{k}} R(a^*). \tag{3.1}$$

Proof: By the definition of k-optimality, any assignment $\tilde{a}$ such that $d(a, \tilde{a}) \leq k$ must have reward $R(\tilde{a}) \leq R(a)$. We call this set of assignments $\tilde{A}$. Now consider any non-null subset $\hat{A} \subset \tilde{A}$. For any assignment $\hat{a} \in \hat{A}$, the constraints θ in the DCOP can be divided into three discrete sets, given a and $\hat{a}$:

- $\theta_1(a, \hat{a}) \subset \theta$ such that $\forall S \in \theta_1(a, \hat{a}), S \subset D(a, \hat{a})$.
- $\theta_2(a, \hat{a}) \subset \theta$ such that $\forall S \in \theta_2(a, \hat{a}), S \cap D(a, \hat{a}) = \emptyset$.
- $\theta_3(a, \hat{a}) \subset \theta$ such that $\forall S \in \theta_3(a, \hat{a}), S \notin \theta_1(a, \hat{a}) \cup \theta_2(a, \hat{a})$.

$\theta_1(a, \hat{a})$ contains the constraints that include only the variables in $\hat{a}$ which have deviated from their values in a; $\theta_2(a, \hat{a})$ contains the constraints that include only the variables in $\hat{a}$ which have not deviated from their values in a; and $\theta_3(a, \hat{a})$ contains the constraints that include at least one of each. Thus:

$$R(\hat{a}) = \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{S \in \theta_2(a,\hat{a})} R_S(\hat{a}) + \sum_{S \in \theta_3(a,\hat{a})} R_S(\hat{a}).$$

The sum of rewards of all assignments $\hat{a}$ in $\hat{A}$ is:

$$\begin{aligned}\sum_{\hat{a}\in\hat{A}} R(\hat{a}) &= \sum_{\hat{a}\in\hat{A}} \Big(\sum_{S\in\theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) + \sum_{S\in\theta_3(a,\hat{a})} R_S(\hat{a}) \Big) \\ &= \sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_3(a,\hat{a})} R_S(\hat{a}).\end{aligned}$$

Given the previous assumption that all rewards are nonnegative, we obtain:

$$\sum_{\hat{a}\in\hat{A}} R(\hat{a}) \geq \sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}).$$

Since $R(a) \geq R(\hat{a}), \forall \hat{a} \in \hat{A}$,

$$R(a) \geq \frac{\sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a})}{|\hat{A}|}. \tag{3.2}$$

Now, for any $\hat{A}$, if the two numerator terms and the denominator can be expressed in terms of $R(a^*)$ and $R(a)$, then we have a bound on $R(a)$ in terms of $R(a^*)$.

To do this, we consider the particular $\hat{A}$, denoted $\hat{A}^{a,k}$, which contains all possible assignments $\hat{a}$ such that

- $d(a, \hat{a}) = k$
- $\hat{a}_i = a^*_i, \forall i \in D(a, \hat{a})$.

This means that $\hat{A}^{a,k}$ contains all assignments $\hat{a}$ where exactly k variables in $\hat{a}$ have deviated from their values in a and these variables are taking the same values that they take in a^*. Note that, given the definition of $\hat{A}^{a,k}$, that $i \in D(a, \hat{a}) \Rightarrow a_i \neq a_i^*$ (no variable that has the same value in a as in a^* can be in $D(a, \hat{a})$).

There are $d(a, a^*)$ variables whose values are different between a and a^*, and $\hat{A}^{a,k}$ contains each assignment where exactly k of these variables are deviating from a to take the same values they take in a^*. Thus, there are $\binom{d(a,a^*)}{k}$ such assignments $\hat{a} \in \hat{A}^{a,k}$, and so the denominator term from Equation 3.2 is equal to $\binom{d(a,a^*)}{k}$. Note that $d(a, a^*) > k$, otherwise a would not be k-optimal.

We can now re-express the first numerator term in Equation 3.2 in terms of $R(a^*)$. Given any $k \in N$, any $\hat{A}^{a,k}$, any $S \in \theta$, and any assignment a, let $\Gamma_1^{a,k}(S)$ denote the set of all $\hat{a} \in \hat{A}^{a,k}$ where $S \in \theta_1(a, \hat{a})$. Now, we have $\forall \hat{a} \in \Gamma_1^{a,k}(S), d(a, \hat{a}) = k$ (exactly k variables are in the deviating group $D(a, \hat{a})$). By definition of $\theta_1(a, \hat{a})$, $S \subset D(a, \hat{a})$. Thus, there are exactly $d(a, a^*) - |S|$ variables not in S for which it is possible to be an element of $D(a, \hat{a})$, where $|S|$ represents the cardinality (arity) of S. Therefore, there are exactly $\binom{d(a,a^*)-|S|}{k-|S|}$ possible deviating groups $D(a, \hat{a})$, and therefore $\binom{d(a,a^*)-|S|}{k-|S|}$ assignments in $\Gamma_1^{a,k}(S)$. Finally, note that $\forall a, \forall S \in \theta, \forall \hat{a} \in \Gamma_1^{a,k}(S), R_S(\hat{a}) = R_S(a^*)$. Therefore,

$$\sum_{\hat{a} \in \hat{A}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) = \sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_1(a,\hat{a})} R_S(a^*) = \sum_{S \in \theta} \sum_{\hat{a} \in \Gamma_1^{a,k}(S)} R_S(a^*) = \sum_{S \in \theta} \binom{d(a, a^*) - |S|}{k - |S|} R_S(a^*).$$

Because $|S| \leq m$, we obtain an inequality for the first numerator term in Equation 3.2 in terms of $R(a^*)$:

$$\sum_{\hat{a} \in \hat{A}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) \geq \binom{d(a, a^*) - m}{k - m} R(a^*).$$

Similarly, we now express the second numerator term in Equation 3.2 in terms of $R(a)$. Given any $k \in N$, any $\hat{A}^{a,k}$, any $S \in \theta$, and any assignment a, let $\Gamma_2^{a,k}(S)$ denote the set of all $\hat{a} \in \hat{A}^{a,k}$ where $S \in \theta_2(a, \hat{a})$. Now, just as for $\Gamma_1^{a,k}(S)$, we have $\forall \hat{a} \in \Gamma_2^{a,k}(S), d(a, \hat{a}) = k$ (exactly k variables are in the deviating group $D(a, \hat{a})$). However, by definition of $\theta_2(a, \hat{a}), S \cap D(a, \hat{a}) = \emptyset$. (no variables in S are in the deviating group $D(a, \hat{a})$). Thus, there are exactly $d(a, a^*) - |S|$ variables not in S for which it is possible to be an element of $D(a, \hat{a})$. Therefore, there are exactly $\binom{d(a,a^*)-|S|}{k}$ possible deviating groups $D(a, \hat{a})$ and therefore $\binom{d(a,a^*)-|S|}{k}$ assignments in $\Gamma_2^{a,k}(S)$. Finally, note that $\forall a, \forall S \in \theta, \forall \hat{a} \in \Gamma_2^{a,k}(S), R_S(\hat{a}) = R_S(a)$. Therefore,

$$\sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) = \sum_{\hat{a}\in\hat{A}^{a,k}} \sum_{S\in\theta_2(a,\hat{a})} R_S(a) = \sum_{S\in\theta} \sum_{\hat{a}\in\Gamma_2^{a,k}(S)} R_S(a) = \sum_{S\in\theta} \binom{d(a, a^*) - |S|}{k} R_S(a).$$

Because $|S| \leq m$, we obtain an inequality for the second numerator term in Equation 3.2 in terms of $R(a^*)$:

$$\sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) \geq \binom{d(a, a^*) - m}{k} R(a).$$

Therefore, from Equation 3.2,

$$R(a) \geq \frac{\binom{d(a,a^*)-m}{k-m} R(a^*) + \binom{d(a,a^*)-m}{k} R(a)}{\binom{d(a,a^*)}{k}} \Rightarrow R(a) \geq \frac{\binom{d(a,a^*)-m}{k-m}}{\binom{d(a,a^*)}{k} - \binom{d(a,a^*)-m}{k}} R(a^*).$$

The ratio of $R(a)$ to $R(a^*)$ is minimized when $d(a, a^*) = n$, representing the case where a has no variable assignments in common with a^*, so Equation 3.1 holds as a guarantee for a k-optimum in any DCOP. We note that it is possible for $k > n - m$; in this case the term $\binom{n-m}{k}$ in Equation 3.1 is technically undefined. In this case, we take this term to be zero, representing the fact that there are zero assignments in $\Gamma_2^{a,k}(S)$ for any $S \in \theta$, since enumerating these assignments would require selecting k variables from a pool of $n-m$ variables. ■

For binary DCOPs ($m = 2$), Equation 3.1 simplifies to:

$$R(a) \geq \frac{\binom{n-2}{k-2}}{\binom{n}{k} - \binom{n-2}{k}} R(a^*) = \frac{k-1}{2n-k-1} R(a^*).$$

Finally, we note that no lower-bound guarantee is possible when $k < m$, such as for a binary DCOP ($m = 2$) where $k = 1$. In these cases Equation 3.1 gives a lower bound of $R(a) \geq 0 \cdot R(a^*)$.

The following example illustrates Proposition 1:

Example 2 *Consider a DCOP with five variables numbered 1 to 5, with domains of {0,1}. Suppose that this DCOP is a fully connected binary DCOP with constraints between every pair of variables (i.e.* $\theta = \{S : |S| = 2\}$*). Suppose that* $a = [0\ 0\ 0\ 0\ 0]$ *is a 3-optimum, and that* $a^* = [1\ 1\ 1\ 1\ 1]$ *is the global optimum. Then* $d(a, a^*) = 5$*, and* $\hat{A}^{a,k}$ *contains* $\binom{d(a,a^*)}{k} = 10$ *assignments, listed below:*

$$[1\ 1\ 1\ 0\ 0], [1\ 1\ 0\ 1\ 0], [1\ 1\ 0\ 0\ 1], [1\ 0\ 1\ 1\ 0], [1\ 0\ 1\ 0\ 1],$$

$$[1\ 0\ 0\ 1\ 1], [0\ 1\ 1\ 1\ 0], [0\ 1\ 1\ 0\ 1], [0\ 1\ 0\ 1\ 1], [0\ 0\ 1\ 1\ 1].$$

Since $R(a) \geq \hat{a}, \forall \hat{a} \in \hat{A}^{a,k}$*, then* $10 \cdot R(a) \geq \sum_{\hat{A}^{a,k}} R(\hat{a})$*. Now,* $\forall S \in \theta, \forall i \in S, a_i^* = \hat{a}_i$ *for exactly* $\binom{n-2}{k-2} = 3$ *assignments in* $\hat{A}^{a,k}$*. For example, for* $S = \{1,2\}, a_1^* = \hat{a}_1 = 1$ *and* $a_2^* = \hat{a}_2 = 1$ *for* $\hat{a} = [1\ 1\ 1\ 0\ 0]$, $[1\ 1\ 0\ 1\ 0]$, *and* $[1\ 1\ 0\ 0\ 1]$*. Therefore,* $\forall S \in \theta, R_S(a^*) = R_S(\hat{a})$ *for these* $\binom{n-2}{k-2} = 3$ *assignments. Similarly,* $\forall S \in \theta, \forall i \in S, a_i = \hat{a}_i$ *for exactly* $\binom{n-2}{k} = 1$ *assignment in* $\hat{A}^{a,k}$*. For example, for* $S = \{1,2\}, a_1 = \hat{a}_1 = 0$ *and* $a_2 = \hat{a}_2 = 0$ *only for* $\hat{a} = [0\ 0\ 1\ 1\ 1]$*. Therefore,* $\forall S \in \theta, R_S(a) = R_S(\hat{a})$ *for* $\binom{n-2}{k} = 1$ *assignment in* $\hat{A}^{a,k}$*. Thus,* $10 \cdot R(a) \geq \sum_{\hat{A}^{a,k}} R(\hat{a}) \geq 3 \cdot R(a^*) + 1 \cdot R(a)$*, and so* $R(a) \geq \frac{3}{10-1} R(a^*) = \frac{1}{3} R(a^*)$.□

We now show that Proposition 1 is tight, i.e. that there exist DCOPs with k-optima of quality equal to the bound.

Proposition 2 *$\forall n, m, k$ such that $m \leq k < n$, there exists some DCOP with n variables, with maximum constraint arity m with a k-optimal assignment, a, such that, if a^* is the globally optimal solution,*

$$R(a) = \frac{\binom{n-m}{k-m}}{\binom{n}{k} - \binom{n-m}{k}} R(a^*). \tag{3.3}$$

Proof: Consider a fully-connected m-ary DCOP where the domain of each variable contains at least two values {0,1} and every constraint R_S contains the following reward function:

$$R_S(a) = \begin{cases} \frac{\binom{n-m}{k-m}}{\binom{n}{k} - \binom{n-m}{k}} & a_i = 0, \forall i \in S \\ 1 & a_i = 1, \forall i \in S \\ 0 & \text{otherwise} \end{cases}$$

The optimal solution a^* is $a^*_i = 1, \forall i$. If a is defined such that $a_i = 0, \forall i$, then Equation 3.3 is true. Now we show that a is k-optimal. For any assignment $\hat{a}$, such that $d(a, \hat{a}) = k$,

$$R(\hat{a}) = \sum_{S \in \theta_1(a,\hat{a})} R(\hat{a}_S) + \sum_{S \in \theta_2(a,\hat{a})} R(\hat{a}_S) + \sum_{S \in \theta_3(a,\hat{a})} R(\hat{a}_S).$$

Given the chosen reward function, $\forall S \in \theta_1(a, \hat{a}), R(\hat{a}_S) = 1$, and $\forall S \in \theta_2(a, \hat{a}), R(\hat{a}_S) = \frac{\binom{n-m}{k-m}}{\binom{n}{k}-\binom{n-m}{k}}$. Note that $|\theta_1(a, \hat{a})| = \binom{k}{m}$ and $|\theta_2(a, \hat{a})| = \binom{n-k}{m}$ because $|S| = m$ and must be chosen from a set of either k or $n - k$ variables, respectively. Finally, for all other $S, R(\hat{a}_S) = 0$. Then, we have:

$$\begin{aligned}
R(\hat{a}) &= \binom{k}{m} + \binom{n-k}{m} \frac{\binom{n-m}{k-m}}{\binom{n}{k} - \binom{n-m}{k}} + 0 \\
&= \frac{\binom{k}{m}\left[\binom{n}{k} - \binom{n-m}{k}\right] + \binom{n-k}{m}\binom{n-m}{k-m}}{\binom{n}{k} - \binom{n-m}{k}} \\
&= \left(\frac{n!}{m!(k-m)!(n-k)!}\right) \bigg/ \left(\frac{n!(n-m-k)! - (n-m)!(n-k)!}{k!(n-k)!(n-m-k)!}\right) \\
&= \frac{n!k!(n-m-k)!}{m!(k-m)![n!(n-m-k)! - (n-k)!(n-m)!]} \\
&= \binom{n}{m} \frac{(n-m)!k!(n-m-k)!}{(k-m)![n!(n-m-k)! - (n-m)!(n-k)!]} \\
&= \binom{n}{m} \frac{\binom{n-m}{k-m} k!(n-m-k)!}{\frac{n!(n-m-k)!}{(n-k)!} - (n-m)!} \\
&= \binom{n}{m} \frac{\binom{n-m}{k-m}}{\binom{n}{k} - \binom{n-m}{k}} \\
&= R(a)
\end{aligned}$$

because in a, each of the $\binom{n}{m}$ constraints in the DCOP are producing the same reward. Since this can be shown for $d(a, \hat{a}) = j, \forall j$ such that $1 \leq j \leq k, a$ is k-optimal. ■

3.2 Graph-Based Quality Guarantees

The guarantee for k-optima in Section 3.1 applies to all possible DCOP graph structures. However, knowledge of the structure of constraint graphs can be used to obtain tighter guarantees. This is done by again expressing the two numerator terms in Equation 3.2 as multiples of $R(a^*)$

and $R(a)$. If the graph structure of the DCOP is known, we can exploit it by refining our definition of $\hat{A}^{a,k}$. We can take $\hat{A}$ from Proposition 1, i.e. $\hat{A}$ which contains all $\hat{a}$ such that $d(a, \hat{a}) = k$ and $\forall \hat{a} \in \hat{A}, \forall \hat{a}_i \in D(a, \hat{a}), \hat{a}_i = a^*_i$. Then, we restrict this $\hat{A}$ further, so that $\forall \hat{a} \in \hat{A}$, the variables in $D(a, \hat{a})$ form a connected subgraph of the DCOP graph (or hypergraph), meaning that any two variables in $D(a, \hat{a})$ must be connected by some chain of constraints. Since the previous section dealt with fully-connected DCOP graphs, this last restriction was actually already being used; however, it only becomes relevant to express it when working with non-fully-connected graph structures, as we will do in this section. With this refinement, we transform Equation 3.2 to express $R(a)$ in terms of $R(a^*)$ as before; however, we can now produce tighter guarantees for k-optima in sparse graphs. As an illustration, provably tight guarantees for binary DCOPs on ring graphs (each variable has two constraints) and star graphs (each variable has one constraint except the central variable, which has $n - 1$) are given below.

Proposition 3 *For any binary DCOP of n agents with a ring graph structure, where all constraint rewards are non-negative, and a^* is the globally optimal solution, then, for any k-optimal assignment, a, where $k < n$,*

$$R(a) \geq \frac{k-1}{k+1} R(a^*). \tag{3.4}$$

Proof: Returning to Equation 3.2, $|\hat{A}| = n$ because $D(a, \hat{a})$ could consist of any of the n connected subgraphs of k variables in a ring. For any constraint $S \in \theta$, there are $k - 1$ assignments $\hat{a} \in \hat{A}$

for which $S \in \theta_1(a, \hat{a})$ because there are $k-1$ connected subgraphs of k variables in a ring that contain S. Therefore,

$$\sum_{\hat{a} \in \hat{A}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) = (k-1)R(a^*).$$

Also, there are $n-k-1$ assignments $\hat{a} \in \hat{A}$ for which $S \in \theta_2(a, \hat{a})$ because there are $n-k-1$ ways to choose S in a ring so that it does not include any variable in a given connected subgraph of k variables. Therefore,

$$\sum_{\hat{a} \in \hat{A}} \sum_{S \in \theta_2(a,\hat{a})} R_S(\hat{a}) = (n-k-1)R(a).$$

So, from Equation 3.2,

$$R(a) \geq \frac{(k-1)R(a^*) + (n-k-1)R(a)}{n}$$

and therefore Equation 3.4 holds. ■

Proposition 4 *For any binary DCOP of n agents with a star graph structure, where all constraint rewards are non-negative, and a^* is the globally optimal solution, then, for any k-optimal assignment, a, where $k < n$,*

$$R(a) \geq \frac{k-1}{n-1} R(a^*). \tag{3.5}$$

Proof: The proof is similar to the previous proof. In a star graph, there are $\binom{n-1}{k-1}$ connected subgraphs of k variables, and therefore $|\hat{A}| = \binom{n-1}{k-1}$. Every constraint $S \in \theta$ includes the central variable and one other variable, and thus there are $\binom{n-2}{k-2}$ connected subgraphs of k variables that contain S, and therefore

$$\sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_1(a,\hat{a})} R_S(\hat{a}) = \binom{n-2}{k-2} R(a^*).$$

Finally, there are no ways to choose S so that it does not include any variable in a given connected subgraph of k variables. Therefore,

$$\sum_{\hat{a}\in\hat{A}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) = 0 \cdot R(a).$$

So, from Equation 3.2,

$$R(a) \geq \frac{\binom{n-2}{k-2} R(a^*) + 0 \cdot R(a)}{\binom{n-1}{k-1}}$$

and therefore Equation 3.5 holds. ■

Tightness can be proven by constructing DCOPs on ring and star graphs with the same rewards as in Proposition 2, except with $R_S(a) = (k-1)/(k+1)$ in the first case for rings, and $R_S(a) = (k-1)/(n-1)$ in the first case for stars. The bound for rings can also be applied to chains, since any chain can be expressed as a ring where all rewards on one constraint are zero.

Finally, bounds for DCOPs with arbitrary graphs and non-negative constraint rewards can be found using a linear-fractional program (LFP). In this method, key rewards on the constraints in the DCOP are treated as variables in the LFP. When the LFP is solved optimally, these variables are instantiated with values that, if used as constraint rewards, would produce the DCOP whose k-optima have lowest reward with respect to the optimal solution. Thus, this method gives a tight bound for any graph, since it instantiates the rewards for all constraints, but requires a globally optimal solution to the LFP, in contrast to the constant-time guarantees of Equations 3.1, 3.4 and 3.5. An LFP such as this is reducible to a linear program (LP) [Boyd and Vandenberghe, 2004] which is solvable in polynomial time with respect to the number of variables in the LP (the same as the number of variables in the LFP). The objective is to minimize $\frac{R(a)}{R(a^*)}$ such that $\forall \tilde{a} \in \tilde{A}, R(a) - R(\tilde{a}) \geq 0$, given $\tilde{A}$ as defined in Proposition 1. Note that $R(a^*)$ and $R(a)$ can be expressed as $\sum_{S \in \theta} R_S(a^*)$ and $\sum_{S \in \theta} R_S(a)$. We can now transform the DCOP so that every $R(\tilde{a})$ can also be expressed in terms of sums of $R_S(a^*)$ and $R_S(a)$, without changing or invalidating the guarantee on $R(a)$. Therefore, the LFP will contain only two variables for each $S \in \theta$, one for $R_S(a^*)$ and one for $R_S(a)$, where the domain of each one is the set of non-negative real numbers. The transformation is to set all reward functions $R_S(\cdot)$ for all $S \in \theta$ to 0, except for two cases: when all variables $i \in S$ have the same value as in a^*, or when all $i \in S$ have the same value as in a. This has no effect on $R(a^*)$ or $R(a)$, because $R_S(a^*)$ and $R_S(a)$ will be unchanged for all $S \in \theta$. It also has no effect on the optimality of a^* or the k-optimality of a, since the only change is to reduce the global reward for assignments other than a^* and a. Thus, the tight lower bound on $\frac{R(a)}{R(a^*)}$ still applies to the original DCOP.

3.3 Quality Guarantees for DCOPs with Hard Constraints

The previous sections in this chapter presented quality guarantees for DCOPs in which all rewards were restricted to being non-negative. A DCOP with both costs and rewards could be normalized to one that met this restriction as long as it contained no hard constraints (constraints with an infinitely large cost). However, in many domains, hard constraints exist, and solutions that violate a hard constraint are not useful. For example, a schedule where two people disagree on the time of their meeting with each other may be considered useless, no matter how good the schedule is for other people. This section shows how, in some cases, we can guarantee the solution quality of *k*-optimal DCOP solutions even when the DCOP contains hard constraints.

To obtain these guarantees we will assume that, in the DCOPs with hard constraints that we are considering, there always exists a solution that does not violate any hard constraints (i.e. that the globally optimal solution is a feasible solution). Given this assumption, we will show how the methods for obtaining guarantees given in the previous sections can be modified to allow for the existence of hard constraints. We define a hard constraint as a constraint in which at least one combination of values produces a sufficiently large cost (a negative reward many times larger than the sum of all positive rewards in the DCOP). Finally, we will assume that we know *a priori* which constraints in the DCOP graph are hard constraints (but not which combinations of values on these constraints cause them to be violated).

One complicating issue is that in some DCOPs with hard constraints, a *k*-optimal solution may be infeasible; that is, the *k*-optimal solution violates at least one hard constraint, and any deviation of *k* or fewer agents also results in an infeasible solution. If a *k*-optimum is infeasible, then it is impossible to guarantee its solution quality with respect to the global optimum. Therefore, to avoid these cases we will restrict our analysis to the following kind of DCOP: Consider a subgraph *H* of the DCOP constraint graph that consists all the hard constraints in the DCOP only. We will only consider DCOPs where the largest connected subgraph of *H* contains *k* or fewer agents (nodes). That is to say, we will not consider any DCOP for which a connected subgraph of *H* exists that contains more than *k* agents. In the DCOPs we are considering, no *k*-optimum could be infeasible. To see this, suppose an infeasible *k*-optimum did exist in such a DCOP. This means that at least one hard constraint in one of these such subgraphs is being violated. Since the number of agents in the subgraph is k or less, all these agents could change their values to the values that they take in the optimal solution. This change would ensure that no constraints in the subgraph were being violated, improving the overall reward. However, if k or fewer agents can improve the reward of a given assignment, it cannot be a k-optimum in the first place.

We now show how our methods for calculating quality guarantees in the previous section can be adapted to handle DCOPs with hard constraints. Previously, in the proof for the quality guarantees in Proposition 1, it was shown that, given a k-optimal solution a and any of the dominated assignments $\hat{a} \in \hat{A}^{a,k}$ (in which k agents are deviating from the k-optimum), the set of constraints θ in the DCOP could be divided into the sets $\theta_1(a, \hat{a})$, $\theta_2(a, \hat{a})$, and $\theta_3(a, \hat{a})$. As before, we can express the sum of the global rewards of all assignments $\hat{a} \in \hat{A}^{a,k}$ as:

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_2(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_3(a,\hat{a})} R_S(\hat{a}).$$

We know from our assumptions that

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) > 0$$

and

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_2(a,\hat{a})} R_S(\hat{a}) > 0$$

because otherwise the globally optimal solution or the k-optimal solution would contain a violated hard constraint, and that is not possible in the DCOPs we are considering. However, it is no longer certain that $\sum_{\hat{a}\in\hat{A}^{a,k}} \sum_{S\in\theta_3(a,\hat{a})} R_S(\hat{a}) > 0$ and so we cannot obtain Equation 3.2 as before. However, if we modify the description of $\hat{A}^{a,k}$ from Proposition 1 to exclude all assignments $\hat{a} \in \hat{A}^{a,k}$ where $S_{hard} \in \theta_3(a,\hat{a})$ for any hard constraint S_{hard}, then we can apply the methods of the previous section to obtain quality guarantees. Excluding all assignments $\hat{a} \in \hat{A}^{a,k}$ where $S_{hard} \in \theta_3(a,\hat{a})$ means including only the assignments $\hat{a} \in \hat{A}^{a,k}$ where $S_{hard} \in \theta_1(a,\hat{a}) \cup \theta_2(a,\hat{a})$, meaning only the assignments where all variables involved in all hard constraints are taking the same values as in the optimal solution, or where all variables involved in all hard constraints are taking the same values as in the k-optimal solution.

To see this, first consider a DCOP of six variables with domains of {0,1} on a star-shaped graph with agent 1 at the center (i.e. $\theta = \{S : 1 \in S, |S| = 2\}$). Suppose that $a = [0\ 0\ 0\ 0\ 0\ 0]$ is a 4-optimum, and that $a^* = [1\ 1\ 1\ 1\ 1\ 1]$ is the global optimum. In the case of no hard constraints, $d(a,a^*) = 6$, and $\hat{A}^{a,k}$ contains $\binom{d(a,a^*)-1}{k-1} = 10$ assignments, listed below:

[1 1 1 1 0 0], [1 1 1 0 1 0], [1 1 1 0 0 1], [1 1 0 1 1 0], [1 1 0 1 0 1],

[1 1 0 0 1 1], [1 0 1 1 1 0], [1 0 1 1 0 1], [1 0 1 0 1 1], [1 0 0 1 1 1].

By Proposition 4, $R(a) = \frac{3}{5}R(a^*)$.

Now suppose that $S_{hard} = \{1,2\}$ is a hard constraint. As mentioned above, we modify $\hat{A}^{a,k}$ to exclude all assignments $\hat{a} \in \hat{A}^{a,k}$ where $S_{hard} \in \theta_3(a,\hat{a})$. This means removing all assignments where $a_1 = 0$ and $a_2 = 1$ and all assignments where $a_1 = 1$ and $a_2 = 0$. Now, we are left with 6 assignments in $\hat{A}^{a,k} =$

[1 1 1 1 0 0], [1 1 1 0 1 0], [1 1 1 0 0 1], [1 1 0 1 1 0], [1 1 0 1 0 1],[1 1 0 0 1 1].

Since $R(a) \geq \hat{a}, \forall \hat{a} \in \hat{A}^{a,k}$, then $6 \cdot R(a) \geq \sum_{\hat{A}^{a,k}} R(\hat{a})$. Now, $\forall S \neq S_{hard} \in \theta, \forall i \in S, a_i^* = \hat{a}_i$ for exactly $\binom{n-2}{k-2} = 3$ assignments in $\hat{A}^{a,k}$. For example, for $S = \{1,3\}, a_1^* = \hat{a}_1 = 1$ and $a_2^* = \hat{a}_2 = 1$ for $\hat{a} = [1\ 1\ 1\ 1\ 0\ 0], [1\ 1\ 1\ 0\ 1\ 0]$, and $[1\ 1\ 1\ 0\ 0\ 1]$. Therefore, $\forall S \in \theta, R_S(a^*) = R_S(\hat{a})$ for these $\binom{n-2}{k-2} = 3$ assignments. Since this is a star graph, $\forall S \in \theta, \forall i \in S, a_i = \hat{a}_i$ for zero assignments in $\hat{A}^{a,k}$. Thus, $6 \cdot R(a) \geq \sum_{\hat{A}^{a,k}} R(\hat{a}) \geq 3 \cdot R(a^*) + 0 \cdot R(a)$, and so $R(a) \geq \frac{3}{6-0} R(a^*) = \frac{1}{2} R(a^*)$.□

For other graph types, we can apply an modification to the linear fractional program (LFP) method in the previous section. The original LFP was a minimization of $\frac{R(a)}{R(a^*)}$ such that $\forall \tilde{a} \in \tilde{A}, R(a) - R(\tilde{a}) \geq 0$, given $\tilde{A}$ as defined in Proposition 1. With the existence of hard constraints in the DCOP that can have an infinitely large negative reward, this constraint in the LFP no longer holds for all $\tilde{a}$. Instead this constraint holds only for those $\tilde{a} \in \tilde{A}$ where $S_{hard} \in \theta_1(a,\tilde{a}) \cup \theta_2(a,\tilde{a})$ for all hard constraints S_{hard}. This occurs because any assignment $\tilde{a}$ that does not meet this condition could violate a hard constraint, resulting in a large negative value for $R(\tilde{a})$. Not including these assignments produces a smaller $\tilde{a}$ and thus a smaller set of constraints in the LFP, resulting in a lower guarantee in the presence of hard constraints. This can be implemented by, when constructing the LFP, omitting any constraint in the LFP which corresponds to

The following proposition gives a more general proof for the case of more than one hard constraint in a star-shaped graph.

