KOPT : Distributed DCOP Algorithm for Arbitrary k-optima with Monotonically Increasing Utility

Hideaki Katagishi, Jonathan P. Pearce

University of Southern California
Computer Science Department
{katagish@usc.edu, jppearce@usc.edu}

Abstract. A distributed constraint optimization problem (DCOP) is a formalism that captures the rewards and costs of local interactions within a team of agents. Because complete algorithms to solve DCOPs are unsuitable for some dynamic or anytime domains, researchers have explored incomplete DCOP algorithms that result in locally optimal solutions. One type of categorization of such algorithms, and the solutions they produce, is k-optimality; a k-optimal solution is one that cannot be improved by any deviation by k or fewer agents. There are no k-optimal algorithms (k>3) so far. In addition, the quality of solution existing algorithm can produce is fixed. We need different algorithms for different optimality. This paper introduces the first DCOP algorithm which can produce arbitrary k-optimal solutions.

1 Introduction

Distributed Constraint Optimization Problems (DCOP) has been an important research area for a long time since many real world problems can be modeled by them. Traditionally, researchers have focused on figuring out a globally optimal solution to DCOPs by using complete algorithms, such as ADOPT[Modi et al., 2005], OptAPO[Mailler and Lesser, 2004] or DPOP[Petcu and Falings, 2005]. However, as the size of problems becomes larger, we cannot underestimate the significant computation and communication cost. Thus, incomplete algorithms, which can generate sub-optimal solutions, are the center of attention. Those algorithms include DSA[Fitzpatrick and Meertens, 2003], DBA[Yokoo and Hirayama, 1998], MGM-1[Maheswaran et al., 2004](a simplified version of DBA), MGM-2[Maheswaran et al., 2004], SCA-2[Maheswaran et al., 2004].

Recently, new measure of optimality in DCOP, **k-optimality** [Pearce et al., 2007] was introduced. A k-optimal solution is one that cannot be improved by any deviation by k or fewer agents. Existing DCOP algorithms can be classified by this measure. DSA, DBA and MGM-1 produce 1-optimal solutions. SCA-2 and MGM-2 produce 2-optimal solutions. Complete algorithms produce n-optimal solutions.

However, **there are no k-optimal(3<k<n) algorithms so far**. And also, the optimality of solutions each algorithm can generate is fixed. We need to use different algorithms for different levels of optimality.

Now, we can calculate **lower bounds on solution quality for arbitrary k-optima**[Pearce et al., 2007]. If we have an incomplete algorithm which can generate arbitrary k-optimal solution and combine those algorithms with quality guarantee calculation, it must be a very strong tool for finding reasonable DCOP solutions. Thus, finding tunable k-optimal algorithms is a critical issue.

In this paper, we propose the **first incomplete algorithm for arbitrary k-optima**, called KOPT. This is a **partially distributed** and **synchronous** communication algorithm. The **global utility is strictly increasing** by using KOPT. In the rest of this paper, we present a formalization of the DCOP problem and an explanation of k-optimality. Then, we explain the algorithm with a 3-optimal example. Next, we present the result of experiments and lastly we discuss the conclusion.

2 DCOP and k-optima

We consider a DCOP in which each agent controls a variable to which it must assign a value. Constraints exist on subsets of these variables; each constraint generates a cost or reward to the team based on the values assigned to each variable in the corresponding subset. We assume in this paper that each agent controls a single variable.

Formally, a DCOP is a set of variables (one per agent) $N := \{1,...,n\}$ and a set of domains $A := \{A_1,...,A_n\}$, where the i_{th} variable takes value $a_i \in A_i$. We denote the assignment of the multi-agent team by $a = [a_1 \cdots a_n]$. Valued constraints exist on various subsets $S \subset N$ of these variables. A constraint on S is expressed as a reward function $R_S(a)$. This function represents the reward generated by the constraint on S when the agents take assignment a; costs are expressed as negative rewards. θ is the set of all such subsets S on which a constraint exists, and no $S \in \theta$ is a subset of any other $S \in \theta$. For convenience, we will refer to these subsets S as "constraints" and the functions $R_S(\cdot)$ as "constraint reward functions." The solution quality for a particular complete assignment a is the sum of the rewards for that assignment from all constraints in the DCOP: $R(a) = \sum_{S \in \theta} R_S(a)$.

