Asynchronous Algorithms for Approximate Distributed Constraint Optimization with Quality Bounds

Christopher Kiekintveld, Zhengyu Yin, Atul Kumar, and Milind Tambe
University of Southern California
{kiekintv, zhengyuy, atulk, tambe}@usc.edu

ABSTRACT

Distributed Constraint Optimization (DCOP) is a popular framework for cooperative multi-agent decision making. DCOP is NP-hard, so an important line of work focuses on developing fast incomplete solution algorithms for large-scale applications. One of the few incomplete algorithms to provide bounds on solution quality is k-size optimality, which defines a local optimality criterion based on the size of the group of deviating agents. Unfortunately, the lack of a general-purpose algorithm and the commitment to forming groups based solely on group size has limited the use of k-size optimality.

This paper introduces t-distance optimality which departs from k-size optimality by using graph distance as an alternative criteria for selecting groups of deviating agents. This throws open a new research direction into the tradeoffs between different group selection and coordination mechanisms for incomplete DCOP algorithms. We derive theoretical quality bounds for t-distance optimality that improve known bounds for k-size optimality. In addition, we develop a new efficient asynchronous local search algorithm for finding both k-size and t-distance optimal solutions — allowing these concepts to be deployed in real applications. Indeed, empirical results show that this algorithm significantly outperforms the only existing algorithm for finding general k-size optimal solutions, which is also synchronous. Finally, we compare the algorithmic performance of k-size and t-distance optimality using this algorithm. We find that t-distance consistently converges to higher-quality solutions in the long run, but results are mixed on convergence speed; we identify cases where k-size and t-distance converge faster.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

General Terms

Algorithms, Theory, Experimentation

Keywords

DCOP, distributed constraint optimization, asynchronous algorithms, approximate, incomplete, DALO, experimentation, bounds, k-Optimality, t-Distance Optimality

Cite as: Asynchronous Algorithms for Approximate Distributed Constraint Optimization with Quality Bounds, Christopher Kiekintveld, Zhengyu Yin, Atul Kumar, and Milind Tambe, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 133-140

1. INTRODUCTION

Distributed constraint optimization (DCOP) is a common formalism for representing multi-agent systems in which agents cooperate to optimize a global objective. It is well-suited to domains where the primary interactions are between local subsets of agents, such as sensor networks [17], peer-to-peer networks [5], and meeting scheduling [14]. There are many complete DCOP algorithms that guarantee global optimality, including ADOPT [12], DPOP [14], OptAPO [11], and AFB [8]. However, DCOP is NP-hard [12] so it is necessary to consider faster incomplete methods for large-scale applications.

Two broad trends have emerged in the literature on incomplete DCOP algorithms. First, researchers have developed algorithms that focus on individual agents, varying the information agents collect from others and the decision procedure. Algorithms that fall into this category include MGM/DBA [13, 17], ALS_DisCOP [18], and DSA [7]. These algorithms generally do not provide any guarantees on the quality of solutions they compute, which may be an important consideration in some domains. Recent work on max-sum algorithms [6] has initiated a new trajectory in this area that offers bounds on solutions quality for specific problem instances.

The second trend is to study approaches for coordinating the decisions of local groups of agents, instead of having each agent make an individual choice. The main research thread here is k-size-optimality — which guarantees that the solution cannot be improved by any group of k or fewer agents changing their decision [13, 3]. k-size-optimality is the first incomplete method to offer solution quality bounds that are independent of the problem instance (in contrast, the bounds for max-sum [6] depend on the specific problem, including both the constraint graph and reward matrices). Furthermore, by varying the group size using the k parameter, this class of algorithms provides the user with a way to trade off better solution quality (larger k) against the computational overhead of coordinating larger groups.

While k-size optimality is promising, so far it is the only approach along this second avenue. There are several fundamental questions that must be explored to progress this line of work. First, it is not clear the size of the group is the only way (or the best way) to define agent groups to coordinate actions. There may be other criteria that offer better bounds, faster algorithms, or other advantages. Second, "KOPT" [10] is the only existing algorithm for k-size optimality that works for arbitrary settings of k; most are limited to $k = 1$, with one exception up to $k = 3$ [13]. Unfortunately, KOPT is synchronous and has significant inefficiencies, which we show in our experimental results; as a result, applications of higher settings of k have not been viable.

We contribute four important theoretical, algorithmic, and empirical advances for DCOP. First, we introduce an alternative crite-

ria for local optimality—t-distance optimality— that forms groups based on the distance between nodes in the constraint graph instead of strict limits on group size. We derive bounds for t-distance optimality that are stronger than comparable quality bounds for k-size optimality. This opens the door to future exploration of a broader class of incomplete algorithms with quality guarantees that vary the mechanisms for group formation and coordination. Second, we develop an asynchronous local search algorithm for computing either k-size or t-distance optimality, opening up applications of arbitrary values of k and t. Third, we evaluate our methods using a new asynchronous DCOP simulation testbed, using metrics that vary the relative cost of computation and communication. The experiments show that our algorithm significantly outperforms KOPT. Finally, we investigate empirical performance tradeoffs between k-size and t-distance optimality. While t-distance consistently converges to higher quality solutions in the long run, we find that relative convergence speed depends on properties of the constraint graph and the relative costs of computation and communication.