Proposition 5 *For any binary DCOP of n agents with a star graph structure, where all constraint rewards are non-negative except for h hard constraints, and a^* is the globally optimal solution, then, for any k-optimal assignment, a, where $k < n$ and* $0 < h < n - 1$.

$$R(a) \geq \frac{k-h-1}{n-h-1} R(a^*). \tag{3.6}$$

Proof: The proof is similar to the proof of Proposition 4. In a star graph, there are $\binom{n-1}{k-1}$ connected subgraphs of k variables, and so for the case of no hard constraints, $|\hat{A}^{a,k}| = \binom{n-1}{k-1}$ because each assignment $\hat{a} \in \hat{A}^{a,k}$ corresponds to one of those connected subgraphs. However, with hard constraints, we only include assignments in $\hat{A}^{a,k}$ where $S_{hard} \in \theta_1(a, \hat{a}) \cup \theta_2(a, \hat{a})$ for all hard constraints S_{hard}. In a star graph, there are $\binom{n-h-1}{k-h-1}$ connected subgraphs of k variables that include all variables involved in a hard constraint (corresponding to the assignments $\hat{a}$ where $S_{hard} \in \theta_1(a, \hat{a})$ for all hard constraints S_{hard}). Additionally, because all hard constraints must involve the central variable, there are (in the worst case, where $d(a, a^*) = n$) zero connected subgraphs of k variables that include all variables in the k-optimal solution($S_{hard} \in \theta_2(a, \hat{a})$ for all hard constraints S_{hard}). Therefore, $|\hat{A}^{a,k}| = \binom{n-h-1}{k-h-1}$. For every non-hard constraint S, there are $\binom{n-h-2}{k-h-2}$ connected subgraphs of k variables that contain S, and therefore,

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) = \binom{n-h-2}{k-h-2} R(a^*).$$

Finally, just as in the case without hard constraints, there are no ways to choose a constraint S so that it does not include any variable in a given connected subgraph of k variables. Therefore,

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_2(a,\hat{a})} R_S(\hat{a}) = 0 \cdot R(a).$$

and therefore Equation 3.6 holds. ■

When $h = n - 1$ (all constraints are hard constraints), it is easy to see that only $k = n$ will guarantee optimality; otherwise, no guarantee is possible.

Finally, we note that, these guarantees for k-optima in standard DCOPs with hard constraints also apply to multiply-constrained DCOPs as introduced in [Bowring et al., 2006]. In these problems, in addition to the standard DCOP constraint cost functions, there exist additional cost functions on certain agents, known as g-constraints. A g-constraint on an agent contains a function, whose inputs are the agent's value and all its neighbors' values, and whose output is a cost known as a g-cost (which is not related to the standard costs and rewards in the DCOP). Any assignment where an agent's g-cost exceeds its preassigned g-budget is infeasible. In other words, a g-constraint on an agent with m neighbors is a shorthand expression for the more general notion of a hard $m + 1$-ary constraint between the agent and all its neighbors (where any assignment where the g-budget is exceeded is considered infeasible). Thus, the methods of this section can be applied to obtain guarantees on k-optima in multiply-constrained DCOPs as well; this is shown in the experimental results later in the chapter.

3.4 Domination Analysis of *k*-Optima

In this section we now provide a different type of guarantee: lower bounds on the proportion of all possible DCOP assignments which any k-optimum must dominate in terms of solution quality. This proportion, called a *domination ratio*, provides a guide for how difficult it may be to find a solution of higher quality than a k-optimum; this metric is commonly used to evaluate heuristics for combinatorial optimization problems [Gutin and Yeo, 2005].

For example, suppose for some k, the solution quality guarantee from Section 3.2 for any k-optimum was 50% of optimal, but, additionally, it was known that any k-optimum was guaranteed to dominate 95% of all possible assignments to the DCOP. Then, at most only 5% of the other assignments could be of higher quality, indicating that it would likely be computationally expensive to find a better assignment, either with a higher k algorithm, or by some other method, and so a k-optimal algorithm should be used despite the low guarantee of 50% of the optimal solution quality. Now suppose instead for the same problem, the k-optimum was guaranteed to dominate only 20% of all assignments. Then it becomes more likely that a better solution could be found quickly, and so the k-optimal algorithm might not be recommended.

To find the domination ratio, observe that any k-optimum a must be of the same or higher quality than all $\tilde{a} \in \tilde{A}$ as defined in Proposition 1. So, the ratio is:

$$\frac{1 + |\tilde{A}|}{\prod_{i \in N} |\mathcal{A}_n|}. \tag{3.7}$$

If the constraint graph is fully connected (or not known, and so must be assumed to be fully connected), and each variable has q values, then $|\tilde{A}| = \sum_{j=1}^{k} \binom{n}{j}(q-1)^j$ and $\prod_{i \in N} |\mathcal{A}_n| = q^n$.

If the graph is known to be not fully connected, then the set $\tilde{A}$ from Equation 3.7 can be expanded to include assignments of distance greater than k from a, providing a stronger guarantee on the ratio of the assignment space that must be dominated by any k-optimum. Specifically, if a is k-optimal, then any assignment where *any number* of disjoint subsets of size $\leq k$ have deviated from a must be of the same or lower quality as a, as long as no constraint includes any two agents in different such subsets; this idea is illustrated below:

Example 3 *Consider a binary DCOP of five variables, numbered 1 to 5, with domains of two values, with unknown constraint graph. Any 3-optimum must be of equal or greater quality than* $\left(1 + \binom{5}{1} + \binom{5}{2} + \binom{5}{3}\right)/2^5 = 81.25\%$ *of all possible assignments, i.e. where 0, 1, 2, or 3 agents have deviated.*

Now, suppose the graph is known to be a chain with variables ordered by number. Since a deviation by either the variables {1,2} or {4,5} cannot increase global reward, and no constraint exists across these subsets, then neither can a deviation by {1,2,4,5}, even though four variables are deviating. The same applies to {1,3,4,5} and {1,2,3,5}, since both are made up of subsets of three or fewer variables that do not share constraints. So, a 3-optimum is now of equal or greater quality than $\left(1 + \binom{5}{1} + \binom{5}{2} + \binom{5}{3} + 3\right)/2^5 = 90.63\%$ *of all assignments.* $\square$

An improved guarantee can be found by enumerating the set $\tilde{A}$ of assignments $\tilde{a}$ with equal or lower reward than a; this set is expanded due to the DCOP graph structure as in the above example. The following proposition makes this possible; we introduce new notation for it: If we define n different subsets of agents as D_i for $i = 1 \ldots n$, we use $D^m = \cup_{i=1}^m D_i$, i.e. D^m is the union of the first m subsets. The proof is by induction over each subset D_i for $i = 1 \ldots n$.

Proposition 6 *Let a be some k-optimal assignment. Let* $\tilde{a}^n$ *be another assignment for which* $D(a, \tilde{a}^n)$ *can be expressed as* $D^n = \cup_{i=1}^n D_i$ *where:*

- $\forall D_i, |D_i| \leq k.$ (subsets contain k or fewer agents)
- $\forall D_i, D_j, D_i \cap D_j = \emptyset.$ (subsets are disjoint)
- $\forall D_i, D_j, \nexists i \in D_i, j \in D_j$ *such that* $i, j \in S$*, for any* $S \in \theta.$ (no constraint exists between agents in different subsets)

Then, $R(a) \geq R(\tilde{a}^n)$.

Proof:

Base case: If $n = 1$ then $D^n = D_1$ and $R(a) \geq R(\tilde{a}^n)$ by definition of k-optimality.

Inductive step: $R(a) \geq R(\tilde{a}^{n-1}) \Rightarrow R(a) \geq R(\tilde{a}^n)$.

The set of all agents can be divided into the set of agents in D^{n-1}, the set of agents in D_n, and the set of agents not in D^n. Also, by inductive hypothesis, $R(a) \geq R(\tilde{a}^{n-1})$. Therefore,

$$\begin{aligned} R(a) &= \sum_{S \in \theta : S \cap D^{n-1} \neq \emptyset} R_S(a) + \sum_{S \in \theta : S \cap D_n \neq \emptyset} R_S(a) + \sum_{S \in \theta : S \cap D^n = \emptyset} R_S(a) \\ &\geq \sum_{S \in \theta : S \cap D^{n-1} \neq \emptyset} R_S(\tilde{a}_S^{n-1}) + \sum_{S \in \theta : S \cap D_n \neq \emptyset} R_S(\tilde{a}^{n-1}) + \sum_{S \in \theta : S \cap D^n = \emptyset} R_S(\tilde{a}^{n-1}), \end{aligned}$$

and so

$$\sum_{S \in \theta : S \cap D^{n-1} \neq \emptyset} R_S(a) \geq \sum_{S \in \theta : S \cap D^{n-1} \neq \emptyset} R_S(\tilde{a}^{n-1})$$

because the rewards from agents outside D^{n-1} are the same for a and $\tilde{a}^{n-1}$.

Let a' be an assignment such that $D(a, a') = D_n = D(\tilde{a}^{n-1}, \tilde{a}^n)$. Because a is k-optimal, $R(a) \geq R(a')$; therefore,

$$\begin{aligned} R(a) &= \sum_{S \in \theta : S \cap D^{n-1} \neq \emptyset} R_S(a) + \sum_{S \in \theta : S \cap D_n \neq \emptyset} R_S(a) + \sum_{S \in \theta : S \cap D^n = \emptyset} R_S(a) \\ &\geq \sum_{S \in \theta : S \cap D^{n-1} \neq \emptyset} R_S(a') + \sum_{S \in \theta : S \cap D_n \neq \emptyset} R_S(a') + \sum_{S \in \theta : S \cap D^n = \emptyset} R_S(a'), \end{aligned}$$

and so

$$\sum_{S \in \theta : S \cap D_n \neq \emptyset} R_S(a) \geq \sum_{S \in \theta : S \cap D_n \neq \emptyset} R_S(a')$$

because the rewards from agents outside D_n are the same for a and a'.

We also know that

$$\sum_{S\in\theta:S\cap D^n=\emptyset} R_S(a) = \sum_{S\in\theta:S\cap D^n=\emptyset} R_S(\tilde{a}^n)$$

because the rewards from agents outside D^n are the same for a and $\tilde{a}^n$; therefore,

$$\begin{aligned} R(a) &\geq \sum_{S\in\theta:S\cap D^{n-1}\neq\emptyset} R_S(\tilde{a}^{n-1}) + \sum_{S\in\theta:S\cap D_n\neq\emptyset} R_S(a') + \sum_{S\in\theta:S\cap D^n=\emptyset} R_S(\tilde{a}^n) \\ &= \sum_{S\in\theta:S\cap D^{n-1}\neq\emptyset} R_S(\tilde{a}^n) + \sum_{S\in\theta:S\cap D_n\neq\emptyset} R_S(\tilde{a}^n) + \sum_{S\in\theta:S\cap D^n=\emptyset} R_S(\tilde{a}^n) \end{aligned}$$

because the rewards from D^{n-1} are the same for $\tilde{a}^{n-1}$ and $\tilde{a}^n$, and the rewards from D_n are the same for a' and $\tilde{a}^n$. Therefore, $R(a) \geq R(\tilde{a}^n)$. ■

3.5 Experimental Results

While the main thrust of this chapter is on theoretical guarantees for k-optima, this section provides an illustration of these guarantees in action, and how they are affected by constraint graph structure. Figures 3.1a, 3.1b, and 3.1c show quality guarantees for binary DCOPs with fully connected graphs, ring graphs, and star graphs, calculated directly from Equations 3.1, 3.4 and 3.5. Figure 3.1d shows quality guarantees for DCOPs whose graphs are binary trees, obtained using the LFP from Section 3.2. Constructing these NLPs and solving them optimally with LINGO 8.0 global solver took about two minutes on a 3 GHz Pentium IV with 1GB RAM. The x-axis plots the value chosen for k, and the y-axis plots the lower bound for k-optima as a percentage of the optimal solution quality for systems of 5, 10, 15, and 20 agents. These results show how the worst-case benefit of increasing k varies depending on graph structure. For example, in a five-agent DCOP, a 3-optimum is guaranteed to be 50% of optimal whether the graph is a star or a ring. However, moving to $k = 4$ means that worst-case solution quality will improve to 75% for a star, but only to 60% for a ring. For fully connected graphs, the benefit of increasing k goes up as k increases; whereas for stars it stays constant, and for chains it decreases, except for when $k = n$. Results for binary trees are mixed.

Figure 3.2 shows quality guarantees in the presence of hard constraints for fully connected graphs, ring graphs, and star graphs. These experiments began with a DCOP containing all soft constraints and no hard constraints, and gradually more and more soft constraints were made into hard constraints. The left column shows the effect of one and two hard constraints in a DCOP of five agents, and the right column shows the effect of two and four hard constraints in a DCOP of 10 agents. The constraints that were set as hard constraints were, in order, {0,1}, {2,3}, {4,5}, and {6,7}. This methodology was chosen so that no agent would be subject to more than one hard constraint, and so that k-optimal solutions would always be feasible. For star graphs, the guarantee from Proposition 5 was used; while for the others, the LFP method was used.

Figure 3.3 shows quality guarantees for a multiply-constrained DCOP with 30 agents, arranged in a ring structure. These experiments begin with a DCOP containing no agents with g-constraints. The number of agents with g-constraints was gradually increased by assigning a g-constraint to every third agent, starting with agent 0, until there were 10 agents with g-constraints. Similar to the previous experiments, this methodology was chosen so that k-optimal solutions would always be feasible. The LFP method was used to calculate the guarantees. Figure 3.3 shows guarantees for up to 4 agents with g-constraints as k increases. For the cases of 5 to 10 agents with g-constraints the guarantee was the same as for 4 agents with g-constraints.

Figure 3.4 shows the guarantees for *k*-optima from Section 3.4, expressed as percentile rankings over all possible assignments, for DCOPs where variables have domains of two values. This figure, when considered with Figure 3.1, provides insight into the difficulty of finding a solution of higher quality than a *k*-optimum. For example, a 7-optimum in a fully connected graph of 10 agents (Figure 3.1a) is only guaranteed to be 50% of optimal; however this 7-optimum is guaranteed to be of higher quality than 94.5% of all possible assignments to that DCOP (Figure 3.4a), which suggests that finding a better solution may be difficult. In contrast, a 3-optimum in a ring of 10 agents (Figure 3.1b) has the same guarantee of 50% of optimal solution, but this 3-optimum is only guaranteed to be of higher quality than 69% of all possible assignments, which suggests that finding a better solution may be easier.

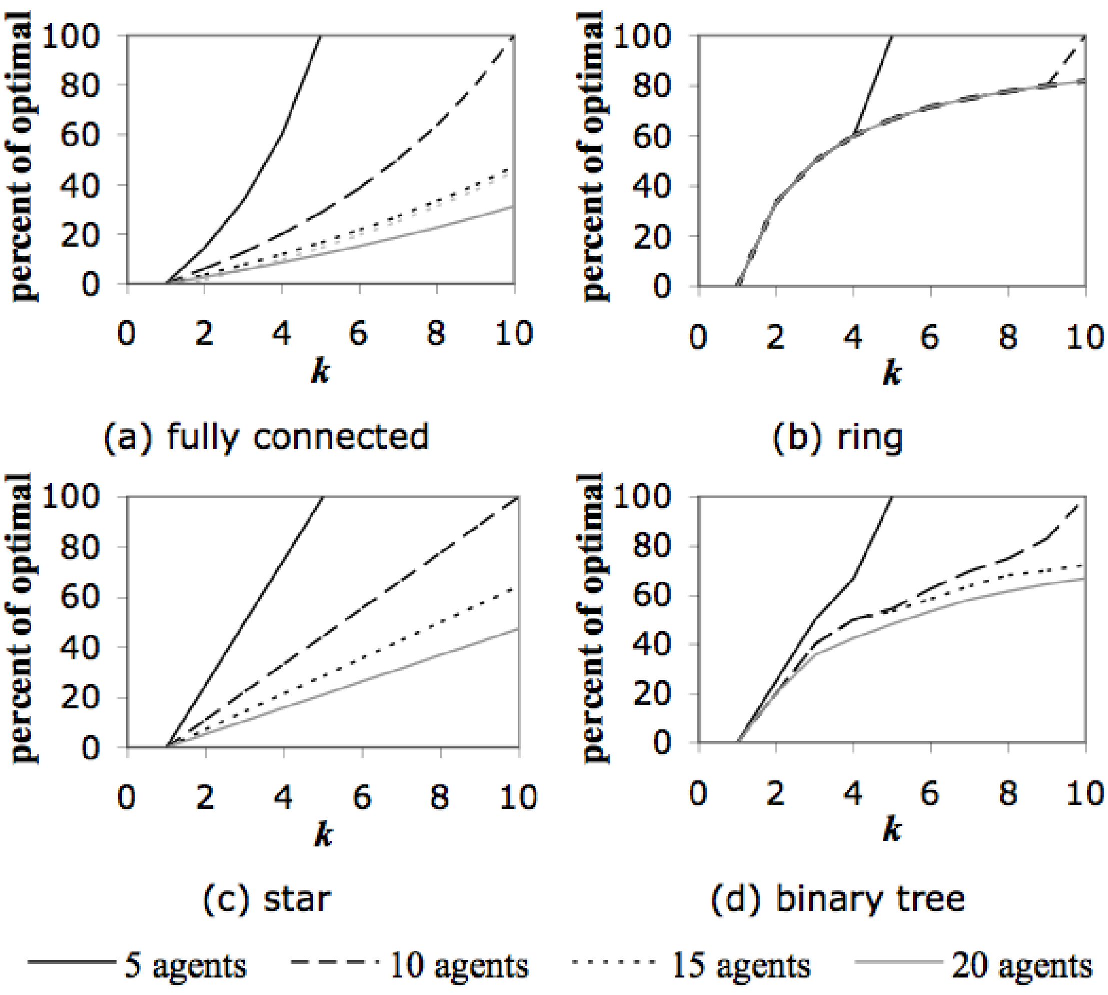


Figure 3.1: Quality guarantees for *k*-optima with respect to the global optimum for DCOPs of various graph structures.

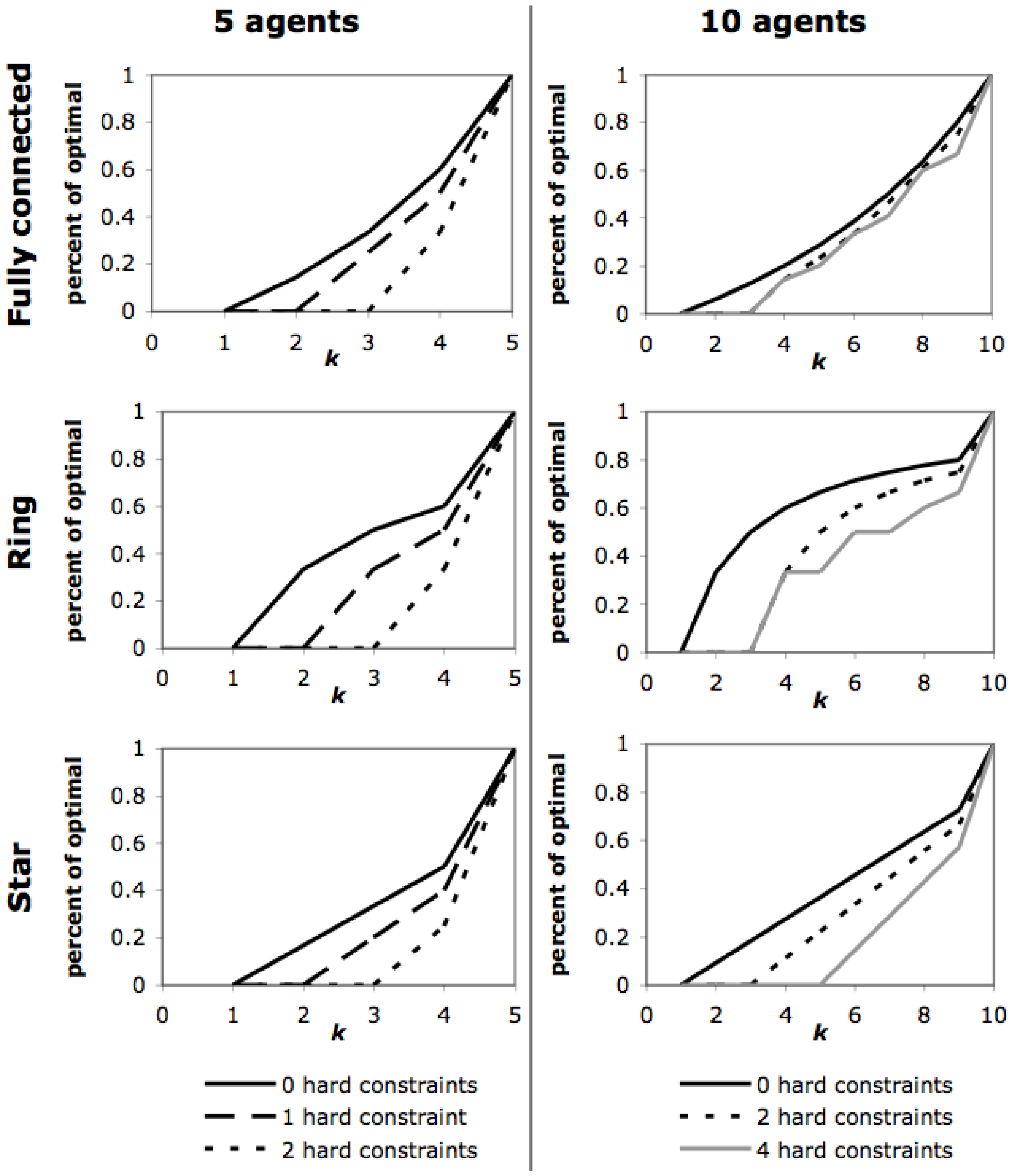


Figure 3.2: Quality guarantees for *k*-optima in DCOPs containing hard constraints.

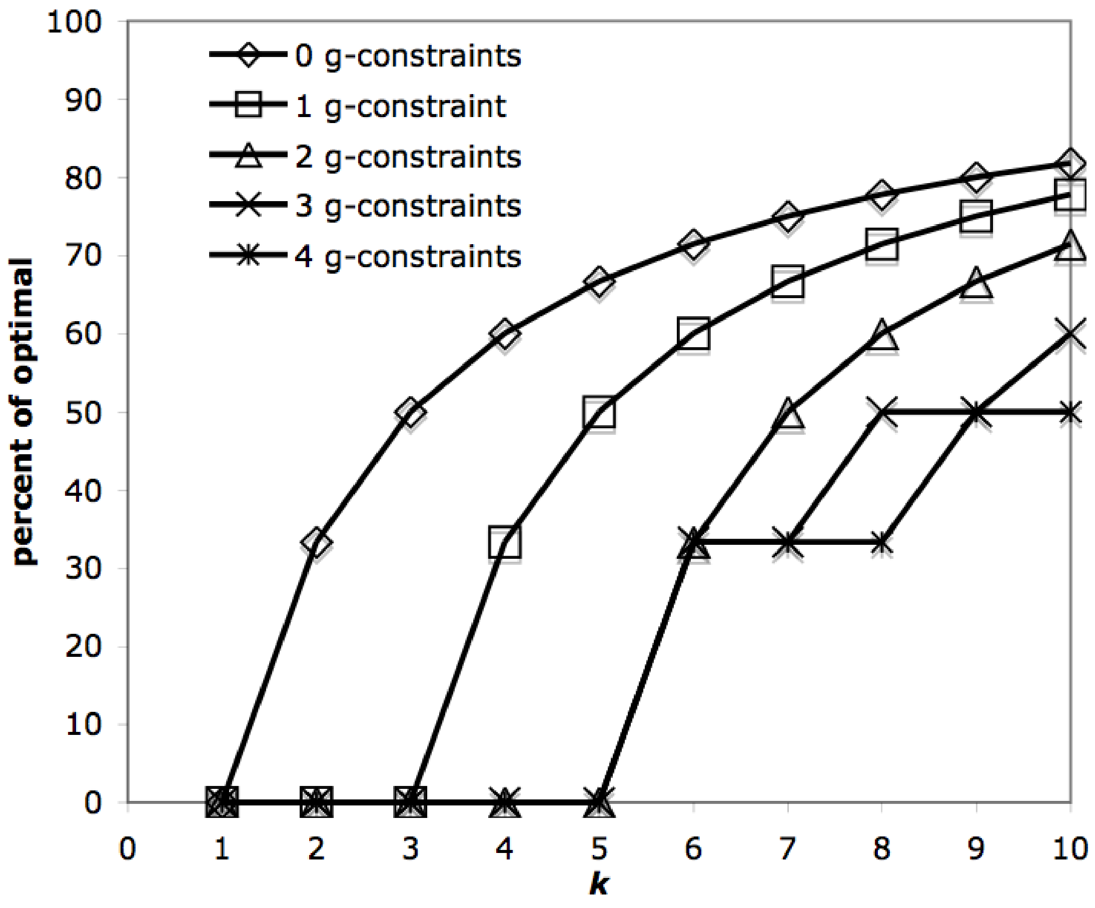


Figure 3.3: Quality guarantees for *k*-optima in a multiply-constrained ring DCOP.

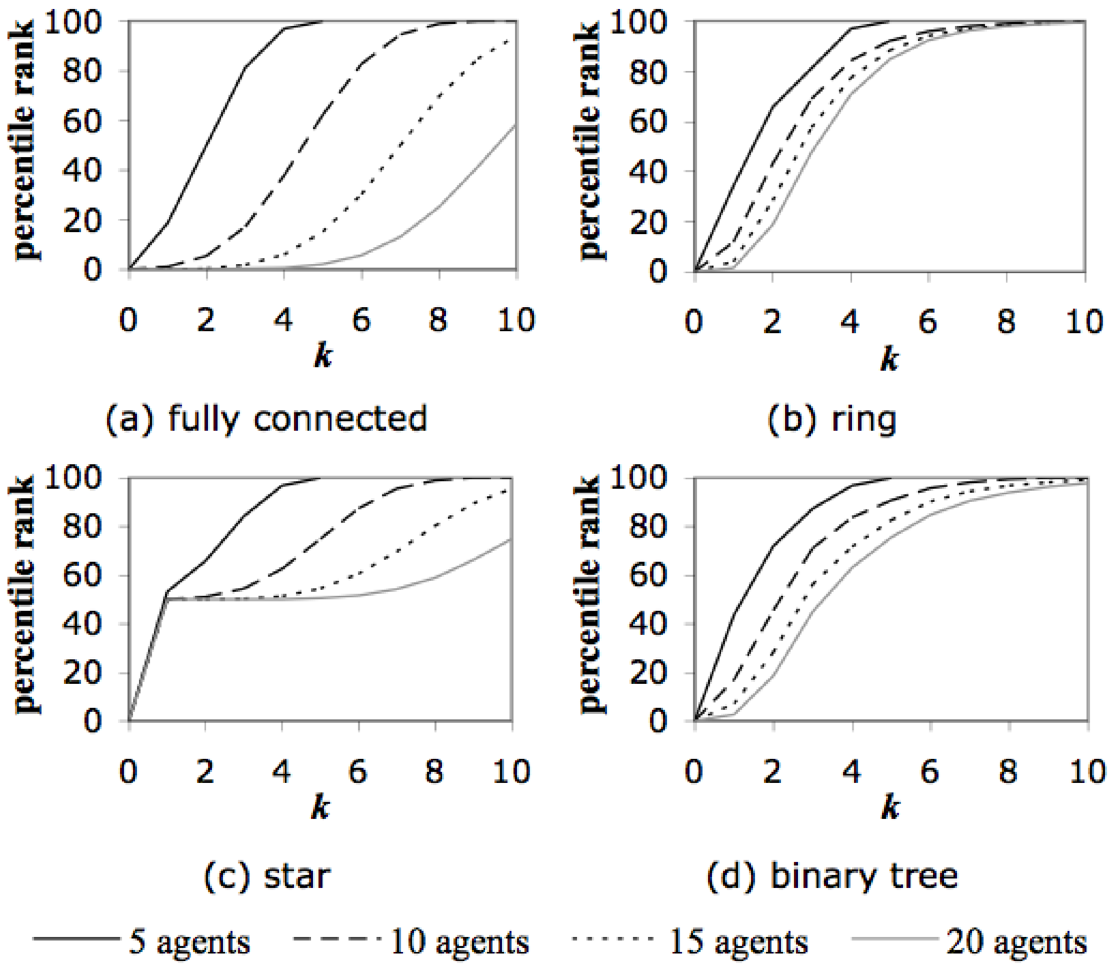


Figure 3.4: Quality guarantees for *k*-optima with respect to the space of dominated assignments for various graph structures.

