On K-optimal Distributed Constraint Optimization Algorithms: New Bounds and Algorithms

Emma Bowring*, Jonathan P Pearce+, Christopher Portway+, Manish Jain+, Milind Tambe+
*University of the Pacific, Stockton, CA 95211
+University of Southern California, Los Angeles, CA 90089
ebowring@pacific.edu
{jppearce,portway,manish.jain,tambe}@usc.edu

ABSTRACT

Distributed constraint optimization (DCOP) is a promising approach to coordination, scheduling and task allocation in multi agent networks. In large-scale or low-bandwidth networks, finding the global optimum is often impractical. K-optimality is a promising new approach: for the first time it provides us a set of locally optimal algorithms with quality guarantees as a fraction of global optimum. Unfortunately, previous work in k-optimality did not address domains where we may have prior knowledge of reward structure; and it failed to provide quality guarantees or algorithms for domains with hard constraints (such as agents' local resource constraints). This paper addresses these shortcomings with three key contributions. It provides: (i) improved lower-bounds on k-optima quality incorporating available prior knowledge of reward structure; (ii) lower bounds on k-optima quality for problems with hard constraints; and (iii) k-optimal algorithms for solving DCOPs with hard constraints and detailed experimental results on large-scale networks.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Design, Theory

Keywords

Constraint reasoning, DCOP, Multi Agent Systems, k-optimality

1. INTRODUCTION

Distributed Constraint Optimization(DCOP) [7, 6, 13] is a major approach within cooperative multiagent systems for distributed planning, scheduling and task allocation, and it has been applied to multi-agent plan coordination[2] , sensor networks [13, 4] , meeting scheduling [10] and RoboCup soccer [11]. In DCOP, teams of agents coordinate their individual actions to achieve joint goals, but the utility of an agent's action depends on the action choices of a subset of the other agents. Traditionally, DCOP has focused on obtaining a single, globally optimal solution through complete algorithms like Adopt [7], OptAPO [6], and DPOP [10]. However, DCOP has been shown to be NP-hard[7], so in large-scale domains, complete algorithms incur tremendous computation and communication costs. In contrast, incomplete algorithms, in which agents optimize locally, are easier to scale up [3, 12, 8, 9].

K-optimal algorithms are an important class of incomplete algorithms[3, 12, 8, 9] where agents dynamically form local groups to coordinate action choices. A *k-optimum* occurs when no group of k or fewer agents can improve the solution. k-optimal algorithms are the first set of incomplete algorithms that provide theoretical guarantees such as the worst-case solution quality and upper bounds on the number of k-optima in a DCOP. These guarantees are important: not only do they guarantee a solution quality as a fraction of the global optimum when running a k-optimal algorithm, but they can aid in algorithm selection and network structure selection in situations when coordination costs must be weighed against solution quality. If increasing k will significantly increase the guaranteed solution quality, the extra computation or communication cost incurred by the k-optimal algorithm with higher k may be justified.

Unfortunately, previous work in k-optimality suffers from three key shortcomings which limits applicability to newer domains. First, it did not address domains where we may have prior knowledge of DCOP reward structure, and thus provided pessimistic quality guarantees. Yet in many domains we may have some information about the range of rewards that may be obtained. For example, in sensor networks[13], we may know the maximum reward of observing a phenomenon and the minimum reward when we have no observations. Second, previous work failed to provide quality guarantees for domains with hard constraints. 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. Additionally, previous k-optimality work did not consider multiply constrained DCOPs where the hard constraints represent travel budgets[1]. Third, previous work failed to provide algorithms for domains involving such hard resource constraints. This paper addresses these shortcomings with three key contributions. It provides: (i) lower-bounds on k-optima quality incorporating available prior knowledge of reward structure; (ii) lower bounds on k-optima quality for problems with hard constraints; and (iii) k-optimal algorithms for solving DCOPs with hard constraints, along with experimental results for networks of 1000 agents.

Cite as: On K-optimal Distributed Constraint Optimization Algorithms: New Bounds and Algorithms, Emma Bowring, Jonathan Pearce, Christopher Portway, Manish Jain, Milind Tambe, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes,MüllerandParsons(eds.),May,12-16.,2008,Estoril,Portugal, pp.607-614.

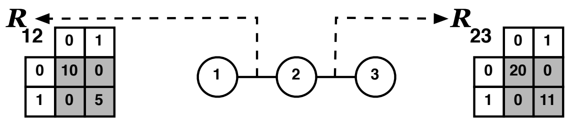


Figure 1: DCOP example

2. BACKGROUND

2.1 DCOP

A DCOP is a set of variables (one per agent) $\mathcal{N} := \{1, \ldots, n\}$ 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 joint action (or assignment) of a subgroup of agents $S \subset N$ by $a_S := \times_{i \in S} a_i \in \mathcal{A}_S$ where $\mathcal{A}_S := \times_{i \in S} \mathcal{A}_i$ and the joint action of the multi-agent team by $a = [a_1 \cdots a_n]$. (Since we assume each agent controls a single variable, we will use the terms "agent" and "variable" interchangeably.)

Valued constraints exist on various subsets $S \subset N$ of DCOP variables. 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_S. We refer to these subsets S as "constraints" and the functions $R_S(\cdot)$ as "reward functions". A DCOP can be depicted graphically with each node representing an agent and each edge representing a constraint (or hyper-edge in the case of constraints on more than two agents). Given that DCOPs are motivated by agents' privacy and communication overheads, non-neighboring agents in DCOPs do not communicate with (or directly reveal their values to) each other. The solution quality for a particular complete assignment a, $R(a)$, is the sum of the rewards for that assignment from all constraints (captured in the set denoted by θ) in the DCOP:$R(a) = \sum_{S \in \theta} R_S(a_S)$.

Example 1: Figure 1 shows a binary DCOP in which agents choose actions from the domain $\{0, 1\}$, with rewards shown for the two constraints $S_{1,2} = \{1, 2\}$ and $S_{2,3} = \{2, 3\}$. If both agents 2 and 3 choose the value 1, the team gets a reward of 11. If agent 1 now chooses 1 as well, the total solution quality of this complete assignment is 16.