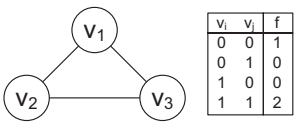
2. DCOP AND k-SIZE OPTIMALITY

2.1 DCOP Definition

A DCOP is defined by sets of variables $V := \{v_1, \ldots, v_n\}$ and constraints $C := \{c_1, \ldots, c_m\}$. Variables have finite domains and each variable is controlled by a separate agent. A joint *assignment* $A := \{a_1, \ldots, a_n\}$ specifies a value for each variable. 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 agents' objective is to compute an assignment A maximizing the sum of rewards, $R(A) = \sum_{c \in C} c(a_i, a_j)$. 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 shows an example DCOP with 3 variables and 3 constraints with identical reward tables. $A = \{1, 1, 1\}$ is the optimal assignment with a reward value of 6.

2.2 k-Size Optimality

Pearce et. al. [13] recently introduced k-size optimality (called k-optimality in the original work) as a local optimality criteria that offers theoretical guarantees on solution quality. The key idea is that any solution that cannot be improved by simultaneously changing the assignment of a subset of the agents is a local optimum. Varying the size of the deviating group yields stronger or weaker solutions, with more or less computational effort.



v_i	v_j	f
0	0	1
0	1	0
1	0	0
1	1	2

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

DEFINITION 1. *Let $D(A, A')$ denote the set of nodes with a different assignment in A and A'. A DCOP assignment A is* k-size optimal *if $R(A) \geq R(A')$ for all A' for which $|D(A, A')| \leq k$.*

Consider the DCOP in Figure 1. The assignment $\{0, 0, 0\}$ is a k-size optimal solution for $k = 1$ or $k = 2$ (with reward of 3), but not $k = 3$. It is 1-size optimal because the reward is reduced to 1 if any single variable changes assignment. If two variables change to 1 the reward decreases to 2 from 3. However, if all three change to 1 the reward increases to 6, so $\{0, 0, 0\}$ is not 3-size optimal. 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 [13]. This bound does not depend on the graph structure, but tighter bounds are possible given additional information about the problem [13, 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. The only existing algorithm for k-size optimality with arbitrary k is Katagishi and Pearce's synchronous "KOPT" algorithm [10].

3. T-DISTANCE OPTIMALITY

We introduce a novel local optimality criteria, t-distance optimality, that defines locality based on a group of surrounding nodes within a fixed distance of a central node. We begin with a formal definition, and then establish bounds on solution quality. Finally, we discuss the relationship between k and t optimality.

DEFINITION 2. *Let $T(v_i, v_j)$ be the distance between two variables in the constraint graph. We denote by $\Omega_t(v) = \{u | T(u, v) \leq t\}$ the t-group centered on v. 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-groups in the constraint graph, centered on the n variables. There may be fewer than n distinct groups if some $\Omega_t(v)$ comprise identical sets of nodes. Consider again the DCOP in Figure 1. Assignment $\{0, 0, 0\}$ is 0-distance optimal, because each t-group contains a single node, equivalent to $k = 1$. However, $\{0, 0, 0\}$ is not 1-distance optimal. A $t = 1$ group for any variable includes both other variables, so all three can change to assignment 1 and improve the reward to 6.

3.1 General Solution Quality Bound

We derive a lower bound on the quality of a t-distance optimal solution, regardless of the graph structure.[1]

PROPOSITION 1. *Consider a DCOP with n variables, minimum constraint arity m, non-negative constraint rewards, and globally optimal assignment A^*. Any t-distance optimal assignment A_{topt} with $t > 0$ and $m + t - 1 \leq n$ has a solution quality $R(A_{topt}) \geq \frac{m+t-1}{n} R(A^*)$.*

PROOF. Let $R_c(A)$ denote the reward on constraint c for any assignment A. For any subset of constraints S, $R_S(A) = \sum_{c \in S} R_c(A)$. Let $\sigma(c)$ be the set of variables in c, and $\pi(W)$ be the set of constraints across a subset of nodes $W \subseteq V$ ($c \in \pi(W)$ 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 t-distance optimal, $R(A_{topt}) \geq R(A'(v))$, and since constraint values are non-negative, $R(A'(v)) \geq R_{\pi(\Omega_t(v))}(A'(v))$. Furthermore, $A'(v)$ is identical to A^* over $\Omega_t(v)$, so $R_{\pi(\Omega_t(v))}(A'(v)) = R_{\pi(\Omega_t(v))}(A^*)$. Therefore, $R(A_{topt}) \geq R_{\pi(\Omega_t(v))}(A^*)$. 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. For an arbitrary variable v in $\sigma(c)$, if the t-group $\Omega_t(v)$ is identical

[1]The bound applies only for the maximization version of DCOP.

to V then $R(A_{topt}) = R(A^*)$ (thus proving proposition 1). 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 $i > 1$, $v_i \notin \sigma(c)$.

