Local Optimal Solutions for DCOP: New Criteria, Bound, and Algorithm

Zhengyu Yin, Christopher Kiekintveld, Atul Kumar, and Milind Tambe
Computer Science Department
University of Southern California, Los Angeles, CA, 90089
{zhengyuy, kiekintv, atulk, tambe}@usc.edu

ABSTRACT

Distributed constraint optimization (DCOP) is a popular formalism for modeling cooperative multi-agent systems. In large-scale networks, finding a global optimum using complete algorithms is often impractical, which leads to the study on incomplete algorithms. Traditionally incomplete algorithms can only find locally optimal solution with no quality guarantees. Recent work on k-*size-optimality* has established bounds on solution quality, but size is not the only criteria for forming local optimization groups. In addition, there is only one algorithm for computing solutions for arbitrary k and it is quite inefficient. We introduce t-*distance-optimality*, which offers an alternative way to specify optimization groups. We establish bounds for this criteria that are often tighter than those for k-optimality. We then introduce an asynchronous local search algorithm for t-*distance-optimality*. We implement and evaluate the algorithm for both t and k optimality that offer significant improvements over KOPT – the only existing algorithm for k-*size-optimality*. Our experiment shows t-*distance-optimality* converges more quickly and to better solutions than k-*size-optimality* in scale-free graphs, but k-*size-optimality* has advantages for random graphs.

1. INTRODUCTION

In various cooperative multi-agent domains, agents have limited information and can directly communicate with only a subset of other agents. Typically, in these domains, the utility generated by an individual action of one agent depends only on the actions of a set of nearby agents. Distributed constraint optimization (DCOP) is a popular formalism for modeling such cooperative multi-agent systems in which agents work together to optimize a global objective. The objective function can be decomposed into constraints with associated utility matrixes across local subsets. There are lots of applications of DCOP including multi-agent plan coordination [5], sensor networks [15], allocation of resources in peer-to-peer networks [6], and meeting scheduling [13].

Various complete algorithms have been developed for finding globally optimal solution to DCOPs, e.g. ADOPT [10], OptAPO [9], DPOP [13], and NCBB [4]. However, DCOP is NP-hard [10]. It is hard for complete algorithms to scale up because the computation burden might increase exponentially with the increasing number of variables. Therefore, researchers have been studying incomplete algorithms, in which agents normally form local groups with small size and optimize within these groups. Incomplete algorithm have better scalability and robustness in dynamic environments. Existing incomplete algorithms include MGM/DBA [12, 15] and DSA [7]. Recent studies with k-*size-optimality* by Pearce et al. [11] have begun to provide theoretical quality guarantees of locally optimal solutions with certain properties. k-*size-optimality* is characterized by the size of local groups, i.e. the minimum number of variables which must change their values to improve the overall solution quality.

Unfortunately, k-*size-optimality* suffers from several shortcomings. First, our theoretical analysis shows the quality lower bound for k-*size-optimality* is inversely related to the graph density. The worst case is complete graphs, where the lower bound for k-*size-optimality* decreases to $\frac{k-1}{2n-k-1}$ [11]. As n increases, this bound gets unacceptably low for some domains. Second, k-*size-optimality* considers all types of k-size groups, among which some may have nodes that are far apart from each other. In such groups, coordination to perform a joint move can be expensive, as messages have to be delivered to some nodes far away from the center node. Third, the number of possible k-size groups in the graph grows combinatorially with increasing values of k. Basically, any combination of k connected nodes can form a k-size group. The complexity of enumerating and optimizing all k-size groups makes it difficult to design efficient algorithms for k-*size-optimality* with $k \geq 3$. This explains why most local search algorithms have limited group size to 1 or 2. "KOPT" [8], recently introduced by Katagishi and Pearce, is the only known algorithm for k-*size-optimality* that supports arbitrary values of k. However it suffers from significant messaging overhead, particularly in highly dense graphs.

In this paper, we introduce a new locally optimal criteria – t-*distance-optimality* which has fixed number of local optimization groups defined by graph distance. We provide formal quality bounds for t-*distance-optimal* solutions with and without prior knowledge about graph structure, and show these bounds can yield significantly better guarantees than comparable k-*size-optimal* solutions. Furthermore, we present an asynchronous local search algorithm for t-*distance-optimality* based on standard lock/commit protocol, and show its significant improvements over KOPT in various metrics. Finally we conducted comprehensive experiments to evaluate tradeoffs between algorithms for t-*distance-optimality* and k-*size-optimality* considering a variety of metrics. Our experiments show t-*distance-optimality* converges more quickly and to better solutions than k-*size-optimality* in scale-free graphs, but k-*size-optimality* has advantages in random graphs.

2. BACKGROUND

Cite as: Local Optimal Solutions for DCOP: New Criteria, Bound, and Algorithm, Z. Yin, C. Kiekintveld, A. Kumar, and M. Tambe, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

2.1 DCOP Definition

A finite *distributed constraint optimization* (DCOP) problem comprises sets of variables $V := \{v_1, \ldots, v_n\}$ and constraints $C := \{c_1, \ldots, c_m\}$. Variables have finite domains and are each controlled by a decision-making agent (for convenience we assume one variable per agent). A joint assignment for all variables is given by $A := \{a_1, \ldots, a_n\}$. We follow the convention in the literature and consider only binary constraints. For some pair of variables (v_i, v_j), a constraint c defines a real-valued reward for all possible joint assignments, $c(a_i, a_j)$. The reward function $R(\cdot)$ of an assignment A is defined as the sum of rewards, $R(A) = \sum_{c \in C} c(a_i, a_j)$. The agents' objective is to find an assignment A^* that maximizes the reward function, i.e. $A^* = \arg\max_A R(A)$. The *constraint graph* has a node for each variable and an edge for each constraint. In the sequel, we use the terms node, variable and agent interchangeably. Agents initially know only their own constraints, and can communicate only with neighbors in the constraint graph.

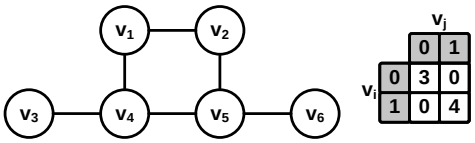


Figure 1: An example DCOP with six binary variables. Each constraint has the same reward table.