In [Pearce et al., 2006], the *deviating group* between two assignments, a and $\tilde{a}$, was defined as $D(a, \tilde{a}) := \{i \in N : a_i \neq \tilde{a}_i\}$, i.e. the set of variables whose values in $\tilde{a}$ differ from their values in a. The *distance* between two assignments was defined as $d(a, \tilde{a}) := |D(a, \tilde{a})|$ where | · | denotes the size of the set. An assignment a is classified as a *k-optimum* if $R(a) - R(\tilde{a}) \geq 0 \forall \tilde{a}$ such that $d(a, \tilde{a}) \leq k$. Equivalently, at a k-optimum, no subset of k or fewer agents can improve the overall reward by choosing different values; every such subset is acting optimally given the values of the others.

Example *Fig.1 is a binary DCOP in which agents choose values from {0, 1}, with constraints S_1 = {1, 2} and S_2 = {2, 3} with rewards shown. The assignment a = [1 1*

1] is 1-optimal because any single agent that deviates reduces the team reward. However, [1 1 1] is not 2-optimal because if the group {2, 3} deviated, making the assignment $\tilde{a}$ *= [1 0 0], team reward would increase from 16 to 20. The globally optimal solution,* a^* *= [0 0 0] is k-optimal for all k*$\in$*{1, 2, 3}.*

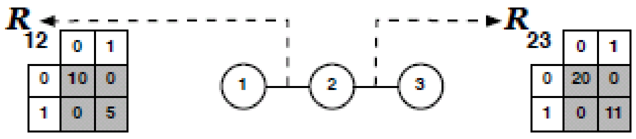


Fig. 1. DCOP Example

3 KOPT

3.1 KOPT Overview

KOPT is the first DCOP incomplete algorithm for arbitrary k-optima. This is partially distributed algorithm and requires synchronous communication. KOPT yields monotonically increasing global utility. This feature is important because in the domain where communication may be halted arbitrarily and each agent cannot change its value any more, randomized algorithms whose utility are not monotonically increasing may end up with highly undesirable solutions.

This algorithm consists of 3 phases. In phase 1, every agent gathers information from nearby agents. In phase 2, every agent calculates the best value assignment to the nearby agents by using the information acquired in phase 1. Then every agent broadcasts it to the agents which belong to the assignment. In phase 3, every agent selects the best value assignment which has the highest utility among the assignments sent by its nearby agents. If all the agents in the assignment X know that all the agents in X have chosen X, they will change their variables according to X. At this point, those agents are called "**committed**". Otherwise, none of agents in X will move.

These three phase make up one iteration. Before proceeding to the detail explanation of each phase, we introduce several important notions in KOPT.

3.2 Group and Mediator

First, let us introduce the notion of group and its mediator. (Figure 2)

At every iteration, every agent forms groups. The groups are not exclusive but overlapping each other, that is, most agents can belong to more than one group. There is one mediator per group, and it gathers the information from its group members (phase1), calculates the best value assignment inside the group (phase2) and broadcasts it to all group members (phase2). It is important to mention that every agent is a mediator of its own group, that is, every agent has its own group.

Every mediator suggests the optimal assignment inside its group and if the group achieves the highest utility among overlapping groups, all the group members can change their value at the end of phase3. Thus, how to choose the group member at each iteration is very important to produce k-optimal solution. If all possible combinations of k agents are covered by groups, KOPT can generate k optimal solutions. (See section 3.4)