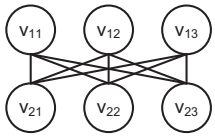
Consider two cases. First, if $v_1 \in \sigma(c)$, c appears in $\pi(\Omega_t(v'))$ for all $v' \in \sigma(c) \cup \{v_2, v_3, ..., v_t\}$. This is because for any variable v'' in $\sigma(c)$, $T(v'', v_i) \leq T(v'', v_1) + T(v_1, v_i) = 1 + i - 1 \leq t$. Therefore in the rhs of inequality 1, c is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

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

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

□

PROPOSITION 2. *The lower bound in proposition 1 is tight for $t = 1$ in DCOPs with binary constraints.*



v_{1i}	v_{2j}	f
0	0	1
0	1	0
1	0	0
1	1	h

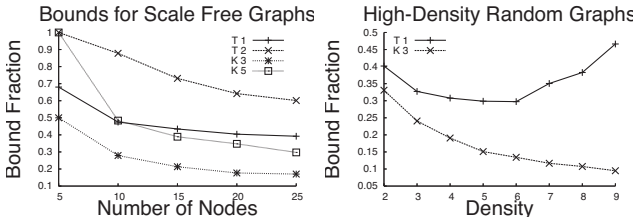
Figure 2: Example showing tightness for $t = 1$.

PROOF. Consider a complete bipartite graph with $2h$ binary variables and constraint payoffs as shown in Figure 2 (for $h = 3$). Let $S_1 = \{v_{11}, v_{12}, ..., v_{1h}\}$ and $S_2 = \{v_{21}, v_{22}, ..., v_{2h}\}$. For any $1 \leq i, j \leq h$, there is a constraint between v_{1i} and v_{2j}, for a total of h^2 constraints. The global optimum is $\{1, \ldots, 1\}$ with quality h^3. Proposition 1 gives a lower bound of $\frac{2}{2h} h^3 = h^2$ for $t = 1$. We claim that $\bar{A} = \{0, 0, ..., 0\}$ is 1-distance optimal with quality of h^2. Consider only variable v_{11}, w.l.o.g. due to symmetry. $\Omega_1(v_{11})$ contains all variables in S_2 and none in S_1. Suppose first that the value assigned to $v_{11} = 0$, and $1 \leq b \leq h$ variables from S_2 change to 1. Then the reward decreases to $h(h - b) < h^2$. Now suppose v_{11} is assigned 1 and $0 \leq b \leq h$ variables in S_2 change to 1. Then the reward is $bh + (h-1)(h-b) = h^2 - h + b \leq h^2$. Therefore, $\bar{A}$ is 1-distance optimal. □

3.2 Graph-Specific Bounds

In previous work on k-size optimality, linear fractional programming (LFP) was used to find tighter bounds for specific graphs [13]. 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 $D(A_{topt}, A') \subseteq \Omega_t(v)$ for some $v \in V$. 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 agents 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 $\sum_c R_c(A^*)$. We can transform the DCOP so that every $R(A')$ can also be expressed in terms of sum of $R_c(A_{topt})$ and $R_c(A^*)$.

Figure 3 shows the average graph-specific quality bound over 30 samples for two classes of graphs (see Section 5 for details on problem generation). The y-axis is the average lower bound, expressed as a fraction of the optimal solution quality (e.g., a bound of 0.7 implies that T=1 guarantees a minimum of 70% on average of the global optimum for a 5-node graph). Figure 3(a) varies the number of nodes for a class of DCOPs with scale-free [1] constraint graphs, and Figure 3(b) varies the density for 20-node graphs with uniform random edges. t-distance optimality provides stronger lower bounds on average than k-size optimality for comparable t and k ("comparable" t and k explained below), and higher settings of t and k provide substantially stronger bounds. The differences shown are statistically significant (for example, the p-value for a comparison of $t = 1$ and $k = 3$ on 25-node graphs is 4.58×10^{-24}). In addition, k-size optimal bounds tend to degrade more quickly as density increases, while t-distance optimality is more stable and improves for very high densities.



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

Figure 3: Average graph-specific quality bounds.

3.3 Comparison of k and t Optimality

There is a distinct tradeoff between k-size and t-distance optimality. In k-size optimality, the number of nodes in each individual group is strictly bounded, but the number of distinct k-groups may be very large, especially in dense graphs. For t-distance optimality the situation is reversed; the number of groups is bounded by the number of variables, but the size of an individual t-group is unbounded and may be large in dense graphs. 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 link between k-size and t-distance, we examine some special cases. First, we note that the criteria are identical for the degenerate case of $k = 1$ and $t = 0$, in which both select groups consisting of individual nodes. The criteria are also equivalent for appropriate values of k and t in ring graphs. Consider $k = 3$ and $t = 1$ (similarly for any $k = 2t + 1$ pair). Every section of 3 connected nodes in the ring is a group for both $k = 3$ and $t = 1$, and no additional groups exist for either case.

There are reasons to believe that t-distance optimality offers benefits over k-size optimality in some cases. First, a t-distance optimal solution always guarantees a k-size optimal solution for $k = 2t + 1$, since every k-group of this size is contained in a t-group. The reverse is not true; there are k-size optimal solutions that are *not* t-distance optimal for $t = (k-1)/2$. For example, consider a complete graph. A 1-distance optimal solution trivially guarantees an optimal solution to the DCOP. The lower bound on 3-size optimal solutions in Section 2.2 is known to be tight for this case, so k-size does not guarantee optimality in this case. Complete graphs are an extreme (and somewhat artificial) example, but we might expect similar advantages for t-distance optimality when there are hub nodes with many connections or densely-connected subgraphs.