DCOP shown in figure 1 contains 6 variables, and 6 constraints with the same reward matrix. The optimal assignment of this problem is $A = \{1, 1, 1, 1, 1, 1\}$ with a reward $R(A) = 24$.

2.2 k-Size Optimality

Pearce et. al. recently introduced *k-size-optimality* (as "k-optimality" in the original work) as a local optimality criteria that offers theoretical guarantees on solution quality. In their approach, agents form groups of one or more agents until no group of k or fewer agents can possibly improve the DCOP solution. This type of local optimum is defined as *k-size-optimal*. Let A be a DCOP assignment and $A(i)$ be the value of v_i in A.

DEFINITION 1. *The deviation group $D(A, A')$ is defined as the set of nodes with a different assignment in A and A', i.e. $D(A, A') = \{v_i | A(i) \neq A'(i)\}$.*

DEFINITION 2. *A DCOP assignment A is* k-size optimal *if $R(A) \geq R(A')$ for all A' for which $|D(A, A')| \leq k$.*

In Figure 1, the assignment $A = \{0, 0, 0, 0, 0, 0\}$ with reward $R(A) = 18$ is a k-size optimal solution for $k = 1, 2, 3, 4$, but not for $k = 5, 6$. It is 1-size optimal because the reward is reduced if any single variable changes assignment, and by carefully examining all possibilities it can also be proved to be $2, 3, 4$-size optimal. However, it is not 5-size-optimal because if we change the values of v_1, v_2, v_3, v_4, and v_5 from 0 to 1, the solution reward is improved to 20. For any binary DCOP with n variables, a k-size optimal solution A has quality $R(A) \geq \frac{k-1}{2n-k-1} R(A^*)$, where A^* is the globally optimal solution [11]. This bound is independent on the graph structure and reward structure, but tighter bounds are possible given additional information about the problem [3]. Many incomplete DCOP algorithms including MGM and DSA yield 1-size optimal solutions. General *k-size-optimality* offers a spectrum of solutions with stronger guarantees in exchange for greater computation [11].

3. T-DISTANCE OPTIMALITY

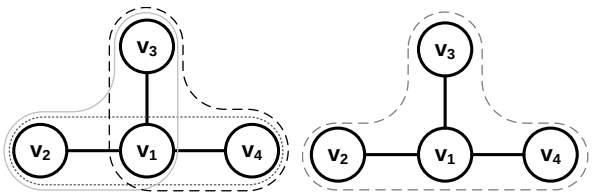
We introduce a novel local optimality criteria, *t-distance-optimality*, that defines local optimization groups based on graph distance. We begin with a formal definition, and discuss the relationship between *k-size-optimality* and *t-distance-optimality*. Then, we give a general lower bound on solution quality of *t-distance-optimality* regardless of graph structure. Finally, we present graph-specific bounds on sets of graphs with varying properties.

For a pair of variables u and v, let $T(u, v)$ be the shortest distance between them in the constraint graph.

DEFINITION 3. *Denote by $\Omega_t(v)$ the group of variables that can be reached from v within t hops, i.e. $\Omega_t(v) = \{u | T(u, v) \leq t\}$.*

DEFINITION 4. *A DCOP assignment A is* t-distance optimal *if $R(A) \geq R(A')$ for all A', where $D(A, A') \subseteq \Omega_t(v)$ for some $v \in V$.*

There are at most n distinct t-distance groups centered on n variables. Some groups may be redundant when they are subsumed by or equivalent to others. For example, in a complete graph, only one group is not redundant because all n 1-distance groups comprise the full set of variables and are thus identical. Furthermore, consider the example shown in Figure 1. No two 1-distance groups are identical. However, $\Omega_1(v_3) = \{v_3, v_4\}$ is subsumed by $\Omega_1(v_4) = \{v_3, v_4, v_1, v_5\}$, so $\Omega_1(v_3)$ is redundant. To better understand the difference between t-groups and k-groups, we consider another example shown in Figure 2. Figure 2(a) shows all three 3-size groups in the graph: $\{v_1, v_2, v_3\}$, $\{v_1, v_3, v_4\}$, and $\{v_1, v_2, v_4\}$. Figure 2(b) shows the only non-redundant 1-distance group: $\{v_1, v_2, v_3, v_4\}$.



(a) 3-size groups (b) 1-distance groups

Figure 2: k-size groups and t-distance groups

Consider, again, the example in Figure 1, where we show assignment $A = \{0, 0, 0, 0, 0, 0\}$ is 0-distance optimal with quality $R(A) = 18$. In 0-distance optimality, each group contains exactly one variable, which is equivalent to 1-size optimality. Changing the value of any single variable from 0 to 1 will reduce the quality. This is because flipping the value of either v_3 or v_6 leads to reward 15, flipping the value of either v_1 or v_2 leads to reward 12, and flipping the value of either v_4 or v_5 leads to reward 9. Since none of these groups has an incentive to deviate, A is 0-distance optimal. Similarly we can test whether A is 1-distance optimal by checking all 1-distance groups. In this example, there are four non-redundant 1-distance groups: $\Omega_1(v_4) = \{v_1, v_3, v_4, v_5\}$, $\Omega_1(v_1) = \{v_1, v_2, v_4\}$, $\Omega_1(v_2) = \{v_1, v_2, v_5\}$, and $\Omega_1(v_5) = \{v_2, v_4, v_5, v_6\}$. After enumerating all possible deviations in these groups, we find no group can improve solution quality by local optimization. Therefore A is also 1-distance optimal. However, it is not 2-distance optimal because $\{1, 1, 1, 1, 1, 1\}$ is a better assignment whose deviation group is fully comprised by $\Omega_2(v_4)$. In this example, 2-distance optimality already guarantees global optimality because $\Omega_2(v_4)$ contains exactly all variables in the network.