2.2 Deviating Groups and k-optimality

For two assignments, a and $\tilde{a}$, the *deviating group* $D(a, \tilde{a}) := \{i \in \mathcal{N} : a_i \neq \tilde{a}_i\}$, is the set of agents whose assignment in $\tilde{a}$ differs from in a. In Figure 1, for assignments [1 1 1] and [0 1 0], the deviating group $D([1\ 1\ 1], [0\ 1\ 0]) = \{1, 3\}$. The *distance* $d(a, \tilde{a})$ is the cardinality of $D(a, \tilde{a})$. The *relative reward* of a with respect to $\tilde{a}$ is

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

Here θ is the set of all constraints, S is a particular constraint, R_S is the reward for that constraint and $R(a)$ is the total reward for the assignment a. Only constraints involving deviating agents need be considered, since the other rewards remain unchanged between assignments.

An assignment a is *k-optimal* if $\Delta(a, \tilde{a}) \geq 0\ \forall \tilde{a}$ where $d(a, \tilde{a}) \leq k$. In a k-optimal, no subgroup of cardinality $\leq k$ can improve the team reward by changing their assignment. In Figure 1, the assignment $a = [1\ 1\ 1]$ has a reward of 16 and is 1-optimal because deviation by any single agent reduces the team reward. If agent 1 changes value from 1 to 0, the reward on $S_{1,2}$ decreases from 5 to 0. Similar decreases occur if agents 2 or 3 individually switch values. However, [1 1 1] is not 2-optimal because if $\{2, 3\}$ switched to [1 0 0], the team reward would increase from 16 to 20. The global optimal, $a^* = [0\ 0\ 0]$ is k-optimal for all k.

3. QUALITY GUARANTEES

A key proven property of k-optimal solutions is the lower bound on solution quality :- assuming no knowledge of reward structure except for that they are non-negative and the lack of hard constraints, [8] provide bounds on the worst case quality of a k-optimal solution as a fraction of the global optimal. For example, for a completely connected binary DCOP graph of 6 nodes, assuming non-negative non-hard constraints, a 4-optimal solution (i.e. $k = 4$) is guaranteed to provide a solution quality of within $\frac{3}{7}$ of the global optimal. It is now crucial to improve upon these bounds for applications in newer domains.

3.1 Assumptions on Reward Structure

Previous work on k-optimal quality guarantees failed to consider the case where we may have some prior knowledge of the reward structure. Yet in many domains we do have some such knowledge. In particular, we may know *a priori* that the minimum reward on any constraint is a certain fraction β of the maximum reward on any constraint. For example, if we consider sensor agent networks[13], we may know the maximum and minimum reward possible for scanning any region covered by the network, and thus know that the minimum reward is a fraction β of the maximum (β is the ratio of the least minimum reward to the maximum reward). This prior knowledge already allows us to conclude that any k-optimal assignment is guaranteed to be at least a fraction β of the global optimal. However, can we provide a guarantee for a k-optimal assignment that is better than both β and the guarantee provided in [8]? The following answers the question in the affirmative.

PROPOSITION 1. *For any DCOP of n agents, with constraint arity of m, where all constraint rewards are non-negative, where a^* is the globally optimal solution, and where β is the least minimum reward to maximum reward ratio among all constraints, then, for any k-optimal assignment a where $m \leq k < n$,*

$$R(a) \geq \frac{\binom{n-m}{k-m} + \beta \sum_{i=1}^{m-1} \binom{m}{i} \binom{n-m}{k-i}}{\binom{n}{k} - \binom{n-m}{k}} R(a^*).$$

Proof: Given the k-optimal assignment a and the global optimal a^*, let the set $\hat{A}^{a,k}$ contain all assignments $\hat{a}$ where exactly k agents have switched from their values in a to their values in a^*. Since a is k-optimal, $R(a) \geq R(\hat{a}), \forall \hat{a} \in \hat{A}^{a,k}$.

For any assignment $\hat{a} \in \hat{A}^{a,k}$, the DCOP constraints θ can be divided into three sets: $\theta_1, \theta_2, \theta_3$. The set $\theta_1(a, \hat{a})$ contains constraints where all agents in $\hat{a}$ have deviated from their values in a; $\theta_2(a, \hat{a})$ contains constraints where no agents have deviated from their values in a; and $\theta_3(a, \hat{a})$ contains the remaining constraints.

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

where $D(a, \hat{a})$ represents the deviating group between a and $\hat{a}$. The sum of rewards of all assignments $\hat{a}$ in $\hat{A}^{a,k}$ is

$$\sum_{\hat{a} \in \hat{A}^{a,k}} R(\hat{a}) = \sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_2(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_3(a,\hat{a})} R_S(\hat{a}) \qquad (1)$$

Since a is k-optimal, $R(a) \geq R(\hat{a})$ for any $\hat{a}$ where at most k agents have deviated from the k-optimal assignment. Hence

$R(a) \geq R(\hat{a}), \forall \hat{a} \in \hat{A}^{a,k}$. Therefore,

$$|\hat{A}^{a,k}| R(a) \geq \sum_{\hat{a} \in \hat{A}^{a,k}} R(\hat{a}) \tag{2}$$

Thus to obtain a bound on $R(a)$, we will establish the value of the right hand side of the Equation 2. In [8], the first two terms of Equation 1 have been established to satisfy the inequalities:

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_1(a,\hat{a})} R_S(\hat{a}) \geq \binom{d(a,a^*) - m}{k - m} R(a^*) \tag{3}$$

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_2(a,\hat{a})} R_S(\hat{a}) \geq \binom{d(a,a^*) - m}{k} R(a) \tag{4}$$

However, [8] assumed that the third term in the Equation 1 was zero, i.e. no assumptions were made about reward structure yielding a pessimistic bound.

Using β, we will now express the third term in Equation 1 in terms of $R(a^*)$. Let $\Gamma_3^{a,i,k}(S)$ denote the set of all $\hat{a} \in \hat{A}^{a,k}$ where m-ary constraint $S \in \theta_3(a,\hat{a})$, and exactly i agents in S have deviated from the k-optimal solution (note that $1 \leq i \leq m-1$, else if $i = 0, S \in \theta_2$ and if $i = m, S \in \theta_1$). Thus the third term of Equation 1 can be written as

$$\sum_{\hat{a} \in \hat{A}^{a,k}} \sum_{S \in \theta_3(a,\hat{a})} R_S(\hat{a}) = \sum_{S \in \theta} \sum_{i=1}^{m-1} \sum_{\hat{a} \in \Gamma_3^{a,i,k}(S)} R_S(\hat{a}_S) \tag{5}$$