Chapter 4

Upper Bounds on the Number of *k*-Optima in a DCOP

Traditionally, researchers have focused on obtaining a single DCOP solution, expressed as a single assignment of actions to agents. However, in this section, we consider a multi-agent system that generates a *set* of *k*-optimal assignments, i.e. multiple assignments to the same DCOP. Generating sets of assignments is useful in domains such as disaster rescue (to provide multiple rescue options to a human commander) [Schurr et al., 2005], patrolling (to execute multiple patrols in the same area) [Ruan et al., 2005], training simulations (to provide several options to a student) and others [Tate et al., 1998].

Domains requiring repeated patrols in an area by a team of UAVs (unmanned air vehicles), UGVs (unmanned ground vehicles), or robots, for peacekeeping or law enforcement after a disaster, provide one key illustration of the utility of *k*-optimality. For example, given a team of patrol robots in charge of executing multiple joint patrols in an area as in [Ruan et al., 2005], each robot may be assigned a region within the area. Each robot is controlled by a single agent, and hence, for one joint patrol, each agent must choose one of several possible routes to patrol within its region. A joint patrol is an assignment, where each agent's action is the route it has chosen to patrol, and rewards and costs arise from the combination of routes patrolled by agents in adjacent or overlapping regions. For example, if two nearby agents choose routes that largely overlap on a low-activity street, the constraint between those agents would incur a cost, while routes that overlap on a high-activity street would generate a reward. Agents in distant regions would not share a constraint. If reward alone is used as a metric to select joint patrols, then all selected joint patrols could be the same, except for the action of one agent. This set of patrols would be repetitive and predictable to adversaries. If we pick some diverse joint patrols at random, they may be very low-quality patrols. Using k-optimality directly addresses such circumstances; if no ties exist between the rewards of patrols a distance k or fewer apart, k-optimality ensures that all joint patrols differ by at least $k + 1$ agents' actions, as well as ensuring that this diversity would not come at the expense of obviously bad joint patrols, as each is optimal within a radius of at least k agents' actions.

Our key contribution in this section is addressing efficient resource allocation for the multiple assignments in a k-optimal set, by defining tight upper bounds on the number of k-optimal assignments that can exist for a given DCOP. These bounds are necessitated by two key features of the typical domains where a k-optimal set is applicable. First, each assignment in the set consumes some resources that must be allocated in advance. Such resource consumption arises because: (i) a team actually executes each assignment in the set, as in our patrolling example above, or (ii) the assignment set is presented to a human user (or another agent) as a list of options to choose from, requiring time. In each case, resources are consumed based on the assignment set size. Second, while the existence of the constraints between agents is known *a priori*, the actual rewards and costs on the constraints depend on conditions that are not known until runtime, and so resources must be allocated before the rewards and costs are known and before the agents generate the k-optimal assignment set. In the patrolling domain, constraints are known to exist between patrol robots assigned to adjacent or overlapping regions. However, their costs and rewards depend on recent field reports of adversarial activity that are not known until the robots are deployed. At this point the robots must already be fueled in order for them to immediately generate and execute a set of *k*-optimal patrols. The resource to be allocated to the robots is the amount of fuel required to execute each patrol; thus it is critical to ensure that enough fuel is given to each robot so that each assignment found can be executed, without burdening the robots with wasted fuel that will go unused. Consider another domain involving a team of disaster rescue agents that must generate a set of *k*-optimal assignments in order to present a set of diverse options to a human commander, where each option represents the best assignment within a neighborhood of similar assignments. The commander will choose one assignment for the team to actually execute. Constraints exist between agents whose actions must be coordinated (i.e. members of subteams) but their costs and rewards depend on conditions on the ground that are unknown until the time when the agents must be deployed. Here, the resource is the time the commander has to make the decision. Presenting too many options will cause the commander to run out of time before considering them all, and presenting too few may cause high-quality options to be omitted.

Because each assignment consumes resources, knowing the maximal number of *k*-optimal assignments that could exist for a given DCOP would allow us to allocate sufficient resources for a given level of *k*. Unfortunately, we cannot predict this number because the costs and rewards for the DCOP are not known in advance. Despite this uncertainty, reward-independent bounds can be obtained on the size of a *k*-optimal assignment set, i.e. to safely allocate enough resources for a given value of *k* for any DCOP with a particular graph structure. We first identify a mapping to coding theory, yielding bounds independent of both reward and graph structure. We then provide a method to use the structure of the DCOP graph (or hypergraph of arbitrary arity) to obtain significantly tighter bounds.

The third key contribution in this section is to establish a connection to noncooperative settings by proving that our bounds for 1-optima also apply to the number of pure-strategy Nash equilibria in any graphical game on a given graph, which remains an open problem. In addition to their uses in resource allocation, these bounds provide insight into the problem landscapes that can exist in both cooperative and noncooperative settings.

4.1 Upper Bounds on k-Optima

Upper bounds on the number of possible k-optimal assignments, $|A_q(n,k)|$ are useful for two reasons: they yield resource savings in domains where a particular level of k-optimality is desired, and help determine the appropriate level of k-optimality to prevent guaranteed waste of resources (fuel, time, etc.) in settings with fixed resources.

First, a particular level of k-optimality may be desired for an assignment set: a high k will include assignments that are more diverse, and optimal within a larger radius, but high-k algorithms have significantly higher coordination/communication overheads either in the number of messages passed or in the space and time required to centrally compute partial solutions to the DCOP [Modi et al., 2005; Mailler and Lesser, 2004; Maheswaran et al., 2004a]; hence lower k is preferable under time pressure. Lower k may also be preferable if an agent team or a human user wants a more detailed set of assignments, for example, more joint patrols, more rescue options, etc. For a given level of k-optimality, bounds indicate the maximum resource requirement for any k-optimal assignment set. Thus, tighter bounds provide savings by allowing fewer resources

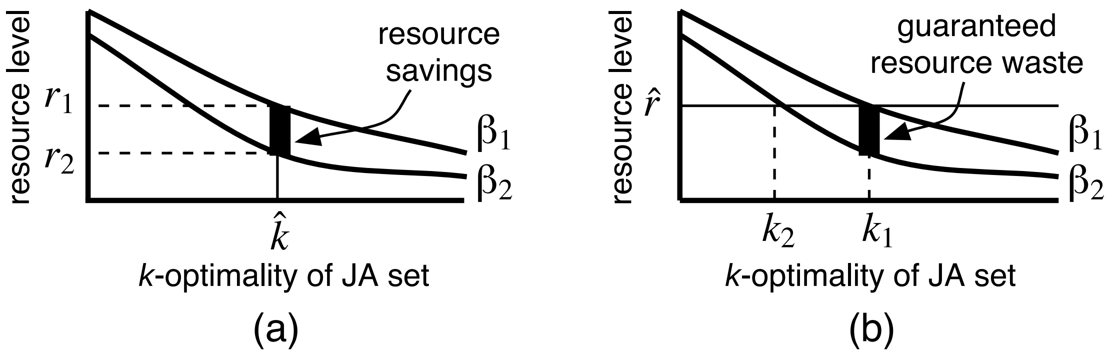


Figure 4.1: Hypothetical example illustrating the advantages of tighter bounds

to be allocated *a priori* while ensuring enough will be available for all k-optimal assignments, regardless of the rewards and costs on the constraints. Figure 4.1 is a hypothetical example, with k on the x-axis and the number of resources to be allocated on the y-axis. β_1 and β_2 are two different upper bounds on the number of k-optimal assignments that can exist for a given DCOP. Part (a) shows how the tighter bound β_2 indicates that a resource level of r_2 is sufficient for all $\hat{k}$-optimal assignments, if each assignment consumes one resource, yielding a savings of $r_1 - r_2$ over using β_1.

Second, if resource availability is fixed, tighter bounds help us choose an appropriate level of k-optimality. If k is too low, we may exhaust our resources on bad assignments (similar assignments with poor relative quality). In contrast, fewer k-optimal assignments can exist as k increases, and so if k is too high, available resources that could be spent on additional assignments are guaranteed to go unused. Tighter bounds provide a more accurate measure of this kind of guaranteed waste and thus, allow a more appropriate k to be chosen. In Figure 4.1(b), under fixed resource level $\hat{r}$, the looser bound β_1 hides the resources guaranteed to go unused when k_1 is used. This waste is revealed by β_2, with the thick line indicating the resources that, if allocated, will never be used, as there cannot exist enough k-optima to use them all; instead, we now see that using k_2 will reduce this guaranteed waste.

To find the first upper bounds on the number of k-optima for a given DCOP graph, we discovered a correspondence to coding theory [Ling and Xing, 2004]. In error-correcting codes, a set of codewords must be chosen from the space of all possible words, where each word is a string of characters from an alphabet. All codewords are sufficiently different from one another so that transmission errors will not cause one to be confused for another. Finding the maximum possible number of k-optima can be mapped to finding the maximum number of codewords in a space of q^n words where the minimum distance between any two codewords is $d = k + 1$. We can map DCOP assignments to words and k-optima to codewords as follows: an assignment a taken by n agents each with a domain of cardinality q is analogous to a word of length n from an alphabet of cardinality q. The distance $d(a, \tilde{a})$ can then be interpreted as a Hamming distance between two words. Then, if a is k-optimal, and $d(a, \tilde{a}) \leq k$, then $\tilde{a}$ cannot also be k-optimal by definition. Thus, any two k-optima must be separated by distance $\geq k + 1$.

Three well-known bounds [Ling and Xing, 2004] on codewords are Hamming:

$$\beta_H = q^n / \left(\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{n}{j} (q-1)^j \right),$$

Singleton:

$$\beta_S = q^{n-k},$$

and Plotkin:

$$\beta_P = \left\lfloor \frac{k+1}{k+1-(1-q^{-1})n} \right\rfloor,$$

which is only valid when $(1-1/q)n < k+1$. Note that for the special case of $q = 2$ (all constraints are binary), it is possible to use the relation $\beta_H(n,k,q) = \beta_H(n-1,k-1,q)$[Ling and Xing, 2004] to obtain a tighter bound for odd k using the Hamming bound. Now, to find a reward-independent bound on the number of 1-optima for three agents with $q = 2$, (e.g., the system in Example 1), we obtain $\min\{\beta_H, \beta_S, \beta_P\} = \beta_H = 4$, without knowing R_{12} and R_{23} explicitly.

Unfortunately, problems of even d (odd k), are not of interest for error-correcting codes, and β_H, the Hamming bound, is very loose or useless for DCOP when $q > 2$, e.g., for 1-optima (solutions reached by DSA) the bound is equal to the number of possible assignments in this case. Hence, for DCOP, we pursue an improved bound for $q > 2$ and odd k. β_H is derived by using a sphere-packing argument stating that the total number of words q^n must be greater than the number of codewords $A_q(n,k)$ multiplied by the size of a sphere of radius $\lfloor k/2 \rfloor$ centered around each codeword. A sphere $S_A(a^*, r)$ with center a^* and radius r is the set of assignments $\tilde{a}$ such that $d(a^*, \tilde{a}) \leq r$, and represents words that cannot be codewords (except for its center). It can be shown that $S_A(a^*, \lfloor k/2 \rfloor)$ contains exactly $\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{n}{j}(q-1)^j$ words. If k is even, the tightest packing occurs with spheres of radius $k/2$ and each word can be uniquely assigned to the sphere of its closest codeword. If k is odd, it is possible for a word to be equidistant from two codewords and it is unclear how to assign it to a sphere. The Hamming bound addresses this issue by using the bound for $k-1$ when k is odd, which leads to smaller spheres and a bound larger than necessary. This ignores the contribution of a word that lies on the "boundary" between several spheres. These boundary assignments can be appropriately partitioned to achieve a tighter bound on the number of k-optima for odd k, called the *Modified Hamming bound*.

Proposition 7 *For odd k,* $A_q(n,k) \le \min\{A_1, A_2\}$ *where*

$$\begin{aligned} A_1 &= \frac{q^n - \binom{n}{(k+1)/2}(q-1)^{(k+1)/2}}{\sum_{j=0}^{\lfloor k/2 \rfloor}\binom{n}{j}(q-1)^j} \\ A_2 &= \frac{q^n}{\sum_{j=0}^{\lfloor k/2 \rfloor}\binom{n}{j}(q-1)^j + \binom{n}{(k+1)/2}(q-1)^{(k+1)/2}(n^{-1})} \end{aligned}$$

Proof. Any word that has Hamming distance $\lfloor k/2 \rfloor$ or less from a codeword belongs in that codeword's sphere, because belonging to more than one sphere would violate the code's distance requirement. Given an odd value of k, each codeword will have $\binom{n}{(k+1)/2}(q-1)^{(k+1)/2}$ words that are a distance of $(k+1)/2$ away from it. It cannot claim all these words for its sphere exclusively, as they may be equidistant from other codewords. We do know however that each of these words can be on the boundary of at most n spheres (i.e. can be equidistant from at most n codewords) because they are of length n. Furthermore, each of these words can be equidistant from at most $A_q(n,k)$ codewords, i.e. the total number of codewords in the space. Thus, each codeword can safely incorporate $1/\min\{n, A_q(n,k)\}$ of each of these boundary words into its sphere without any portion being claimed by more than one sphere. Aggregating over all the words on the boundary, we can increase the volume of the sphere by $\binom{n}{(k+1)/2}(q-1)^{(k+1)/2}/\min\{n, A_q(n,k)\}$.

Using the sphere-packing argument with the portions of the boundary words added to each sphere, if $A_q(n,k) \leq n$, we have

$$q^n \geq A_q(n,k)\Bigg[\sum_{j=0}^{\lfloor k/2 \rfloor}\binom{n}{j}(q-1)^j + \frac{\binom{n}{(k+1)/2}(q-1)^{(k+1)/2}}{A_q(n,k)}\Bigg]$$

$$\Rightarrow A_q(n,k) \leq \frac{q^n - \binom{n}{(k+1)/2}(q-1)^{(k+1)/2}}{\sum_{j=0}^{\lfloor k/2 \rfloor}\binom{n}{j}(q-1)^j} \equiv A_1,$$

and if $A_q(n,k) \geq n$, we have

$$q^n \geq A_q(n,k)\Bigg[\sum_{j=0}^{\lfloor k/2 \rfloor}\binom{n}{j}(q-1)^j + \frac{\binom{n}{(k+1)/2}(q-1)^{(k+1)/2}}{n}\Bigg]$$

$$\Rightarrow A_q(n,k) \leq \frac{q^n}{\sum_{j=0}^{\lfloor k/2 \rfloor}\binom{n}{j}(q-1)^j + \binom{n}{(k+1)/2}(q-1)^{(k+1)/2}(n^{-1})} \equiv A_2.$$

We have $A_q(n,k) \leq n \Rightarrow A_q(n,k) \leq A_1$ and $A_q(n,k) \geq n \Rightarrow A_q(n,k) \leq A_2$. We can show that $A_1 \odot n \Leftrightarrow A_2 \odot n$, $\forall \odot \in \{<,>,=\}$. Furthermore, $A_1 \odot n, A_2 \odot n \Leftrightarrow A_1 \odot A_2$. Thus, when $A_1 \leq n$, then $A_2 \leq n$ and $A_1 \leq A_2$. So, $A_q(n,k) \leq A_1 = \min\{A_1, A_2\}$ when $A_1 \leq n$. And, when $A_1 > n$, then $A_2 > n$ and $A_1 > A_2$. So, $A_q(n,k) \leq A_2 = \min\{A_1, A_2\}$ when $A_1 > n$. Therefore, $A_q(n,k) \leq \min\{A_1, A_2\}$.■

We call the Modified Hamming bound β_{MH} and define $\beta_{HSP} = \min\{\beta_H, \beta_S, \beta_P, \beta_{MH}\}$, including the relation for β_H for $q = 2$; i.e. β_{HSP} gives the best of all the (graph-independent) bounds so far.

4.2 Graph-Based Analysis: k-Optima

The β_{HSP} bound and its components depend only on n, k and q, regardless of how the team reward is decomposed onto constraints; i.e., the bounds are the same for all θ. For instance, the bound on 1-optima for Example 1 (found to be 4 in the previous section) ignored the fact that agents 1 and 3 do not share a constraint, and yields the same result independent of the DCOP graph structure. However, exploiting this structure (as captured by θ) can significantly tighten the bounds on $\{|A_q(n,k)|\}_{k=1}^{n}$. In particular, in obtaining the bounds in Section 4.1, pairs of assignments were mutually exclusive as k-optima (only one of the two could be k-optimal) if they were separated by a distance $\leq k$. We now show how some assignments separated by a distance $\geq k+1$ must also be mutually exclusive as k-optima.

We define $D_G(a,\tilde{a}) := \{i \in G : a_i \neq \tilde{a}_i\}$ and $V(G) := \cup_{S\in\theta : G\cap S\neq\emptyset} S$. Intuitively, $D_G(a,\tilde{a})$ is the set of agents within the subgroup G who have chosen different actions between a and $\tilde{a}$, and $V(G)$ is the set of agents (including those in G) who are a member of some constraint $S \in \theta$ incident on a member of G (e.g., G and the agents who share a constraint with some member of G). Then, $V(G)^C$ is the set of all agents whose contribution to the team reward is independent of the values taken by G.

Proposition 8 *Let there be an assignment $a^* \in A_q(n,k)$ and let $\tilde{a} \in A$ be another assignment for which $d(a^*,\tilde{a}) > k$. If $\exists\, G \subset \mathcal{N}$, $G \neq \emptyset$ for which $|G| \leq k$ and $D_{V(G)}(a^*,\tilde{a}) = G$, then $\tilde{a} \notin A_q(n,k)$. In other words, if $\tilde{a}$ contains some group G that is facing the same context as it does in a^*, but some agent in G is choosing a different value than it does in a^*, then a^* and $\tilde{a}$ cannot both be k-optimal.*

Proof. Given a^*, $\tilde{a}$, and G with the properties stated above, we have that $\forall a : d(a^*, a) \leq k$, $\Delta(a^*, a) > 0$. In other words, any assignment a that is a distance of k or less from a^* must have a lower reward than a^*. Let us define a particular a such that $a_i = \tilde{a}_i$ for $i \in V(G)$ and $a_i = a_i^*$ for $i \notin V(G)$; i.e., the agents in $V(G)$ take the same values as in $\tilde{a}$ and the other agents take the same values as in a^*. Then, by our initial conditions, $D(a^*, a) = D_{V(G)}(a^*, \tilde{a}) = G$, and therefore $d(a^*, a) \leq k$ which implies that a^* has a higher reward than a:

$$\Delta(a^*, a) = \sum_{S \in \theta : S \cap D(a^*, a) \neq \emptyset} R_S(a_S^*) - R_S(a_S) = \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(a_S^*) - R_S(a_S)$$
$$= \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(a_S^*) - R_S(\tilde{a}_S) > 0.$$

Now let us define a new assignment, $\hat{a}$, such that $\hat{a}_i = a_i^*$ for $i \in V(G)$ and $\hat{a}_i = \tilde{a}_i$ for $i \notin V(G)$. In other words, the agents in $V(G)$ take the same values as in a^* and the other agents take the same values as in $\tilde{a}$. Then $D(\tilde{a}, \hat{a}) = D_{V(G)}(\tilde{a}, a^*) = G$ and $d(\tilde{a}, \hat{a}) \leq k$, which implies that $\tilde{a}$ has a lower reward than $\hat{a}$:

$$\Delta(\tilde{a}, \hat{a}) = \sum_{S \in \theta : S \cap D(\tilde{a}, \hat{a}) \neq \emptyset} R_S(\tilde{a}_S) - R_S(\hat{a}_S) = \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(\tilde{a}_S) - R_S(\hat{a}_S)$$
$$= \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(\tilde{a}_S) - R_S(a_S^*) < 0.$$

Thus, $\tilde{a} \notin A_q(n, k)$ because $\Delta(\tilde{a}, \hat{a}) < 0$ and $d(\tilde{a}, \hat{a}) \leq k$. ($\tilde{a}$ has a lower reward than $\hat{a}$ which is a distance of k or less away, and thus cannot be k-optimal if a^* is k-optimal.) ■

Proposition 8 provides conditions where if a^* is k-optimal then $\tilde{a}$, which may be separated from a^* by a distance greater than $k + 1$ may not be k-optimal, thus tightening bounds on k-optimal assignment sets. With Proposition 8, since agents are typically not fully connected to all

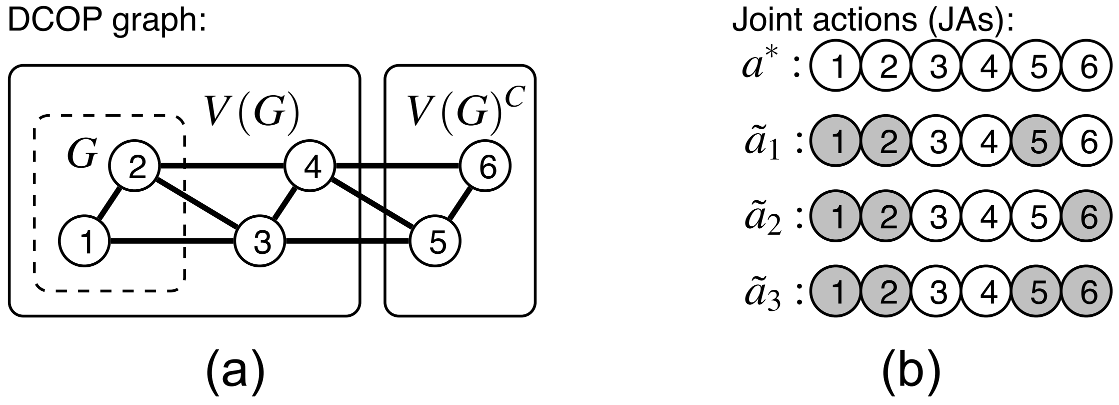


Figure 4.2: A visual representation of the effect of Proposition 8.

other agents, the *relevant context* a subgroup faces is not the entire set of other agents. Thus, the subgroup and its relevant context form a view (captured by $V(G)$) that is not the entire team. We consider the case where an assignment $\tilde{a}$ has $d(a^*, \tilde{a}) > k$. We also have group G of size k within whose view $V(G)$, G are the only deviators between a^* and $\tilde{a}$ (although agents outside the view must also have deviated, because $d(a^*, \tilde{a}) > k$). We then know that $\tilde{a}$ contains a group G of size k or less that has taken a suboptimal subgroup assignment of values to variables with respect to its relevant context and thus $\tilde{a}$ cannot be k-optimal, i.e. if the group G chose a^*_G instead of $\tilde{a}_G$ under its relevant context $V(G) \setminus G$ for $\tilde{a}$, then team reward would increase. Finally, we note that this proposition does not imply any relationship between the reward of a^* and that of $\tilde{a}$.

Figure 4.2(a) shows G, $V(G)$, and $V(G)^C$ for a sample DCOP of six agents with a domain of two actions, white and gray. Without Proposition 8, $\tilde{a}_1$, $\tilde{a}_2$, and $\tilde{a}_3$ could all potentially be 2-optimal. However, Proposition 8 guarantees that they are not, leading to a tighter bound on the number of 2-optima that could exist. To see the effect, note that if a^* is 2-optimal, then $G = \{1, 2\}$, a subgroup of size 2, must have taken an optimal subgroup joint action (all white) given its relevant context (all white). Even though $\tilde{a}_1$, $\tilde{a}_2$, and $\tilde{a}_3$ are a distance greater than 2 from a^*, they cannot be 2-optimal, since in each of them, G faces the same relevant context (all white) but is now taking a suboptimal subgroup joint action (all gray).

To explain the significance of Proposition 8 to bounds, we introduce the notion of an *exclusivity relation* $E \subset \mathcal{N}$ which captures the restriction that if deviating group $D(a, \tilde{a}) = E$, then at most one of a and $\tilde{a}$ can be k-optimal. An *exclusivity relation set* for k-optimality, $\mathcal{E}_k \subset \mathcal{P}(\mathcal{N})$, is a collection of such relations that limits $|A_q(n,k)|$, the number of assignments that can be k-optimal in a reward-independent setting (otherwise every assignment could potentially be k-optimal). In particular, the set $\mathcal{E}_k$ defines an *exclusivity graph* H_k where each node corresponds uniquely to one of all q^n possible assignments. Edges are defined between pairs of assignments, a and $\tilde{a}$, if $D(a, \tilde{a}) \in \mathcal{E}_k$. The size of the maximum independent set (MIS) of H_k, the largest subset of nodes such that no pair defines an edge, gives an upper bound on $|A_q(n,k)|$. Naturally, an expanded $\mathcal{E}_k$ implies a more connected exclusivity graph and thus a tighter bound on $|A_q(n,k)|$.

Without introducing graph-based analysis, β_{HSP} for each k provides a bound on the MIS of H_k when

$$\mathcal{E}_k = \bigcup_{E \subset \mathcal{N} : 1 \le |E| \le k} E.$$

This set $\mathcal{E}_k$ captures only the restriction that no two assignments within a distance of k can both be k-optimal. Consider Example 1, but with unknown rewards on the links. Here, the exclusivity relation set for 1-optima without considering the DCOP graph is $\mathcal{E}_1 = \{\{1\}, \{2\}, \{3\}\}$, meaning that no two assignments differing only by the action taken by either agent 1, 2, or 3, can both be 1-optimal. This leads to the exclusivity graph in Figure 4.3(a) whose MIS implies a bound of 4.

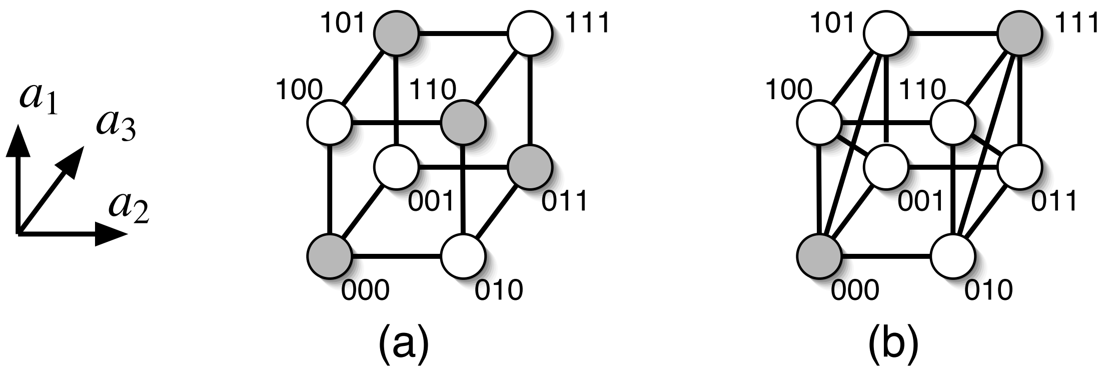


Figure 4.3: Exclusivity graphs for 1-optima for Example 1 with MIS shown in gray, (a) not using Proposition 8 and (b) using it.

The significance of Proposition 8 is that it provides additional exclusivity relations for solutions separated by distance $\geq k + 1$, which arise only because we considered the structure of the DCOP graph, which will allow a tighter bound to be computed. This graph-based exclusivity relation set is

$$\widetilde{\mathcal{E}}_k = \bigcup_{E \subset \mathcal{N}: 1 \leq |E| \leq k} \bigcup_{F \in \mathcal{P}(V(E)^C)} [E \cup F]$$

which is a superset of $\mathcal{E}_k$. Additional relations exist because multiple different exclusivity relations ($\bigcup_{F \in \mathcal{P}(V(E)^C)} [E \cup F]$) appear the same to the subgroup E because of its reduced view $V(E)$. Now, for Example 1, the exclusivity relation set for 1-optima when considering the DCOP graph is $\widetilde{\mathcal{E}}_1 = \{\{1\}\{2\}, \{3\}, \{1, 3\}\}$, which now has the additional relation {1,3}. This relation, included because of the realization that agents 1 and 3 are not connected, says that no two assignments can both be 1-optimal if they differ only in the actions of both agent 1 and agent 3. This leads to the exclusivity graph in Figure 4.3(b) whose MIS implies a bound of 2. Algorithms for obtaining bounds using $\widetilde{\mathcal{E}}_k$ will be discussed in Section 4.4.

4.3 Application to Nash Equilibria

Our graph-based bounds can be extended beyond agent teams to noncooperative settings. It is possible to employ the same exclusivity relations for 1-optimal DCOP assignments to bound the number of pure-strategy Nash equilibria in a graphical game (of the same graph structure) using any of our bounds for $|A_q(n, 1)|$. Bounds on Nash equilibria [McLennan and Park, 1999] are useful for design and analysis of mechanisms as they predict the maximum number of outcomes of a game.

We begin with a set of noncooperative agents $\mathcal{N} = \{1, \ldots n\}$, where the i^{th} agent's utility is

$$U^i(a_i; a_{\{\mathcal{N}\setminus i\}}) = \sum_{S_i \in \theta_i} U^i_{S_i}(a_i; a_{\{S_i \setminus i\}})$$

which is a decomposition into an aggregation of component utilities generated from minimal subgroups. Note that the combination of actions taken by any subgroup of agents may generate utility for any agent i, therefore the subgroups are denoted as S_i rather than S, as in the cooperative case, where the utility went to the entire team. The notation a_i and $a_{\{G\setminus i\}}$ refers to the i^{th} agent's action and the actions of the group G with i removed, respectively. We refer to a as an assignment, with the understanding that it is composed of actions motivated by individual utilities. Let the *view* of the i^{th} agent in a noncooperative setting to be $V(i) = \cup_{S_i \in \theta_i} S_i$. The deviating group with respect to G is: $D_G(a, \tilde{a}) := \{i \in G : a_i \neq \tilde{a}_i\}$. Assuming every player has a unique optimal response to its context, then if a^* is a pure-strategy Nash equilibrium, and $d(a^*, a) = 1$, $i = D(a^*, a)$, we know that

$$U^i(a^*_i; a^*_{\{\mathcal{N}\setminus i\}}) > U^i(a_i; a^*_{\{\mathcal{N}\setminus i\}})$$

and a is not a pure-strategy Nash equilibrium. However, applying the graph (or hypergraph) structure of the game, captured by the sets $\{\theta_i\}$, we get exclusivity relations between assignments with distance > 1 as follows.