3.1 Comparing t and k Optimality

Both *t-distance-optimality* and *k-size-optimality* are criteria for local optimality, but there is a key distinction between them. In *k-size-optimality*, the size of local optimization group is fixed, but the number of possible k-size groups may be very large, especially in dense graphs. In contrast, in *t-distance-optimality*, the number of optimization groups is fixed, but the size of t-distance groups can be very large, particularly in dense graphs. For example, in a complete graph with n variables, there are $\binom{n}{3}$ distinct 3-size groups where each has a fixed size 3. However, there is only one unique 1-distance group comprising all n variables. One of our primary contributions in this paper is to empirically test the implications of this tradeoff for local search methods.

To further understand the connection between k-optimality and t-optimality, we examine some special cases. The first observation is that 1-size optimality is equivalent to 0-distance optimality. In both criteria, one single node forms the local optimization group. Furthermore, in ring graphs, *t-distance-optimality* is also equivalent to *k-size-optimality* for $k = 2t + 1$. For example, 3-size optimality is exactly the same as 1-distance optimality in a ring graph, because every section of 3 connected nodes is an optimization group for both $k = 3$ and $t = 1$ and no additional groups exist for either case.

There are several reasons to believe *t-distance-optimality* potentially offers benefits over *k-size-optimality* as a criterion for distributed local search algorithms. First of all, a t-distance optimal solution always guarantees $(2t + 1)$-size optimality, since every $(2t + 1)$-size group must be contained in some t-distance group. But the reverse is not true; there are $(2t + 1)$-size optimal solutions that are not t-distance optimal. For example, in a complete graph, t-distance optimal solution always guarantees global optimality for $t \geq 1$, while a k-size optimum can possibly reach the lower bound given in Section 2.2 which is known to be tight for complete graphs. While complete graphs are a somewhat artificial example, we might expect to find similar advantages for t-distance optimality when there are hub nodes with many connections or subgraphs that are densely connected. Second, *t-distance-optimality* naturally captures graph locality which might help improving efficiency of local search algorithms particularly in distributed environment where communication delay is the dominating cost. Also, algorithms for *t-distance-optimality* may reduce privacy loss as private information of an agent can only be obtained by those within a fixed distance.

3.2 General Lower Bound

We derive a general lower bound on solution quality for *t-distance-optimality*, regardless of the graph structure.

PROPOSITION 1. *Consider a DCOP with n variables, minimum constraint arity m, where all constraint rewards are non-negative, and A^* is the globally optimal solution, then, for any t-*distance optimal *assignment A_{topt}, where $t > 0$ and $m+t-1 \leq n$, we have $R(A_{topt}) \geq \frac{m+t-1}{n} R(A^*)$.*

PROOF. Let $R_c(A)$ denote the reward on constraint c for any assignment A. For any set of constraints S, let $R_S(A) = \sum_{c \in S} R_c(A)$. Let $\sigma(c)$ be the set of variables in constraint c, and $\pi(W)$ be the set of constraints across a subset of variables $W \subseteq V$ ($c \in \pi(W)$ iff $\sigma(c) \subseteq (W)$).

Let $A'(v)$ be an assignment derived from A_{topt} by changing all assignments in $\Omega_t(v)$ to their corresponding values in A^*. Since A_{topt} is a *t-distance optimal* assignment, $R(A_{topt}) \geq R(A'(v))$. Because all constraint values in the DCOP are non-negative, $R(A'(v)) \geq R_{\pi(\Omega_t(v))}(A'(v))$. Furthermore $A'(v)$ has the identical assignment as A^* over $\Omega_t(v)$, so

$$R(A_{topt}) \geq R_{\pi(\Omega_t(v))}(A'(v)) = R_{\pi(\Omega_t(v))}(A^*)$$

Summing over all t-groups, we have:

$$nR(A_{topt}) \geq \sum_{i=1}^{n} R_{\pi(\Omega_t(v_i))}(A^*) \quad (1)$$

Now we count the contribution of each constraint c to the rhs. Because c has an arity of at least m, $|\sigma(c)| \geq m$. Pick an arbitrary variable v in $\sigma(c)$, if the t-group $\Omega_t(v)$ is identical to V, i.e. comprising all variables in the graph, then $R(A_{topt}) = R(A^*)$. Otherwise, there exists a $v_j \in V$ such that $T(v, v_j) > t$. Write the first $t+1$ variables on the shortest path from v to v_j as $v, v_1, \ldots, v_t$. $T(v, v_i) = i$, which implies that for $i > 1$, $v_i \notin \sigma(c)$.

Consider two cases. First, if $v_1 \in \sigma(c)$, c appears in $\pi(\Omega_t(v'))$ for all $v' \in \sigma(c) \cup \{v_2, v_3, ..., v_t\}$. Certainly c appears $\pi(\Omega_t(v'))$ for $v' \in \sigma(c)$ as $t \geq 1$. c also appears $\pi(\Omega_t(v'))$ for $v' \in \{v_2, v_3, ..., v_t\}$ because for any variable v'' in $\sigma(c)$ and any $2 \leq i \leq t, T(v'', v_i) \leq T(v'', v_1) + T(v_1, v_i) = 1 + i - 1 \leq t$. Therefore in the rhs of inequality 1, c is counted at least $|\sigma(c)| + t - 1 \geq m + t - 1$ times.

Now consider the other case where $v_1 \notin \sigma(c)$. c appears in $\pi(\Omega_t(v'))$ for all $v' \in \sigma(c) \cup \{v_1, v_2, ..., v_{t-1}\}$. c appears $\pi(\Omega_t(v'))$ for $v' \in \{v_1, v_2, ..., v_{t-1}\}$ because for any v'' in $\sigma(c)$ and any $1 \leq i \leq t - 1, T(v'', v_i) \leq T(v'', v) + T(v, v_i) = 1 + i \leq t$. Therefore, c will be also counted at least $|\sigma(c)| + t - 1 \geq m + t - 1$ times in this case. Since c is counted at least $m + t - 1$ times in the rhs of inequality 1 in both cases we have,

$$R(A_{topt}) \geq \frac{\sum_c (m + t - 1) R_c(A^*)}{n} = \frac{(m + t - 1)}{n} R(A^*)$$