We now estimate the size of set $\Gamma_3^{a,i,k}$, so that we can compute $\sum_{\hat{a} \in \Gamma_3^{a,i,k}} R_S(\hat{a}_S)$. For any assignment $\hat{a} \in \Gamma_3^{a,i,k}(S)$, there are exactly i agents deviating in S. We can choose the i deviating agents within constraint S in $\binom{m}{i}$ ways, because the arity of S is known to be m. This means that we have now assigned a value to all these m agents of assignment $\hat{a}$, with exactly i of them chosen to deviate and rest $m-i$ as to not deviate. However, in our DCOP of n agents, there have to be k deviating agents because $D(a,\hat{a}) = k$ by design. Therefore, $n-m$ agents remain from which we have to choose the remaining of the $k-i$ deviating agents. This can be done in $\binom{n-m}{k-i}$ ways. Therefore, the number of assignments in $\Gamma_3^{a,i,k}(S)$ is $\binom{m}{i}\binom{n-m}{k-i}$. Therefore,

$$\sum_{S \in \theta} \sum_{i=1}^{m-1} \sum_{S \in \Gamma_3^{a,i,k}} R_S(\hat{a}_S) = \sum_{S \in \theta} \sum_{i=1}^{m-1} \binom{m}{i}\binom{n-m}{k-i} R_S(\hat{a}_S)$$

Now, we know that for this constraint, $R_S(a_S) \geq \beta R_S(a_S^*)$.

$$\sum_{S \in \theta} \sum_{i=1}^{m-1} \sum_{S \in \Gamma_3^{a,i,k}} R_S(\hat{a}_S) \geq \beta \sum_{S \in \theta} \sum_{i=1}^{m-1} \binom{m}{i}\binom{n-m}{k-i} R_S(a_S^*)$$

$$\geq \beta \sum_{i=1}^{m-1} \binom{m}{i}\binom{n-m}{k-i} R(a^*) \ \ [\text{since} \sum_{S \in \theta} R_S(a_S^*) = R(a^*)] \tag{6}$$

Substituting terms from Equations 1, 3, 4 and 6 in Equation 2,

$$\binom{d(a,a^*)}{k} R(a) \geq \binom{d(a,a^*) - m}{k - m} R(a^*) + \binom{d(a,a^*) - m}{k} R(a)$$
$$+ \beta \sum_{i=1}^{m-1} \binom{m}{i}\binom{n-m}{k-i} R(a^*)$$

which is minimized when $d(a,a^*) = n$, representing the case when a has no agent assignments in common with a^*. Therefore,

$$R(a) \geq \frac{\binom{n-m}{k-m} R(a^*) + \binom{n-m}{k} R(a) + \beta \sum_{i=1}^{m-1} \binom{m}{i}\binom{n-m}{k-i} R(a^*)}{\binom{n}{k}} \tag{7}$$

which on solving proves the Proposition 1 ■

Example 2: We focus on an example from [8] for illustration, which considers a DCOP with five agents ($n = 5$) numbered 1 to 5, with domains of {0,1}. Suppose that this DCOP is a fully connected binary DCOP ($m = 2$) with non-negative constraints between every pair of agents. We are to look for guarantees for a 3-optimum for this graph, and we are given a $\beta = 0.5$. In [8], which does not take into account β, the 3-optimum is only guaranteed to be 1/3 of the global optimal, i.e. $R(a) \geq \frac{1}{3}R(a^*)$. Only taking into account $\beta = 0.5$, we can only guarantee $R(a) \geq \frac{1}{2}R(a^*)$. We now show the result of Proposition 1.

To illustrate how the proof of the Proposition 1 works, 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 $d(a,a^*) = 5$. We now consider the set $\hat{A}^{a,k}$ discussed above. We can show that $\hat{A}^{a,k}$ contains $\binom{d(a,a^*)}{k} = 10$ assignments, which are: [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]. These are all the assignments that deviate from a by 3 actions and take the value from the optimal solution in those deviations. Given this $|\hat{A}^{a,k}|$, from Equations 3 and 4 we can conclude that the first two terms of Equation 1 are greater than $\binom{5-2}{3-2}R(a^*)$ and $\binom{5-2}{3}R(a)$ respectively.

Now focus on the third term of Equation 1. Here we only focus on the constraints from $\theta_3(a,\hat{a})$, and calculate the number of assignments in the set $\hat{A}^{a,k}$ where exactly one agent in a particular constraint deviates from the k-optimal solution a, i.e. the size of $\Gamma_3^{a,1,3}$. For our binary DCOP this value is $2 \cdot \binom{n-2}{k-1} = 6$. For example, for $S = \{1,2\}, \hat{a_1} = a_1^* = 1$ and $a_2 = 0$ for $\hat{a} = [1\ 0\ 1\ 1\ 0]$, $[1\ 0\ 1\ 0\ 1]$, and $[1\ 0\ 0\ 1\ 1]$. Similarly, $a_1 = 0$ and $\hat{a_2} = a_2^* = 1$ for $\hat{a} = [0\ 1\ 1\ 1\ 0]$, $[0\ 1\ 1\ 0\ 1]$, and $[0\ 1\ 0\ 1\ 1]$. Since we assume the value of β to be 0.5, by combining these constraints in $\theta_3(a,\hat{a})$, the third term of Equation 1 $\geq 0.5 \cdot 6R(a^*) = 3R(a^*)$

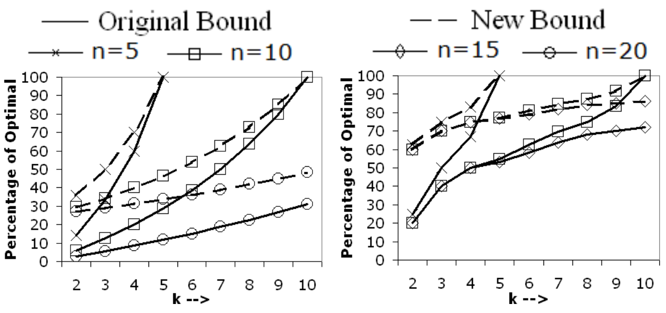
Thus Equation 7 gives us $10 \cdot R(a) \geq 3 \cdot R(a^*) + 1 \cdot R(a) + 0.5 \cdot 6 \cdot R(a^*)$, and hence $R(a) \geq \frac{3+3}{10-1}R(a^*) = \frac{2}{3}R(a^*)$. We have thus improved upon the bound provided by either of the earlier two methods. □