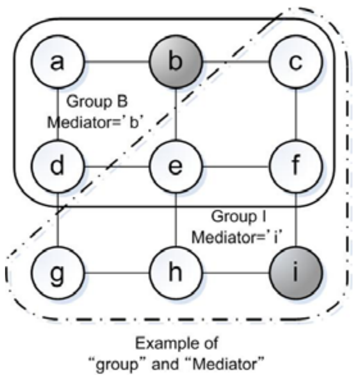


Fig. 2. Group and its Mediator

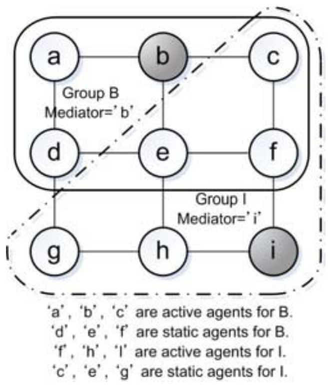

a
b
c
Group B
Mediator=' b'
d
e
f
Group I
Mediator=' i'
g
h
i
'a' , 'b' , 'c' are active agents for B.
'd' , 'e' , 'f' are static agents for B.
'f' , 'h' , 'I' are active agents for I.
'c' , 'e' , 'g' are static agents for I.

Fig. 3. Active and Static agents

3.3 Active agents and Static agents

In each group, there are two kinds of agents, **active agents** and **static agents**. (Figure 3)

Active agents are the agents which can change their value to achieve the highest utility in the group. Static agents surround the active agents and are located at the outer layer of the group.

In order to ensure that the global utility is always strictly increasing in KOPT, they cannot change their values. Before changing its value, every agent in group X confirms that all X's group members have chosen X's best assignment. (phase3) However, they do not care about the outside of the group to make decisions about the next movement. Thus, if active agents are not surrounded by static agents in the group and neighboring groups move at the same time, there can be unexpected combination movements which can cause negative rewards.

3.4 Group formation

How to choose group members is important to guarantee the k-optimality of the solution. If all possible combinations of k agents are covered by groups, KOPT can generate k optimal solutions. It is obvious that the number of active agents in the group should be k.

For every possible group, there are central points (graph theory). Only the agent at the graph center can be the mediator of that group. Thus, there is a clear division of roles. Each mediator has the set of potential groups it is responsible for. It also minimizes the number of communications steps between the mediator and its group members. At each iteration, each mediator chooses the group randomly from its potential groups. After enough iterations, any combination of k active agents has been covered and tried to optimize by the group.

3.5 Detail explanation of each phase

Here, we will explain the detail of each phase by using the 3-optimal example below.(figure 4) The top of the figure is the actual DCOP, the bottom is an illustration of the constraint rewards. The agents take 2 values, black and white. We measure the number of steps by the number of communication. It is common in the DCOP literature since it is assumed that communication is the speed bottleneck. One communication adds the number of steps by one.

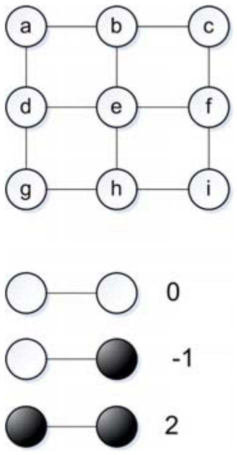


Fig. 4. 3-optimal example

Phase1

Step1-1
Each agent broadcasts it own <value> to all its neighbors (Figure 5).

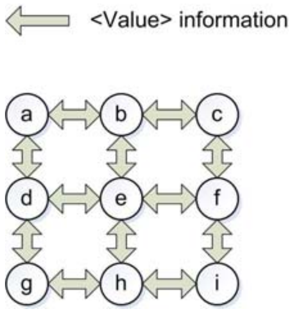


Fig. 5. Step1-1

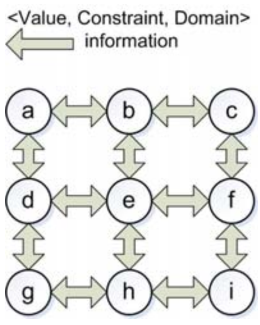
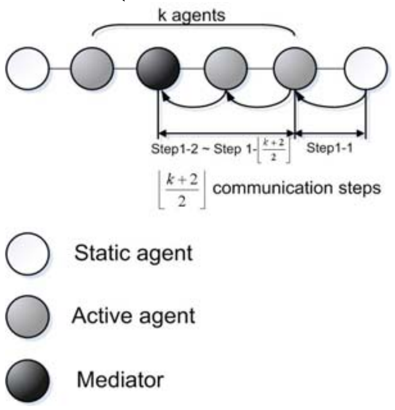


Fig. 6. Step1-2 ~

Step 1-2 ~ Step 1-$\left\lfloor \frac{k+2}{2} \right\rfloor$

Receive <value, constraint, domain> information (<value> in step1-2) from its neighbors and store it into its own storage. Then, broadcast its own <value, constraint, domain> information and the received <value, constraint, domain> information (<value> in step1-2) to all the neighbors (Figure 6).