Proposition 9 *If a^* is a pure-strategy Nash equilibrium, $\tilde{a} \in A$ such that $d(a^*, \tilde{a}) > 1$, and $\exists i \in \mathcal{N}$ such that $D_{V(i)}(a^*, \tilde{a}) = i$, then $\tilde{a}$ is not a pure-strategy Nash equilibrium.*

Proof. We have:

$$\begin{aligned} U^i(\tilde{a}_i; \tilde{a}_{\{\mathcal{N} \setminus i\}}) &= \sum_{S_i \in \theta_i} U^i_{S_i}(\tilde{a}_i; \tilde{a}_{\{S_i \setminus i\}}) = \sum_{S_i \in \theta_i} U^i_{S_i}(\tilde{a}_i; a^*_{\{S_i \setminus i\}}) \\ &< \sum_{S_i \in \theta_i} U^i_{S_i}(a^*_i; a^*_{\{S_i \setminus i\}}) = \sum_{S_i \in \theta_i} U^i_{S_i}(a^*_i; \tilde{a}_{\{S_i \setminus i\}}) = U^i(a^*_i; \tilde{a}_{\{\mathcal{N} \setminus i\}}). \end{aligned}$$

The first and last equalities are by definition. The second and third equalities are because

$$D_{V(i)}(a^*, \tilde{a}) = i.$$

The inequality is because a^* is a pure-strategy Nash equilibrium. The result is that $\tilde{a}_i$ is not an optimal response to $\tilde{a}_{\{\mathcal{N} \setminus i\}}$ and thus cannot be a Nash equilibrium. ■

Proposition 9 states that a^* and $\tilde{a}$ cannot both be Nash equilibria if $\exists i,\ \ D_{V(i)}(a^*, \tilde{a}) = i$, which is identical to the condition that prevents two assignments (in a team setting) from being 1-optimal. The commonality is that in both the cooperative and noncooperative settings, agents have optimal actions for any given context, and in both settings there is a notion of relevant context, $V(i) \setminus i$, which can be a subset of other agents $\{\mathcal{N} \setminus i\}$. The difference is that the views are generated in different manners: $V(i) = \cup_{S \in \theta : i \cap S \neq \emptyset} S$ in a cooperative setting, while $V(i) = \cup_{S_i \in \theta_i} S_i$ in a noncooperative setting. Given the views, we can generate the exclusivity relation set in the same manner, $\mathcal{E}_1 = \bigcup_{i \in \mathcal{N}} \bigcup_{F \in \mathcal{P}(V(i)^C)} [i \cup F]$. Given the exclusivity relation set, we can create an exclusivity graph for a noncooperative setting in a fashion similar to the one in Section 4.2. Thus, the bound on the number of Nash equilibria for a noncooperative graphical game is identical to the bound on 1-optimal assignments for a cooperative DCOP, if both share the same exclusivity relation set $\mathcal{E}_1$.

4.4 Graph-Based Bounds

As seen earlier, the graph structure expands the exclusivity relation set for k-optimality in cooperative (DCOP) settings and Nash equilibria in noncooperative (graphical-game) settings. This set defines exclusivity graph H_k whose maximum independent set (MIS) provides a bound for the number of k-optimal assignments (or alternatively, for the number of Nash equilibria). Finding the size of the MIS is NP-complete in the general case [Alon and Kahale, 1998], so we investigated heuristic techniques to obtain an upper bound on $|A_q(n,k)|$. We observe that any fully-connected subset (clique) of H_k can contain at most one k-optimum. Thus, the number of cliques in any clique partitioning of H_k also provides an upper bound on $|A_q(n,k)|$, where a partitioning yielding fewer cliques will provide a tighter bound. Hence, our first approach is the polynomial-time F_{CLIQUE} clique-partitioning algorithm, shown in [Kim and Shin, 2002] to outperform several competitors.

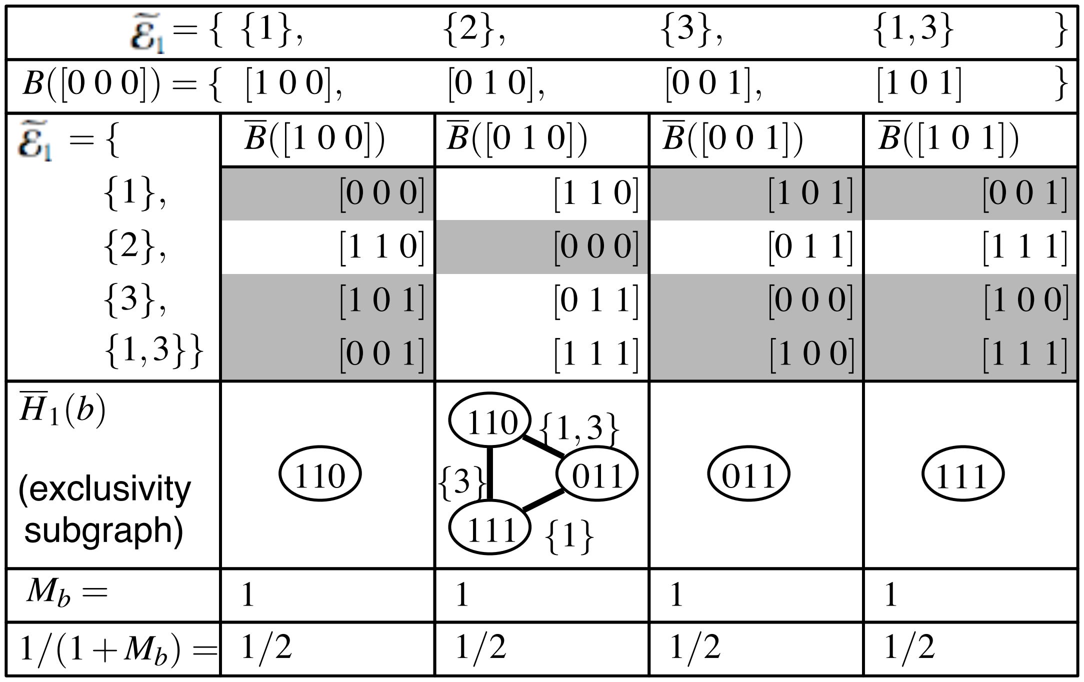
Our second heuristic technique to find a graph-based bound is Algorithm 1, the *Symmetric Region Packing bound*, β_{SRP}, which uses a packing method analogous to Proposition 7, where each k-optimum claims a region of the space of all possible assignments (the nodes of H_k). Because these regions are constructed to be disjoint and have identical volumes, dividing the space

Algorithm 1 Algorithm for Symmetric Region Packing (SRP) bound

```
 1: $\widetilde{\mathcal{E}}_k = \bigcup_{E \subset \mathcal{I}: 1 \leq |E| \leq k} \bigcup_{F \in \mathcal{P}(V(E)^C)} [E \cup F]$
 2: a = [0 0 0]
 3: $|A_k| = 1$
 4: $B(a) = \cup_{E \in \widetilde{\mathcal{E}}_k} f(a, E)$
 5: for all b ∈ $B(a)$ do
 6:     $\overline{B}(b) = (\cup_{E \in \widetilde{\mathcal{E}}_k} f(b, E)) \setminus (a \cup B(a))$
 7:     $\overline{H}_k(b)$.addNodes($\overline{B}(b)$)
 8:     for all $\overline{b_1}, \overline{b_2} \in \overline{B}(b)$ do
 9:         if $D(\overline{b_1}, \overline{b_2}) \in \widetilde{\mathcal{E}}_k$ then
10:             $\overline{H}_k(b)$.addEdge($\overline{b_1}, \overline{b_2}$)
11:     $M_b$ = |cliquePartition($\overline{H}_k(b)$)|
12:     $|A_k| = |A_k| + 1/(1 + M_b)$
13: $\beta_{SRP} = (q^I)/|A_k|$
```

of all assignments by this volume yields a bound. Figure 4.4 shows β_{SRP} computed for 1-optima for Example 1. We choose an arbitrary assignment $a \in A$ which we assume to be k-optimal (a = [0 0 0] in Figure 4.4), around which we will construct a region claimed by a.

Applying the exclusivity relations from $\widetilde{\mathcal{E}}_k$, we generate a set $B(a) = \cup_{E \in \widetilde{\mathcal{E}}_k} f(a, E)$ where $f(a, E)$ yields the assignment that is excluded from being k-optimal by a and E. The first two rows of Figure 4.4 show $\widetilde{\mathcal{E}}_1$ and the set B([0 0 0]). Applying the exclusivity relations again for each $b \in B(a)$, and discarding assignments already included in a or $B(a)$, we generate a set $\overline{B}(b) = \cup_{E \in \widetilde{\mathcal{E}}_k} f(b, E)$ which contains all assignments that potentially exclude b from being k-optimal. In Figure 4.4, we apply $\widetilde{\mathcal{E}}_1$ to find $\overline{B}(b)$ for all $b \in B(a) = \{[1\ 0\ 0], [0\ 1\ 0], [0\ 0\ 1], [1\ 0\ 1]\}$ where the grayed out assignments are those discarded for being in $\{a\} \cup B(a)$. To ensure that the region that a claims is disjoint from the regions claimed by other k-optima, a should only claim a fraction of each $b \in B(a)$. This can be achieved if a shares each b equally with all other k-optima that might exclude b. These additional k-optima are contained within $\overline{B}(b)$. However, not all $\overline{b} \in \overline{B}(b)$ can actually be k-optimal as they might exclude each other. If we construct a graph $\overline{H}_k(b)$ with nodes for all $\overline{b} \in \overline{B}(b)$ and edges formed using $\widetilde{\mathcal{E}}_k$, and we find M_b, the size of the MIS, then a can safely claim $1/(1 + M_b)$ of b. We again use clique partitioning to safely estimate



$\widetilde{\mathcal{E}}_1 = \{$	$\{1\},$	$\{2\},$	$\{3\},$	$\{1,3\}\quad\}$
$B([0\,0\,0]) = \{$	$[1\,0\,0],$	$[0\,1\,0],$	$[0\,0\,1],$	$[1\,0\,1]\quad\}$
$\widetilde{\mathcal{E}}_1 = \{$	$\overline{B}([1\,0\,0])$	$\overline{B}([0\,1\,0])$	$\overline{B}([0\,0\,1])$	$\overline{B}([1\,0\,1])$
$\{1\},$	[0 0 0]	[1 1 0]	[1 0 1]	[0 0 1]
$\{2\},$	[1 1 0]	[0 0 0]	[0 1 1]	[1 1 1]
$\{3\},$	[1 0 1]	[0 1 1]	[0 0 0]	[1 0 0]
$\{1,3\}\}$	[0 0 1]	[1 1 1]	[1 0 0]	[1 1 1]
$\overline{H}_1(b)$ (exclusivity subgraph)	110	110 {1,3} {3} 011 111 {1}	011	111
$M_b =$	1	1	1	1
$1/(1+M_b) =$	1/2	1/2	1/2	1/2

Figure 4.4: Computation of β_{SRP} for Example 1

M_b. In Figure 4.4, for $b = [0\ 1\ 0]$, $\overline{B}([0\ 1\ 0])$ leads to a three-node, three-edge exclusivity graph $\overline{H}_k([0\ 1\ 0])$. By adding the values of $1/(1+M_b)$ for all $b \in B(a)$ (plus one for itself), we obtain that a can safely claim a region of size 3, which implies $\beta_{SRP} = \lfloor 2^3/3 \rfloor = 2$. Algorithm 1's runtime is polynomial in the number of possible assignments, which is a comparatively small cost for a bound that applies to every possible instantiation of rewards to actions. An exhaustive search for the MIS of H_k would be exponential in this number (doubly exponential in the number of agents).

4.5 Experimental Results

We performed five evaluations in addition to the experiment described at the beginning of the section. The first evaluates the impact of k-optimality for higher values of k. For each of the three DCOP graphs from Figure 2.2(a-c), Figure 4.5(a-c) shows key properties for 1-, 2- and 3-optima.

	$\lvert\tilde{A}\rvert$	avg. reward
1-opt.	10	.850
2-opt.	55	.964
3-opt.	175	.993

(a)

	$\lvert\tilde{A}\rvert$	avg. reward
1-opt.	10	.809
2-opt.	55	.961
3-opt.	175	.986

(b)

	$\lvert\tilde{A}\rvert$	avg. reward
1-opt.	9	.832
2-opt.	45	.977
3-opt.	129	.982

(c)

Figure 4.5: 1-optima vs. assignment sets chosen using other metrics

The first column of each table shows $|\tilde{A}|$, the size of the neighborhood containing all assignments within a distance of k from a k-optimal assignment a, and hence of lower reward than a. For example, in the joint patrol domain described at the beginning of the section, Figure 4.5(a) shows that, if agents are arranged as in the DCOP graph from Figure 2.2 (a), any 1-optimal joint patrol must have a higher reward than at least 10 other joint patrols. We see that as k increases, the k-optimal set contains assignments that each individually dominate a larger and larger neighborhood. The second column shows, for each of the three graphs, the average reward of each k-optimal assignment set found over 20 problem instances, generated by assigning rewards to the links from a uniform random distribution. We define the reward of a k-optimal assignment set as the mean reward of all k-optimal assignments that exist for a particular problem instance; each figure in the second column is therefore a mean of means. As k was increased, leading to a larger neighborhood of dominated assignments, the average reward of the k-optimal assignment sets show a significant increase (T-tests showed the increase in average reward as k increased was significant within 5%.)

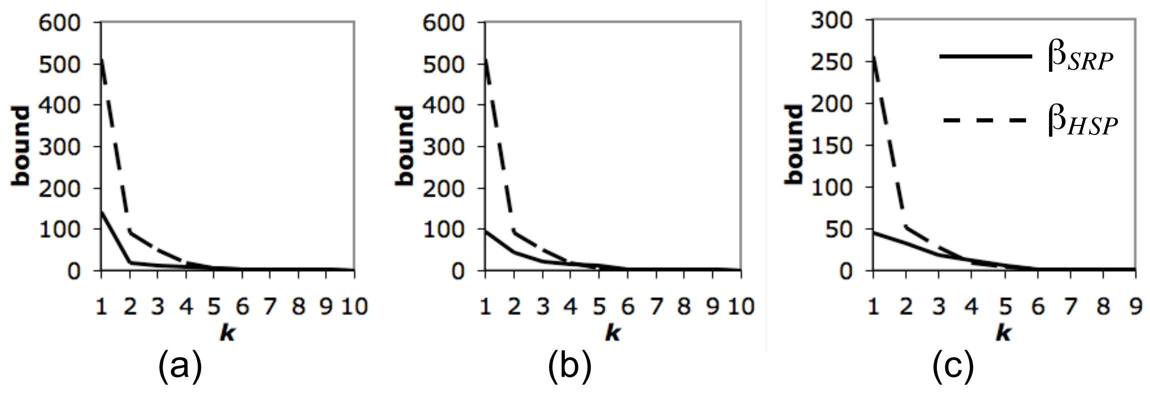


Figure 4.6: β_{SRP} vs. β_{HSP} for DCOP graphs from Figure 2.2

However as k increases, the number of possible k-optimal assignments decreases, and hence the next four evaluations explore the effectiveness of the different bounds on the number of k-optima. For the three DCOP graphs shown in Figure 2.2, Figure 4.6 provides a concrete demonstration of the gains in resource allocation due to the tighter bounds made possible with graph-based analysis. The x axis in Figure 4.6 shows k, and the y axis shows the β_{HSP} and β_{SRP} bounds on the number of k-optima that can exist. To understand the implications of these results on resource allocation, consider a patrolling problem where the constraints between agents are shown in the 10-agent DCOP graph from Figure 2.2(a), and all agents consume one unit of fuel for each assignment taken. Suppose that $k = 2$ has been chosen, and so at runtime, the agents will use MGM-2 [Maheswaran et al., 2004a], repeatedly, to find and execute a set of 2-optimal assignments. We must allocate enough fuel to the agents *a priori* so they can execute up to all possible 2-optimal assignments. Figure 4.6(a) shows that if β_{HSP} is used, the agents would be loaded with 93 units of fuel to ensure enough for all 2-optimal assignments. However, β_{SRP} reveals that only 18 units of fuel are sufficient, a five-fold savings. (For clarity we note that on all three graphs, both bounds are 1 when $k = n$ and 2 when $n - 3 \leq k < n$.)

To systematically investigate the impact of graph structure on bounds, we generated a large number of DCOP graphs of varying size and density. We started with complete binary graphs (all pairs of agents are connected) where each node (agent) had a unique ID. To gradually make each graph sparser, edges were repeatedly removed according to the following two-step process: (1) Find the lowest-ID node that has more than one incident edge. (2) If such a node exists, find the lowest-ID node that shares an edge with it, and remove this edge. Figure 4.7 shows the β_{HSP} and β_{SRP} bounds for k-optima for $k \in \{1, 2, 3, 4\}$ and $n \in \{7, 8, 9, 10\}$. For each of the 16 plots shown, the y axis shows the bounds and the x-axis shows the number of links removed from the graph according to the above method. While $\beta_{HSP} < \beta_{SRP}$ for very dense graphs, β_{SRP} provides significant gains for the vast majority of cases. For example, for the graph with 10 agents, and 24 links removed, and a fixed $k = 1$, β_{HSP} implies that we must equip the agents with 512 resources to ensure that all resources are not exhausted before all 1-optimal actions are executed. However, β_{SRP} indicates that a 15-fold reduction to 34 resources will suffice, yielding a savings of 478 due to the use of graph structure when computing bounds.

A fourth experiment compared β_{HSP} and β_{SRP} to the bound obtained by applying F_{CLIQUE}, $\beta_{FCLIQUE}$ to DCOP graphs from the previous experiment. Selected results are shown in Figure 4.8 for graphs of 8 and 9 agents. While $\beta_{FCLIQUE}$ is marginally better for $k = 1$, β_{SRP} has clear gains for $k = 4$. Identifying the relative effectiveness of various algorithms that exploit our exclusivity relation sets is clearly an area for future work.

Finally, Figure 4.9 compares the constant-time-computable graph-independent bounds from Section 4.1, in particular, showing the improvement of β_{MH} over $\min\{\beta_H, \beta_S, \beta_P\}$ for selected odd values of k, given three possible actions for each agent ($q = 3$). The x-axis shows n, the number of agents and the y-axis shows $100 \cdot (\min\{\beta_H, \beta_S, \beta_P\} - \beta_{MH}) / \min\{\beta_H, \beta_S, \beta_P\}$. For odd values

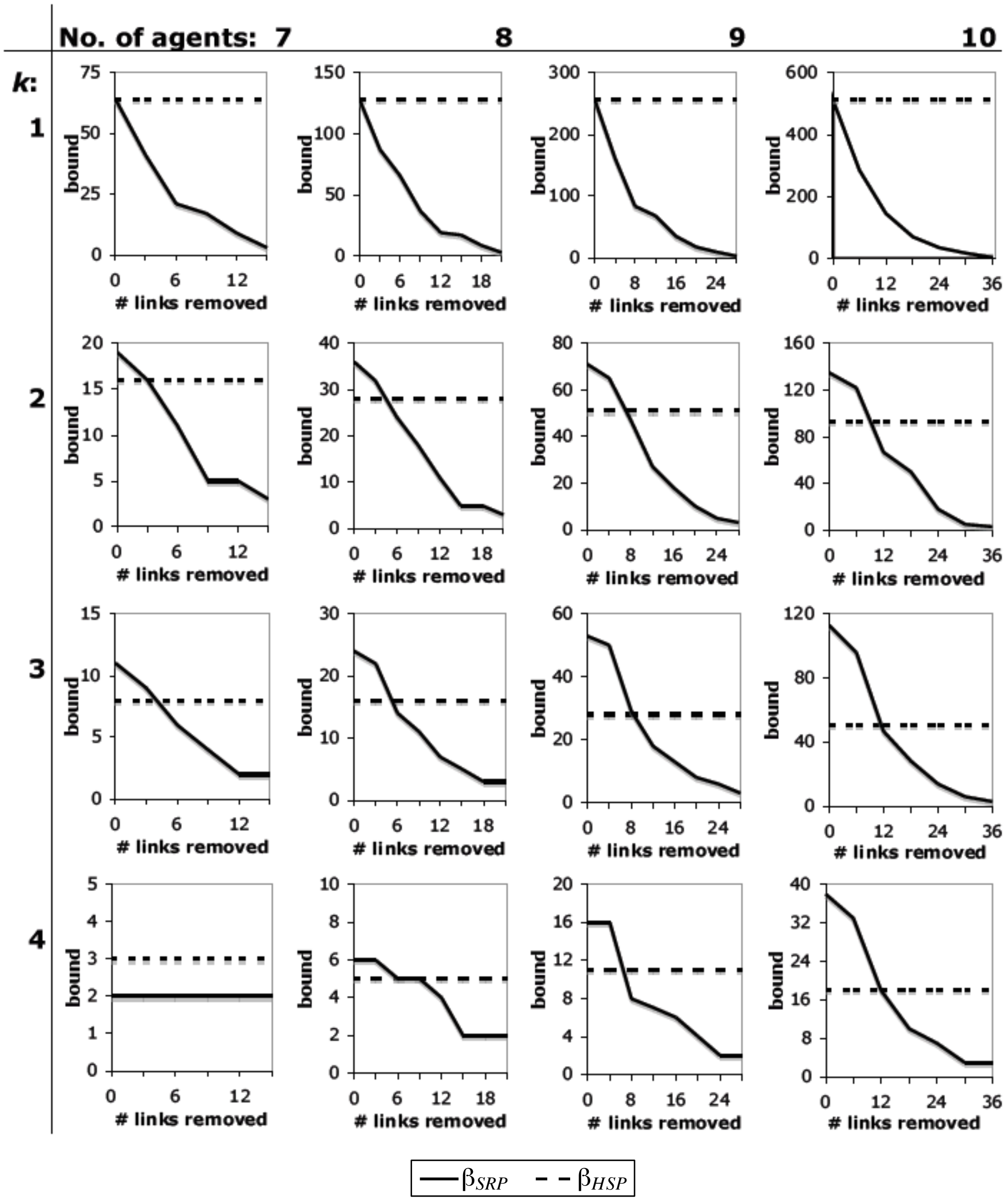


Figure 4.7: Comparisons of β_{SRP} vs. β_{HSP}

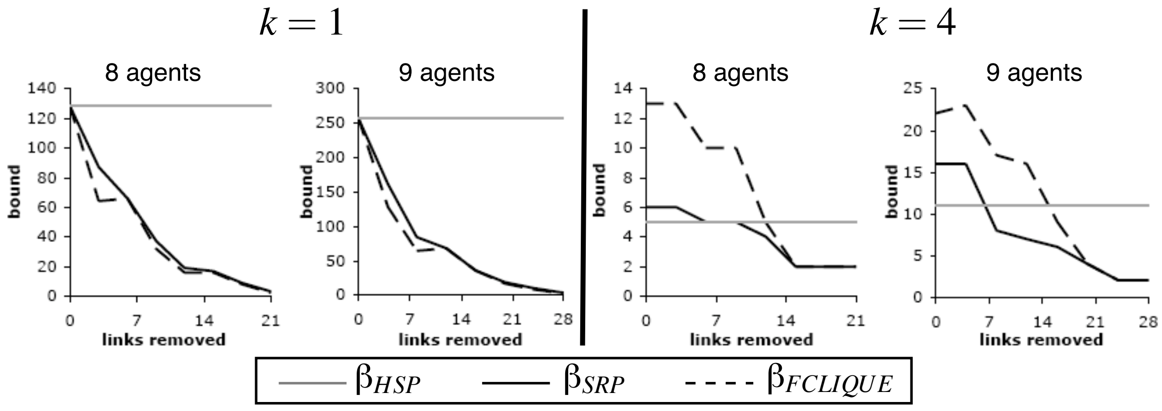


Figure 4.8: Comparisons of $\beta_{SRP}, \beta_{HSP}, \beta_{FCLIQUE}$

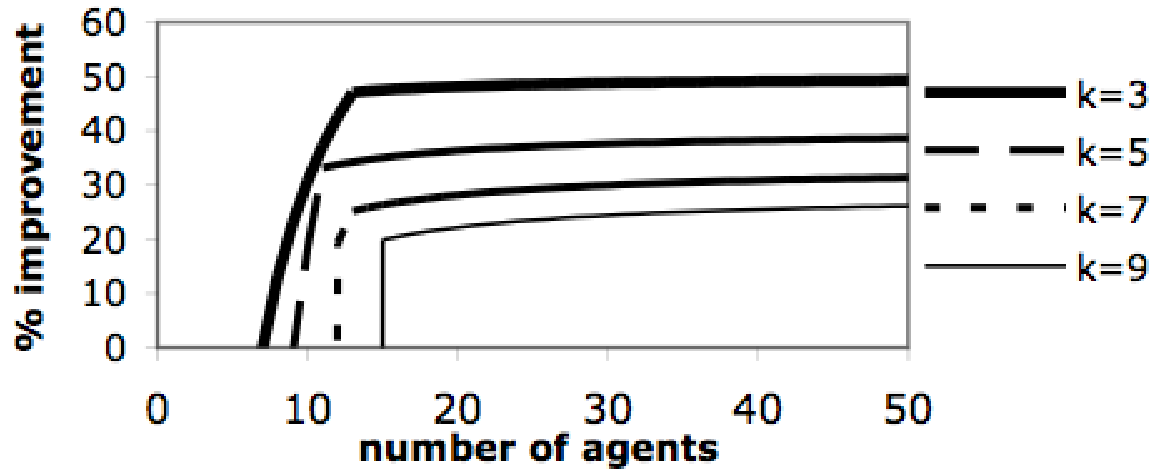


Figure 4.9: Improvement of β_{MH} on $\min\{\beta_H, \beta_S, \beta_P\}$

of $k > 1$, as n increased, β_{MH} provided a tighter bound on the number of k-optima. The most improvement was for $k = 3$; as n increased, β_{MH} gave a bound 50% tighter than the others.

Chapter 5

Algorithms

This chapter contains a description of existing 1-optimal algorithms, new 2- and 3-optimal algorithms, as well as a theoretical analysis of key properties of these algorithms and experimental comparisons. We discuss the existing 1-optimal algorithms here, rather than in the discussion of related work in Chapter 8 because our new 2- and 3-optimal algorithms build on the frameworks of these basic algorithms.

5.1 1-Optimal Algorithms

We begin with two algorithms that only consider unilateral actions by agents in a given context. The first is the MGM (Maximum Gain Message) Algorithm which is a modification of DBA (Distributed Breakout Algorithm) [Yokoo and Hirayama, 1996] focused solely on gain message passing. MGM is not a novel algorithm, but simply a name chosen to describe DBA without the changes on constraint costs that DBA uses to break out of local minima. We note that DBA itself cannot be applied in an optimization context, as it would require global knowledge of solution quality (it can be applied in a satisfaction context because any agent encountering a violated constraint would know that the current solution is not a satisfying solution). The second is DSA

(Distributed Stochastic Algorithm) [Fitzpatrick and Meertens, 2003], which is a homogeneous stationary randomized algorithm. Our analysis will focus on synchronous applications of these algorithms.

Let us define a *round* as the duration to execute one run of a particular algorithm. This run could involve multiple broadcasts of *messages*. Every time a messaging phase occurs in a round, we will count that as one *cycle* and cycles will be our performance metric for speed, as is common in DCOP literature. Let $x^{(n)} \in X$ denote the assignments at the beginning of the n-th round. We assume that every algorithm will broadcast its current value to all its neighbors at the beginning of the round taking up one cycle. Once agents are aware of their current contexts, they will go through a process as determined by the specific algorithm to decide which of them will be able to modify their value. Let $M^{(n)} \subseteq \mathcal{N}$ denote the set of agents allowed to modify the values in the n-th round. For MGM, each agent broadcasts a gain message to all its neighbors that represents the maximum change in its local utility if it is allowed to act under the current context. An agent is then allowed to act if its gain message is larger than all the gain messages it receives from all its neighbors (ties can be broken through variable ordering or another method) [Yokoo and Hirayama, 1996]. MGM is outlined in Algorithm 7. For DSA, each agent generates a random number from a uniform distribution on $[0, 1]$ and acts if that number is less than some threshold p [Fitzpatrick and Meertens, 2003]. We note that MGM has a cost of two cycles per round while DSA only has a cost of one cycle per round. Pseudocode for DSA and MGM is given in Algorithms 2 and 3 respectively.

We are able to prove the following monotonicity property of MGM.

Algorithm 2 DSA (myNeighbors, myValue)

```
1: SendValueMessage(myNeighbors, myValue)
2: currentContext = GetValueMessages(myNeighbors)
3: [gain,newValue] = BestUnilateralGain(currentContext)
4: if Random(0,1) < Threshold then
5:     myValue = newValue
```

Algorithm 3 MGM (myNeighbors, myValue)

```
1: SendValueMessage(myNeighbors, myValue)
2: currentContext = GetValueMessages(myNeighbors)
3: [gain,newValue] = BestUnilateralGain(currentContext)
4: SendGainMessage(myNeighbors,gain)
5: neighborGains = ReceiveGainMessages(myNeighbors)
6: if gain > max(neighborGains) then
7:     myValue = newValue
```

Proposition 10 *When applying MGM, the global utility $\overline{U}(x^{(n)})$ is strictly increasing with respect to the round (n) until $x^{(n)} \in X_E$.*

Proof. We assume $M^{(n)} \neq \emptyset$, otherwise we would be at a 1-optimum. When utilizing MGM, if $i \in M^{(n)}$ and $E_{ij} = 1$, then $j \notin M^{(n)}$. If the i-th variable is allowed to modify its value in a particular round, then its gain is higher than all its neighbors gains. Consequently, all its neighbors would have received a gain message higher than their own and thus, would not modify their values in that round. Because there exists at least one neighbor for every variable, the set of agents who cannot modify their values is not empty.We have $x_i^{(n+1)} \neq x_i^{(n)}\ \forall i \in M^{(n)}$ and $x_i^{(n+1)} = x_i^{(n)}\ \forall i \notin M^{(n)}$.