4. ALGORITHM DESCRIPTION

Both k-size and t-distance optimal solutions can be computed using local search, and existing algorithms such as MGM[13] and DSA [7] apply local search to find 1-size (0-distance) optimal solutions. We introduce *DALO (Distributed Asynchronous Local Optimization)*, an algorithm which can compute either k-size or t-distance optimal solutions for any setting of k or t (allowing a tradeoff in solution quality and computation/communication cost). DALO-k and DALO-t have the same high-level design, and share most of their source code in our implementation.

DALO is an anytime algorithm that starts from a random initial assignment and monotonically improves solution quality.[2] Nodes belong to groups, defined by either k-size or t-distance criteria, and each group has a unique leader. *Fringe nodes* of a group are are directly connected to a group member, but are not members themselves. DALO has three phases that we describe below:

1. **Initialization:** Agents send initialization messages to nearby agents, which are used to find all of the k or t groups in the constraint graph and assign each group a unique leader.

2. **Optimization:** Each group leader computes a new optimal assignment for the group, assuming that all fringe nodes maintain their current assignment.

3. **Implementation:** The group leader implements the new assignment if if is an improvement, using an asynchronous locking/commitment protocol.

The initialization phase is executed once at the beginning, while phases 2 and 3 are executed in parallel by all groups until quiescence. Pseudocode for DALO is presented in Algorithm 1. The code is structured as a set of message handlers that act on arriving messages, updating internal state as necessary.

4.1 Initialization and Group Selection

There are two goals for the initialization phase. The first is to identify all k-size or t-distance groups in the constraint graph and define a unique leader for each group. The second is to provide group leaders with the information necessary to compute new optimal assignments in the optimization phase. Agents begin by broadcasting discovery information for a limited number of hops. Full constraint tables are broadcast to a distance of $\lfloor \frac{k}{2} \rfloor$ or t hops (line 3), so that group leaders have all constraints for possible group members. The initial random assignments are sent for $\lfloor \frac{k}{2} \rfloor + 1$ or $t + 1$ hops (line 2), so that group leaders know the current assignment of all fringe nodes. No messages are ever sent further than these initialization messages. From these messages agents construct a local subgraph of the constraint graph. This is used to determine which groups the agent could be a leader of, and to compute a breadth-first spanning tree for future communication.

For k-size groups, we are only interested in connected groups of size exactly k, since all others are redundant (they are strict subsets of a larger k-group). A connected group is a set of agents for which the subgraph containing only these agents is connected. The leader of the group is one of the central nodes of the group, to minimize communication costs. If there are multiple central nodes, the one with the minimum unique id number is the leader.[3] Each node uses a brute-force approach to compute all possible k-size groups that

[2]In our implementation quality may go down briefly as new assignment for large groups are implemented, but this could be avoided.

[3]We assume that all nodes have a unique id, included in all messages (e.g., a MAC address).

Algorithm 1: DALO: Distributed Asynchronous Local Optimization

```
1  Initialization ;
2     Send_ValueMsg(myNeighbors, myID, myValue, t + 1);
3     Send_ConstraintMsg(myNeighbors, myID, myConstraint, t);
4  When received ValueMsg(ID, Value, TTL) ;
5     Update_LocalView(VariableList, ID, Value);
6     if TTL > 0 then
7           Send_ValueMsg(myNeighbors, ID, Value, TTL - 1);
8     end
9     if VariableList[ID].Value != Value and VariableList[ID].IsFringe() then
10          changeFlag = true;
11    end
12 When received ConstraintMsg(ID, Constraint, TTL) ;
13    Update_Constraint(ConstraintList, ID, Constraint);
14    if TTL > 0 then
15          Send_ConstraintMsg(myNeighbors, ID, Constraint, TTL - 1);
16    end
17 When fringe node value changed (changeFlag is true);
18    Solution = Solve(VariableList, ConstraintList);
19    LockTreeRoot = ConstructBreadthFirstTree(VariableList, Solution);
20    changeFlag = false;
21    if find gain then
22          if lockingFlag then
23                forall child in LockTreeRoot.Children do
24                      Send_UnlockMsg(VariableList[child.ID], myID, child);
25                end
26                RelockTimer.activate();
27                lockingFlag = false;
28          end
29          gainFlag = true;
30    end
31 When there is an unrealized local gain (gainFlag is true);
32    if !lockingFlag and RelockTimer is expired then
33          LockValue = LockTreeRoot.LockValue;
34          forall Node in LockTreeRoot.Children do
35                Send_LockMsg(VariableList[Node.ID], myID, Node);
36          end
37          lockingFlag = true;
38    end
39 When received LockMsg(ID, Node);
40    if not locked or LockValue == Node.LockValue then
41          LockSet.add(ID);
42          LockValue = Node.LockValue;
43          forall child in Node.Children do
44                Send_LockMsg(VariableList[child.ID], ID, child);
45          end
46          Send_AcceptMsg(VariableList[Node.Parent.ID], myID);
47    else
48          Send_RejectMsg(VariableList[Node.Parent.ID], myID);
49    end
50 When received UnlockMsg(ID, Node);
51    LockSet.remove(ID);
52    forall child in Node.Children do
53          Send_UnlockMsg(VariableList[child.ID], ID, child);
54    end
55 When received AcceptMsg(ID);
56    AcceptSet.add(ID);
57    if AllAccepted(AcceptSet, LockTreeRoot) then
58          ChangeValue(LockValue);
59          Send_ValueMsg(myNeighbors, myID, myValue, t + 1);
60          forall child in LockTreeRoot.Children do
61                Send_CommitMsg(VariableList[child.ID], myID, child);
62          end
63          gainFlag = false;
64    end
65 When received RejectMsg(ID);
66    forall child in LockTreeRoot.Children do
67          Send_UnlockMsg(VariableList[child.ID], myID, child);
68    end
69    RelockTimer.activate();
70    ClearAll();
71 When received CommitMsg(ID, Node);
72    LockSet.remove(ID);
73    ChangeValue(LockValue);
74    Send_ValueMsg(myNeighbors, myID, myValue, t + 1);
75    forall child in Node.Children do
76          Send_CommitMsg(VariableList[child.ID], ID, child);
77    end
```