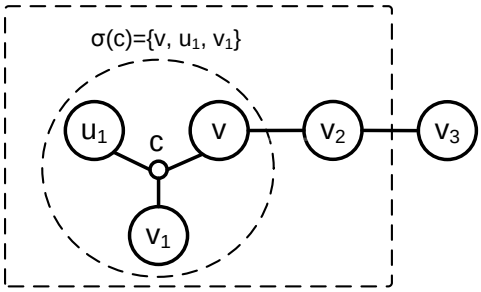
□

To demonstrate the two cases in the proof above, we consider the example shown in Figure 3, where $|c| = 3$ and $t = 2$. In this example, we show constraint c appears in $|c| + t - 1 = 4$ distinct 2-distance groups. Figure 3(a) demonstrates the situation where $v_1 \in \sigma(c)$. In this case, constraint c appears in four groups centered on v, u_1, v_1, and v_2 respectively. Figure 3(b) shows the other situation where $v_1 \notin \sigma(c)$. In this case, constraint c appears in four groups centered on v, u_1, u_2, and v_1 respectively. In both figures, variables inside the dashed circle represent $\sigma(c)$ and variables inside the dashed box represent those whose t-groups have constraint c. As we can see in both cases, c appears in at least 4 different 2-distance groups.

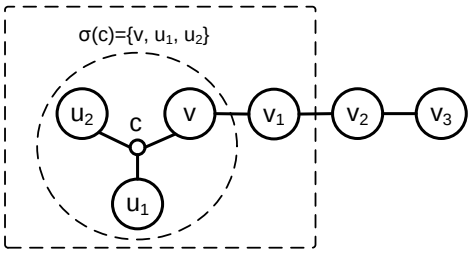
PROPOSITION 2. *For binary constraint DCOP with n variables, the lower bound for 1-distance optimality in proposition 1 is tight.*

PROOF. Consider a complete bipartite graph with $2h$ variables (figure 4 shows an example of $h = 3$). Let $S_1 = \{v_{11}, v_{12}, \ldots, v_{1n}\}$ and $S_2 = \{v_{21}, v_{22}, \ldots, v_{2n}\}$. For any $1 \leq i, j \leq n$, there is a constraint between v_{1i} and v_{2j}. Variables can take a value of either 0 or 1. All constraints have the same reward matrix as shown in figure 4. The global optimum is $\{1, 1, \ldots, 1\}$ with quality h^3. Proposition 1 gives the lower bound for 1-distance optimum of $\frac{2}{2h} h^3 = h^2$. We claim that assignment $\{0, 0, \ldots, 0\}$ is an 1-distance optimum with quality h^2. Consider only variable v_{11}, w.l.o.g. due to symmetry. $\Omega_1(v_{11})$ contains all h variables in S_2 and none in S_1.

i. Suppose the value of v_{11} remains 0, and $1 \leq b \leq n$ variables in S_2 change to 1, then the total quality will decrease to $h(h - b)$.



(a) In the first case, constraint c appears in groups centered on v, u_1, v_1, and v_2



(b) In the second case, constraint c appears in groups centered on v, u_1, u_2, and v_1

Figure 3: Example showing the two cases

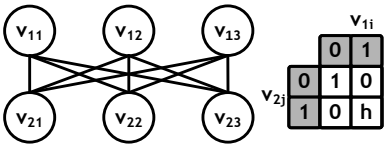


Figure 4: Example showing bound tightness for $t = 1$

ii. Suppose the value of v_{11} is changed to 1, and $0 \leq b \leq n$ variables in S_2 change to 1, then the total quality will be to $bh + (h-1)(h-b) = h^2 - h + b \leq h^2$.

In either case, the solution quality can't be improved. Therefore $\{0, 0, \ldots, 0\}$ is 1-distance optimal. □

3.3 Graph-Specific Lower Bound

In previous work on *k-size-optimality*, linear fractional programming (LFP) was used to find tighter bounds for specific graphs [11]. We use a similar method for t-distance optimality. One LFP variable $R_c(A_{topt})$ represents the reward on c in the t-distance optimal solution, and a second $R_c(A^*)$ represents the reward in the optimal solution. By definition $R(A_{topt}) \geq R(A')$ for all A', where the deviation group between A_{topt} and A' is comprised in some t-group. Let Θ be the set of assignments such that $A' \in \Theta$ iff $D(A_{topt}, A') \subseteq \Omega_t(v)$ for some $v \in V$ and variables in $D(A_{topt}, A')$ take the same value as in A^*. The objective is to minimize $\frac{R(A_{topt})}{R(A^*)}$ such that $\forall A' \in \Theta, R(A_{topt}) - R(A') \geq 0$. Note that $R(A_{topt})$ and $R(A^*)$ can be expressed as $\sum_c R_c(A_{topt})$ and $\sum_c R_c(A^*)$. $R(A')$ can also be represented as the sum over all constraints, $R(A') = \sum_{c \in C} R_c(A')$. So far we can write down the LFP:

$$
\begin{aligned}
Min \quad & \frac{R(A_{topt})}{R(A^*)} \\
s.t. \quad & R(A_{topt}) = \sum_{c \in C} R_c(A_{topt}) \\
s.t. \quad & R(A^*) = \sum_{c \in C} R_c(A^*) \\
s.t. \quad & R(A') = \sum_{c \in C} R_c(A'), \forall A' \in \Theta \\
s.t. \quad & R(A_{topt}) \geq R(A'), \forall A' \in \Theta
\end{aligned}
$$

In the LFP above we still need to represent $R_c(A')$. Consider any constraint c, suppose it has two variables v_i and v_j. Denote by $A(i)$ the value of v_i in assignment A. Then,

i. $R_c(A') = R_c(A_{topt})$, if $A'(i) = A_{topt}(i) \wedge A'(j) = A_{topt}(j)$.

ii. $R_c(A') = R_c(A^*)$, if $A'(i) = A^*(i) \wedge A'(j) = A^*(j)$.