Repeat this step $\left\lfloor \frac{k}{2} \right\rfloor$ times. Mediators have to know the information about all the potential groups they are responsible for. We have to think of the worst case, which is a chain graph (Figure 7). Thus, it needs $\left\lfloor \frac{k+2}{2} \right\rfloor$ communications steps including step1-1 for the mediator to receive the information from all the potential group members. (Remember that the mediator has to be the graph center of the group.)



k agents
Step1-2 ~ Step 1-$\left\lfloor \frac{k+2}{2} \right\rfloor$
Step1-1
$\left\lfloor \frac{k+2}{2} \right\rfloor$ communication steps
Static agent
Active agent
Mediator

Fig. 7. Worst Case

Example

- By the end of Phase1, agent 'a' knows the value of 'b', 'c', 'd', 'e', 'g' and the constraint and domain of 'b', 'd'.

Phase2

Step 2-1

At the beginning of Phase2, in addition to itself, each agent chooses $k-1$ active agents randomly among the agents it has <value, constraint, domain> information and calculates the best value assignment to them using received information. The group is actually formed here. Non-active neighbor of active agents become static agents for this group.

The central calculation method includes the enumeration of all possible value assignments, branch-and-bounds, ADOPT[Modi et al., 2005] etc.. We can apply **any DCOP complete algorithm** to this central calculation.

Example

- 'a' can calculate best value assignments by changing the values of 'a', 'b', 'd'. (Figure 8)
- 'b' can calculates the best value assignments by changing the values of 'a', 'b', 'c'.(Figure 9)

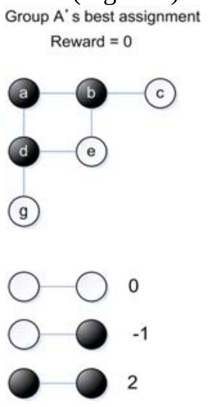


Fig. 8.

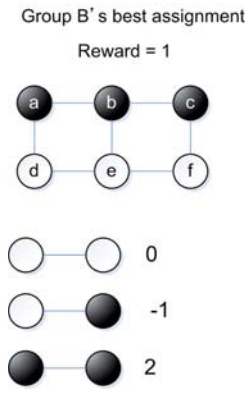


Fig. 9.

Step 2-1(continued)

Broadcast <best value assignment, reward>, the result of central calculation, to all its neighbors.(Figure 10)

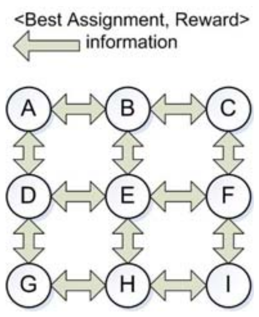


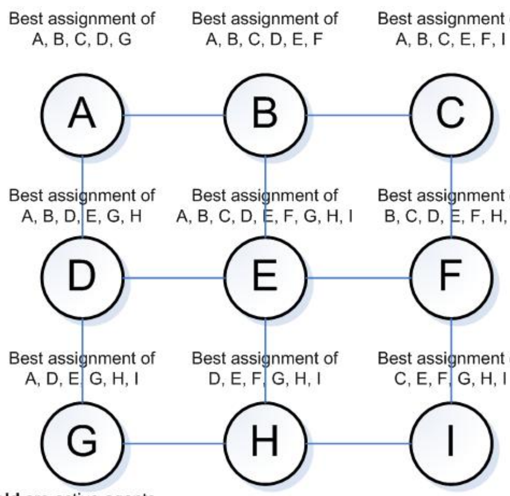
Fig. 10. Step2

Step 2-2 ~ Step 2-$\left\lfloor \frac{k+2}{2} \right\rfloor$

Receive <best value assignment, reward> and forward them to all its neighbors. Repeat Step2-2 $\left\lfloor \frac{k}{2} \right\rfloor$ times. This means "Repeat this step until the best assignment information reaches to all group members". A chain graph is the worst case where the most communications are needed. Thus, it takes $\left\lfloor \frac{k+2}{2} \right\rfloor$ communications for all group members to receive the information from its mediator.