Also, $u_i(x_i^{(n+1)}; x_{-i}^{(n)}) > u_i(x_i^{(n)}; x_{-i}^{(n)})\ \forall i \in M^{(n)}$, otherwise the i-th player's gain message would have been zero. Looking at the global utility, we have

$$
\begin{aligned}
&\overline{U}\left(x^{(n+1)}\right) \\
&= \sum_{i,j:E_{ij}=1} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) \\
&= \sum_{\substack{i,j:i\in M^{(n)},\\ j\in M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\in M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) \\
&\quad + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\in M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{\substack{i,j:i\in M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n)}\right) + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\in M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n)}\right) \\
&= \sum_{i\in M^{(n)}} u_i\left(x_i^{(n+1)}; x_{-i}^{(n)}\right) + \sum_{j\in M^{(n)}} u_j\left(x_j^{(n+1)}; x_{-j}^{(n)}\right) + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n)}\right) \\
&> \sum_{i\in M^{(n)}} u_i\left(x_i^{(n)}; x_{-i}^{(n)}\right) + \sum_{j\in M^{(n)}} u_j\left(x_j^{(n)}; x_{-j}^{(n)}\right) + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n)}\right) \\
&= \sum_{\substack{i,j:i\in M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n)}\right) + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\in M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n)}\right) + \sum_{\substack{i,j:i\notin M^{(n)},\\ j\notin M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n)}\right) \\
&= \overline{U}\left(x^{(n)}\right).
\end{aligned}
$$

The second equality is due to a partition of the summation indexes. The third equality utilizes the properties that there are no neighbors in $M^{(n)}$ and that the values for variables corresponding to indexes not in $M^{(n)}$ in the $(n+1)$-th round are identical to the values in the n-th round. The strict inequality occurs because agents in $M^{(n)}$ must be making local utility gains. The remaining equalities are true by definition. Thus, MGM yields monotonically increasing global utility until equilibrium. ■

Why is monotonicity important? In anytime domains where communication may be halted arbitrarily and existing strategies must be executed, randomized algorithms risk being terminated at highly undesirable assignments. Given a starting condition with a minimum acceptable global utility, monotonic algorithms guarantee lower bounds on performance in anytime environments. Consider the following example.

Example 4 The Traffic Light Game. *Consider two variables, both of which can take on the values red or green, with a constraint that takes on utilities as follows:*

- $U(red, red) = 0$.
- $U(red, green) = U(green, red) = 1$.
- $U(green, green) = -1000$.

Turning this DCOP into a game would require the agent for each variable to take the utility of the single constraint as its local utility. If (red, red) *is the initial condition, each agent would choose to alter its value to green if given the opportunity to move. If both agents are allowed to alter their value in the same round, we would end up in the adverse state* $(green, green)$*. When using DSA, there is always a positive probability for any time horizon that* $(green, green)$ *will be the resulting assignment.*

In domains such as independent path planning of trajectories for UAVs or rovers, in environments where communication channels are unstable, bad assignments could lead to crashes

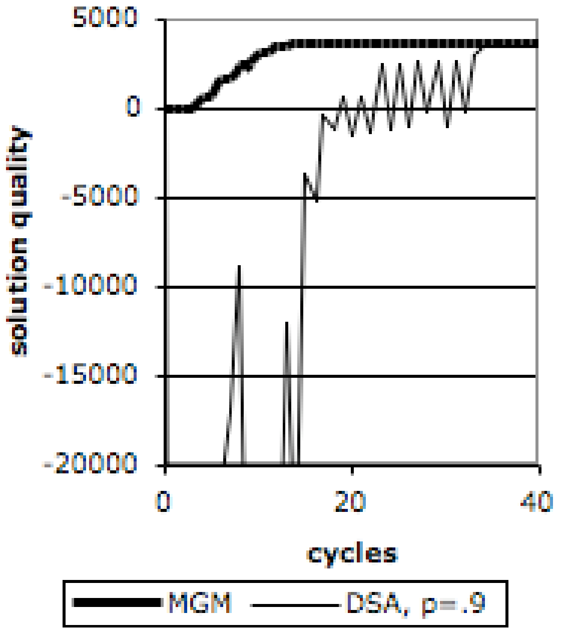


Figure 5.1: Sample Trajectories of MGM and DSA for a High-Stakes Scenario

whose costs preclude the use of methods without guarantees. This is illustrated in Figure 5.1 which displays sample trajectories for MGM and DSA with identical starting conditions for a high-stakes scenario described in Section 5.4. The performance of both MGM and DSA with respect to various graph coloring problems are investigated and discussed in Section 5.4.

5.2 2-Optimal Algorithms

When applying 1-optimal algorithms, the evolution of the assignments will terminate at a 1-optimum within the set X_E described earlier. One method to improve the solution quality is for agents to coordinate actions with their neighbors. This allows the evolution to follow a richer space of trajectories and alters the set of terminal assignments. In this section we introduce two 2-optimal algorithms, where agents can coordinate actions with one other agent. Let us refer to the set of terminal states of the class of 2-optimal algorithms as X_{2E}, i.e. neither a unilateral nor a bilateral modification of values will increase sum of all constraint utilities connected to the acting agent(s) if $x \in X_{2E}$.

We now introduce two algorithms that allow for coordination while maintaining the underlying distributed decision making process and the same constraint graph: MGM-2 (Maximum Gain Message-2) and SCA-2 (Stochastic Coordination Algorithm-2).

Both MGM-2 and SCA-2 begin a round with agents broadcasting their current values. The first step in both algorithms is to decide which subset of agents are allowed to make *offers*. We resolve this by randomization, as each agent generates a random number uniformly from $[0, 1]$ and considers themselves to be an *offerer* if the random number is below a threshold q. If an agent is an offerer, it cannot accept offers from other agents. All agents who are not offerers are considered to be *receivers*. Each offerer will choose a neighbor at random (uniformly) and send it an offer message which consists of all coordinated moves between the offerer and receiver that will yield a gain in local utility to the offerer under the current context. The offer message will contain both the suggested values for each player and the offerer's local utility gain for each value pair. Each receiver will then calculate the global utility gain for each value pair in the offer message by adding the offerer's local utility gain to its own utility change under the new context and (very importantly) subtracting the difference in the link between the two so it is not counted twice. If the maximum global gain over all offered value pairs is positive, the receiver will send an *accept* message to the offerer with the appropriate value pair and both the offerer and receiver are considered to be committed. Otherwise, it sends a *reject* message to the offerer, and neither agent is committed.

At this point, the algorithms diverge. For SCA-2, any agent who is not committed and can make a local utility gain with a unilateral move generates a random number uniformly from

$[0, 1]$ and considers themselves to be *active* if the number is under a threshold p. At the end of the round, all committed agents change their values to the committed offer and all active agents change their values according to their unilateral best response. Thus, SCA-2 requires three cycles (value, offer, accept/reject) per round. In MGM-2 (after the offers and replies are settled), each agent sends a gain message to all its neighbors. Uncommitted agents send their best local utility gain for a unilateral move. Committed agents send the global gain for their coordinated move. Uncommitted agents follow the same procedure as in MGM, where they modify their value if their gain message was larger than all the gain messages they received. Committed agents send their partners a *confirm* message if all the gain messages they received were less than the calculated global gain for the coordinated move and send a *deconfirm* message, otherwise. A committed agent will only modify its value if it receives a go message from its partner. MGM-2 is outlined in Algorithm 4. We note that MGM-2 requires five cycles (value, offer, accept/reject, gain, confirm/deconfirm) per round, and has less concurrency than SCA-2 (since no two neighboring groups will ever move together). Given the excess cost of MGM-2, why would one choose to apply it? We can show that MGM-2 is monotonic in global utility.

Proposition 11 *When applying MGM-2, the global utility $\overline{U}(x^{(n)})$ is strictly increasing with respect to the round (n) until $x^{(n)} \in X_{2E}$.*

Proof. We begin by introducing some notation. At the end of the n-th round, let $C^{(n)} \subset \mathcal{N}$ denote the set of agents who are committed, $M^{(n)} \subset \mathcal{N}$ denote the set of uncommitted agents who are active, and $S^{(n)} \equiv \{C^{(n)} \cup M^{(n)}\}^C \subset \mathcal{N}$ denote the uncommitted agents who are inactive. Let

$p(i) \in C^{(n)}$ denote the partner of a committed agent $i \in C^{(n)}$. The global utility can then be expressed as:

$$
\begin{aligned}
&\overline{U}\left(x^{(n+1)}\right) \\
&= \sum_{i,j:E_{ij}=1} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) \\
&= \sum_{\substack{i,j:i\in C^{(n)},\\ j\in C^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\in C^{(n)},\\ j\in S^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\in S^{(n)},\\ j\in C^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) \\
&\quad + \sum_{\substack{i,j:i\in S^{(n)},\\ j\in S^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\in M^{(n)},\\ j\in S^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\in S^{(n)},\\ j\in M^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) \\
&= \sum_{i\in C^{(n)}} U_{ip(i)}\left(x_i^{(n+1)}, x_{p(i)}^{(n+1)}\right) + \sum_{i\in C^{(n)}} \sum_{j\in\mathcal{N}_i\setminus\{p(i)\}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{j\in C^{(n)}} \sum_{i\in\mathcal{N}_j\setminus\{p(j)\}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \\
&\quad \left(+ \sum_{j\in C^{(n)}} U_{jp(j)}\left(x_{p(j)}^{(n+1)}, x_j^{(n+1)}\right) - \sum_{j\in C^{(n)}} U_{jp(j)}\left(x_{p(j)}^{(n+1)}, x_j^{(n+1)}\right)\right) + \sum_{\substack{i,j:i\in S^{(n)},\\ j\in S^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \\
&\quad + \sum_{i\in M^{(n)}} u_i\left(x_i^{(n+1)}, x_j^{(n+1)}\right) + \sum_{j\in M^{(n)}} u_j\left(x_i^{(n+1)}, x_j^{(n+1)}\right) \\
&= \sum_{i\in C^{(n)}} U_{ip(i)}\left(x_i^{(n+1)}, x_{p(i)}^{(n+1)}\right) + \sum_{i\in C^{(n)}} \sum_{j\in\mathcal{N}_i\setminus\{p(i)\}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n)}\right) + \sum_{j\in C^{(n)}} \sum_{i\in\mathcal{N}_j\setminus\{p(j)\}} U_{ij}\left(x_i^{(n+1)}, x_j^{(n)}\right) + \\
&\quad + \sum_{j\in C^{(n)}} U_{jp(j)}\left(x_{p(j)}^{(n+1)}, x_j^{(n+1)}\right) - \sum_{j\in C^{(n)}} U_{jp(j)}\left(x_{p(j)}^{(n+1)}, x_j^{(n+1)}\right) + \sum_{\substack{i,j:i\in S^{(n)},\\ j\in S^{(n)},E_{ij}=1}} U_{ij}\left(x_i^{(n)}, x_j^{(n)}\right) + \\
&\quad + \sum_{i\in M^{(n)}} u_i\left(x_i^{(n+1)}, x_j^{(n)}\right) + \sum_{j\in M^{(n)}} u_j\left(x_i^{(n)}, x_j^{(n+1)}\right)
\end{aligned}
$$

$$= \sum_{i \in C^{(n)}} u_i(x_i^{(n+1)}; \mu_{-i}(x_{p(i)}^{(n+1)}, x_{-ip(i)}^{(n)})) + \sum_{j \in C^{(n)}} u_j(x_j^{(n+1)}; \mu_{-j}(x_{p(j)}^{(n+1)}, x_{-jp(j)}^{(n)})) - \sum_{j \in C^{(n)}} U_{jp(j)} \left(x_{p(j)}^{(n+1)}, x_j^{(n+1)} \right)$$

$$+ \sum_{\substack{i,j: i \in S^{(n)}, \\ j \in S^{(n)}, E_{ij}=1}} U_{ij} \left(x_i^{(n)}, x_j^{(n)} \right) + \sum_{i \in M^{(n)}} u_i \left(x_i^{(n+1)}, x_j^{(n)} \right) + \sum_{j \in M^{(n)}} u_j \left(x_i^{(n)}, x_j^{(n+1)} \right)$$

$$> \sum_{i \in C^{(n)}} u_i(x_i^{(n)}; \mu_{-i}(x_{p(i)}^{(n)}, x_{-ip(i)}^{(n)})) + \sum_{j \in C^{(n)}} u_j(x_j^{(n)}; \mu_{-j}(x_{p(j)}^{(n)}, x_{-jp(j)}^{(n)})) - \sum_{j \in C^{(n)}} U_{jp(j)} \left(x_{p(j)}^{(n)}, x_j^{(n)} \right)$$

$$+ \sum_{\substack{i,j: i \in S^{(n)}, \\ j \in S^{(n)}, E_{ij}=1}} U_{ij} \left(x_i^{(n)}, x_j^{(n)} \right) + \sum_{i \in M^{(n)}} u_i \left(x_i^{(n)}, x_j^{(n)} \right) + \sum_{j \in M^{(n)}} u_j \left(x_i^{(n)}, x_j^{(n)} \right) = \overline{U} \left(x^{(n)} \right).$$

The first equality is by definition. The second equality partitions the indexes into update class, eliminating cross indexes of $M^{(n)}$ with anything other than $S^{(n)}$. In the third equality, we simplify the summations involving committed agents using expressions for partners and neighbors, we insert a zero value term in parenthesis, and transform the summations involving active agents into local utilities. In the fourth equality, we modify the round index for those agents who are inactive. In the fifth equality, we transform the summations involving committed agents into local utilities. The inequality is due to the fact that the global utility on the links of the committed partners and the local utility of the active agents must increase due to the positive gain messages. The key is that by setting $j = p(i)$ in the second and third summations, we recover the gain message of the committed teams. Note the subtraction of the utility gain on the link between partners to avoid double counting. The final equality can be achieved by reversing the transformation to yield the global utility at the previous round. Thus, MGM-2 yields monotonically increasing global utility until equilibrium is reached. ■

Example 5 Meeting Scheduling. *Consider two agents trying to schedule a meeting at either 7:00 AM or 1:00 PM with the constraint utility as follows:* $U(7,7) = 1, U(7,1) = U(1,7) =$

Algorithm 4 MGM2 (myNeighbors, myConstraints, myValue)

```
Initialization step: (Send out value messages)
 1: for all neighbor ∈ myNeighbors do
 2:     sendMsg(neighbor, <VALUE, myValue>)
Cycle 1: (Some agents become offerers. Offerers send out offers)
 1: offerer? = FALSE; confirmed? = FALSE
 2: for all neighbor ∈ myNeighbors do
 3:     myContext.add(receiveMsg(neighbor, <VALUE, neighborValue>))
 4: if  Random(0,1) < offererThreshold then
 5:     offerer? = TRUE;
 6:     partner = pickRandom(myNeighbors)
 7:     sendMsg(partner, <OFFER, allCoordinatedMoves>)
 8: [ myIndividualMove, myIndividualGain ] = computeBestMove(myConstraints, myContext)
Cycle 2: (Agents respond to offerers and join their groups)
 1: for all neighbor ∈ myNeighbors do
 2:     myOffers.add(receiveMsg(neighbor, <OFFER, allCoordinatedMoves>))
 3: if NOT empty(myOffers) then
 4:     [partner, ourMove, ourGain ] = computeBestMove(myConstraints, myContext, myOffers)
 5:     if offerer? == FALSE then
 6:         committed? = TRUE
 7:         sendMsg(partner, <ACCEPT, TRUE, ourMove.partnersMove, ourGain>)
 8: for all neighbor ∈ myOffers \ partner do
 9:     sendMsg(neighbor, <ACCEPT, FALSE, null, null>)
Cycle 3: (Group members send gain message (for group move) to neighbors outside the group)
 1: receiveMsg(partner, <ACCEPT, partnerCommitted?, ourMove.myMove, ourGain>)
 2: if offerer? then
 3:     if partnerCommitted? == TRUE then
 4:         committed? = TRUE
 5: if committed? then
 6:     myGain = ourGain
 7: else
 8:     myGain = myIndividualGain
 9: for all neighbor in myNeighbors \ partner do
10:     sendMsg(neighbor, <GAIN, myGain>)
Cycle 4: (Based on neighbors' gains, agents confirm or deconfirm with partner)
 1: for all neighbor ∈ myNeighbors do
 2:     neighborsGains.add(receiveMsg(neighbor, <GAIN, neighborsGain>))
 3: if myGain > max(neighborsGains) then
 4:     confirmed? = TRUE // ties broken using node ordering
 5: if committed? then
 6:     sendMsg(partner, <CONFIRM, confirmed?>)
Cycle 5: (Confirmed agents move and send out value messages)
 1: receiveMsg(partner, <CONFIRM, partnerConfirmed?>)
 2: if committed then
 3:     if confirmed? then
 4:         if partnerConfirmed? then
 5:             myValue = ourMove.myValue
 6:         else
 7:             myValue = myIndividualMove
 8: else if confirmed? then
 9:     myValue = myIndividualMove
10: for all neighbor ∈ myNeighbors do
11:     sendMsg(neighbor, <VALUE, myValue>)
```

$-100, U(1,1) = 10$. *If the agents started at* $(7,7)$*, any 1-coordinated algorithm would not be able to reach the global optimum, while 2-coordinated algorithms would.*

It is not true that 2-optimal algorithm will yield a solution with higher quality than a 1-optimal algorithm in all situations. In fact, there are DCOPs and initial conditions for which a given 1-optimal algorithm will yield a better solution than a given 2-optimal algorithm. The complexity lies in that we cannot predict exactly what trajectory the evolution will follow. Given certain initial conditions (beginning the algorithm at a 1-optimum), 2-optimal algorithms will always perform at least as well as 1-optimal algorithms (or better) as outlined in the following corollary. For other initial conditions, the experiments shown in Section 5.4 show that for the domains investigated, 2-optimal algorithms did outperform 1-optimal algorithms on average with respect to solution quality.

Corollary 1 *For every initial DCOP assignment* $x_0 \in X_E \setminus X_{2E}$*, MGM-2 will yield a better solution than either MGM or DSA.*

Proof. Since $x_0 \in X_E$, neither MGM nor DSA will move and the solution quality will be that obtained at the assignment x_0. However, since $x_0 \notin X_{2E}$, MGM-2 will continue to evolve from x_0 until it reaches an assignment in X_{2E}. Because $MGM - 2$ is monotonic in global utility, whatever solution in reaches in X_{2E} will have a higher global utility than x_0. ■

Thus, MGM-2 dominates DSA and MGM for initial conditions in $X_E \setminus X_{2E}$ and is identical to DSA and MGM on X_{2E} (as neither algorithm will evolve from there). The unknown is the behavior on $X \setminus X_E$. It is difficult to analyze this space because one cannot pinpoint the trajectories due to the probabilistic nature of their evolution. If we assume that iterations beginning in $X \setminus X_E$ are taken to points in X_E in a relatively uniform manner on average with all algorithms, then we might surmise that the dominance of MGM-2 should yield a better solution quality. The performance of both MGM-2 and SCA-2 with respect to a various graph coloring problems are investigated and discussed in Section 5.4.

5.3 3-Optimal Algorithms

Analogous algorithms for 3-optimality are detailed in this section. MGM-3 is presented in detail; SCA-3 can be implemented through small adjustments to MGM-3. The main complication with moving to 3-optimality is the following: With 2-optimal algorithms, the offerer could simply send all information the receiver needed to compute the optimal joint move in the offer message itself. With groups of three agents, this is no longer possible, and thus two more message cycles are needed.

Just as in 1- and 2-optimal versions, we begin with all agents broadcasting value messages to their neighbors in an initialization step. Then, in the first cycle, as in MGM-2 and SCA-2, each agent generates a random number uniformly from $[0, 1]$ and considers themselves to be an *offerer* if the random number is below a threshold q and a receiver otherwise. Now, instead of a single neighbor, each offerer will choose two neighbors at random (uniformly) and send offer messages to both. These offer messages will not contain any suggested move; rather, they are simple invitations to join the offerer's group. In the second cycle, upon receiving a set of offer messages from its neighbors, all receivers will accept a randomly chosen offer from this set and decline the others. The receiver will return an accept message to the chosen offerer that includes all of the constraints on the receiver as well as the context it faces (the current values of its own neighbors). Offerers will decline all offers from the set. In the third cycle, the offerer will receive

Algorithm 5 MGM3 (myNeighbors, myConstraints, myValue)

```
 1: for all neighbor ∈ myNeighbors do
 2:     sendMsg(neighbor, <VALUE, myValue>)
```

Cycle 1: (Some agents become offerers. Offerers send out offers)

```
 1: offerer? = FALSE; committed? = FALSE; confirmed? = FALSE
 2: for all neighbor ∈ myNeighbors do
 3:     myContext.add(receiveMsg(neighbor, <VALUE, neighborValue>))
 4: if  Random(0,1) < offererThreshold then
 5:     offerer? = TRUE; committed? = TRUE
 6:     partner1 = pickRandom(myNeighbors); partner2 = pickRandom(myNeighbors \ partner1)
 7:     sendMsg(partner1, <OFFER>); sendMsg(partner2, <OFFER>)
```

Cycle 2: (Agents respond to offerers and join their groups)

```
 1: for all neighbor ∈ myNeighbors do
 2:     myOffers.add(receiveMsg(neighbor, <OFFER>))
 3: if NOT offerer? then
 4:     if empty(myOffers) then
 5:         offerer? = TRUE
 6:     else
 7:         bestOffer = selectRandom(myOffers); committed? = TRUE
 8:         sendMsg(bestOffer.neighbor, <ACCEPT, TRUE, myConstraints, myContext>)
 9: for all neighbor ∈ myOffers \ bestOffer.neighbor do
10:     sendMsg(neighbor, <ACCEPT, FALSE>);
```

Cycle 3: (Offerer computes best move for group, sends it to group members)

```
 1: for all neighbor ∈ myNeighbors do
 2:     myAccepts.add(receiveMsg(neighbor, <ACCEPT, TRUE, neighborsConstraints, neighborsContext>)
 3: [ ourMove, ourGain ] = computeBestMove(myConstraints, myContext, myAccepts);
 4: for all neighbor ∈ myAccepts do
 5:     sendMsg(neighbor, <MOVE, ourMove, ourGain>); myGroup.add(neighbor)
```

Cycle 4: (Group members send gain message (for group move) to neighbors outside the group)

```
 1: for all neighbor ∈ myNeighbors do
 2:     myGroup.add(receiveMsg(neighbor, <MOVE, ourMove, ourGain>))
 3: for all neighbor in myNeighbors \ myGroup do
 4:     sendMsg(neighbor, <GAIN, ourGain>)
```

Cycle 5: (Based on neighbors' gains, agents commit or decommit to group move, notify offerers)

```
 1: for all neighbor ∈ myNeighbors do
 2:     neighborsGains.add(receiveMsg(neighbor, <GAIN, neighborsGain>))
 3: if ourGain > max(neighborsGains) then
 4:     committed? = TRUE // ties broken using node ordering
 5: if NOT offerer? then
 6:     sendMsg(myGroup.offerer, <COMMIT, committed?>)
```

Cycle 6: (Offerers send final confirmation (or non-confirmation) to receivers)

```
 1: for all neighbor ∈ myGroup do
 2:     committedGroup = receiveMsg(neighbor, <COMMIT, TRUE>)
 3: if committedGroup = myGroup then
 4:     confirmed? = TRUE
 5: for all neighbor ∈ myGroup do
 6:     sendMsg(neighbor, <CONFIRM, confirmed?>)
```

Cycle 7: (If whole group is confirmed, all group members move, otherwise don't move)

```
 1: confirmed? = receiveMsg(myGroup.offerer, <CONFIRM, confirmed?>)
 2: if confirmed? = TRUE then
 3:     myValue = ourMove.myValue
 4: for all neighbor ∈ myNeighbors do
 5:     sendMsg(neighbor, <VALUE, myValue>)
```

these accept (or decline) messages and will calculate the best possible move for the group formed by itself and the agents that accepted its offers, assuming all other agents keep the same values. It will communicate this joint move, as well as the potential gain in reward by taking this move, to the receivers in the group. In the fourth cycle, the receivers will receive the move and the gain, and now, just as in the 1- and 2-optimal versions of MGM, all agents in the group (offerer and receivers) will send this gain in a gain message to their neighbors outside the group. Then in the fifth cycle, as the agents receive similar gain messages from their neighbors, they consider themselves committed if their gains are greater than all those reported by their neighbors outside the group. Receivers send commit messages to the offerer to report this status. In the sixth cycle, the offerer checks the commit messages from the receivers. If all agents in the group are committed, the offerer sends a confirm message to the receivers; otherwise a deconfirm message is sent. Then, in the seventh cycle, offerers who have just sent a confirm message, as well as receivers who have just received one, change their values to the values specified in the joint move, and send out new value messages to all their neighbors as the algorithm repeats.

We omit the formal proof of monotonicity for MGM-3, but it is straightforward to see that, just as in MGM-2, the gain message cycle and the confirmation process prevent two neighboring agents not in the same group will ever move at the same time. The main difference is that now, since we have groups of three agents, the confirmation process requires two cycles (commit and confirm) where in MGM-2 it only required one (confirm). An additional cycle is also needed at the beginning since, while the offerer still initiates the formation of the group, it is now the offerer who centralizes information from the two receivers rather than the receiver centralizing information from a single offerer, for a total of seven cycles. We implemented SCA-3 by simply replacing the gain, commit, and confirm cycles of MGM-3 (cycles 4, 5, and 6) with a stochastic step added to cycle 3 in which the offerer decides whether or not the group will move at all by choosing a random number. SCA-3 thus requires four message cycles.

5.4 Experiments

We performed two groups of experiments - one for "medium-sized" DCOPs of forty variables (the largest problem size considered in [Modi et al., 2005]) and one for DCOPs of 1000 variables, larger than any problems considered in papers on complete DCOP algorithms.

5.4.1 Medium-Sized DCOPs

We considered three different domains for our first group of experiments. The first was a standard graph-coloring scenario, in which a cost of one is incurred if two neighboring agents choose the same color, and no cost is incurred otherwise. Real-world problems involving sensor networks, in which it may be undesirable for neighboring sensors to be observing the same location, are commonly mapped to this type of graph-coloring scenario. The second was a fully randomized DCOP, in which every combination of values on a constraint between two neighboring agents was assigned a random reward chosen uniformly from the set $\{1, \ldots, 10\}$. In both of these domains, we considered ten randomly generated graphs with forty variables, three values per variable, and 120 constraints. For each graph, we ran 100 runs of each algorithm, with a randomized start state. The third domain was chosen to simulate a high-stakes scenario, in which miscoordination is very costly. In this enviroment, agents are negotiating over the use of resources. If two agents decide to use the same resource, the result could be catastrophic. An example of such a scenario might be a set of unmanned aerial vehicles (UAVs) negotiating over sections of airspace, or rovers negotiating over sections of terrain. In this domain, if two neighboring agents take the same value, there is a large penalty incurred (-1000). If two neighboring agents take different values, they obtain a reward chosen uniformly from $\{10, \ldots, 100\}$. Because miscoordination is costly, we introduced a *safe* (zero) value for all agents. An agent with this value is not using any resource. If two neighboring agents choose zero as their values, neither a reward nor a penalty is obtained. In such a high-stakes scenario, a randomized start state would be a poor choice, especially for an anytime algorithm, as it would likely contain many of the large penalties. So, rather than using randomized start states, all agents started with the zero value. However, if all agents start at zero, then DSA and MGM would be useless, since no agent would ever want to move alone. So, a reward of one was introduced for the case where one agent has the zero value, and its neighbor has a nonzero value. In the high-stakes domain, we also performed 100 runs on each of 10 randomly generated graphs with forty variables and 120 constraints, but due to the addition of the safe value, the agents in these experiments had four possible values.

For each of the three domains, we ran: MGM, DSA with $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, MGM-2 with $q \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and SCA-2 with all combinations of the above values of p and q (where q is the probability of being an offerer and p is the probability of an uncommited agent acting). We also ran MGM-3 with $q = 0.5$. Each table shows an average of 100 runs on ten randomly generated examples with some selected values of p and q.

We used communication cycles as the metric for our experiments, as is common in the DCOP literature, since it is assumed that communication is the speed bottleneck. However, we note that, as we move from 1-optimal to 2-optimal algorithms, the computational cost each agent i must incur can increase by a factor of as much as $\sum_j |X_j|$ as the agent can now consider the combination of its and all its neighbors' moves. However, in the 2-optimal algorithms we present, each agent randomly picks a single neighbor j to coordinate with, and so its computation is increased by a factor of only $|X_j|$. As we move to 3-optimal algorithms, the cost increases further, as each agent considers the combination of its own move and two of its neighbors' moves; analogously, each agent randomly picks two neighbors to coordinate with. Although each run was for 256 cycles, most of the graphs display a cropped view, to show the important phenomena.

Figure 5.2 shows a comparison between MGM and DSA for several values of p. For graph coloring, MGM is dominated, first by DSA with $p = 0.5$, and then by DSA with $p = 0.9$. For the randomized DCOP, MGM is completely dominated by DSA with $p = 0.9$. MGM does better in the high-stakes scenario as all DSA algorithms have a negative solution quality (not shown in the graph) for the first few cycles. This happens because at the beginning of a run, almost every agent will want to move. As the value of p increases, more agents act simultaneously, and thus, many pairs of neighbors are choosing the same value, causing large penalties. Thus, these results show that the nature of the constraint utility function makes a fundamental difference in which algorithm dominates. Results from the high-stakes scenario contrast with [Zhang et al., 2003] and show that DSA is not necessarily the algorithm of choice when compared with DBA across all domains.