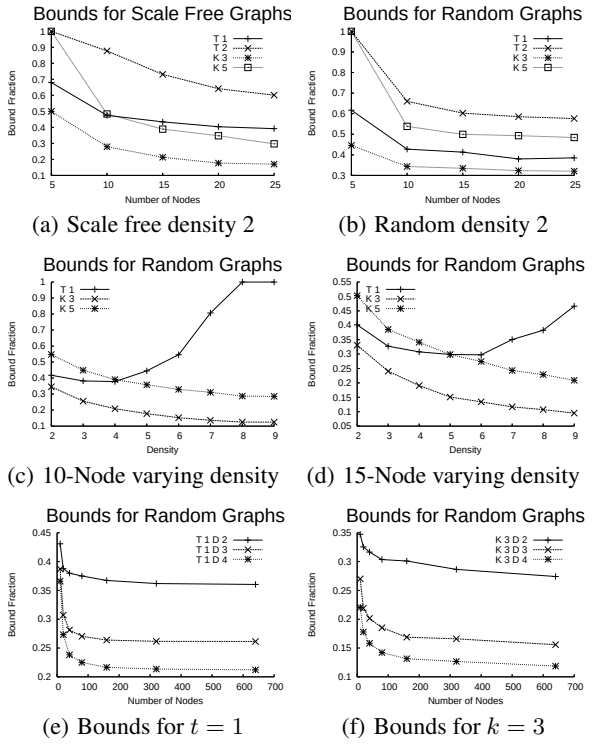
iii. $R_c(A') = 0$, otherwise.

Figure 5 shows the average graph-specific bounds over 30 samples. (see Section 5 for details on graph generation). In Figure 5(a) and Figure 5(b), we see that *t-distance-optimality* provides much stronger lower bounds on average than comparable *k-size-optimality* on both scale free and random graphs. We note $t = 2$ provides a substantial improvement over $t = 1$. Comparing lower bounds on different types of graphs, we also note *t-distance-optimality* offers more benefits on scale free graphs than on random graphs. The lower bound for $t = 1$ is generally better than that for $k = 5$ on scale free graphs while $k = 5$ constantly outperforms $t = 1$ on random graphs. The differences shown are statistically significant (for example, the p-value for a comparison of $t = 1$ and $k = 3$ on 25-node scale free graphs is 4.58×10^{-24}). We also did tests on graphs with varying density. Figure 5(c) and Figure 5(d) show the results for $t = 1$, $k = 3$, and $k = 5$ on 10-node and 15-node random graphs with varying density respectively. We note that k-size optimal bounds tend to degrade more quickly with increasing density, which t-distance optimality is more stable and eventually improves for very high densities. Finally, we tested bounds on large graphs. Figure 5(e) and Figure 5(f) show the bounds for $t = 1$ and $k = 3$ on random graphs with varying graph size. We can see both *t-distance-optimality* and *k-size-optimality* bounds degrade quickly at relatively small graph size. However, after graph size reaching 100, both bounds begin to stabilize and eventually converge to some fixed value. We can see $t = 1$ on average guarantees 21.2% of the global optimum on density 4 graphs with 640 nodes while $k = 3$ guarantees 11.9%.

4. ASYNCHRONOUS ALGORITHM

In this section, we introduce our novel asynchronous algorithm for DCOP based on *t-distance-optimality*. It is a variant of distributed local search which improves quality monotonically over time from a random initial assignment. It is fully distributed and uses asynchronous methods for both computation and coordination.

We give an overview of the algorithm before discussing key stages in more detail. First, agents broadcast local information about the graph structure and use this to determine group members. During the main phase, all groups compute new optimal assignments in parallel, assuming nodes outside of the group maintain unchanged. If an improvement is found, the group leader attempts to implement the new assignment by sending out requests. This can cause



(a) Scale free density 2 (b) Random density 2

(c) 10-Node varying density (d) 15-Node varying density

(e) Bounds for $t = 1$ (f) Bounds for $k = 3$

Figure 5: Graph-specific bounds comparison

conflicts among overlapping groups, which is resolved by an asynchronous locking and commitment protocol in our approach.

4.1 Initialization

An agent begins by picking a random value from its domain as its initial assignment. It then composes a message containing all its constraints and broadcasts the message to a distance of t hops. After that, it broadcasts its initial value to a distance of $t + 1$ hops in a separate message. Here we assume each agent has an unique ID (e.g. a MAC address) which can be included in the message to identify the sender.

In the initial stage, the center node of each group will receive full constraint information and values of nodes inside the group. Since the initial values are broadcast an additional hop, the center node will also receive the values of all fringe nodes of the group. Fringe nodes of a group are those outside the group but directly linked to some node inside the group. Local optimization of a group is performed under the assumption that all the fringe nodes remain static.

Every agent in the network will automatically be the leader of a t-distance group centered on it. However, some groups may be subsumed by or equivalent to others, leading to redundant computations. We remove some redundant groups using a simple optimization. Each agent computes a shortest distance matrix within its local view ($t + 1$ hops). Its t-group is redundant if

i. there is an agent whose shortest distance to every other agent is strictly less than $t + 1$;

ii. there is an agent with a lower id whose shortest distance to every other agent is less than or equal to $t + 1$.

4.2 Computation for Local Optimum

In the previous stage, the leader has gathered all the information required for computing the local optimal solution. During the main phase of the algorithm new optimal assignments for each group are computed in parallel. In principle, any complete DCOP solver can be adapted to this purpose. We implement a centralized approach motivated by OptAPO [9] for our experiments.

Our implementation uses a variable elimination algorithm comparable to a centralized version of DPOP [13] if the average density of the local perspective is less than $\frac{n}{2}$ and a branch-and-bound solver otherwise. Variable elimination methods like DPOP are exponentially complex with respect to the width of the pseudo tree. Therefore, the centralized DPOP solver can be very efficient in low density graphs. However, a branch-and-bound solver has advantage in dense graphs thanks to the pruning on solution space based on reward structure. To perform this computation, the leader needs to know the current assignments of the fringe nodes. Each time a node commits a change and switches to a new assignment, it broadcasts the new value to a distance of $t + 1$ hops to ensure that all leaders have the required information. Once the leader receives new assignment information, it immediately gives up the ongoing lock attempt (if there is one), unlocks all group members, and recomputes a new optimal assignment.