While Proposition 1 provides a lower bound on solution quality in constant time, it ignores the DCOP graph structure. To provide guarantees taking into account DCOP graph structure, we use the linear fractional program (LFP) technique, earlier used in [8]. However, β changes the nature of this LFP. In particular, for each m-ary constraint S in the DCOP, we have multiple variables in the LFP. The first variable represents the reward on S in the k-optimal solution, that is when there are no deviating agents in S. The next set of m variables represents the reward on S when exactly one agent in S deviates, and each variable corresponds to the deviation of a different agent in S. The next $\binom{m}{2}$ variables represent the reward on S when exactly two agents in S deviate. Similary, we have variables corresponding to the deviation of every proper subset of agents in S. The final variable represents the reward on S in the global optimal solution, i.e. considering the case when all agents in S deviate. The objective of the LFP is to find the set of rewards to bring about the worst possible k-optimal solution to get a lower bound. Thus the objective function is to minimize

$$\frac{R(a)}{R(a^*)} = \frac{\sum_{S \in \theta} R_S(a)}{\sum_{S \in \theta} R_S(a^*)}$$

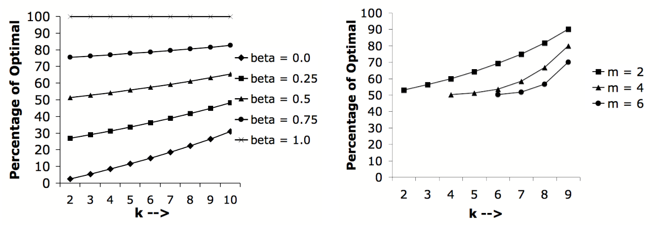
subject to the LFP constraints that (i) $\forall \tilde{a} \in \tilde{A}, R(a) - R(\tilde{a}) \geq 0$, where $\tilde{A}$ contains any assignment $\tilde{a}$ such that $d(a,\tilde{a}) \leq k$ and (ii) all rewards on S obey our β ratio Note that the reward on a^*, a and $\tilde{a}$ can be decomposed into sum of a combination of variables mentioned above.

We will now provide results for Proposition 1 and the LFP. Fig-



(a) Fully Connected Graph (b) Binary Tree

Figure 2: Quality guarantees for k-optima under Reward Structure Assumptions



(a) Variation with β (b) Variation with m

Figure 3: Quality guarantees for m-ary graphs with variation in β

ure 2 shows improved quality guarantees due to β i.e. what is the worst case quality a k-optimal solution obtains given β, as a fraction of the global optimum. Figure 2(a) shows the improvement in bounds for fully connected binary DCOP graphs if we assume $\beta = 0.5$ (i.e. the minimum reward for any constraint is at least half the maximum reward on the corresponding constraint). The bounds were calculated using Proposition 1. n refers to the number of agents in the DCOP graph. On the x-axis is the variation in the value of k and on the y-axis is the percentage of optimal determined by the proposition. The dashed lines show the bounds achieved by Proposition 1 whereas the solid lines show the original bounds as presented in [8]. For example, for $k = 3$ and $n = 10$, the original bound was 12.5% whereas the new bound is 56.25%. Similarly, Figure 2(b) shows the improvement of bounds for binary DCOP trees. The bounds were calculated using the LFP. On the x-axis is the variation in k and on the y-axis is the percentage of global optimal obtained. Again, the dashed lines show the bounds achieved by Proposition 1 whereas the solid lines show the original bounds as presented in [8]. Since we have assumed β to be 0.5, all new bounds start from a minimum of 50% of the global optimal. However, they don't add 50% directly to the original bound. For example, for $k = 4$ and $n = 10$, the original bound was 50.0% whereas the new bound is 75.0%. From the graphs, we can conclude that the improvement observed is significant.

Figure 3(a) shows variation in the percentage of optimal with variation and k and β for a fully connected binary DCOP graph with 20 nodes. The x-axis represents the value of k where as the y-axis represents the percentage of optimal. The graph shows that we get better and better bounds as we increase the value of β, till we reach $\beta = 1$ when the reward of the k-optimal reward is equal to the reward of the globally optimal solution. For example, when k=5 the solution quality is $11.76\%, 33.82\%$ and 77.94% when β is $0, 0.25$ and 0.75 respectively. Similarly, Figure 3(b) shows the percentage of optimality for a fully connected graphs of 20 nodes with $\beta = 0.5$ when the constraint arities are $2, 4$ and 6. x-axis represents the value of k and the y-axis represents the percentage of optimal. For example, for $k = 1$, the k-optimal solution is 69.23% optimal in case of binary constraints and 53.59% optimal for a constraint arity of 4.

3.2 DCOPs with Hard Constraints

Previous work[8] focussed exclusively on DCOPs with non-negative rewards and presented quality guarantees for them. When we have no hard constraints, by adding a sufficiently high positive number to all rewards, DCOPs meet the requirement of non-negative rewards. However, no guarantees are available for DCOPs with hard constraints yet and in many domains, hard constraints exist — these are constraints where at least one combination of values incurs a reward of $-\infty$. For example, in meeting scheduling, if two users disagree on their meeting time the reward is $-\infty$.

This section presents quality guarantees in the presence of such hard constraints. To obtain these guarantees we assume that there exists a solution to the DCOP that does not violate any hard constraints (i.e. the globally optimal solution is feasible, else there can be no guarantee). We also assume that we know *a priori* which constraints in the DCOP are hard constraints, but not which values satisfy them. Finally, we wish to avoid problems where the k-optimal is 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. This would again preclude our providing any guarantees with respect to the global optimum. 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). In such DCOPs, no k-optimum would be infeasible (violating hard constraint). In the following we provide an illustrative proof of quality guarantees with hard constraints on specific graph structures followed by an LFP for guarantees for arbitrary graphs.

PROPOSITION 2. *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 $0 < h < n-1$.*

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

Proof: We define as before the set $\hat{A}^{a,k}$ such that exactly k agents deviate from the k-optimal solution but now require all the deviating agents to be connected. We will again define θ_1, θ_2 and θ_3 as defined in proof of Proposition 1. Now we will consider only the subset of $\hat{A}^{a,k}$ in which $\forall S_{hard}, S_{hard} \in \theta_1$($S_{hard}$ are the hard constraints). Let us call this set $\hat{A}^{a,h,k}$. Now,

$$R(a) \geq \frac{\sum_{\hat{a} \in \hat{A}^{a,h,k}} R(\hat{a})}{|A^{a,h,k}|} \tag{8}$$

because $R(a) \geq R(\hat{a}) \, \forall \hat{a} \in \hat{A}^{a,h,k}$.