it leads. Each node leads exactly one t-distance group, which contains all nodes within t hops (i.e., all nodes that the leader received constraint information messages from). However, some groups are redundant because they are subsumed by or equivalent to another. We eliminate redundant groups by having each agent send one additional message to its neighbors containing the ids of the group members. When groups are identical only the one with minimum leader id is used.

4.2 Computing Optimal Group Assignments

After initialization, all group leaders compute new optimal assignments for their groups in parallel. The optimization is performed using the same method for DALO-k and DALO-t— the leader node uses a centralized variable elimination algorithm comparable to DPOP [14] to solve the subproblem for the local group. To do this optimization, the leader must assume that all fringe nodes maintain their current assignment, known from the initialization messages. However, nodes can change assignments at any time while the algorithm is running because they are simultaneously part of many overlapping groups. When a nodes changes assignment, it broadcasts the new assignment to a distance of $\lfloor \frac{k}{2} \rfloor + 1$ or $t + 1$ to ensure that all group leaders have the latest assignment (line 59 and line 74). As soon as a group leader receives and update assignment message from any fringe node, the optimization is restarted using the new information (lines 18–30), where the leader computes the new optimal assignment in the group and constructs a tree rooted on $LockTreeRoot$ for forwarding the lock requests. After that, it switches the $changeFlag$ off to indicate the change on fringe nodes has been handled. The leader will attempt to lock the entire group if the new assignment is strictly better and the leader is not locking. The leader needs to give up the previous lock attempt by unlocking the entire group if it is locking while receiving updates from fringe nodes. In that case, it needs to switch the $gainFlag$ on so that it will consider to implement the new assignment later on. The computation is not affected by updates to group members, since the optimization selects the best possible assignment for the group nodes regardless of the current assignment.

There are several reasons that we adopt a centralized solver for local groups in DALO-k and DALO-t— providing limited partial centralization [11] — though in principle any DCOP solver could be used. It is relatively simple to implement, and allows us to focus the current analysis on the differences between k-size and t-distance optimality, rather than the choice of optimization algorithm. The centralized approach also seems to be quite efficient in this application, where the group sizes are typically small. A single group may need to optimize several times before the algorithm converges, and sending all of the constraint information once during initialization may be more efficient than sending constraint information several times as it is needed.

4.3 Implementing Assignments

Once a group leader has found a new optimal assignment, it needs to communicate it to the members for them to implement. Many groups are trying to find and implement new assignments in parallel, and we want solution quality to improve monotonically (which also ensures convergence). To ensure monotonicity, we must ensure that overlapping groups (including fringe nodes) do not try to change assignments at the same time, which would result in implementing new assignments based on stale information and potentially degrade overall solution quality.

Our approach implements new assignments using an asynchronous protocol based on a standard lock/commit pattern. The leader sends locks to all group members and fringe nodes (lines 34–36), which accept the request unless they have already locked on a different assignment (multiple locks on the same assignment are acceptable, line 40). Each node maintains a $LockSet$ which represents the set of nodes that have locked it. A node is added to $LockSet$ when its lock request is accepted (line 41). If all nodes accept, the leader sends a commit message and the assignment is implemented (lines 56–64), where the leader maintains an $AcceptSet$ – the set of nodes that have accepted the lock request. It needs to switch the $gainFlag$ off when the group optimal assignment is implemented. Otherwise, the leader unlocks all nodes and backs off to prevent deadlock (line 65), waiting for a random interval before attempting to lock again (line 69). Leaders also delay for a random interval before the first lock attempt to minimize conflicts. When a node commits the assignment it broadcasts the new assignment to the necessary group leaders and is able to accept new lock requests. Leaders receiving new assignment information unlock any nodes they have locked and start computing a new optimal assignment.

In our experimental results, we find that our asynchronous approach is much more efficient than the existing synchronous algorithm. However, the costs of locking and conflicts are still a very important factor in the performance of the algorithm. We implemented two techniques to improve the locking protocol:

Subset Locking (SL): Subset locking requests locks only from the subset of nodes that are changing assignment and their immediate neighbors, reducing the probability of conflicts.