Figure 5.3 shows a comparison between MGM and MGM-2, for several values of q. In all domains, MGM-2 eventually reaches a higher solution quality after about thirty cycles, despite the algorithms' initial slowness. The stair-like shape of the MGM-2 curves is due to the fact that agents are changing values only once out of every five cycles, due to the cycles used in communication. Of the three values of q shown in the graphs, MGM-2 rises fastest when $q = 0.5$, but eventually reaches its highest average solution quality when $q = 0.9$, for each of the three domains. We note that, in the high-stakes domain, the solution quality is positive at every cycle,

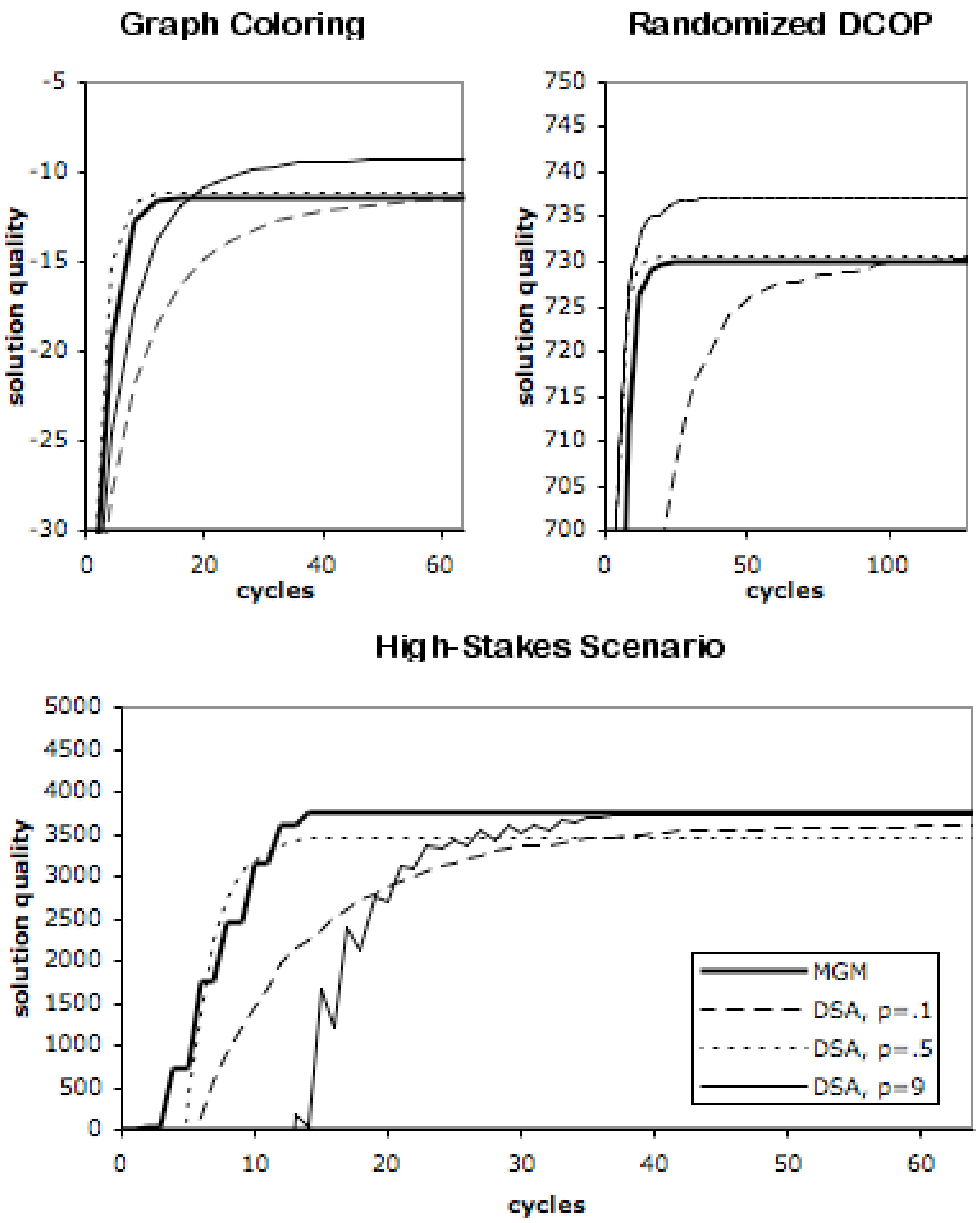


Figure 5.2: Comparison of the performance of MGM and DSA

due to the monotonic property of both MGM and MGM-2. Thus, these experiments clearly verify the monotonicity of MGM and MGM-2, and also show that MGM-2 reaches a higher solution quality as expected.

Figure 5.4 shows a comparison between DSA and SCA-2, for $p = 0.9$ and several values of q. DSA starts out faster, but SCA-2 eventually overtakes it. The result of the effect of q on SCA-2 appears inconclusive. Although SCA-2 with $q = 0.9$ does not achieve a solution quality above zero for the first 65 cycles, it eventually achieves a solution quality comparable to SCA with lower values of q.

Figure 5.5 compares MGM, MGM-2, and MGM-3 for $q = 0.5$. In all three cases, MGM-3 increases at the slowest rate, but eventually overtakes MGM-2. In the graph coloring and random DCOP domains, the time window where MGM-2 provides the highest solution is very small.

Figure 5.6 compares DSA, SCA-2, and SCA-3 for $p = 0.9$ and $q = 0.5$. The performance gains for increasing k are similar to the MGM case, except for the graph-coloring domain, where the gains are smaller moving from $k = 2$ to $k = 3$, likely because both SCA-2 and SCA-3 are approaching the optimal solution in almost every case.

Figure 5.7 contains a graph and a pie-chart for each of the three domains, providing a deeper justification for the improved solution quality of MGM-2 and SCA-2. The graph shows a probability mass function (PMF) of solution quality for three sets of assignments: the set of all assignments in the DCOP (X), the set of 1-optima (X_E), and the set of 2-optima (X_{2E}). Here we considered scenarios with twelve variables, 36 constraints, and three values per variable (four for the high-stakes scenario to include the zero value) in order to investigate tractably explorable domains. In all three domains, the solution quality of the set of 2-optima (the set of equilibria to

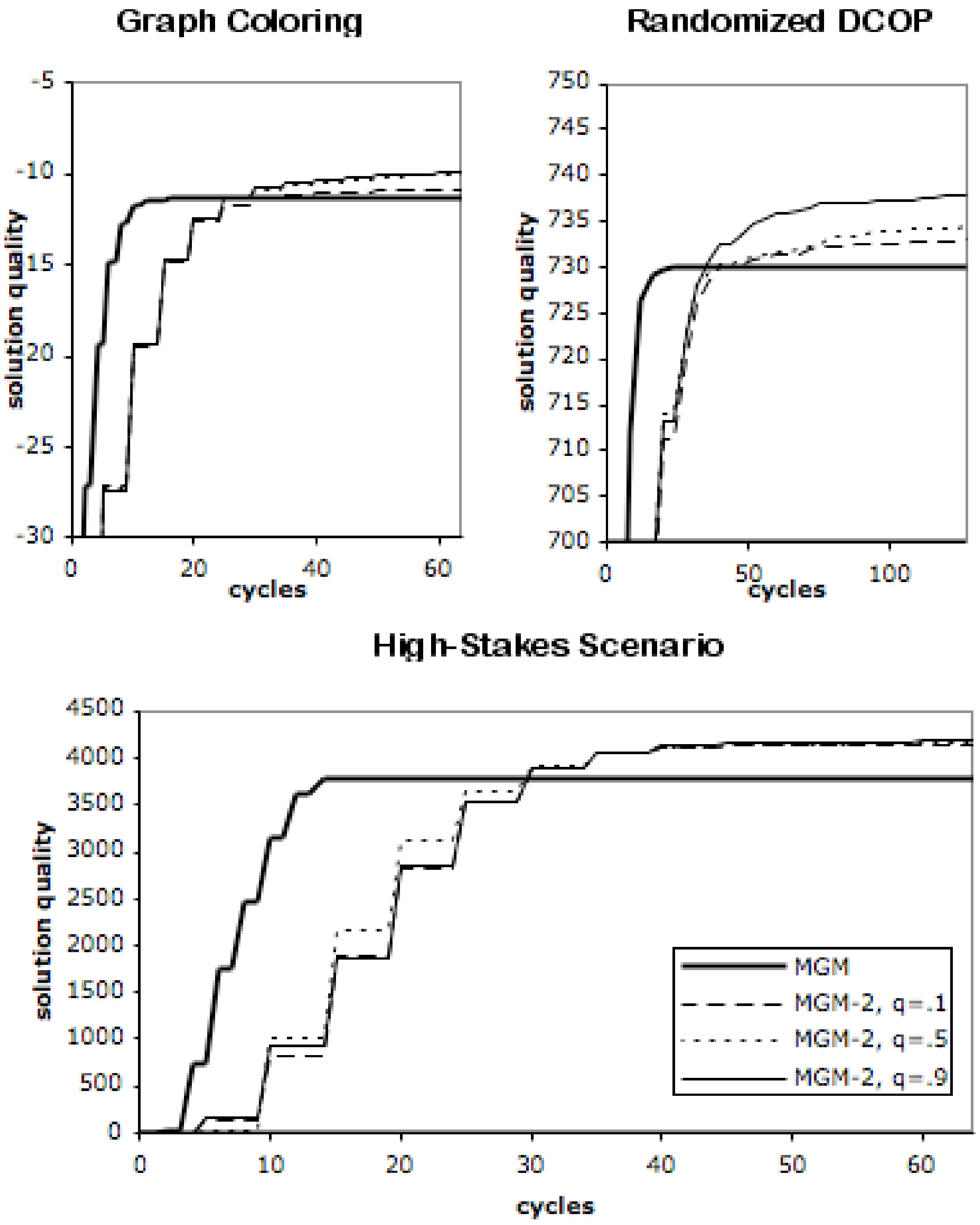


Figure 5.3: Comparison of the performance of MGM and MGM-2

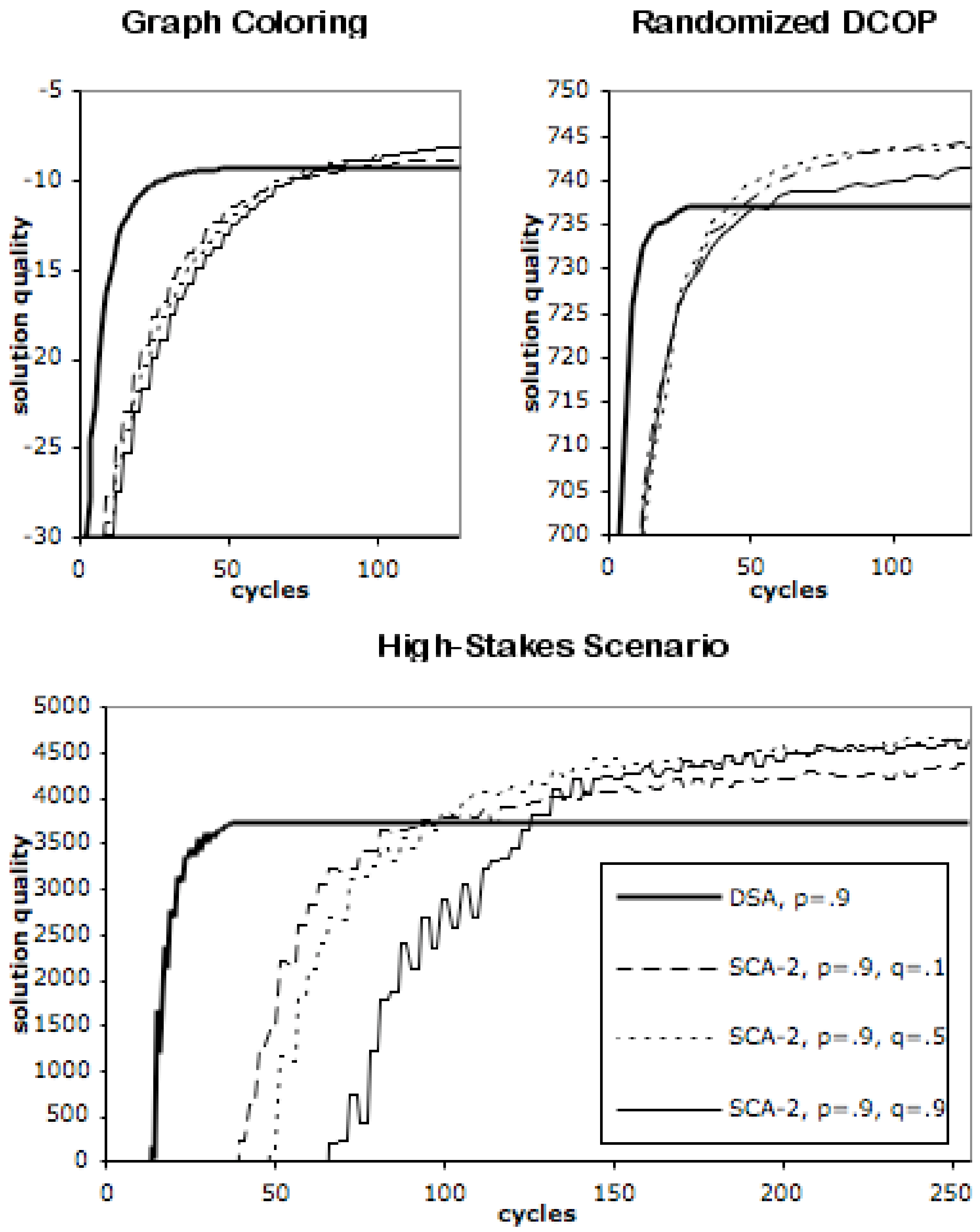


Figure 5.4: Comparison of the performance of DSA and SCA-2

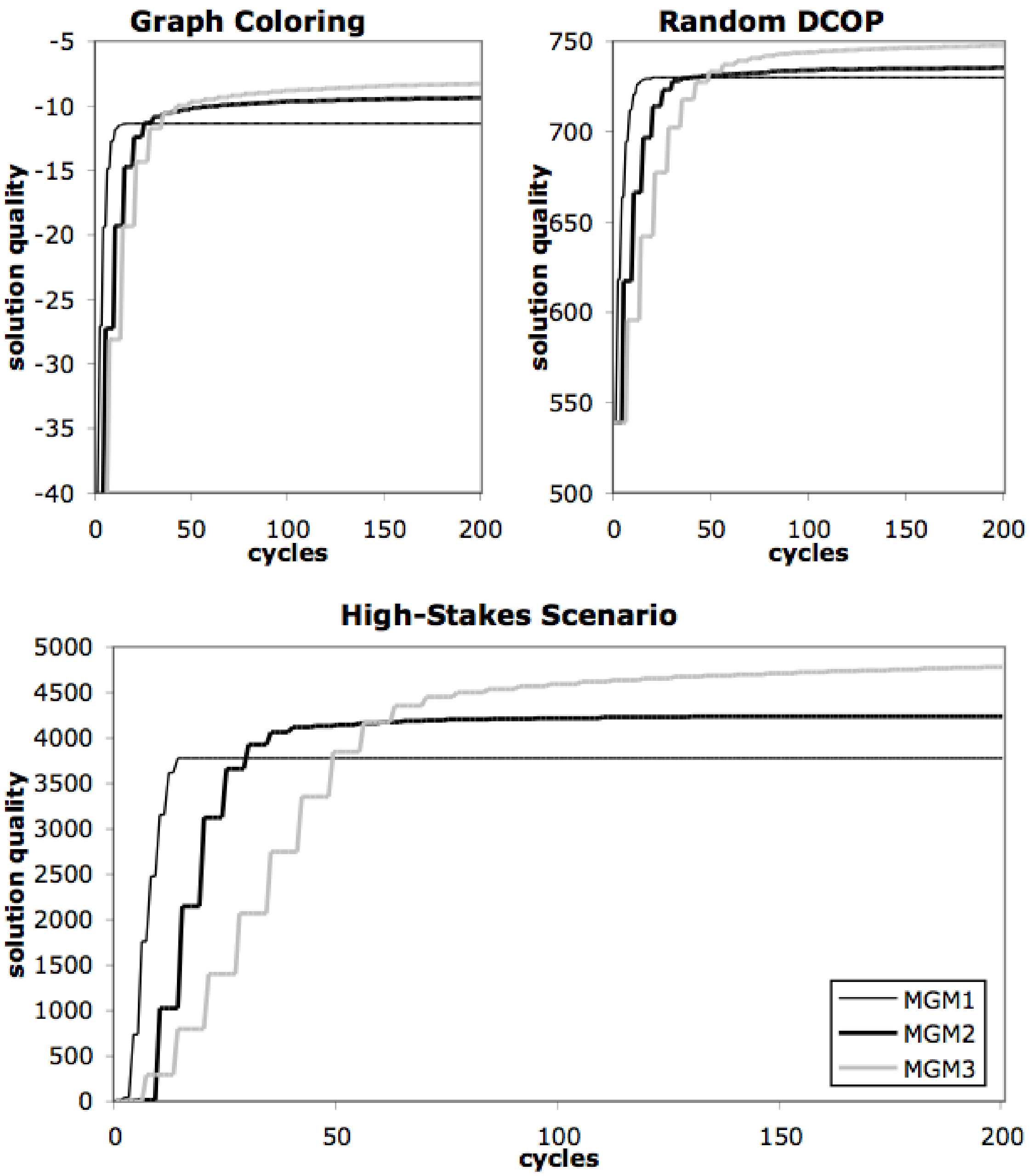


Figure 5.5: Comparison of the performance of MGM, MGM-2, and MGM-3

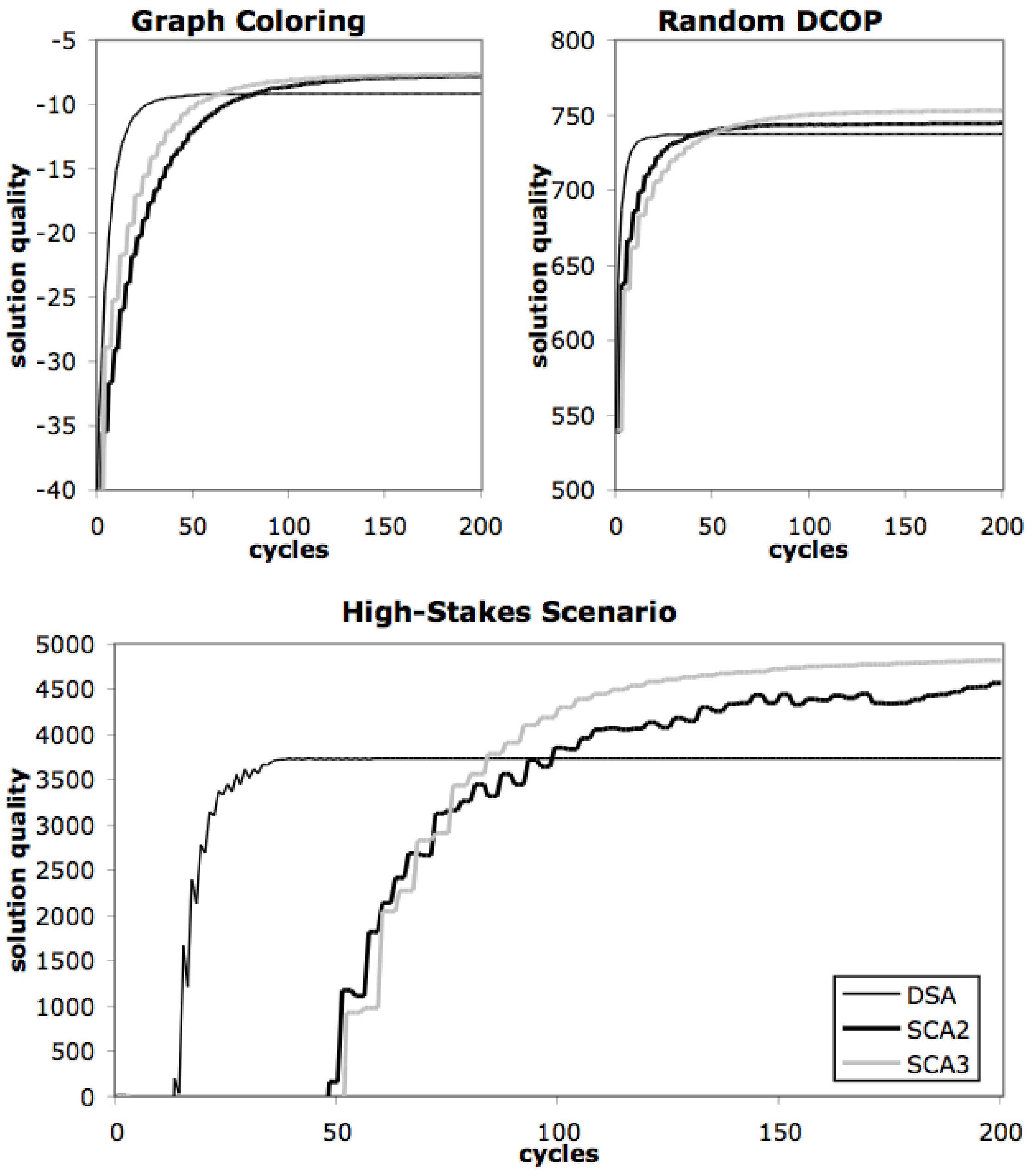


Figure 5.6: Comparison of the performance of DSA, SCA-2, and SCA-3

which MGM-2 and SCA-2 must converge) is, on average, higher than the set of 1-optima. In the high-stakes DCOP, 99.5% of assignments have a value less than zero (not shown on the graph.)

The pie chart shows the proportion of the number of 2-optima to the number of 1-optima that are not also 2-optima. Notice that in the case of the randomized DCOP, most 1-optima are also 2-optima. Therefore, there is very little difference between the PMFs of the two sets of *k*-optima on the corresponding graph. We also note that the phase transition mentioned in [Zhang et al., 2003] (where DSA's performance degrades for $p > 0.8$) is not replicated in our results. In fact, our solution quality gets better as $p > 0.8$, though with slower convergence.

5.4.2 Large DCOPs

For our second group of experiments, we considered DCOPs of 1000 variables using the graph-coloring and random DCOP domains. The main purpose of these experiments was to demonstrate that the *k*-optimal algorithms quickly converge to a solution even for very large problems such as these.

A random DCOP graph was generated for each domain, for link densities ranging from 1 to 5, and results for MGM and MGM-3 are shown in the following tables. The tables shown represent an average of 100 runs (from a random initial set of values) for each DCOP. The solution quality shown in the tables is the total reward in the DCOP divided by the number of constraints in the DCOP (so that the solution quality ranges from 0 to 1 in all cases for ease of comparison). Note that a solution quality of 1.000 does not represent the optimal solution to the DCOP; rather it represents a "theoretical" optimum, where all constraint rewards are 1 (i.e. no constraints are violated in the graph-coloring domain).

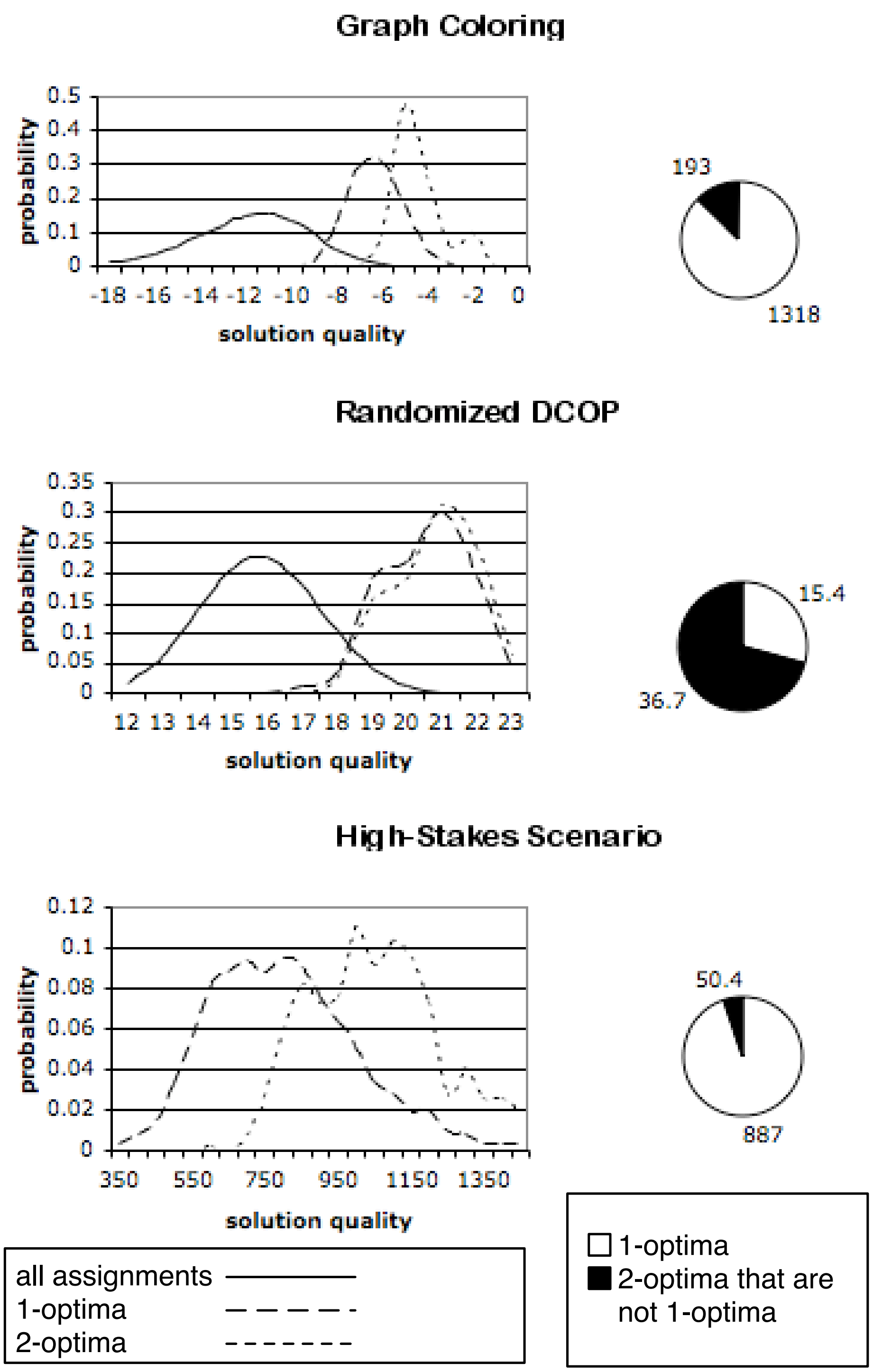


Figure 5.7: Distributions of Solution Quality for X, X_E, X_{2E} and Cardinality of X_{2E} as a proportion of X_E

Density	Quality (MGM)	Quality (MGM-3)	Cycles (MGM)	Cycles (MGM-3)
1	0.985	0.995	7.12	270.62
2	0.947	0.981	11.74	3277.89
3	0.915	0.948	15.58	4708.06
4	0.891	0.919	19.92	5220.46
5	0.874	0.897	23.30	5448.10

Table 5.1: Results for MGM and MGM-3 for large DCOPs: Graph Coloring

Density	Quality (MGM)	Quality(MGM-3)	Cycles (MGM)	Cycles (MGM-3)
1	0.872	0.927	8.54	1233.82
2	0.804	0.852	12.84	3993.15
3	0.759	0.795	17.20	4845.96
4	0.738	0.766	21.08	5685.47
5	0.708	0.731	24.96	5786.55

Table 5.2: Results for MGM and MGM-3 for large DCOPs: Random Rewards

Chapter 6

k-Optimality and Team Formation

6.1 Problem Formulation

To illustrate the benefits of applying *k*-optimality to real-world problems, I investigated the problem of human team formation, in which a team of people must be assembled from a large pool of possible candidates, in order to accomplish a specific task. These problems can be formulated as centralized COPs or as DCOPs, where each candidate is represented by a distributed agent. Either way, once the problem is formulated, several options must be generated rapidly to present to a human supervisor to make the final team selection. Finding *k*-optimal teams was a suitable choice for this domain, because it ensured that the set of choices would be diverse, and each possible team would be optimal within a region of similar teams that could be formed. Real data from three separate domains was used; in all three cases, the problem formulation was essentially the same. Each problem consisted of a set of candidates for the team and a set of roles on the team that must be filled. In each problem, there were from 40-70 candidates for a team of six to seven roles. With an agent for each candidate, this could represent a problem with as many as $8^{70} = 1.65 \times 10^{63}$ possible assignments of candidates to roles (including an additional "null" role, where a candidate is not assigned to the team) and finding a globally optimal team is NP-hard in general [Modi et al., 2005]. In addition, each problem consisted of various hard and soft constraints on the assignment of candidates to positions that are further detailed in the domain sections. To express these problems as DCOPs, each candidate was treated as an agent, and the agent's domain consisted of the set of roles that the candidate is allowed to take on plus a "null" value, representing the candidate's exclusion from the team. Not every candidate is allowed to take on every role. Hard constraints existed between agents whose domains shared a value; such that if both agents chose the same value, the constraint was considered violated. Other constraints are detailed in the domain sections.

6.2 Sample Domain 1: Task force for Homeland Security Exercise

The first set of data comes from the Tabletop Exercise (TTX) planning phase of the large-scale, multi-agency, AWI-07N anti-terrorist exercise being conducted in the Seattle-Tacoma area by the US Navy Center for Asymmetric Warfare. The AWI-07N exercise is planned as a week-long live simulation exercise in July 2007 during which forty participants from about thirty defense, law enforcement, security and operational agencies, organizations and companies will respond to and recover from a series of terrorist attacks in the Seattle-Tacoma port and maritime areas by forming a central command organization and a variety of special- purpose teams operating under that command.