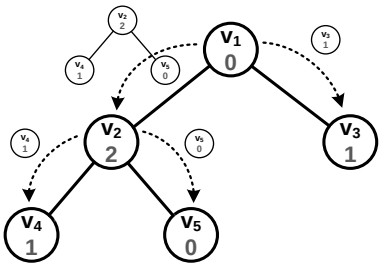
4.3 Asynchronous Assignment Implementation

When the leader finds the local optimum of the group that can yield improved solution quality, it will attempt to implement the new assignment. While implementing changes in a single group guarantees monotonically increasing quality, multiple overlapping groups implementing assignments simultaneously might lead to degradations. Conflicts happen when some agent receives exclusive value changing requests from multiple group leaders. Existing incomplete DCOP solvers, including KOPT [8], typically requires synchronization to resolve such conflicts. However, in real problems, synchronization over the whole network is not trivial to achieve in the first place. Furthermore, synchronous algorithms can be fragile in dynamic environments where message loss, node failures, and other unpredictable events happen frequently. For example, when a node loses connection to the network, synchronous algorithm completely stops working due to the failure of synchronization. In our algorithm, we use an asynchronous protocol, which is generally more robust to various types of noise in real applications. In asynchronous algorithms, when a single node failure happens, only a small set of nodes are affected and the remaining part of the network still works as normal.

The protocol is based on a standard lock/commit pattern. The leader sends lock requests to all group members and fringe nodes, which accept the request unless they have already locked on a different assignment (multiple locks for the same assignment are acceptable). If all nodes accept, the leader sends a commit message and the assignment is implemented. Otherwise, the leader unlocks all nodes and backs off to prevent deadlock.

It is trivial for the leader to lock the group when all lock requests are sent to its direct neighbors. However, we need a protocol to forward lock requests in large groups where the leader cannot directly communicate with all group and fringe nodes. A simple flooding approach based on broadcasting is inefficient, especially in dense graphs. Our algorithm fully utilizes the graph information procured in the initial stage to reduce number of messages. The leader first performs a breadth-first search and builds a BFS tree rooted on itself, in which a tree node represents an agent. A tree node is assigned an **(ID, Val)** pair, indicating agent **ID** is locked on **Val**. Each neighbor of the leader corresponds to a child of the root in the BFS tree. The leader sends the corresponding subtree to every neighbor

as a lock request. An agent that receives a lock message extracts the lock value from the root, and forwards the subtrees to its children. Figure 6 is a demonstration for this protocol.



6: Example demonstrating lock message forwarding protocol. Each node represents a variable v_i. The gray number in the node indicates the value to be locked on. Each arrow associated with a subtree that represents a lock message.

If a previous lock attempt fails, the leader will wait for a random interval between 0 and the maximum interval I_m before sending lock requests again. Failure of each lock attempt doubles I_m until reaching a threshold (256 time units). To reduce conflicts in the beginning, every leader waits for a random number of time units between 0 and 32 before the first lock attempt. When a node commits to an assignment, it broadcasts the new assignment to a distance of $t + 1$ hops and is able to accept new lock requests. Leaders receiving new assignment information unlock nodes if necessary and recompute a new optimal assignment. Conflicts have a substantial cost, so we implemented several techniques to improve locking performance.

Subset Locking (SL): If a new assignment does not change the assignment for every node, it is not necessary to lock all group and fringe nodes. Subset locking only requests locks from nodes changing assignment and their neighbors, rather than the full group, reducing the chance for conflicts.

Partial Synchronization (PS): A benefit of synchronized conflict resolution is that nodes can implement heuristics to commit to groups with a high gain for changing assignment. We approximate this in the asynchronous setting by waiting for a fixed time period to pool lock requests at each node. Once the timer expires, a node selects the lock message with the largest gain to accept (group gain is included in the lock message when using PS).

5. EXPERIMENTAL EVALUATION

In addition to the algorithm for *t-distance-optimality* described in Section 4, we implement a comparable algorithm for *k-size-optimality* using the same asynchronous coordination protocol. We explain some non-trivial details. First, in the initialization stage, constraints are broadcast to a distance of $\lfloor \frac{k}{2} \rfloor$ instead of t hops. And after changing value, agents broadcast the new value to a distance of $\lfloor \frac{k}{2} \rfloor + 1$ instead of $t + 1$ hops. Second, each agent can be the leader for multiple k-size groups. Each k-size group selects the most centralized agent (using ID as secondary comparator) to be the leader. Third, each agent computes the locally optimal solution for each k-size group it leads and locks the one that yields the maximum gain.

We test our algorithms in simulation, using the DAJ toolkit [14] which provides low-level support for simulating distributed algorithms in Java. This toolkit provides a simple programming interface that allows to develop distributed algorithms based on a message passing model. The primary performance metric we use is solution quality per unit of global time. Global time unit is motivated by the metric of cycles commonly used in evaluating synchronous DCOP algorithms. We made some small modifications to the DAJ toolkit so that in a single global time unit, each node can process all messages in the incoming queue, and send as many messages as desired to neighbors. Messages are delivered on the subsequent time steps, e.g. a message sent at time x will be received at time $x + 1$.

We generate DCOPs for three main classes of constraint graphs:

i. $G(n, M)$ random graphs [2]. In this model, the graph is generated by randomly adding M edges. Each is picked out of $\binom{n}{2}$ possible choices with equal probability. However, by excluding disconnected graphs to guarantee connectivity, our random graphs may be slightly different from the original model.