Partial Synchronization (PS): In cases where multiple overlapping groups want to change to a new assignment, a very useful heuristic is to allow the group with the highest gain to change first. This is easy to implement in a synchronous algorithm, but more difficult in an asynchronous environment. We can achieve many of the benefits of this heuristic by "pooling" lock requests at each node for a small fixed time window. Once the timer expires, the node accepts the lock request with the largest gain (which is sent in the initial lock request).

5. EXPERIMENTAL EVALUATION

We test DALO in simulation, using a novel asynchronous testbed and performance metrics.[4] In addition to comparing against KOPT we examine tradeoffs between k-size and t-distance optimality.

5.1 Testbed and Performance Metrics

Both synchronous and asynchronous DCOP algorithms are commonly tested using synchronous simulations. Our experiments use an asynchronous simulator to provide a more accurate performance evaluation and highlight differences between synchronous and asynchronous algorithms. We developed a simulator based on DAJ [15], an open source toolkit that provides low-level messaging support and visualization for distributed algorithms.

A challenge in evaluating DCOP algorithms is that performance depends on both messaging and computation costs, which vary across hardware platforms. Some algorithms may be suited to applications with relatively cheap messaging, or vice versa. Cycle-Based Runtime (CBR) [4] is a measure designed to capture both communication and computation latencies.[5] We adapt CBR to an asynchronous context to define the *Computation/Communication Ratio* (CCR) metric. This metric is based on the concept of *global time* (instead of synchronized cycles, which do not exist in our sim-

[4]Code for the DALO algorithm, the testbed framework, and random problem instance generators are posted online in the USC DCOP repository at `http://teamcore.usc.edu/dcop`.

[5]Silaghi et al [16] propose an alternative metric, but it is non-trivial to adapt to our simulation framework.

ulations). During a global time step every node is allowed to process all incoming messages in it's queue and send as many messages as desired to it's immediate neighbors. Messages are delivered on the next time step (with no message loss), so communicating with distant nodes requires multiple time units.

A key difference in our work with synchronous simulators is that computations may also take more than one time step to complete. Our measure of computation cost is the number of constraint checks required (as in CBR), where a check queries the value of a constraint for a single assignment. The CCR setting defines the number of constraints assignments that may be evaluated in a global time step (as defined above). For example, CCR = 0.01 allows each node to process up to 100 checks in a time step. The results of computations are delayed the requisite number of time steps, and may be interrupted in the interim. We vary the setting of CCR in our experiments to test algorithms across a broad range of possible settings with different relative cost for sending messages and computation. The setting CCR=0 (zero-cost computation) is closest to the synchronous setting, although KOPT makes some additional assumptions as discussed below.

5.2 Benchmark KOPT Algorithm

Katagishi and Pearce's KOPT [10] is the only existing algorithm for computing k-size optima for any k. We use it as a benchmark in our experiments because it has comparable functionality and has outperformed MGM, DSA, and other incomplete algorithms in prior experiments [10]. The structure of KOPT is similar to DALO, in that it finds k-size groups in the graph, computes optimal assignments for these groups, and implements these assignments to monotonically improve global reward. However, KOPT is based on synchronous mediation, and the details of how each phase operates are quite different from our methods. We obtained the source code for KOPT from the authors and modified it as little as possible to run in the DAJ testbed; the content of messages and computations was not altered. The KOPT algorithm synchronizes nodes so they are all executing the same operations at once. This would generally require additional messages, but we allow the algorithm to synchronize without messages so as not to penalize it in our experiments (if anything, it has an advantage).

5.3 Results

We present results for three classes of DCOPs with constraint graphs generated using (1) $G(n, M)$ random graphs [2] (2) Barabasi-Albert (BA) scale-free graphs [1], and (3) non-linear preferential attachment (NLPA) graphs based on the BA model, but with a stronger bias towards having many nodes with few connections. All graphs have 100 nodes and density four (an average of four edges per node) unless otherwise noted. Variables have domain size 10, and rewards are integers drawn from U[0,10000]. All results are averaged over 50 sample instances.

We compare KOPT, DALO-k, and DALO-t for CCR setting ranging from low to high computation costs: 0, 0.01, and 0.1. The results appear in Figures 4(a)–4(j). We plot normalized solution quality against global time for each algorithm. All algorithms start from the same random initial assignment. Quality is normalized by subtracting the reward for the initial assignment and dividing by the best known reward for each instance (a proxy for the global optimum). Error bars are omitted for readability, but spot t-tests are all significant for the comparisons described below. Both DALO-k and DALO-t use subset locking and partial synchronization with a window size of 7 time steps (selected after initial testing).

Plots 4(a)–4(h) test both KOPT and DALO for a wide range of graph classes and CCR settings, but only include two comparable settings of k and t (for readability). NLPA with CCR=0.01 is omitted for space, but shows the same pattern as the other two NLPA plots. We include three different settings of k and t in plots 4(i) and 4(j), but omit KOPT for readability. There are two desirable features of algorithm performance that we focus on. The first is converging to a high final solution quality in the limit (at the right edge of a plot). The second is converging more quickly to good solutions, evidenced by higher quality at lower values of global time.

Table 1: Statistics for k-size and t-distance groups over 50 instances. Num k is the total number of connected k-groups, and Max k is the maximum number of groups assigned to any leader. Size t is the average size of a t-group, Max t is the largest t-group, and Unique t is the number of non-redundant t-groups (all k-groups are unique by definition).