Example

- 'a', 'c', 'd', 'e', 'f' have to know Group B's best value assignment. (Figure 9)
- At the end of Phase2, each agent knows the value assignments of all the groups it belongs to. (Figure 11)


Best assignment of A, B, C, D, G
Best assignment of A, B, C, D, E, F
Best assignment of A, B, C, E, F, I
A
B
C
Best assignment of A, B, D, E, G, H
Best assignment of A, B, C, D, E, F, G, H, I
Best assignment of B, C, D, E, F, H, I
D
E
F
Best assignment of A, D, E, G, H, I
Best assignment of D, E, F, G, H, I
Best assignment of C, E, F, G, H, I
G
H
I

Bold are active agents

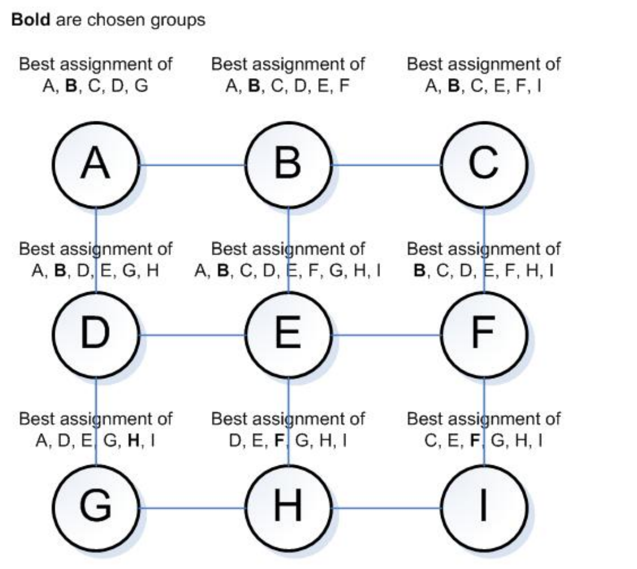
Group A' s members	: **a**, **b**, c, **d**, e, g	best assignment' s reward = 0
Group B' s members	: **a**, **b**, **c**, d, e, f	best assignment' s reward = 1
Group C' s members	: a, **b**, **c**, e, **f**, i	best assignment' s reward = 0
Group D' s members	: a, b, **d**, **e**, f, **g**, h	best assignment' s reward = 0
Group E' s members	: b, c, d, **e**, **f**, g, **h**, i	best assignment' s reward = 0
Group F' s members	: b, **c**, e, **f**, h, **i**	best assignment' s reward = 1
Group G' s members	: a, **d**, e, **g**, **h**, i	best assignment' s reward = 0
Group H' s members	: d, e, f, **g**, **h**, **i**	best assignment' s reward = 1
Group I' s members	: c, e, **f**, g, **h**, **i**	best assignment' s reward = 0

Fig. 11. At the end of Phase2

Phase3

Step 3-1
At the beginning of phase3, each agent chooses the best group whose assignment has the highest utility among the groups it belongs to. Each agent will choose the group which has the smallest group ID if those utility are the same.

Example
B, F, H's assignments have the same reward in agent 'e'. So, 'e' will choose B. (Figure 12)


Bold are chosen groups
Best assignment of A, **B**, C, D, G
Best assignment of A, **B**, C, D, E, F
Best assignment of A, **B**, C, E, F, I
A
B
C
Best assignment of A, **B**, D, E, G, H
Best assignment of A, **B**, C, D, E, F, G, H, I
Best assignment of **B**, C, D, E, F, H, I
D
E
F
Best assignment of A, D, E, G, **H**, I
Best assignment of D, E, **F**, G, H, I
Best assignment of C, E, **F**, G, H, I
G
H
I