We will now try to establish the numerator and the denominator separately. Let us first consider the denominator. By definition of $|A^{a,h,k}|$, it is the total number of possible assignments when exactly k agents are deviating and $S_{hard} \in \theta_1(a, \hat{a})$. Since $S_{hard} \in \theta_1(a, \hat{a})$ and all constraints connect to the center agent of the star graph, therefore we have already decided that the center agent and the other h agents with hard-constraints are deviating. Hence, we have to chose the rest of $(k - (h+1))$ deviating agents from $(n - (h+1))$ remaining agents. Therefore,

$$|\hat{A}^{a,h,k}| = \binom{n-h-1}{k-h-1} \quad (9)$$

Let us now consider the numerator of Equation 8. The global reward of all assignments $\hat{a} \in \hat{A}^{a,h,k}$ is:

$$\sum_{\hat{a}\in\hat{A}^{a,h,k}} R(\hat{a}) = \sum_{\hat{a}\in\hat{A}^{a,h,k}} \sum_{S\in\theta_1(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a}\in\hat{A}^{a,h,k}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) + \sum_{\hat{a}\in\hat{A}^{a,h,k}} \sum_{S\in\theta_3(a,\hat{a})} R_S(\hat{a}). \quad (10)$$

Now we will determine the bounds for each of these terms. Note that in our definition of $|\hat{A}^{a,h,k}|$, we have fixed the values of all agents linked to hard constraints. Hence, when we choose any non-hard constraint S, if $S \in \theta_1(a, \hat{a})$, one more agent is set to deviate and hence there are $\binom{n-h-2}{k-h-2}$ possible assignments. Therefore,

$$\sum_{\hat{a}\in\hat{A}^{a,h,k}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) = \binom{n-h-2}{k-h-2} R(a^*) \quad (11)$$

No S can be part of $\theta_2(a, \hat{a})$ because the center agent deviates, and all constraints connect to the center agent. Therefore,

$$\sum_{\hat{a}\in\hat{A}^{a,h,k}} \sum_{S\in\theta_2(a,\hat{a})} R_S(\hat{a}) = 0.R(a) \quad (12)$$

Finally, since no hard constraint is a part of the set θ_3,

$$\sum_{\hat{a}\in\hat{A}^{a,h,k}} \sum_{S\in\theta_3(a,\hat{a})} R_S(\hat{a}) \geq 0. \quad (13)$$

Hence, following Equation 2,

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

which on solving proves Proposition 2. ■

Example 3: Consider a star-shaped binary DCOP involving six agents with one hard constraint. For a 4-optimal solution, the total number of assignments in $\hat{A}^{a,h,k}$ is $\binom{6-1-1}{4-1-1} = 6$ assignments as given by Equation 9. Similarly, the sum of constrains defined by the first term of Equation 10 is greater than or equal to $\binom{6-1-2}{4-1-2} R(a^*) = 3R(a^*)$ as given by Equation 11. Thus, with Proposition 2, we get that for this star graph, $R(a) \geq \frac{3R(a^*)+0\cdot R(a)}{6} = \frac{1}{2}R(a^*)$.□

For other graph types, we modify the LFP mentioned earlier. The LFP remains a minimization of $\frac{R(a)}{R(a^*)}$ such that $\forall \tilde{a} \in \tilde{A}, R(a) - R(\tilde{a}) \geq 0$, given $\tilde{A}$, where $\tilde{A}$ contains any assignment $\tilde{a}$ such that $d(a, \tilde{a}) \leq k$, however with key differences. Firstly, now the LFP variables corresponding to violated hard constraints are allowed to take infinitely large negative rewards, whereas previously they were always non-negative. Also, the β conditions are removed.

Figure 5 shows quality guarantees in the presence of hard constraints for fully connected, ring and star DCOP 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 x-axis of the graph represents the value of k where as the y-axis shows the ratio of the k-optimal to the globally optimal solution. 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 2 was used; while for the others, the LFP method was used. For example, for *star* graphs with n = 5, for k = 4, and 2 hard constraints, the $R(a)$ to $R(a^*)$ ratio is 0.5, where as for a *ring* graph with 10 agents, k = 5, and 4 hard constraints, it is 0.33.

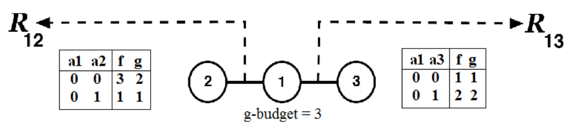


Figure 4: Multiply Constrained DCOP example

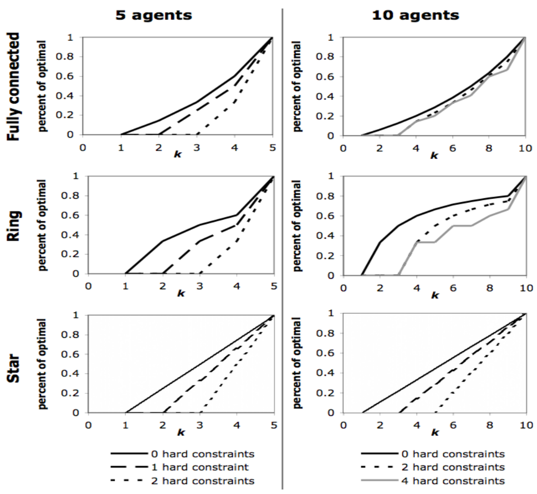


Figure 5: Quality guarantees for k-optima in DCOPs containing hard constraints.

3.3 Guarantees in multiply constrained DCOP

These LFP guarantees for k-optima in standard DCOPs with hard constraints also apply to multiply-constrained DCOPs (MC-DCOP) where multiple agents are connected via a single constraint. MC-DCOP extends DCOP to address hard resource constraints[1], and is shown to be valuable in addressing domains where agents may have resource constraints such as travel budgets or overtime constraints. In MC-DCOP, the DCOP reward function remains (henceforth referred to as "f") and a new cost function g_{ij} is added on a subset of agent i's links along with a g-budget G_i which the accumulated g-cost must not exceed. A g-constraint on an agent with $m-1$ neighbors is a shorthand expression for the more general notion of a hard m-ary constraint between the agent and all its neighbors (where any assignment where the g-budget is exceeded is considered infeasible). While the agent has full knowledge of this m-ary hard constraint, its neighbors may have no knowledge of all the neighbors involved in the constraint. Figure 4 shows an example MC-DCOP, with a g-budget of 3 on agent 1. Each link has both a f-reward and a g-cost. When agents 1,2 and 3 take values 0,0 and 1 respectively, the assignment yields an optimal f-reward 4, but the g-cost is 4 which violates agent 1's g-budget.