	Num k	Max k	Size t	Max t	Unique t
Random	298	8	5.1	10	96
Scale-free	297	27	5.1	25	90
NLPA	293	65	4.9	53	62

The first important result is that both DALO-k and DALO-t substantially outperform KOPT for all eight conditions shown in 4(a)–4(h), converging both more quickly and to higher final solution quality. When computation is instantaneous (CCR=0) our environment is close to a synchronous setting, and even in this case both DALO algorithms outperform KOPT. The difference is greatest for high computation costs because KOPT wastes time waiting for slow computations to finish.

We now turn to comparing DALO-k and DALO-t. DALO-t has a higher final solution quality in every test case, though in some the difference is small. Convergence speed depends on both the graph properties and CCR setting. DALO-k tends to converge faster in random graphs (4(a)–4(c)), convergence is roughly equivalent in scale free graphs (4(d)–4(f)), and DALO-t converges faster in NLPA graphs (4(g)–4(f)). DALO-t tends to improve relative to DALO-k as computation cost increases (higher CCR settings). For example, DALO-k converges noticeably faster for CCR 0 on random graphs, but only slightly faster for CCR 0.01 on random graphs. One factor in these trends is the difference in how DALO-k and DALO-t handle nodes with many connections, which are more prevalent in scale-free and NLPA graphs. In DALO-k these nodes result in many overlapping k-groups, while DALO-t merges these into a single t-group which is more efficient in many cases.

Figures 4(i) and 4(j) show additional settings of k and t for scale-free graphs of density 2 and CCR settings of 0 and 0.1. Note that $k = 1$ is equivalent to $t = 0$. As the group size increases with larger values of k or t, convergence time becomes significantly slower, but final solution quality improves. The theoretical quality bounds also improve with higher settings of k and t. We observe a large improvement in solution quality for moving from $k = 1$ to $k = 3$ or $t = 0$ to $t = 1$, and a positive but diminishing return for additional increases in group size.

The primary tradeoffs between k-size and t-distance are in the number of groups, their size, and how the group calculations are handled. Table 1 presents statistics about the number and size of groups for each class of graphs. For these problems the raw number of t-groups and the size of k-groups are always 100 and 3, respectively. The number of k-groups and the average size of a t-groups remains relatively constant for the different cases, but the other values change significantly. NLPA graphs are characterized by large "hub" nodes with many connections. This results in large t-groups, and more redundant groups (though the problems remain significantly decentralized). It also results in a very large number

Table 2: Number of messages sent and locking conflicts, averaged over 50 random graphs; CCR setting in parentheses.

	Msg(0)	Conf(0)	Msg(.01)	Conf(.01)	Msg(.1)	Conf(.1)
t-opt	56189	569	37727	429	22401	199
k-opt	26958	326	20104	259	10720	59
KOPT	145244	0	42857	0	6689	0

of k-groups that are assigned to a single leader (the Max k column). Both of these factors tend to result in large computational burdens on highly-connected nodes in NLPA graphs. We also evaluated messaging statistics and the number of locking conflicts on random graphs, shown in Table 2. DALO-t generally has more locking conflicts, driven by the larger group sizes. KOPT sends fewer messages for higher CCR due to periods of inactivity.

The overhead of detecting and resolving conflicts is a large factor in the performance of DALO. We implemented two techniques to mitigate this: partial synchronization (PS) and subset locking (SL). Results using different combinations of PS and SL are presented in Figure 4(k) for DALO-t. PS and SL substantial improve performance, both individually and in combination. The PS method is particularly interesting, since it is based on the DCOP-specific insight that selecting the group with the highest gain to implement a change is a powerful heuristic in these problems. The final experiment in Figure 4(l) shows the scalability of DALO as we increase the number of nodes tenfold from 100 to 1000 for random graphs. The time necessary for both DALO-k and DALO-t to converge is almost constant across this range of problem size.

6. CONCLUSION

Within the growing literature on incomplete DCOP methods [17, 18, 3, 13, 6] k-size optimality is an important direction because of the focus on group coordination and the availability of reward-independent quality bounds. Unfortunately, considering only group size as a criteria for coordination and the lack of efficient algorithms has so far limited the applicability and impact of this work. Our four main contributions overcome some of these limitations. First, t-distance optimality introduces a novel criterion for group coordination, posing new research questions about how incomplete algorithms that rely on group coordination should select and coordinate groups. One important insight is that k-size and t-distance make different tradeoffs in the overall size and number of groups, impacting theoretical properties and algorithmic performance. Second, solution quality bounds we derive for t-distance improve known bounds for k-size optimality. Third, the asynchronous DALO algorithm provides a general framework for computing both k-size and t-distance optimality, significantly outperforming KOPT in our experiments and making applications of high values of t and k viable. Fourth, DALO allows us to investigate tradeoffs: DALO-t consistently converges to better solutions in practice than DALO-k. DALO-t also converges more quickly that DALO-k in many settings, particularly when computation is costly and the constraint graph has large hub nodes. However, DALO-k converges more quickly on random graphs with low computation costs.

Investigating additional criteria for group selection (e.g., hybrids of k-size and t-distance) is a key avenue for future work. Another is to compare the privacy implications of different grouping criteria, which has been an important research thrust in k-size optimality [9]. Finally, it is important to thoroughly study the differences between k-size/t-distance optimality and the max-sum algorithm [6]. The two classes of algorithms offer different types of guarantees based on different assumptions, and future work could clarify tradeoffs of each approach.