Bold are active agents

Group A' s members	: **a**, **b**, c, **d**, e, g	best assignment' s reward = 0
Group B' s members	: **a**, **b**, **c**, d, e, f	best assignment' s reward = 1
Group C' s members	: a, **b**, **c**, e, **f**, i	best assignment' s reward = 0
Group D' s members	: a, b, **d**, **e**, f, **g**, h	best assignment' s reward = 0
Group E' s members	: b, c, d, **e**, **f**, g, **h**, i	best assignment' s reward = 0
Group F' s members	: b, **c**, e, **f**, h, **i**	best assignment' s reward = 1
Group G' s members	: a, **d**, e, **g**, **h**, i	best assignment' s reward = 0
Group H' s members	: d, e, f, **g**, **h**, **i**	best assignment' s reward = 1
Group I' s members	: c, e, **f**, g, **h**, **i**	best assignment' s reward = 0

Fig. 12. At the beginning of Phase3

Step 3-1(Continued)
Each agent broadcast its <next value> in the best assignment it has chosen. (Figure 13)

Step 3-2 ~ Step 3- $k-1$
Receive <next value> information and forward them to all its neighbors.
Repeat Step 3.2 $k-2$ times. This means "Repeat until **every active member** in group X is informed whether or not **every X's active member** have selected X's assignment." (Figure 14) If so, group X is called **committed** and every X's active member changes its value. Otherwise, none of the members will move.

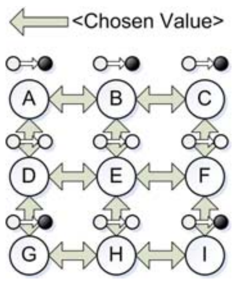


Fig. 13. Step3

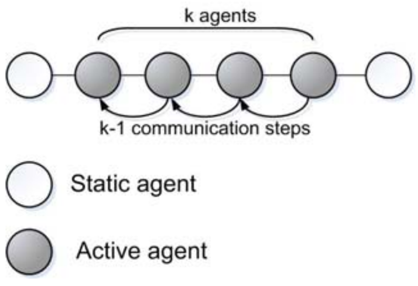


Fig. 14. Worst Case (Chain graph)

Static members do not have to know anything because they will not change their values anyway. However, do the group X's active members need to know whether or not **X's static members** have selected X's assignment? The answer is no. They have to know **only about active agents**. We will explain the reason later.
The phase3 ensures only the best group which has the highest reward among the overlapping groups can change its value. This mechanism help KOPT converge efficiently.

Example

- 'a' has to know whether 'b', 'c' will follow B's best assignment. If so, 'a' can change its value for sure. Otherwise, A cannot change its value. (Figure 15)
- 'g' has to know whether 'h', 'i' will follow H's best assignment. But, 'h' does not follow H's best assignment. So, 'g' cannot change its value. (Figure 15)

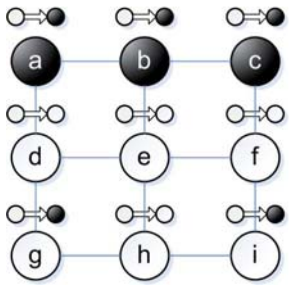


Fig. 15. At the end of Phase3

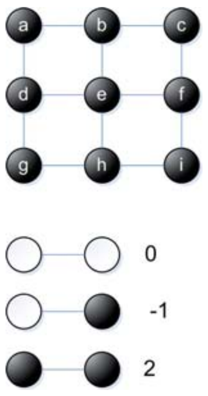


Fig. 16. 3-optimal solution

Repeat

Repeat these three phases until global utility converges. (Figure 16) After 1st Iteration, each agent need not to broadcast constraint and domain information. Those information are invariant over the iterations. Total communication steps in each iteration is $2\left\lfloor \frac{k}{2} \right\rfloor + k + 1$

k=1$\rightarrow$ 2 steps
k=2$\rightarrow$ 5 steps
k=3$\rightarrow$ 6 steps
k=4$\rightarrow$ 9steps
…

KOPT becomes identical to MGM-1 when parameter k is 1. Thus, KOPT is the generalization of MGM-1.