For a key illustration of application of LFP to MC-DCOPs, Figure 6 shows quality guarantees for a MC-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. This methodology was chosen so that k-optimal solutions would always be feasible. The LFP method was used to calculate the guarantees. Figure 6 shows guarantees for up to 4 agents with g-constraints as k increases. On the x-axis of the graph is the value of k where as the y-axis represents the percentage of the optimal solution. For example, with 30 agents arranged in a ring, k=6 and 3 g-constraints, the percentage of optimal guaran-

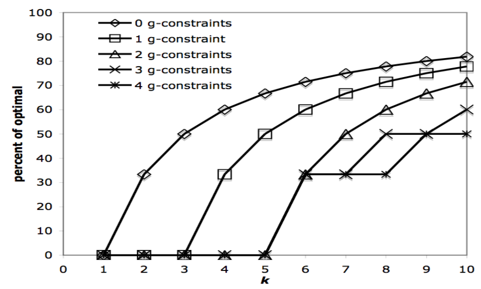


Figure 6: Quality guarantees for k-optima in a multiply-constrained ring DCOP.

teed is 33.33%. For the cases of 5 to 10 agents with g-constraints the guarantee was the same as for 4 agents with g-constraints.

4. HARD CONSTRAINT ALGORITHMS

Unfortunately, despite the importance of hard constraints and in particular the importance of MC-DCOPs [1] in representing resource constraints, current k-optimal algorithms fail to address MC-DCOPs. Consider for example the k-optimal algorithms MGM-1 or MGM-2 [13, 5]. In MGM-1, starting from an initial random assignment, each agent sends a message to all its neighbors indicating the maximum reward gain it could accrue by changing values in the current context. An agent changes values if its gain is larger than those of its neighbors; agents keep changing values in this fashion until convergence. However to apply MGM-1 to MC-DCOP, for an agent with $m-1$ neighbors, we will need to add (m)-ary constraint so agents can convey maximum gain messages to others taking into account g-constraint violations. This approach fails however: (i) sprouting large number of (m)-ary constraints adds significant communication overheads; (ii) agents lose privacy by communicating with new neighbors hoisted upon them due to the new (m)-ary constraints, e.g. this approach would force agent 2 and agent 3 in Figure 4 to communicate values; (iii) parallelism degrades as only one agent in a (m)-ary constraint can change value at a time. These problems are exacerbated in algorithms with higher k.

Multiply-Constrained-MGM-k (MC-MGM-k) are a new set of algorithm that do not add any explicit (m)-ary constraint; agents only communicate with their neighbors avoiding problems of privacy violations or communication overheads. The key insight is to maintain an invariant of g-constraint satisfaction. This invariant is not guaranteed *a priori* before each agent proposes its change of values; instead, an agent with g-budget heuristically blocks others from making changes that violate its g-budget. MC-MGM-k algorithms are designed to terminate in *mc-k-optima*, i.e. an assignment where if k (or fewer) agents deviate they either violate g-constraints or degrade (or do not improve) solution quality.

For MC-MGM-1, agents iteratively run the code shown in Algorithm 1, proceeding in rounds of changes to values. Agents initialize at a dummy value (newly added to their domain) a', which has a g-cost = 0 to start at a satisfying assignment. Agents send value messages to their neighbors informing them of their current value and available-g: total g-budget minus that consumed by all other neighbors (line 1 in Algorithm 1). Upon receipt, each agent calculates its effective domain by removing any values that would violate its neighbors' available-g's and computes the maximum local reward in the effective domain (lines 3-4). It sends a message to all neighbors proposing its move and listing the gain (lines 5-7). Upon receipt of gain messages, agents check to see if its neighbors' proposed moves violate its g-constraint (line 8). An example is shown in Figure 4, where if agent 1 has the value 0 and agent

Algorithm 1 MC-MGM-1 (allNeighbors, currentValue)

```
1: SendValueMessages(allNeighbors, currentValue, available-g-budget)
2: currentContext = GetValueMessages(allNeighbors)
3: **for** newValue in EffectiveDomain(currentContext) **do**
4:     [gain,newValue] = BestUnilateralGain(currentContext)
5: **if** gain $>$ 0 **then**
6:     SendGainMessage(allNeighbors,gain, newValue)
7: neighborGains = ReceiveGainMessages(allNeighbors, NeighborValues)
8: **if** GConstraintViolated(newContext) **then**
9:     n = SelectNeighborsToBlock()
10:    SendBlockMessages(n)
11: **if** gain $>$ max(neighborGains) and !ReceivedBlockMessage() **then**
12:    currentValue = newValue
```

2 and agent 3 propose taking on 1, neither move individually violates agent 1's g-constraint but together they do. If this situation is detected, an agent sends a blocking message to a subset of its neighbors (lines 9-10). If an agent receives a block message, it will not move in the current round. The agents with the highest gain who don't receive blocking messages move. The rounds repeat until no agents propose moves.

MC-MGM-2 operates much like MC-MGM-1 but in addition to individual moves, it allows joint moves(coordinated changes of value) between pairs of agents using the same ideas of *Effective-Domain* and blocking messages to avoid violation of g-constraints. In particular, each round, MC-MGM-2 randomly decides which agents will be *offerers* and which *receivers*. Each offerer chooses a random neighbor and sends a message proposing joint moves that yield a gain in the offerer's local utility. Each receiver then calculates the joint utility gain for the proposed moves. If the joint gain is positive, the receiver will send a message to the offerer committing to the move; otherwise, neither agent is committed. Uncommitted agents consider unilateral moves as in MC-MGM-1. The agent or pair of agents with the highest local gain will move if not blocked.

In MC-MGM 1 and 2, different heuristics are available to select which neighbor to block. We focus on three representative heuristics: (i) *Monotonic*: agent i randomly selects one or more neighbors to block and blocked neighbor does not change value; (ii) *Random Reset*: agent i randomly selects one or more neighbors to block and blocked neighbor reset its value to dummy value a'; (iii) *Self Reset*: agent i sends no blocking messages and resets itself to a'.

We now prove MC-MGM 1 and 2 actually terminate in *mc-1-optima(mc-2-optima)*. Recall the notation from section 2. In addition, let us define $a^{(n)}$ as the assignment of values to agents in the DCOP due to MC-MGM at the beginning of the n^{th} round of the algorithm. Let the global utility of this assignment be $U(a^{(n)})$. Define $Gain_i$ as the change in local utility of agent i due to a unilateral change in its value from a_i to $\hat{a_i}$, given fixed context a_{-i}. In computing $Gain_i$, agent i only considers its *EffectiveDomain*, i.e. values that do not violate its own g-constraint, or the available-g sent by it's neighbors.