Acknowledgements

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

7. REFERENCES

[1] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439), 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*, 2008.

[4] J. Davin and P. J. Modi. Impact of problem centralization in distributed constraint optimization algorithms. In *AAMAS*, 2005.

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

[6] A. Farinelli, A. Rogers, and N. Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *DCR*, 2009.

[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*. Kluwer, 2003.

[8] A. Gershman, A. Meisels, and R. Zivan. Asynchronous forward-bounding for distributed constraints optimization. In *First International Workshop on Distributed and Speculative Constraint Processing*, 2006.

[9] R. Greenstadt. An analysis of privacy loss in k-optimal algorithms. In *DCR*, 2008.

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

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

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

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

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

[15] W. Schreiner. A java toolkit for teaching distributed algorithms. In *ITCSE*, 2002.

[16] M. C. Silaghi, R. N. Lass, E. A. Sultanik, W. C.Regli, T. Matsui, and M. Yokoo. The operation point units of distributed constraint solvers. In *DCR*, 2008.

[17] 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), 2005.

[18] R. Zivan. Anytime local search for distributed constraint optimization. In *AAMAS*, 2008.

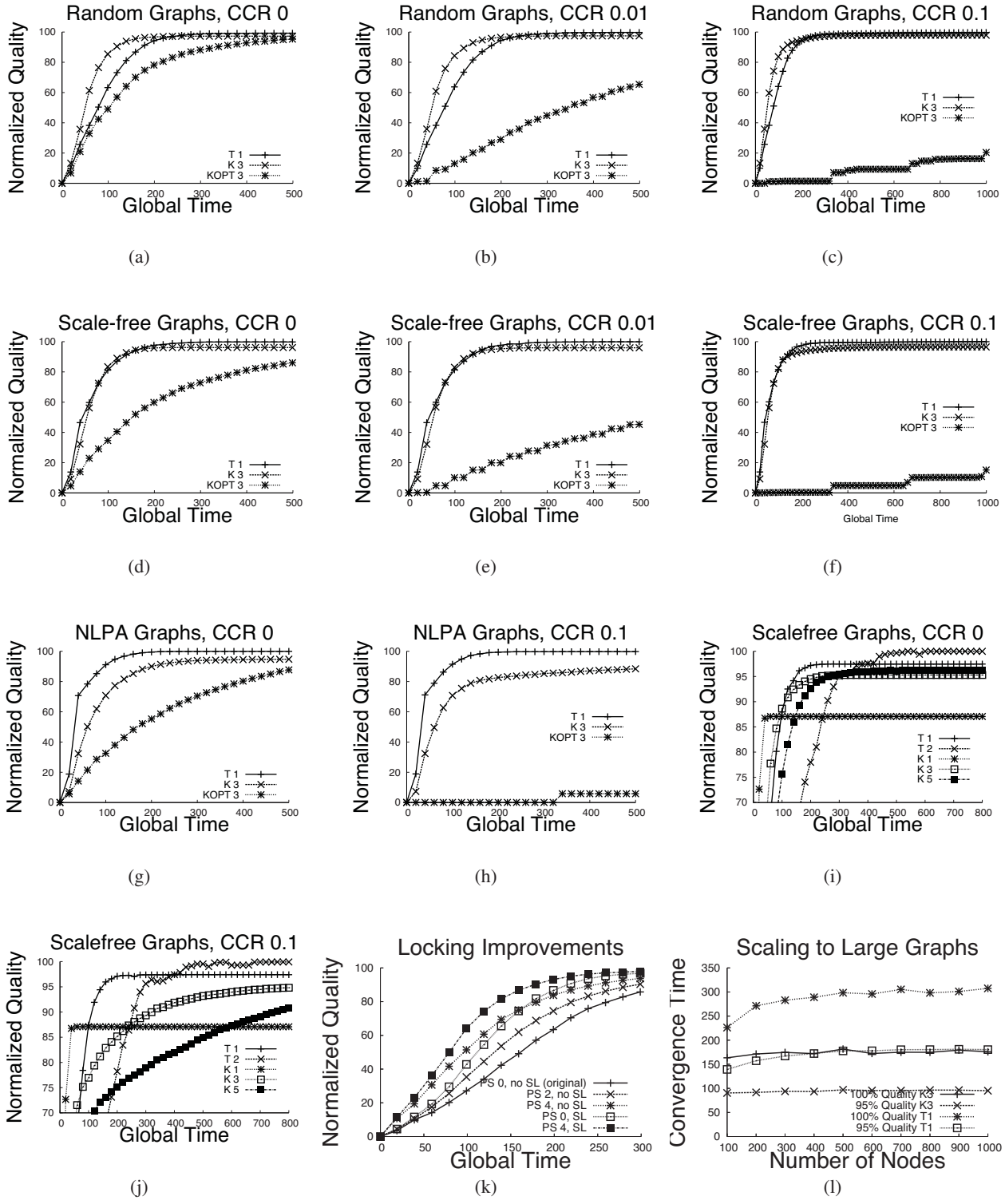


Figure 4: Experimental results comparing DALO-k, DALO-t, and KOPT.