3.6 Committed group in Phase3

In phase 3, do the group X's active members need to know whether or not **X's static members** have chosen X's assignment? The answer is no. However, you may think the static agents in group X might have chosen the other group's active assignment and they might change their value. It might cause negative rewards. Thus, we will show that the proposition, **"Once the active agents in the group X are committed, none of the overlapping groups will not interfere with group X",** is true.

Proof

For simplicity, we can look at the relationship between two overlapping groups at first. There are two ways of overlapping.

One is that two groups are overlapping only by static agents. (Figure 17) In this case, two groups will never interfere with each other because static agents exist between active agents. Since the static members do not change their values, there is no need to identify the group chosen by the static agents.

The other is that two group are overlapping by both active and static agents. (Figure 18)

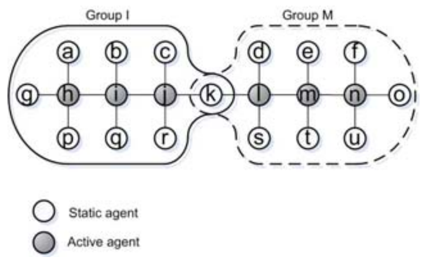


Fig. 17.

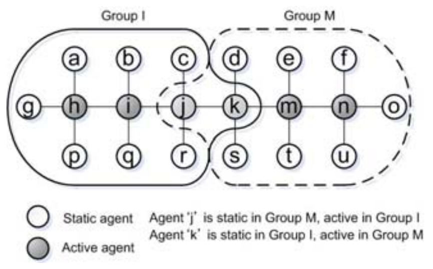

Group I
Group M
a
b
c
d
e
f
g
h
i
j
k
m
n
o
p
q
r
s
t
u
Static agent
Active agent
Agent 'j' is static in Group M, active in Group I
Agent 'k' is static in Group I, active in Group M

Fig. 18.

In the latter case, suppose all I's active agents are committed, we can think about two possible situations.

Case1: all active agents of I recognize 'i' as a mediator.(figure 19)

In this case, Group I has higher utility than Group M. Since all agents in overlap area ('j' and 'k') have the same information, I's static('k') agents will also choose Group I. So, I's static agents will not change their value. This means Group I will never interfere with Group M.

Case2: Some of active agents('j') in group I recognize 'm' as a mediator, however the overlap area of Group I's active assignment('j') happens to be the same as Group M's. So, active agents in Group I are committed. (Figure 20)

In this case, Group M has higher utility than Group I and the overlap area of Group I's active assignment are the same as Group M's assignment. M's assignment should be surrounded by static agents('j'). ('j' has chosen a static assignment of Group M at Phase2. This means 'j' will not change its value whether or not Group M are committed at Phase3.) So, these two assignments will never interfere with each other. So, Group I will never interfere with Group M. In both cases, once all of I's active agents are committed, Group I will never interfere with Group M.