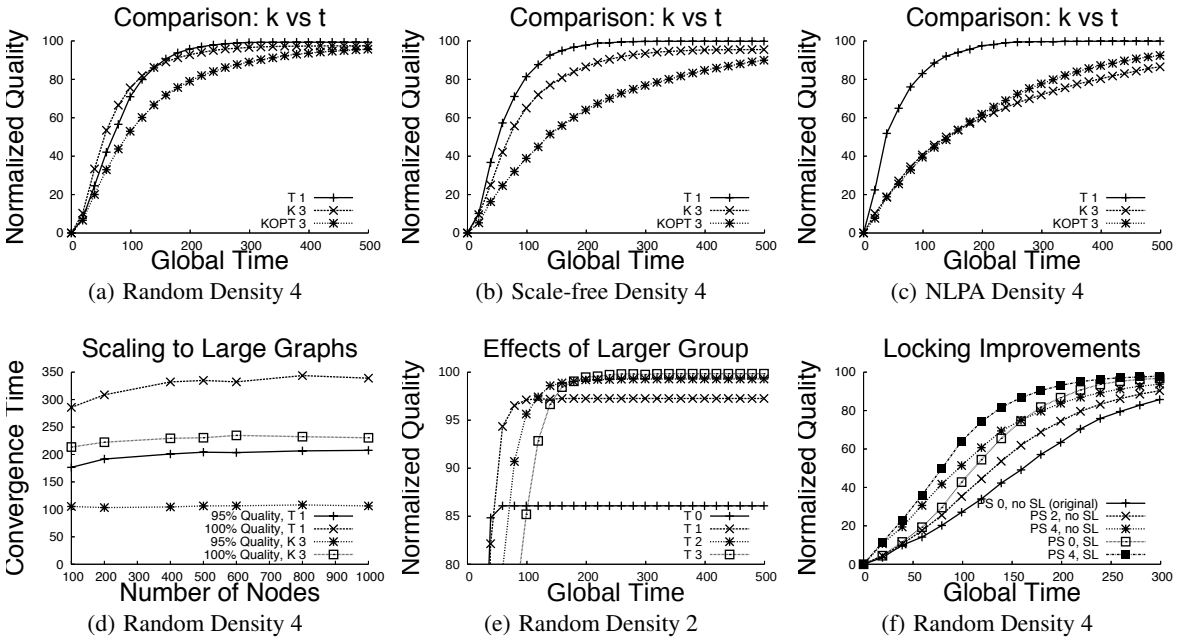
ii. Barabasi-Albert (BA) scale-free graphs [1]. This is one simple algorithm for generating random scale-free graphs using a linear preferential attachment mechanism. In this model, graphs begin with a complete graph of m_0 nodes, where m_0 is a small number but at least 2. New nodes are added to the network one at a time. Each new node is connected to $m \leq m_0$ of the existing nodes with a biased probability proportional to the number of links the existing node already has. The degree distribution resulting from the BA model is scale-free, in particular, it is a power law of the form $P(k) \sim k^{-3}$.

iii. Non-linear preferential attachment (NLPA) graphs based on the BA model, but with a stronger bias towards larger numbers of nodes with few connections. This model is based on the BA model, but instead of adding a new node to existing nodes with probability linear to their degree, we alter the probability to be non-linear to the degree, in particular, degree$^{1.7}$. As a result, heavily linked nodes ("hubs") tend to quickly accumulate even more links than those in scale-free graphs, while nodes with only a few links are less likely to be chosen as the destination for a new link.

All variables have domain size 10 and constraint rewards are all randomly drawn from the uniform distribution of integers in the range $[0 \ldots 10000]$.

Our first experiment compares k-size and t-distance optimality on random, scale free, and NLPA graphs. Each graph has 100 variables. Both k-size and t-distance algorithms use subset locking and partial synchronization. PS window size is selected after testing several possible settings. "KOPT" introduced by Katagishi and Pearce [8] is included as a benchmark. We show solution quality at each global time, averaged over 50 sample graphs. Each algorithm starts from the same random assignment. Quality is normalized by subtracting the value of the initial random assignment and dividing by the maximum value found by any algorithm for each problem instance. We do not include error bars so as not to clutter the graphs, but the primary comparisons described are all highly significant based on paired t-tests.

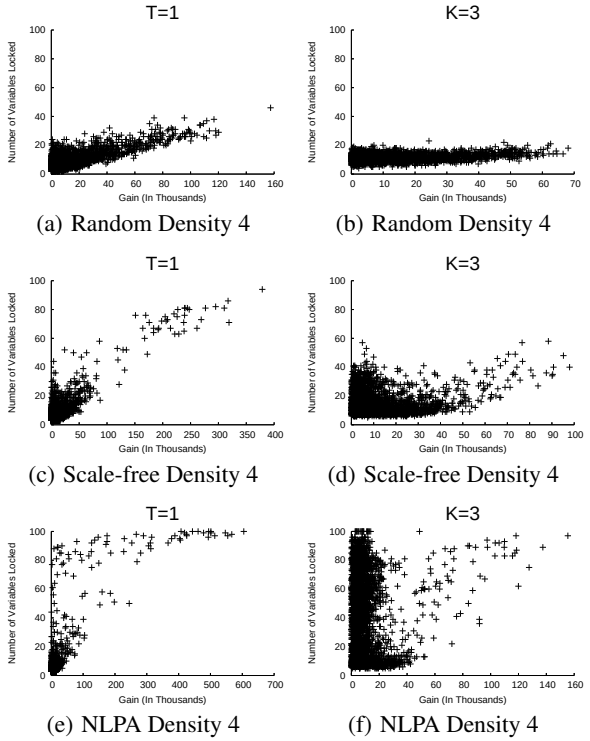
Results of the first experiment are shown in Figures 7(a), 7(b), and 7(c). Settings with $k = 2t + 1$ are comparable and equivalent for some classes of graphs (Section 3.1); we use settings of $k = 3$ and $t = 1$ to examine the tradeoff in local optimization group type. Our algorithms for both t-opt and k-opt clearly outperform KOPT on both random and scale-free graphs. Our *k-size-optimality* algorithm is similar to KOPT on NLPA, but *t-distance-optimality* strongly outperforms both other algorithms in this case. By time 100, its improvement in solution quality is double that of the other algorithms. In scale-free graphs *t-distance-optimality* is also a clear winner in both convergence speed and final quality. For random graphs, *k-size-optimality* has advantages in convergence speed, but converges to a lower final quality.



(a) Random Density 4 (b) Scale-free Density 4 (c) NLPA Density 4

(d) Random Density 4 (e) Random Density 2 (f) Random Density 4

Figure 7: Results of experiments (1) comparing k-size and t-distance optimality, (2) examining scalability to large graphs, (3) exploring the effects of increasing t, and (4) showing the benefits of locking improvements.