PROPOSITION 3. *When running MC-MGM-1 with the monotonic heuristic, the global utility $U(a^{(n)})$ never decreases.*

Proof: There are two separate cases to handle. In the first case, no block messages are issued (by using *EffectiveDomain*, agents managed to avoid violating any other agent's g-budget). Here, if agent i intends to modify its value in round r, then (i) $Gain_i > Gain_j, \forall j \in Neighbors(i)$; and (ii) $Gain_i > 0$. Hence agent i's neighbors will not modify their values in round r. So, when agent i changes its value, $Gain_i$ will add to the global utility $U(a^{(r)})$.

If multiple agents change values simultaneously, they must be non-neighbors, and each of their gains adds to $U(a^{(r)})$.

In the second case, despite using *EffectiveDomain*, a blocking message was issued within a round r, which will block an agent i from changing value, i.e. agent i will realize a gain of 0, which does not decrease $U(a^{(r)})$. Furthermore, since MC-MGM maintains as an invariant that no agents' g-constraint is violated at the end of any round of execution, $U(a^{(r)})$ never decreases due to violation of g-constraints. ■

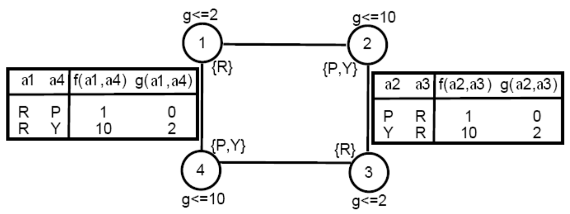


Figure 7: A Deadlock example for MC-MGM-1

We can similarly show monotonicity of MC-MGM-2. Given that MC-MGM sends out blocking messages, a cycle of blocking messages may create a deadlock. Figure 7 shows an example of deadlock in MC-MGM-1. There are four agents, with domains, f reward and g cost as shown (identical f and g functions exist between agent 1 and 2, and between agents 3 and 4). After all agents are initialized to a dummy value 0, agent 1 and agent 3 switch from 0 to value R. Now, agents 2 and 4 propose switching to Y, which gives them each a gain of 20. Also each of agent 2 and agent 4 individually perceive that the change to Y is under the available-g of 2 of agent 1 and agent 3. However, if both agents 2 and 4 switched to the value Y, agent 1 and agent 3's budgets would be violated; so they must send blocking messages. If agent 1 randomly selected agent 2 to block and agent 3 randomly selected agent 4 neither agent 2 or 4 could change values, i.e. a deadlock situation. Fortunately, since the agents being blocked are randomly selected (in the monotonic heuristic), remaining in deadlock indefinitely is impossible. Suppose agents can enter deadlock with probability p, where $p \in [0, 1)$. After n rounds of execution, the probability of remaining in deadlock is p^n. Since execution continues until there are no longer any proposal messages being sent, n approaches ∞ and p^n approaches 0.

PROPOSITION 4. *Given that deadlocks are resolved using randomization, MC-MGM-k algorithms with the monotonic heuristic will terminate at an mc-k-optimal solution.*

Proof: Since by previous Proposition, MC-MGM-k monotonically increases global utility $U(a^{(n)})$, and there is finite globally optimal solution, MC-MGM-k cannot keep increasing $U(a^{(n)})$ forever. Thus, assuming it eventually resolves any deadlocks it enters, MC-MGM-k will terminate. Termination in MC-MGM-k occurs when no k agents are able to propose a move where $Gain_i > 0$ and no g-constraints are violated after the move. This situation is the definition of an mc-k-optimal. ■

5. EXPERIMENTAL RESULTS

We compare MC-MGM-1 and MC-MGM-2 for MC-DCOPs based on solution quality measured by the sum of all f-constraints, and on the runtime based on message cycles. A message cycle is one messaging exchange between agents, and is the traditional metric in DCOP literature for algorithmic efficiency [7, 6, 10]. In particular, we ran four sets of experiments on randomly generated 1000-agent graph coloring problems; this contrasts with complete DCOP algorithms [7, 6, 10] that can only handle 20-agent to 100-agent problems. Networks of agents with average link densities (number of edges per agent) of 3 and 6 were generated, with randomly generated f- and g-constraints. In each case the g-budget assigned to all agents was varied between 0 to 50, as shown on the x-axis. Results were averaged over 30 runs of the algorithms.

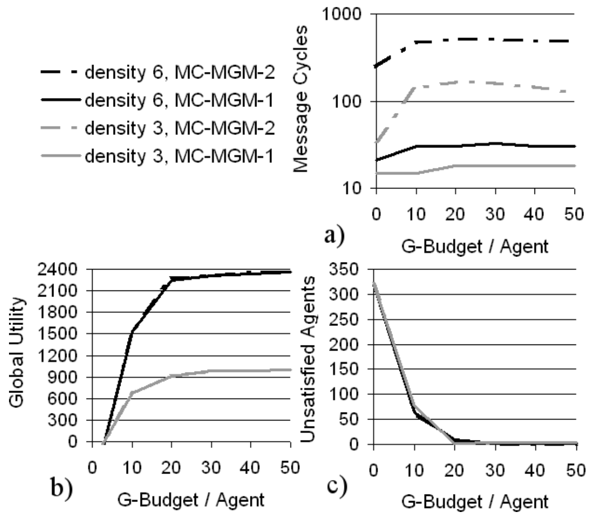


Figure 8: Comparing MC-MGM-1 and 2 for $|H| = 2$

The first experiment ran MC-MGM-1 and MC-MGM-2 using the monotonic heuristic on DCOPs where all of the g-constraints in the DCOP were only allowed on subgraphs of two agents i.e. the largest connected subgraphs H of the DCOP constraint graph that consisted of all the hard constraints were of size two ($|H| = 2$). In Figure 8a, we vary the available g-budget per agent along the x-axis, and show the run-time in cycles on a log-scale along the y-axis. For example, when g-budget per agent is 10, DCOP problems of constraint density 6 take an average of 30 message cycles to run using MC-MGM-1, or 483 using MC-MGM-2. *T*he key conclusion is this: compared to 1000s of cycles taken by complete algorithms attacking MC-DCOPs of 20-30 nodes[1], this is a 50-fold scale-up in number of nodes running with a 10-fold speedup. There is a loss in solution quality in incomplete algorithms, but if we cannot run the complete algorithm at all, then this loss is moot. The results also demonstrate that MC-MGM-1 runs between 2.3 and 9.5 times faster than MC-MGM-2 on problems with a link density of 3, and 11.6 to 17.1 faster when the link density is 6. Figure 8b plots the global utility of the final solution on the y-axis and shows that global utility varied by less than 2% between the algorithms for both densities. Finally, Figure 8c measures the number of agents (on the y-axis) which never left the dummy value a' due to blocking messages. The number of agents remaining at a' was identical for both algorithms below a g-budget of 30, however above that budget, MC-MGM-2 was able to find a solution which satisfied all agents, while MC-MGM-1 always left 2 agents unsatisfied.