6.2.1 Team Roles

The focus of TTX was on recovery operations after a series of varied terrorist attacks in the Seattle and Tacoma port and maritime areas (the Full Scale Exercise will include both immediate response and recovery.) Accordingly, a six-person Command Team was used, having the following roles:

- Officer in Charge (OIC)
- Assistant Officer in Charge (AIC)
- Intelligence Coordinator (IC)
- Operations Coordinator (OC)
- Inter-Agency Coordinator (IAC)
- Information Dissemination Coordinator (IDC)

The Command Team

6.2.2 Data Sources

A Qualification Check-In Form was filled out by 40 candidates from 30 different organizations (DoD, Security, Pierce County, Tacoma City, etc.). The Qualification Check-In Form had three parts, following a request for name, organization and job title; these parts were:

- Experience. Participants were asked to rate their experience in each of 9 technical areas – judged important to port and marine terrorist attacks and recovery – on a scale of 1 (relatively low) to 5 (relatively high) by checking the appropriate box and use a broad definition of each area.

- Team Processes. Participants were asked to answer 19 questions designed to assess their skills related to working in a team.
- Existing Connections. Participants were asked to list the people attending this Tabletop Exercise whom they had communicated with the most during the past several weeks by any means (in person, phone, e-mail, instant messaging, etc.), and provide the approximate number of communications to that person and from that person.

For convenience in modeling, we divided the 30 organizations into 8 types. Job Titles were categorized into levels from 1 (lowest) to 5 (top command level). Candidates were assigned ID numbers for use in the subsequent analyses. The tabulated results of the Experience section of the Check-In Form are shown in Figure 6.1. From left to right, the columns show: the person's ID number, the organization the person works for, the person's job title, the organization type (8 types), the person's rank within the organization (1 to 5) and the person's experience in various different skill areas.

6.2.3 DCOP Constraints

Three types of constraints were generated from this data: soft unary constraints, soft binary constraints, and hard constraints of any arity.

6.2.3.1 Soft Unary Constraints

Each agent had an associated unary constraint, generating a reward based on the value (role) it took, with a zero reward given for the null value. These constraint were generated in the following ways. The answers to the Experience questions on the Check-In Form were used to generate

					Experience								
ID Number	Agency	Title	Group	Rank	HAZMAT	Logistics	1st Resp	Coord	Law/Sec	DefSup	Salvage	C2 Org	Econ
1	WA State DOT	Emer Mgmt. Coordinator	3	3	4	2	3	3	4	1	1	3	2
2	Pierce Emer Mgmnt	Program Coordinator	4	5	5	5	5	5	3	4	3	5	4
3	Husky	AVP/ FSO	6	3	3	3	3	2	3	1	1	1	1
4	US Coast Guard	Contingency Planning	1	3	4	3	4	5	4	5	3	5	3
5	APM Terminals	General Manager	6	4	4	4	4	4	3	3	1	5	2
6	Pierce Co. Sheriff's Dpt	Lt.	4	3	2	3	4	4	5	1	2	4	3
7	Tacoma Police Dpt.	Patrol OFC (MSU/DIVE)	5	2	3	3	5	4	5	2	4	3	2
8	I.L.W.U	Union Security Liaison	6	2	4	4	4	4	3	3	3	4	4
9	Securitas Security	Branch Manager	7	2	2	4	4	2	5	2	1	2	1
10	WA United Terminals	Dir. Safety	6	3	5	4	3	4	2	1	1	1	1
11	DOC/MICC	Assoc. Supt.	3	4	3	3	4	3	3	2	2	4	2
12	PCSD	Deputy	4	2	4	3	4	2	5	3	4	3	2
13	Tacoma Police Dpt.	Lt.	5	3	2	3	5	4	5	2	1	5	2
14	APM Terminals	Security Manager	6	3	4	4	4	4	4	3	2	4	3
15	National Guard Bureau	Analyst	1	1	3	1	2	5	1	5	1	1	3
16	DTRA (Contractor)	Training & Ex.	2	1	1	2	1	4	2	5	1	4	1
17	Husky Terminals	Superintendent	6	4	4	4	4	3	3	2	1	4	2
18	Am Corporate Security	General Manager	7	4	3	4	3	3	5	1	1	5	4
19	PCSD	Deputy SAR Coordinator	4	2	5	5	5	5	5	4	1	5	5
20	Fort Lewis	Plans Officer	1	2	1	1	2	3	1	4	1	2	1
21	TSG	Analyst	2	2	1	2	3	2	3	1	1	2	1
22	CNRNW	ATFP	1	2	1	2	4	3	5	3	1	4	2
23	TOTE	Analyst	6	1	3	3	3	2	3	2	2	3	2
24	WADS	CBRNE OIC	3	3	4	3	4	5	2	5	1	4	2
25	Pierce County Terminal	Facility Security Officer	6	3	3	1	2	1	2	1	2	2	1
26	Madigan Army Hospital	Homeland Security Coord.	1	2	3	3	4	2	2	5	1	3	2
27	PCDRM	Program Manager	4	2	3	3	1	5	5	2	1	5	3
28	PC Ferry Project	Port. Capt.	4	4	2	2	3	2	2	1	2	1	1
29	Temco	Facility Security Officer	6	2	2	3	3	2	4	3	1	4	3
30	US Oil Refining	Manager, Admin. Services	6	2	1	2	1	1	4	1	1	4	2
31	Defense Coord Element	SR. Operations Analyst	2	2	2	2	3	5	1	5	1	2	4
32	Tacoma Fire	Battalion Chief	5	3	5	2	5	3	1	1	1	5	1
33	WA Dept. of Corr	Lietenant	3	3	2	4	5	3	5	2	1	4	2
34	WSDOT Aviation	Aviation ES Coordiator	3	3	2	2	5	4	2	2	1	5	2
35	WSDOT Aviation	Incident Resp Manager	3	2	3	3	5	3	5	3	1	4	3
36	CAW	Ex. Manager	8	2	5	5	5	5	5	5	4	5	5
37	HQ NORAD NJ33N	NBC/RECA CBRN/RECA	1	3	5	3	5	5	2	4	1	5	3
38	Tacoma Fire Dpt.	Battalion Chief	5	3	4	2	5	4	1	1	1	5	1
39	CAW	Exercise Mgr.	8	2	1	3	4	5	3	4	1	5	5
40	CTTSO-TSWG	Program Manager	2	3	1	1	1	1	1	2	4	1	1

Figure 6.1: Tabulated results of Experience section of Check-In Form

scores for every candidate for five mission-related criteria: Organization and Control, Emergency Response, Economic Effects, Defense Support, and Port and Marine Operations. Similarly, the answers to the Team Processes questions were used to generate scores for every candidate for three team-related criteria: Coordination Attitudes, Attitudes to Teamwork, and Leadership Attitudes. Each team role was paired with two sets of predetermined weights; one set contained weights for each of the mission-related areas, and the other set contained weights for each of the team-related areas. Thus, the suitability of a candidate for each role with respect to the mission-related criteria was determined by a weighted sum of the candidate's mission-related scores, using the weights particular to that role. Similarly, the suitability of a candidate for each role with respect to the team-related criteria was determined by a weighted sum of the candidate's team-related scores. In addition, social networking centrality measures, used to estimate leadership skills, were generated based on the Existing Connections section of the form. For the Officer-In-Charge role, the centrality measure, mission-related scores and team-related scores were combined; for all other roles, only the mission-related and team-related scores were used.

6.2.3.2 Soft Binary Constraints

From the Existing Connections section, a second measure was generated, to represent the connectedness between each pair of candidates in the social network. Pairs of agents, representing candidates who were connected in some way, were assigned binary constraints, such that a reward proportional to this connectedness measure was generated if both agents were assigned a non-null value.

6.2.3.3 Hard Constraints

Additional pre-existing constraints on the team composition were expressed as DCOP constraints of any arity with a large negative reward if violated. Single-agent (unary) constraints were expressed simply by removing values from agents' domains. Multi-agent constraints were expressed explicitly with constraints in the DCOP. For this domain, the initial hard constraints used to generate teams were:

- Officer in Charge must be from Groups 4 or 5
- Officer in Charge must have Rank/Position ≥ 3
- Officer in Charge: Leadership ≥ 2.5 + Interpersonal Skills ≥ 2
- Operations Coordinator must have Emergency Response ≥ 2 and Defense Support ≥ 2
- Inter-Agency Coordinator must have Interpersonal Skills ≥ 2 and Team Attitude ≥ 2
- Team must contain at least one member from Group 1
- Team must contain at least one member from Group 3
- Team must contain at least one member with Port and Marine Operations ≥ 3
- Team must contain at least one member with Economic Effects ≥ 3

In addition a relaxed set of constraints were used to generate other teams.

- Officer in Charge must have Rank/Position ≥ 3
- Team must contain at least one member from Groups 4 and 5

6.2.3.4 Weighting the Constraints

Finally, the relative importance of the four types of soft constraints (mission-related unary constraints, team-related unary constraints, centrality unary constraints, and connectedness binary constraints) to the global objective function could be adjusted by further weighting these four measures with respect to each other. In the initial examples used, each of the four factors were weighted equally, with each accounting for 25% of the overall objective.

6.2.4 Formation of *k*-Optimal Teams

2-optimal teams were formed for several cases; each case used a different combination of weights and hard constraints. For each case, 100 random runs of a 2-optimal algorithm were used to generate teams; discarding duplicate teams, the results are shown in Figure 6.2. Each run took approximately 4 minutes on a 2.16 GHz Intel Core Duo machine with 2GB of RAM running Mac OS X. In some cases (1, 4, and 7) all 100 runs produced the same team, strongly suggesting that this was the only 2-optimal team possible under the constraints given (and that it was the globally optimal team). In others, several teams were produced. Since the teams are 2-optimal, any two teams in a single case must have more than two differing assignments of people to roles, providing diversity among the choices.

6.3 Sample Domain 2: Second Homeland Security Exercise

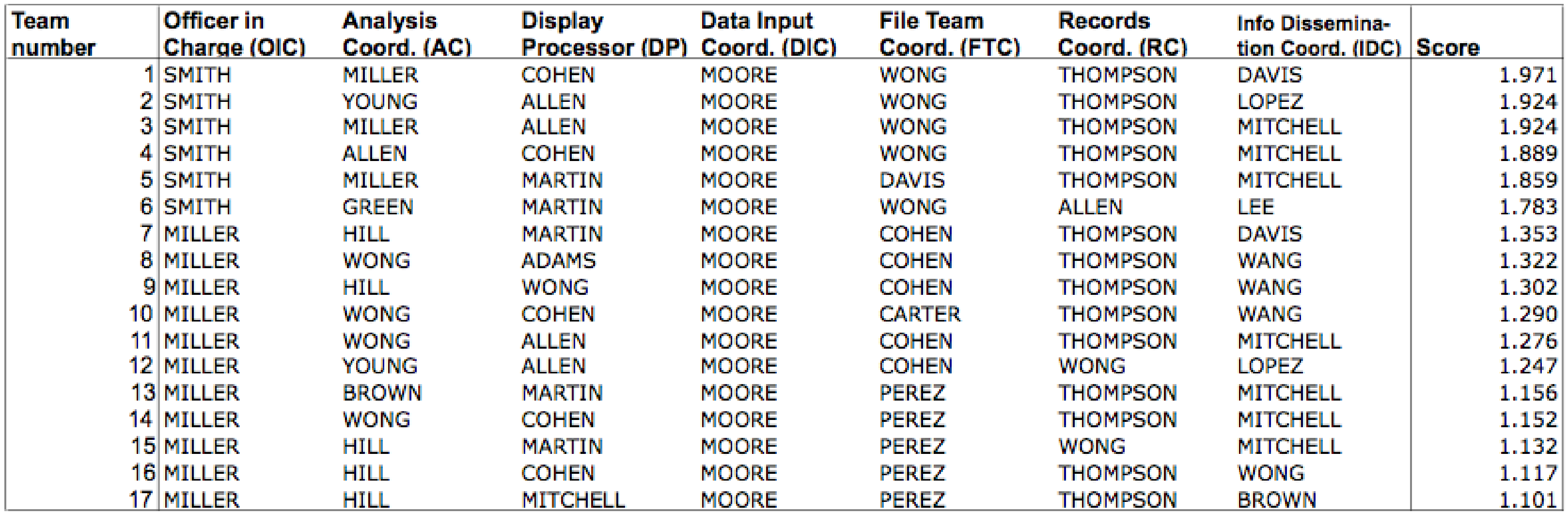
We also applied the same techniques on a larger dataset from a similar exercise in the Los Angeles area. Here, 2-optimal and 3-optimal seven-person teams were formed from a pool of 68 candidates using 100 random runs each. The 2-optimal teams formed are shown in Figure 6.3.

Constraint settings	Team number	Officer in Charge (OIC)	Asst. Officer in Charge (AO)	Intelligence Coord. (IC)	Operations Coord. (OC)	Inter-Agency Coord. (IAC)	Info Dissemina-tion Coord. (IDC)	Score
case 1: 25% mission, 25% team, 25% centrality (for OIC), 25% connectivity, original hard constraints	1	2	19	13	26	17	35	**1.130**
case 2: 50% mission 50% team, original hard constraints	1	2	19	18	36	26	35	**1.073**
	2	2	19	36	26	31	35	**1.072**
	3	2	35	18	26	31	36	**1.050**
case 3: 25% mission, 25% team, 25% centrality (for OIC), 25% connectivity, no hard constraints	1	14	19	13	2	5	17	**1.740**
	2	14	19	2	36	13	17	**1.730**
case 4: 50% mission, 25% centrality (for OIC), 25% connectivity, relaxed multiagent hard constraints	1	14	19	2	13	5	17	**1.668**
case 5: 50% mission, 25% centrality (for OIC), 25% connectivity, original hard constraints	1	2	13	19	4	24	17	**1.243**
	2	2	13	19	17	4	35	**1.231**
case 6: 50% mission, 25% centrality (for AO), 25% connectivity, original hard constraints (mission-	1	2	14	19	4	24	17	**1.711**
	2	2	14	19	17	4	35	**1.700**
case 7: 50% mission, 25% centrality (for OIC), 25% connectivity, relaxed hard constraints	1	14	13	19	36	2	17	**1.852**

Figure 6.2: 2-optimal teams formed according to various criteria

Due to the larger dataset, more 2-optimal teams were found in the search space. Again, any two teams must have more than two differing assignments of people to roles.

When 3-optimality was used, only the first team in the figure was produced, strongly suggesting that this team is the globally optimal team, and the only 3-optimal team.



Team number	Officer in Charge (OIC)	Analysis Coord. (AC)	Display Processor (DP)	Data Input Coord. (DIC)	File Team Coord. (FTC)	Records Coord. (RC)	Info Dissemina-tion Coord. (IDC)	Score
1	SMITH	MILLER	COHEN	MOORE	WONG	THOMPSON	DAVIS	1.971
2	SMITH	YOUNG	ALLEN	MOORE	WONG	THOMPSON	LOPEZ	1.924
3	SMITH	MILLER	ALLEN	MOORE	WONG	THOMPSON	MITCHELL	1.924
4	SMITH	ALLEN	COHEN	MOORE	WONG	THOMPSON	MITCHELL	1.889
5	SMITH	MILLER	MARTIN	MOORE	DAVIS	THOMPSON	MITCHELL	1.859
6	SMITH	GREEN	MARTIN	MOORE	WONG	ALLEN	LEE	1.783
7	MILLER	HILL	MARTIN	MOORE	COHEN	THOMPSON	DAVIS	1.353
8	MILLER	WONG	ADAMS	MOORE	COHEN	THOMPSON	WANG	1.322
9	MILLER	HILL	WONG	MOORE	COHEN	THOMPSON	WANG	1.302
10	MILLER	WONG	COHEN	MOORE	CARTER	THOMPSON	WANG	1.290
11	MILLER	WONG	ALLEN	MOORE	COHEN	THOMPSON	MITCHELL	1.276
12	MILLER	YOUNG	ALLEN	MOORE	COHEN	WONG	LOPEZ	1.247
13	MILLER	BROWN	MARTIN	MOORE	PEREZ	THOMPSON	MITCHELL	1.156
14	MILLER	WONG	COHEN	MOORE	PEREZ	THOMPSON	MITCHELL	1.152
15	MILLER	HILL	MARTIN	MOORE	PEREZ	WONG	MITCHELL	1.132
16	MILLER	HILL	COHEN	MOORE	PEREZ	THOMPSON	WONG	1.117
17	MILLER	HILL	MITCHELL	MOORE	PEREZ	THOMPSON	BROWN	1.101

Figure 6.3: 2-optimal teams

Chapter 7

Other Work

In addition to my work in *k*-optimality, I have also contributed to advances in conceptualizing and measuring privacy in DCOPs [Maheswaran et al., 2006; Greenstadt et al., 2006], as well as in solving DCOPs more efficiently [Maheswaran et al., 2004b].

7.1 Privacy in DCOPs

It is critical that agents deployed in real-world settings, such as businesses, offices, universities and research laboratories, protect their individual users' privacy when interacting with other entities. Indeed, privacy is recognized as a key motivating factor in the design of several multiagent algorithms, such as in distributed constraint reasoning (including both algorithms for distributed constraint optimization (DCOP) and distributed constraint satisfaction (DisCSPs)), and researchers have begun to propose metrics for analysis of privacy loss in such multiagent algorithms. Unfortunately, a general quantitative framework to compare these existing metrics for privacy loss or to identify dimensions along which to construct new metrics was currently lacking.

In [Maheswaran et al., 2006] we presented three key contributions to address this shortcoming. First, we introduced VPS (Valuations of Possible States), a general quantitative framework to express, analyze and compare existing metrics of privacy loss. Based on a state-space model, VPS is shown to capture various existing measures of privacy created for specific domains of DisCSPs. The utility of VPS is further illustrated through analysis of privacy loss in DCOP algorithms, when such algorithms are used by personal assistant agents to schedule meetings among users. In addition, VPS helps identify dimensions along which to classify and construct new privacy metrics and it also supports their quantitative comparison. Second, [Maheswaran et al., 2006] presents key inference rules that may be used in analysis of privacy loss in DCOP algorithms under different assumptions. Third, detailed experiments based on the VPS-driven analysis lead to the following key results: (i) decentralization by itself does not provide superior protection of privacy in DisCSP/DCOP algorithms when compared with centralization; instead, privacy protection also requires the presence of uncertainty about agents' knowledge of the constraint graph. (ii) one needs to carefully examine the metrics chosen to measure privacy loss; the qualitative properties of privacy loss and hence the conclusions that can be drawn about an algorithm can vary widely based on the metric chosen.

In [Greenstadt et al., 2006] we applied a similar analysis to more recent DCOP algorithms, including DPOP and Adopt, to see if they suffered from a similar shortcoming. We found that several of the most efficient DCOP algorithms, including both DPOP and Adopt, provided better privacy protection than the algorithms considered in [Maheswaran et al., 2006], such as SynchBB. Furthermore, we examined, for the first time, the privacy implications of various distributed constraint reasoning design decisions, e.g. constraint-graph topology, asynchrony, message-contents, to provide an improved understanding of privacy-efficiency tradeoffs. Finally, this paper augmented our previous work on system-wide privacy loss, by investigating inequities in individual agents' privacy loss.

7.2 Solving DCOPs Efficiently

To capably capture a rich class of complex problem domains, we introduced the Distributed Multi-Event Scheduling (DiMES) framework and showed how DCOPs could be formulated, using only binary constraints, whose optimal solution is also the optimal solution to the DiMES problem [Maheswaran et al., 2004b]. To approach real-world efficiency requirements, we obtained large speedups for the Adopt algorithm using two preprocessing steps: improving the communication (tree) structure of the agents and precomputing best-case bounds on solution quality. Results were given for meeting scheduling and sensor network domains.

Chapter 8

Related Work

Although *k*-optimality is a fundamentally new concept in distributed constraint reasoning, we can draw parallels to related work.

8.1 Constraint Reasoning

An active area of recent research has been in the developtment of complete "*n*-optimal" algorithms, including Adopt [Modi et al., 2005], OptAPO [Mailler and Lesser, 2004] and DPOP [Petcu and Faltings, 2005].

The Asynchronous Distributed Optimization (Adopt) algorithm [Modi et al., 2005] is the first known algorithm for DCOP; it is an asychnronous algorithm in which a Depth-First Search (DFS) tree is extracted from the DCOP graph, and small messages are passed up and down the tree. The runtime of Adopt was significantly improved using preprocessing techniques in [Maheswaran et al., 2004b] and [Ali et al., 2005]. A second algorithm for DCOP, Optimal Asynchronous Partial Overlay (OptAPO) [Mailler and Lesser, 2004], requires no DFS tree. Instead agents partially centralize the problem by forming subgroups; one agent in the subgroup would receive the constraints on the other agents and would solve the subproblem for that subgroup; in most cases complete centralization was not necessary to reach an optimal solution. A third algorithm, DPOP [Petcu and Faltings, 2005], is a dynamic programming-based algorithm in which a DFS tree is used, as in Adopt. In DPOP, each agent passes just one message up the tree. The agent considers every possible combination of values that could be chosen by the higher-priority agents that have constraints with this agent, and finds the best value it can choose, for each of these combinations; the value and reward for every combination is included in this message. Multiply-constrained DCOPs, for which the bounds on solution quality for *k*-optima in this thesis can also be applied, were introduced in [Bowring et al., 2006], along with a modification of Adopt to solve these problems optimally.

Previous work in incomplete, 1-optimal algorithms for DCOP was described in detail in Chapter 5. The aim of this thesis is to consider the large space in between these two classes of algorithms and solutions.

This thesis also provides a theoretical complement to the experimental analysis of local minima (1-optima) and landscapes in centralized constraint satisfaction problems (CSPs) [Yokoo, 1997] as well as incomplete DCOP algorithms [Zhang et al., 2003]. While this thesis is primarily concerned with the effects of varying *k* and graph structure, the cited works provide insight into the effects of the choice between *k*-coordinated algorithms of the same *k*-level on solution quality and convergence time. Note that *k*-optimality can also apply to centralized constraint reasoning as a measure of the relative quality and diversity of local optima. However, examining properties of solutions that arise from coordinated value changes of small groups of variables is especially useful in distributed settings, given the computational and communication expense of large-scale coordination.

Finally, despite the seeming similarity of *k*-optimality to *k*-consistency [Freuder, 1978] in centralized constraint satisfaction, the two concepts are entirely different, as *k*-consistency refers to reducing the domains of subsets of variables to maintain internal consistency in a satisfaction framework while *k*-optimality refers to comparing fixed solutions where subsets of variables optimize with respect to an external context.

8.2 Distributed Markov Decision Processes

It should also be noted that the bounds on solution quality of *k*-optima in this thesis also apply to networked distributed partially-observable Markov decision processes as defined in [Nair et al., 2005]. These structures can be viewed as DCOPs where each agent's domain represents its entire policy space, and rewards arise from the combinations of policies chosen by subsets of agents. Since the solution quality bounds do not depend on domain size, they can be directly applied to *k*-optimal ND-POMDP policies; in fact, the locally optimal algorithm presented in [Nair et al., 2005] is a 1-optimal algorithm.

8.3 Local Search and Game Theory

My research on upper bounds on the number of *k*-optima in DCOPs is related to work in landscape analysis in local search and evolutionary computing. In particular, [Caruana and Mullin, 1999] and [Whitley et al., 1998] have provided techniques for estimating numbers of local optima in these problems. In contrast, my work provides worst-case bounds on the number of *k*-optima in a DCOP, rather than an estimate based on sampling the solution space, and also exploits the structure of the agent network (constraint graph) to obtain these bounds, which was not done in previous work.

Given that counting the number of Nash equilibria in a game with known payoffs is #$\mathcal{P}$-hard [Conitzer and Sandholm, 2003], upper bounds on this number have indeed been investigated for particular types of games [McLennan and Park, 1999; Keiding, 1995]. Additionally, graph structure is utilized in several different algorithms to expedite finding Nash equilibria for a given graphical game with known payoffs [Kearns et al., 2001; Vickrey and Koller, 2002; Blum et al., 2003; Littman et al., 2002]. However, using graph structure to finding bounds on pure-strategy Nash equilibria over all possible games on a given graph (i.e., reward-independent bounds) remained an open problem; this thesis contains the first known upper bounds on Nash equilibria in such games.

8.4 Local Search in Combinatorial Optimization

The definition of k-optimality in DCOPs is analogous to the *k-opt* family of algorithms for the canonical Traveling Salesman Problem (TSP) [Lin, 1965]. In TSP, a route is considered k-optimal if it cannot be improved by replacing any k or fewer edges with new edges in the route graph. A quality guarantee on 2-optimal TSP tours has been given in [Lin, 1965].

Chapter 9

Conclusions and Future Work

9.1 Conclusions

In addition to new algorithms for distributed constraint optimization, this thesis makes several contributions towards understanding locally optimal states that occur in systems of cooperative agents when the ability to cooperate is bounded.

In Chapter 2, k-optimality was introduced as an algorithm-independent classification for locally optimal solutions to a DCOP. At the same time, based on this definition, any DCOP algorithm can be classified as a k-optimal algorithm for some value of k. We have also shown the existence of several useful properties of k-optimal DCOP solutions. In Chapter 6, a case study of the application of k-optimality to the problem of human team formation was shown, in order to instantiate many of the theoretical ideas presented in the rest of the thesis. k-optimal teams were found using actual data, including social network data, from several real military and civilian task scenarios. In Chapter 5, new families of k-optimal algorithms were designed for $k = 2$ and $k = 3$. Important properties of these algorithms, such as monotonicity, were proven. The

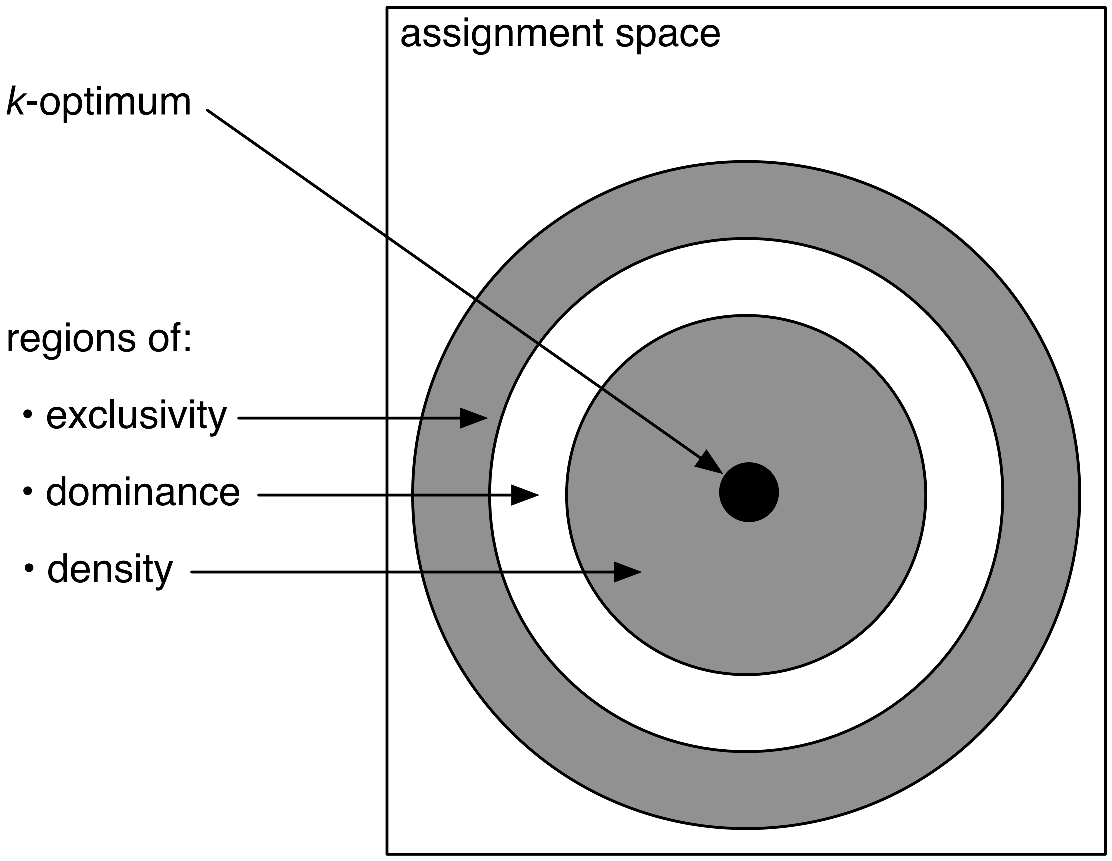


Figure 9.1: A depiction of three key regions in the assignment space, given a *k*-optimum, that allowed for the discovery of various theoretical properties about *k*-optima

algorithms were also analyzed experimentally and shown to outperform existing 1-optimal algorithms. In addition, these algorithms were used to find *k*-optimal solutions to DCOPs larger than those solved optimally in any previous work.

Based on this foundation, this thesis contains three key insights that led to the further contributions of theoretical results about *k*-optimality. We can refer to these insights as *exclusivity*, *dominance*, and *density*. Figure 9.1 depicts a *k*-optimal assignment *a* existing within the space of all possible assignments to a given DCOP. This thesis has shown how any *k*-optimum implies the existence of three particularly defined regions of the assignment space. Certain properties of these regions of the space (the number of assignments they contain, as well as the total reward of those assignments) allow us to determine properties of the *k*-optima themselves. These three regions are shown in the figure, each corresponding to one of the three insights:

1. The region of *exclusivity*, as found in Chapter 4 contains all the assignments that, given the *k*-optimum *a*, cannot also be a *k*-optimum. These assignments occur when any group *G* of *k* or fewer agents faces the same context as in *a*, but some agent in *G* chooses a different value from its value in *a*. Note that it is possible for some of these assignments to actually have a reward higher than that of *a* (due to value changes by other agents outside the view of *G*). Nevertheless, these assignments cannot be *k*-optimal because the agents in *G* (or some subset) can always improve the global reward by changing their values.