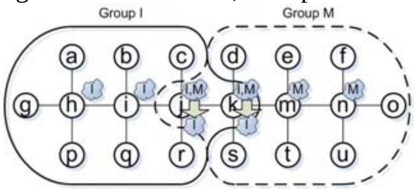


Fig. 19. Case1

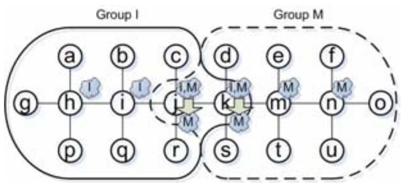
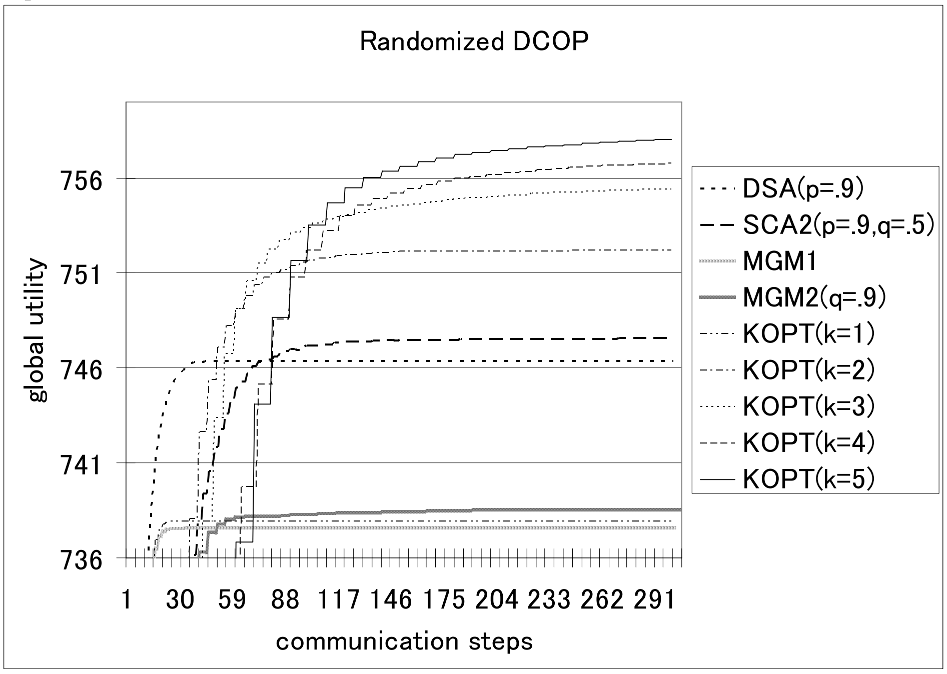


Fig. 20. Case2

We can apply this theory to any two relation between I and I's neighbor group. So we can say, once all of I's active agents are committed, I's movement will never interfere with any overlapping group.

4 Experiment

This is a work in progress report. We considered a randomized DCOP domain used in [Maheswaran et al., 2004]. In this domain, every combination of values on a constraint between two neighboring agents was assigned a random reward chosen uniformly from the set {1, …., 10}. We considered ten randomly generated graphs with forty variables, three values per variables, and 120 constraints. For each graph, we ran 100 runs of each algorithm, with a randomized start state. The algorithms used in this experiment are MGM, DSA(p = 0.9), MGM-2(q=0.9), SCA-2(p=0.9, q=0.5) and KOPT(k=1,2,3,4,5). The central calculation method used in KOPT in this experiment is a simple enumeration of all possible assignments. Please notice that the choice of central calculation method doesn’t affect this experiment’s result as long as it is a DCOP complete algorithm. The parameter p, q in DSA, MGM-2, SCA-2 were chosen such that these algorithms achieved the best performance based on the experiment in [Maheswaran et al., 2004].



As the parameter k of KOPT increased, KOPT achieved higher utility as we expected. KOPT(k>1) achieved higher utility than any other incomplete DCOP algorithm.

MGM1 and KOPT(k=1) produced almost same result because these two algorithms are identical.

Compared to MGM2, KOPT(k=2) achieved higher utility. That is because in MGM2, every agent is either an offerer or a receiver at each iteration. Only the receiver can choose the best offer from offerers. However, in KOPT(k=2), every agent is an offerer(mediator) and receiver at the same time. And every agent can choose the best assignment among the offers including its own offer. Thus, KOPT(k=2) can form a pair of committed agents more efficiently than MGM2.

5 Conclusion

We have found new DCOP algorithm for arbitrary k-optimal, KOPT. Since the quality guarantee for arbitrary k-optima is already found [Pearce et al., 2007], we can analyze the trade-off between the solution quality and the computational cost (in this case, total communication steps), then we can figure out an appropriate parameter k for given situation.

As the parameter k is increasing, the computation cost at each agent is increasing. If k = n, KOPT is equivalent to applying the central calculation method to the entire problem. Thus, if communication is speed bottleneck and the number of k is small enough, KOPT is fast and finds out a reasonable solution compared to other DCOP algorithms.

References

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

[Pearce et al., 2007] J. P. Pearce and M. Tambe, "Quality Guarantees on k-Optimal Solutions for Distributed Constraint Optimization Problems," in Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India, January 6-12, 2007.

[Pearce et al., 2006] J. P. Pearce, R. T. Maheswaran, and M. Tambe. Solution sets for DCOPs and graphical games. In AAMAS, 2006.

[Fitzpatrick and Meertens, 2003] 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.

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

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

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

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