(a) Random Density 4 (b) Random Density 4

(c) Scale-free Density 4 (d) Scale-free Density 4

(e) NLPA Density 4 (f) NLPA Density 4

Figure 8: Analysis on Local Group Changes

To further understand the performance of t-opt and k-opt, we provide analysis on local group changes. A local group change is a successful assignment implementation in an optimization group. For every group change, we collect the quality gain and the number of variables that were locked for implementing the new assignment. We plot all group changes for each set of graphs in a scatter chart, where a data point represents a **(Gain, #Locked Variables)** pair. The efficiency of an algorithm can be measured by the distribution of all group changes. Since larger lock sizes indicate higher chance of conflicts, group changes that have a large gain but small number of variables locked are most favored. The results on random graphs are shown in Figure 8(a) and Figure 8(b). We can see, $t = 1$ often has a larger gain but needs to lock more variables than $k = 3$. In particular, while $k = 3$ never identifies a gain larger than 70,000 and barely locks more than 20 variables, $t = 1$ often finds group changes where the gain is greater than 70,000 and the number of locked variables is greater than 20. On scale free graphs, as shown in Figure 8(c) and Figure 8(d), $t = 1$ often finds larger gains than $k = 3$. Particularly, the largest gain of $t = 1$ is almost quadruple of that of $k = 3$. We also note $k = 3$ often locks a large set of variables. Nearly half of the group changes require to lock more than 20 variables. This explains why $t = 1$ outperforms $k = 3$ in both convergence speed and final quality on scale free graphs. Figure 8(e) and Figure 8(f) show the results on NLPA graphs, where $k = 3$ is the most inefficient because it often locks a huge number of variables but barely identifies large gains.

Figure 7(e) shows the effect of increasing values of t on density 2 random graphs (domain size 5, reward drawn uniformly randomly from 0 to 625). We see that the different levels of t offer increasing final quality but also increasing convergence time. We also note increasing t doesn't give consistent gain in final quality. While $t = 1$ is significantly better than $t = 0$, the gain of $t = 3$ over $t = 2$ is much smaller. We also tested the ability of our algorithms to scale to very large graphs. In Figure 7(d), we show the number of time units required to reach a percentage of the final quality for both *t-distance-optimality* and *k-size-optimality*. We see that increasing the size of the random graph by tenfold to 1000 nodes

barely increases the time necessary for convergence at all, showing impressive scalability. To show the improvements offered by the partial synchronization (PS) and subset locking (SL) methods, we tested the t-distance algorithm for $t = 1$ with different PS window sizes and without SL. As can be seen in Figure 7(f), both techniques improve performance in isolation and in tandem.

Tables 1(a), 1(b), and 1(c) give additional statistics about algorithm performance in the first set of experiments on density 4 graphs. "Msgs" is the average number of messages per agent per time step. "MsgSize" is the average message size per message. "Evals" is the average number of constraint evaluations per agent per time step, which is a rough measure of the computation burden on each node. "Conflicts" is the total number of failed lock attempts. All of these are accumulated for 500 time units. We see that our algorithm for *t-distance-optimality* is generally much more efficient than KOPT in message size and the number of messages sent out, but does place a somewhat higher computational burden on the nodes. *t-distance-optimality* generally requires more computation than *k-size-optimality* because of its larger group size. We also note that *k-size-optimality* generally sends fewer but larger messages, which may offer some additional benefits in real world implementations.

Table 1: Additional Statistics

(a) Statistics for scale-free density 4 graphs.

	Msgs	MsgSize	Evals	Conflicts
T 1	1.27	28.44	187.62	405.64
K 3	0.72	43.99	114.64	411.60
KOPT 3	3.12	2970.83	72.84	0.00

(b) Statistics for random density 4 graphs.

	Msgs	MsgSize	Evals	Conflicts
T 1	0.85	26.68	81.81	445.76
K 3	0.47	41.51	45.51	410.50
KOPT 3	3.20	1209.94	57.29	0.00

(c) Statistics for NLPA density 4 graphs.

	Msgs	MsgSize	Evals	Conflicts
T 1	2.55	36.52	270.65	300.72
K 3	1.90	46.01	338.76	358.54
KOPT 3	3.12	7093.90	100.42	0.00

6. CONCLUSION

We make three key contributions. First, we introduce the novel concept of *t-distance-optimality*, and establish solution quality bounds for this concept that are often tighter than known bounds for *k*-size optimality. Second, we develop asynchronous local search algorithms for *t-distance-optimality* that outperform existing synchronous algorithms for *k-size-optimality*. Finally, in our experimental evaluation we investigate the tradeoff between k and t optimality, showing that *t-distance-optimality* offers considerable advantages for some types of DCOP (notably scale free graphs), while *k-size-optimality* has advantages in others.

7. ACKNOWLEDGEMENT

This research was supported by a subcontract from Perceptronics.

8. REFERENCES

[1] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[2] B. Bollobas. *Random Graphs*. Cambridge University Press, 2nd edition, 2001.

[3] E. Bowring, J. P. Pearce, C. Portway, M. Jain, and M. Tambe. On k-optimal distributed constraint optimization algorithms: New bounds and algorithms. In *AAMAS-08*, 2008.

[4] A. Chechetka and K. Sycara. No-commitment branch and bound search for distributed constraint optimization. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1427–1429, New York, NY, USA, 2006. ACM.

[5] J. S. Cox, E. H. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 821–827, New York, NY, USA, 2005. ACM.

[6] B. Faltings, D. Parkes, A. Petcu, and J. Shneidman. Optimizing streaming applications with self-interested users using M-DPOP. In *COMSOC-06*, 2006.

[7] 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.

[8] H. Katagishi and J. P. Pearce. KOPT: Distributed DCOP algorithm for arbitrary k-optima with monotonically increasing utility. In *DCR-07*, 2007.

[9] R. Mailler and V. Lesser. Using cooperative mediation to solve distributed constraint satisfaction problems. In *AAMAS-04*, 2004.

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

[11] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI-07*, 2007.

[12] J. P. Pearce, M. Tambe, and R. T. Maheswaran. Solving multiagent networks using distributed constraint optimization. *AI Magazine*, 29(3):47–66, 2008.

[13] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. In *IJCAI-05*, 2005.

[14] W. Schreiner. A java toolkit for teaching distributed algorithms. In *ITCSE-02*, pages 111–115, 2002.

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