2. A subset of the region of exclusivity is the region of *dominance*. As found in Section 3.4, this region, given a *k*-optimum, contains all the assignments that not only cannot be a *k*-optimum, but also must have a lower or equal reward to the *k*-optimum.

3. A subset of the region of dominance is the region of highest *density*. We can define the density of a region of assignments $\hat{A}$ as the mean reward of all assignments in $\hat{A}$:

   $$\frac{\sum_{\hat{a}\in\hat{A}} R(\hat{a})}{|\hat{A}|}$$

   In Section 3.1 and 3.2, we are finding, for any *k*-optimum, the *densest* region that this *k*-optimum must dominate (have a higher reward than all assignments in the region). This region is of course a subset of the *k*-optimum's region of dominance. Without knowing the actual costs and rewards on the constraints in the DCOP, we can express the density of this particular region in terms of $R(a)$ (the reward of the k-optimum) and $R(a^*)$ (the reward of the global optimum).

These three regions of assignments are the basis for four remaining contributions of this thesis. Each contribution is a set of theoretical results about k-optima.

1. **Quality guarantees on k-optima.** The region of highest density defined by a k-optimum was used to find the guarantees on the solution quality of a k-optimum, given in Sections 3.1 and 3.2. Since a k-optimum a must dominate this region, it must therefore have reward $R(a)$ higher than the region's density. Since this density can be expressed in terms of $R(a)$ and $R(a^*)$, a lower bound on the reward of the k-optimum can be expressed in terms of the reward of the globally optimal solution, independent of the costs and rewards on the DCOP constraints, as well as the domain size of each variable.

2. **Quality guarantees on k-optima in DCOPs with hard constraints.** Both the region of highest density and the region of dominance are used in Section 3.3 to extend these guarantees to DCOPs that contain hard constraints. In that section, we observe that these guarantees are only possible given certain restrictions on the quantity and placement of the hard constraints in the DCOP. Otherwise it would be possible for a k-optimum to violate a hard constraint, making no guarantee possible for a k-optimum in such a DCOP. In a DCOP withe these restrictions, we showed it is not possible for a k-optimum to violate a hard constraint, because if such a k-optimum existed, it would be dominated by an assignment in its *own* region of dominance; a contradiction. Subsequently, this section showed how the region of highest density could be found in the presence of hard constraints by ensuring that no assignment that violated a hard constraint would be included in the region. If such

an assignment were included, its large negative reward would offset the reward of the other assignments, driving the density of the region below zero and making guarantees on quality impossible for k-optima.

3. **Guarantees on domination ratio of k-optima.** The region of dominance was used in Section 3.4 to calculate the guaranteed domination ratio (the proportion of assignments in the total space that any k-optimum must dominate).

4. **Upper bounds on the number of k-optima in a DCOP.** Finally, the region of exclusivity was used in Chapter 4 to calculate upper bounds on the number of k-optima that can exist, given a DCOP graph. Each k-optimum claimed a region of assignments that could not also be k-optima. Then, each assignment in the region was divided among the maximal number of k-optima whose exclusivity regions it could possibly be in. This provided each k-optimum with a distinct region of assignments (or shares of assignments) that it could claim. Dividing the number of total assignments in the whole assignment space by the size of this region produced the upper bound on the number of k-optima that could exist given a DCOP graph.

9.2 Future Work

While this thesis has begun to explore the effects of agents' bounded ability to cooperate on their performance as a team, the following areas seem especially promising for future work in this area.

1. **Analogs of k-optimality for noncooperative settings: restricted coalition-proof equilibria**. A primary justification for k-optimality is the cost and difficulty of aggregating the

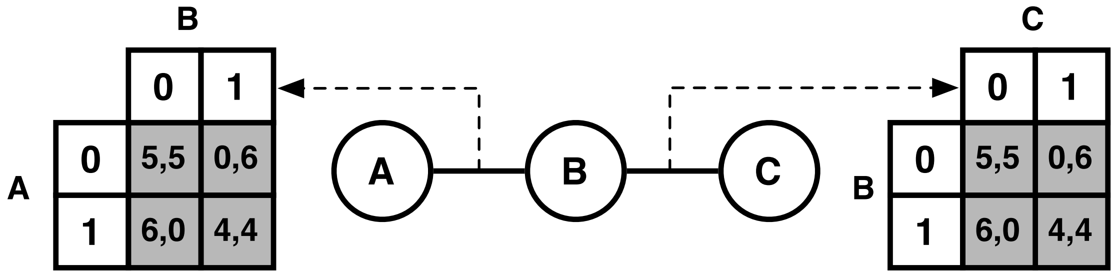


Figure 9.2: Graphical game example

preferences of large groups of agents; if agents are bounded in this ability, up to a certain group size, a *k*-optimum will arise for cooperative settings such as DCOPs. A clear area for investigation is the extension of this idea to noncooperative settings. Figure 9.2 shows a sample graphical game - a game where players' payoffs depend only on the actions of a subset of other players. Edges in the graph show which players affect each others' payoffs; since this relationship is symmetric in this example, the graph is undirected. Here, the payoffs for agent A depend only on its own action and that of agent B (the row player's payoff is listed first). Agent B's payoffs depend on the actions of all three agents (B receives the sum of the payoffs from its two matrices with A and C).

What happens when, due to bounds on computational ability or time, noncooperative agents in graphical games are limited in their ability to form coalitions that contain more than *k* agents? Game theorists have focused on properties of coalition-proof equilibria, but this work generally assumes that all sizes of coalitions are possible. If coalitions are limited to *k* agents, the space of "*k*-coalition-proof" equilibria should become larger, and it may be easier to find such equilibria. These equilibria would be just as robust as standard coalition-proof equilibria in settings where agents do not have the ability to form coalitions of more than *k* agents. I will begin by developing algorithms to find *k*-coalition-proof equilibria in graphical games by using existing techniques to find all Nash equilibria, and then focus on efficiently checking them for *k*-coalition-proofness.

2. **Efficient *k*-optimal algorithms for DCOP**. Now that levels of solution quality can be guaranteed for *k*-optima, another clear area for future work is the development of efficient, distributed *k*-optimal algorithms for $k > 3$. One path along which these algorithms can be developed are the MGM/DSA framework, based on the algorithms in this thesis, where larger groups are formed in each round, at a cost of more message cycles per round. One challenge of this approach is that, for $k > 3$ there is not always an agent in a group that has a constraint with all of the other agents (consider a chain of four agents for example). This possibility would require a detailed system for passing messages in order to ensure that the group would always make coordinated value changes.

   Another framework that could possibly be utilized is the OptAPO algorithm of [Mailler and Lesser, 2004]. In this complete algorithm, agents designated as mediators find optimal solutions to subgraphs of the DCOP. It would seem that if this responsibility of mediating agents were relaxed to require only a *k*-optimal solution to their subproblems, a *k*-optimal solution would result.

3. **Improving bounds on solution quality of *k*-optima**. The current lower bounds on solution quality of *k*-optima depend only on the graph structure of the DCOP, and do not take any information about rewards into account. This makes them applicable over all possible reward structures, and is appropriate given the assumption that no central source has knowledge of all the rewards in the system. However, initial experiments have shown that if the system designer knows some partial information about the rewards on the constraints,

this can be used to add constraints to the LFP which is used to find lower bounds on solution quality for arbitrary graphs. One area for future work is to explore the effect of this kind of partial knowledge on the solution quality bounds for *k*-optima. We focus on partial information because if all rewards are known and collected together, then an exact lower bound could be found by simply enumerating all *k*-optima and their respective rewards.

Bibliography

S. M. Ali, S. Koenig, and M. Tambe. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. In *AAMAS*, 2005.

N. Alon and N. Kahale. Approximating the independence number via the theta-function. *Mathematical Programming*, 80:253–264, 1998.

B. Blum, C. Shelton, and D. Koller. A continuation method for Nash equilibria in structured games. In *IJCAI*, 2003.

E. Bowring, M. Tambe, and M. Yokoo. Multiply-constrained distributed constraint optimization. In *AAMAS*, 2006.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge U. Press, 2004.

R. Caruana and M. Mullin. Estimating the number of local minima in complex search spaces. In *IJCAI Workshop on Optimization*, 1999.

V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *IJCAI*, 2003.

J. Cox, E. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS*, 2005.

S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In V. Lesser, C. L. Ortiz, and M. Tambe, editors, *Distributed Sensor Networks: A Multiagent Perspective*, pages 257–295. Kluwer, 2003.

E. C. Freuder. Synthesizing constraint expressions. *Communications of the ACM*, 21(11):958–966, 1978.

R. Greenstadt, J. P. Pearce, and M. Tambe. Analysis of privacy loss in DCOP algorithms. In *AAAI*, 2006.

G. Gutin and A. Yeo. Domination analysis of combinatorial optimization algorithms and problems. In M. Golumbic and I. Hartman, editors, *Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications*. Kluwer, 2005.

M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *Proc. UAI*, 2001.

H. Keiding. On the maximal number of Nash equilibria in an *n* x *n* bimatrix game. *Games and Economic Behavior*, 21(1-2):148–160, 1995.

J. T. Kim and D. R. Shin. New efficient clique partitioning algorithms for register-transfer synthesis of data paths. *Journal of the Korean Phys. Soc.*, 40(4):754–758, 2002.

S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

S. Ling and C. Xing. *Coding Theory: A First Course*. Cambridge U. Press, 2004.

M. L. Littman, M. Kearns, and S. Singh. An efficient, exact algorithm for solving tree-structured graphical games. In *Proc. NIPS*, 2002.

R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *PDCS*, 2004a.

R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: efficient complete solutions for distributed multi-event scheduling. In *AAMAS*, 2004b.

R. T. Maheswaran, J. P. Pearce, E. Bowring, P. Varakantham, and M. Tambe. Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications. *JAAMAS*, 13:27–60, 2006.

R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS*, 2004.

A. McLennan and I. Park. Generic 4 x 4 two person games have at most 15 Nash equilibria. *Games and Economic Behavior*, 26(1):111–130, 1999.

P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI*, 2005.

A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, 2005.

A. Petcu and B. Faltings. ODPOP: An algorithm for open/distributed constraint optimization. In *AAAI*, 2006.

S. Ruan, C. Meirina, F. Yu, K. R. Pattipati, and R. L. Popp. Patrolling in a stochastic environment. In *10th Intl. Command and Control Research Symp.*, 2005.

T. Sandholm. *Negotiation among Self-Interested Computationally Limited Agents*. PhD thesis, University of Massachusetts at Amherst, Dept. of Computer Science, 1996.

N. Schurr, J. Marecki, P. Scerri, J.P. Lewis, and M. Tambe. The DEFACTO system: Training tool for incident commanders. In *IAAI*, 2005.

A. Tate, J. Dalton, and J. Levine. Generation of multiple qualitatively different plan options. In *Proc. AIPS*, 1998.

D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Proc. AAAI*, pages 345–351, 2002.

N. Vlassis, R. Elhorst, and J. R. Kok. Anytime algorithms for multiagent decision making using coordination graphs. In *Proc. Intl. Conf. on Systems, Man and Cybernetics*, 2004.

D. Whitley, S. Rana, and R. B. Heckendorn. Representation issues in neighborhood search and evolutionary algorithms. In D. Quagliarella, et al., editor, *Genetic Algs. and Evolution Strategies in Eng. and Comp. Sci.*, pages 39–57. Wiley, 1998.

M. Yokoo. How adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs. In *Int'l Conf. on Constraint Programming*, 1997.

M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction and optimization problems. In *ICMAS*, 1996.

W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. An analysis and application of distributed constraint satisfaction and optimization algorithms in sensor networks. In *AAMAS*, 2003.

W. Zhang, G. Wang, Z. Xing, and L. Wittenberg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, 2005a.

Y. Zhang, J. G. Bellingham, R. E. Davis, and Y. Chao. Optimizing autonomous underwater vehicles' survey for reconstruction of an ocean field that varies in space and time. In *American Geophysical Union, Fall meeting*, 2005b.

Curriculum Vitae

Contact Information

Jonathan P. Pearce
University of Southern California
Powell Hall of Engineering, Room 210
Los Angeles, CA 90089
(213) 740 7231
jppearce@usc.edu
http://www-scf.usc.edu/~jppearce

Research Interests

Autonomous Agents, Multi-Agent Systems, Distributed Constraint Optimization, Privacy in Collaborative Environments, Graphical Models for Artificial Intelligence, Game Theory, Reasoning under Uncertainty, Coalition Formation.

Education

University of Southern California, Los Angeles, CA
Ph.D., Computer Science (expected) 09/03–06/07
Thesis: "Local Optimization in Cooperative Agent Networks"
Advisor: Milind Tambe
Committee: Bhaskar Krishnamachari, Victor Lesser, Fernando Ordóñez

Massachusetts Institute of Technology, Cambridge, MA
M.Eng., Electrical Engineering and Computer Science 09/00–06/01
Thesis: "Qualitative Behavior Prediction for Simple Mechanical Systems"
Advisor: Randall Davis

Massachusetts Institute of Technology, Cambridge, MA
S.B., Management Science (Sloan School of Management) 09/96–06/01
S.B., Computer Science and Engineering 09/96–06/00
Minor in Economics

Research Funding Granted

Automated Mission Scheduling and Collaboration Support By Distributed Constraint Optimization for Shared Control of Unmanned Vehicles 3/01/07–3/01/08 (funded: $100k), wrote USC portion of proposal (part of DARPA SBIR Phase I)

Optimizing Team Composition Using Modeling and Multiagent Systems, 12/01/06–12/01/07 (funded: $70k), wrote USC portion of proposal (part of US Army Research Institute SBIR Phase I)

Rapid Formation of Virtual Organizations Using Modeling and Multiagent Systems, 10/01/06–10/01/07 (funded: $100k), wrote USC portion of proposal (part of DARPA STTR Phase I)

Computational Models for Effects Based Operations in Special Forces Teams (in submission), wrote USC portion of proposal (part of DARPA SBIR Phase I)

Reconfigurable Robot Teams for Surface Operations, 2005 (unfunded), assisted with proposal writing (NASA)

Awards

Best Paper Award, Eighth International Workshop on Distributed Constraint Reasoning (DCR–07), Hyderabad, India (at IJCAI–07), for J. P. Pearce and M. Tambe, "Lower Bounds on *k*-Optimal Solutions for Distributed Constraint Optimization Problems," 1/8/07.

Outstanding Research Assistant award, Computer Science Department, USC. Awarded annually to two Ph.D. students for excellence in research (out of over 240 Ph.D. students), 12/4/06.

Chosen as one of three USC graduate students university-wide to present research to US Secretary of Homeland Security Tom Ridge, 1/15/04.

Eta Kappa Nu (Electrical engineering and computer science honor society), 2000.

David A. Chanen Writing Prize, for best-written undergraduate term paper in computer science at MIT (out of over 200 papers), 1998.

Research Experience

Graduate Research Assistant, USC Computer Science Department 8/03–present
Supervisor: Milind Tambe
Developed novel algorithms and theoretical results for multiagent distributed constraint optimization. Applied algorithms to a working system of personal-assistant software agents as part of a multi-institution effort called CALO (Cognitive Agent that Learns and Organizes). Presented results at DARPA PI meetings.

Graduate Research Assistant, MIT AI Lab 9/00–6/01
Undergraduate Research Assistant, MIT AI Lab 1/00–5/00
Supervisor: Randall Davis

Developed a theoretical framework and software engine for qualitative analysis of mechanical diagrams.

Summer Research Associate, Charles River Analytics, Cambridge, MA 6/98–8/98
Supervisor: Magnus Snorrason
Created a terrain-based guidance algorithm for an unmanned aircraft searching for targets. Developed a software tool for testing this algorithm on a variety of scenarios. Presented work to funders at Eglin Air Force Base, Florida. This work was the deliverable in a Phase I proposal which was subsequently approved for Phase II funding.

Undergraduate Research Assistant, MIT AI Lab 1/98–5/98
Supervisor: Dave Cliff
Created an interactive tool for visualizing the results and editing parameters of automated bidding agents in auctions.

Publications

Journal Articles in Submission

J. P. Pearce, M. Tambe and R.T. Maheswaran, "Properties of *k*-Optimal Solutions in Distributed Constraint Optimization," in submission, *Journal of Artificial Intelligence Research*, 33 pages.

Rigorously Refereed Journal Articles and Conference Papers (Full-Length)

P. Paruchuri, J. P. Pearce, F. Ordóñez, S. Kraus and M. Tambe, "An Efficient Heuristic Approach for Security Against Multiple Adversaries," to appear in *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS–07)*, 8 pages. (acceptance rate 22%)

J. P. Pearce and M. Tambe, "Quality Guarantees on *k*-Optimal Solutions for Distributed Constraint Optimization Problems," to appear in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI–07)*, Hyderabad, India, January 6–12, 2007, 6 pages. (acceptance rate 34%)

R. Greenstadt, J. P. Pearce and M. Tambe, "Analysis of Privacy Loss in DCOP Algorithms," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI–06)*, Boston, Massachusetts, July 16–20, 2006, pp. 647–653. (selected for oral presentation, acceptance rate 23%)

J. P. Pearce, R. T. Maheswaran, and M. Tambe, "Solution Sets for DCOPs and Graphical Games," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi–Agent Systems (AAMAS–06)*, Hakodate, Japan, May 8–12, 2006, pp. 577–584. (selected for oral presentation, acceptance rate 11%)

R. T. Maheswaran, J. P. Pearce, P. Varakantham, E. Bowring and M. Tambe, "Privacy Loss in Distributed Constraint Reasoning: A Quantitative Framework for Analysis and its Applications." *Journal of Autonomous Agents and Multi Agent Systems (JAAMAS)*, 13:27–60, 2006.

R. T. Maheswaran, J. P. Pearce, P. Varakantham, E. Bowring and M. Tambe, "Valuation of Possible States: A Unifying Quantitative Framework for Evaluating Privacy in Collaboration," in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS–05)*, Utrecht, The Netherlands, July 25–29, 2005, pp. 1030–1037. (acceptance rate 24%)

R. T. Maheswaran, J. P. Pearce and M. Tambe, "Distributed Algorithms for DCOP: A Graphical-Game-Based Approach," in *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems (PDCS–04)*, San Francisco, CA, September 15–17, 2004, pp. 432–439.

R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce and P. Varakantham, Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS–04)*, New York, NY, July 19–13, 2004, pp. 310–317. (acceptance rate 24%)

Rigorously Refereed Conferences: Short Papers

J. P. Pearce, "Locally Optimal Algorithms and Solutions for Distributed Constraint Optimization," in Member Abstracts and Posters Track, *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI–06)*, Boston, Massachusetts, July 16–20, 2006., 2 pages.

P. M. Berry, C. Albright, E. Bowring, K. Conley, K. Nitz, J. P. Pearce, B. Peintner, S. Saadati, M. Tambe, T. Uribe, and N. Yorke-Smith, "CALO Conflict Negotiation: PTIME and DCOP," demo paper in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS–06)*, Hakodate, Japan, May 8–12, 2006, pp. 1467–8.

J. P. Pearce, "Locally Optimal Algorithms and Solutions for Distributed Constraint Optimization," in Doctoral Consortium, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS–06)*, Hakodate, Japan, May 8–12, 2006, 2 pages.

R. Greenstadt, J. P. Pearce, E. Bowring and M. Tambe "Experimental analysis of privacy loss in DCOP algorithms," short paper in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS–06)*, Hakodate, Japan, May 8–12, 2006, pp. 1424–26.

J. P. Pearce, R. T. Maheswaran and M. Tambe, "How Local Is That Optimum? *k*-optimality for DCOP," short paper in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS–05)*, Utrecht, The Netherlands, July 25–29, 2005, pp. 1303–1304.

Refereed Technical Magazine Articles

M. Tambe, E. Bowring, J. P. Pearce, P. Varakantham, P. Scerri, D. V. Pynadath, "Electric Elves: What Went Wrong and Why," in AAAI Spring Symposium on What Went Wrong and Why: Lessons from AI Research and Applications, to appear, *AI Magazine*, 2007.

Book Chapters

R. T. Maheswaran, J. P. Pearce and M. Tambe, "A Family of Graphical-Game-Based Algorithms for Distributed Constraint Optimization Problems," in P. Scerri, R. Mailler and R. Vincent editors, *Coordination of Large-Scale Multiagent Systems*, Springer, 2005, pp. 127–146.

Invited Papers

P. Paruchuri, E. Bowring, R. Nair, J. P. Pearce, M. Tambe, N. Schurr and P. Varakantham, "Multi-agent Teamwork: Hybrid Approaches," *Communications of the Computer Society of India*, 30(6):19–24, 2006.

M. Tambe, E. Bowring, J. P. Pearce, P. Varakantham, P. Scerri, D. V. Pynadath, "Electric Elves: What Went Wrong and Why," in AAAI Spring Symposium on What Went Wrong and Why: Lessons from AI Research and Applications, Stanford, CA, March 27–29, 2006.

M. Tambe, E. Bowring, H. Jung, G. Kaminka, R. T. Maheswaran, J. Marecki, P. J. Modi, R. Nair, J. P. Pearce, P. Paruchuri, D. Pynadath, P. Scerri, N. Schurr and P. Varakantham, "Conflicts in Teamwork: Hybrids to the Rescue," in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Utrecht, The Netherlands, July 25–29, 2005, pp. 3–10.

Refereed Symposium and Workshop Papers

P. Paruchuri, J. P. Pearce, F. Ordóñez , S. Kraus and M. Tambe, "An Efficient Heuristic for Security Against Multiple Adversaries in Stackelberg Games," to appear in AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents (GTDT), Stanford, CA, March 26-28, 2007.

J. P. Pearce and M. Tambe, "Lower Bounds on *k*-Optimal Solutions for Distributed Constraint Optimization Problems," in The Eighth International Workshop on Distributed Constraint Reasoning (DCR), Hyderabad, India, January 8, 2007 (at IJCAI–07). **(Winner of Best Paper Award)**

R. Greenstadt, J. P. Pearce and M. Tambe, "An Experimental Analysis of Privacy Loss in DCOP Algorithms," in The Seventh International Workshop on Distributed Constraint Reasoning (DCR), Hakodate, Japan, May 8, 2006 (at AAMAS–06).

J. P. Pearce and M. Tambe, "Quality Guarantees on Locally Optimal Solutions for Distributed Constraint Optimization Problems," in The Seventh International Workshop on Distributed Constraint Reasoning (DCR), Hakodate, Japan, May 8, 2006 (at AAMAS–06).

J. P. Pearce, R. T. Maheswaran, and M. Tambe, "Solution Sets for DCOPs and Graphical Games: Metrics and Bounds," in Ninth International Symposium on Artificial Intelligence and Mathematics, Ft. Lauderdale, FL, January 4–6, 2006.

J. P. Pearce, R. T. Maheswaran and M. Tambe, "Local Algorithms for Distributed Constraint Optimization in Dynamic, Anytime Environments," in Workshop on Ambient Intelligence -

Agents for Ubiquitous Environments, Utrecht, The Netherlands, July 26, 2005 (at AAMAS–05).

J. P. Pearce, R. T. Maheswaran and M. Tambe, “Graph-Based Bounds on k-Optimal Joint-Action Sets for Multiple Agents,” in Workshop on Declarative Agent Languages and Technologies (DALT), Utrecht, the Netherlands, July 25, 2005 (at AAMAS–05).

R. T. Maheswaran, J. P. Pearce, P. Varakantham, E. Bowring and M. Tambe, “Valuations of Possible States (VPS): A Quantitative Framework for Analysis of Privacy Loss Among Collaborative Personal Assistant Agents,” in AAAI Spring Symposium on Persistent Assistants: Living and Working with AI, Menlo Park, CA, March 21–23, 2005.

J. P. Pearce, R. T. Maheswaran and M. Tambe, “DCOP Games for Multi-Agent Coordination,” in The Fifth International Workshop on Distributed Constraint Reasoning (DCR), Toronto, Canada, September 27, 2004 (at CP–04).

R. T. Maheswaran, J. P. Pearce and M. Tambe, “Distributed Algorithms for DCOP: A Graphical Game-Based Approach,” (longer version) in Workshop on Challenges in the Coordination of Large-Scale Multi-Agent Systems, New York, NY, July 20, 2004 (at AAMAS–04).

Presentations at Conferences and Workshops

“Lower Bounds on *k*-Optimal Solutions for Distributed Constraint Optimization Problems,” in Eighth International Workshop on Distributed Constraint Reasoning (DCR), Hyderabad, India, January 8, 2007 (at IJCAI–07).

“Solution Sets for DCOPs and Graphical Games,” in Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Hakodate, Japan, May 11, 2006.

“Distributed Algorithms for DCOP: A Graphical-Game-Based Approach,” in 17th International Conference on Parallel and Distributed Computing Systems (PDCS), San Francisco, CA, September 15, 2004. Also given at AAMAS Workshop on Challenges in the Coordination of Large-Scale Multi-Agent Systems, New York, NY, July 20, 2004.

“Quality Guarantees on Locally Optimal Solutions for Distributed Constraint Optimization Problems,” in AAMAS 2006 Workshop on Distributed Constraint Reasoning (DCR), Hakodate, Japan, May 8, 2006.

“Locally Optimal Algorithms and Solutions for Distributed Constraint Optimization,” in Doctoral Consortium, 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Hakodate, Japan, May 8, 2006.

“Solution Sets for DCOPs and Graphical Games: Metrics and Bounds,” in Ninth International Symposium on Artificial Intelligence and Mathematics, Ft. Lauderdale, FL, January 4-6, 2006.

“Local Algorithms for Distributed Constraint Optimization in Dynamic, Anytime Environments,” in AAMAS Workshop on Ambient Intelligence - Agents for Ubiquitous Environments, Utrecht, The Netherlands, July 26, 2005.

"Graph-Based Bounds on *k*-Optimal Joint-Action Sets for Multiple Agents," in AAMAS Workshop on Declarative Agent Languages and Technologies (DALT), Utrecht, the Netherlands, July 25, 2005.

Invited Talks

"Local Optimization in Cooperative Agent Networks," University of California, Irvine, February 6, 2007.

"Local Optimization in Cooperative Agent Networks," University of California, San Diego, April 30, 2007.

Teaching

Course Design Experience

Contributor, Intelligent Agents and Science Fiction (CSCI 499), USC, Fall 2006. Assisted with conceptual design and creation of syllabus for completely new undergraduate class on AI and multi-agent systems, using science fiction movies and literature as inspiration for topics covered. Instructor: Milind Tambe.

Teaching Experience

Teaching Assistant, Software Multi-Agent Systems (CSCI 543), USC, Spring 2005. Instructor: Milind Tambe. Graduate-level course of 50 students. Duties included choosing topics and papers for the course syllabus, preparing lecture slides and assignments, teaching several lectures, mentoring students on final projects, holding regular office hours. Topics included game theory, constraint reasoning, partially observable Markov decision processes, belief-desire-intentions-based agent models, multiagent learning, market-based systems, and coalition formation.

Guest Lecturer, Advanced Artificial Intelligence (CECS 551), California State University, Long Beach, Spring 2006 (4/3/06). Instructor: Colleen Van Lent.

Guest Lecturer, Software Multi-Agent Systems (CSCI 543), USC, Spring 2006 (1/31/06, 2/2/06). Instructor: Milind Tambe.

Guest Lecturer, Advanced Artificial Intelligence (CSCI 573), USC, Spring 2005 (4/11/05). Instructor: Sven Koenig.

Guest Lecturer, Advanced Artificial Intelligence (CSCI 573), USC, Fall 2005 (8/22/05). Instructor: Milind Tambe.

Professional Service

Conferences

Organizer, Ninth International Workshop on Distributed Constraint Reasoning (DCR), 2007

Program Committee, National Conference on Artificial Intelligence (AAAI), 2007
Reviewer, International Joint Conference on Artificial Intelligence (IJCAI), 2007
Reviewer, International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2006
Reviewer, Seventh International Workshop on Distributed Constraint Reasoning (DCR), 2006
Reviewer, International Symposium on Artificial Intelligence and Mathematics, 2006
Reviewer, International Joint Conference on Artificial Intelligence (IJCAI), 2005
Reviewer, Florida Artificial Intelligence Research Society Conference (FLAIRS), 2005
Reviewer, 17th Brazilian Symposium on Artificial Intelligence, 2004

Journals

Reviewer, IEEE Transactions on System, Man, and Cybernetics, 2006–present
Reviewer, IEEE/ACM Transactions on Networking, 2006–present
Reviewer, Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS), 2005–present
Reviewer, ACM Computing Surveys, 2004–present

Other Service

Organizing Committee, USC Computer Science Colloquium Series, 2005–present
Organizer, Distributed Constraint Optimization (DCOP) repository website, where several researchers have made their programs and data sets available for testing and further research. *http://teamcore.usc.edu/dcop*

Industry Experience

Applications Engineer, Oracle Corporation, Redwood Shores, CA 8/01–5/03
Advanced Supply Chain Planning and Scheduling Group
Developed and implemented novel constraint-based algorithms for supply-chain scheduling in Oracle Advanced Supply Chain Planning software. Also worked directly with corporate clients including Sony, Agilent, and Intersil Semiconductor.

Software Engineering Intern, Hewlett-Packard, Cupertino, CA 6/00–8/00
Internet Security and Solutions Group

Intern, Ernst & Young Consulting, New York, NY 6/99–8/99