The second and third experiments focused on DCOP graphs where g-constraints applied to larger subgraphs with $|H| \leq 3, 4$ and 5. Figures 9 and 10 show that the same differences between MC-MGM-1 and 2 are present and become more pronounced as $|H|$ increases. The utility of MC-MGM-2 is seen to be significantly higher.

The fourth experiment compared the performance of three blocking heuristics: monotonic, random reset and self reset, again with

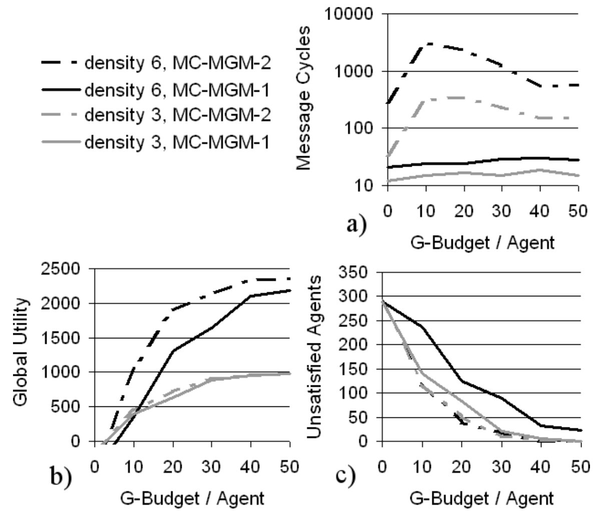


Figure 9: Comparing MC-MGM-1 and 2 for $|H| \leq 3$

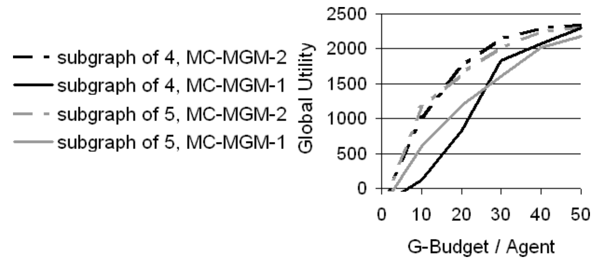


Figure 10: MC-MGM-1 and MC-MGM-2 for $|H| \leq 4$ or 5

the same x-axis and y-axis used in the first experiment for graphs in Figure 11 a and b. Figure 11a shows that random reset performed poorly for both densities; it reached our 6000 cycle cut-off point. The monotonic and self reset heuristics performed within the same range. The monotonic heuristic was more efficient for high g-budgets and self reset was more efficient for low ones. The solution quality results are shown in Figure 11b. The monotonic heuristic reaches a higher utility for higher g-budgets and self reset does better on looser g-budgets.

6. CONCLUSION AND RELATED WORK

DCOP is a major paradigm within cooperative multiagent systems for coordination, scheduling and task allocation in domains such as networks of sensors, unmanned air vehicles or software personal assistants. K-optimal algorithms provide a unique advance in locally optimal DCOP algorithms: these algorithms provide quality guarantees. Improving upon these bounds is now crucial to apply k-optimality to new domains. To that end, this paper provided: (i) lower-bounds on k-optima quality incorporating available prior knowledge of the ratio of minimum to maximum rewards on constraints; (ii) lower-bounds on k-optima quality for problems with hard constraints; and (iii) new k-optimal algorithms for solving multiply-constrained DCOPs, which provide resource bounds as hard constraints, along with experimental results for networks of more than 1000 agents with various connection densities.

We have already discussed related work, particularly weaknesses in prior work on k-optimality, at length[8]. Among globally optimal algorithms, [1] provide a complete MC-DCOP algorithm based on Adopt[7]. Other complete algorithms, OptAPO [6] and DPOP[10] have yet to be applied to MC-DCOPs, where agents face a global objective as well as local resource constraints. We have emphasized incomplete algorithms and quality guarantees including for MC-DCOPs. Nonetheless, research on complete and incomplete algorithms is complementary, and advancing both is essential for advancing multiagent systems.

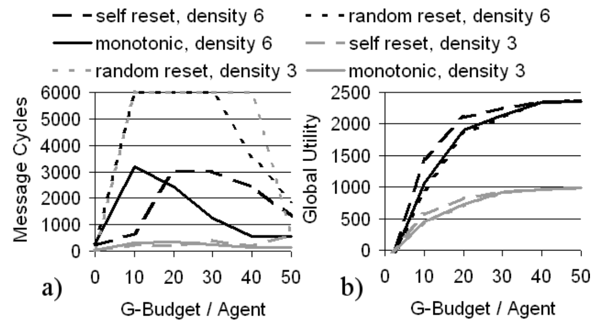


Figure 11: Comparing blocking heuristics: MC-MGM-2 with $|H| \leq 3$

ACKNOWLEDGEMENTS

This research was supported by DARPA STTR Contract W31P4Q-06-C-0410 via a subcontract from Perceptronics Inc.

7. REFERENCES

[1] E. Bowring, M. Tambe, and M. Yakoo. Multiply constrained distributed constraint optimization. In *AAMAS*, 2006.

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

[3] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. Kluwer, 2003.

[4] V. Lesser, C. Ortiz, and M. Tambe. *Distributed sensor nets: A multiagent perspective*. Kluwer Academic Publishers, 2003.

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

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

[7] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 2005.

[8] J. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization. In *IJCAI*, 2007.

[9] J. Pearce, M. Tambe, and R. Maheswaran. Solution sets for dcops and graphical games. In *AAMAS*, 2006.

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

[11] N. Vlassis, R. Elhorst, and J. R. Kok. Anytime algorithms for multiagent decision making using coordination graphs. In *SMC*, 2004.

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

[13] 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*.