DGOPT: Dynamic Group Optimization to Find Better Group Formations in DCOPs

Elnaz Bigdeli, Maryam Rahmaninia, and Mohsen Afsharchi

Institute for Advanced Studies in Basic Sciences
Zanjan, Iran
{e_bigdeli,m_rahmani,afsharchim}@iasbs.ac.ir

Abstract. A substantial amount of study in multi-agent systems has focused on multi-agent coordination for over twenty years. Many challenges in multi-agent coordination can be modeled as Distributed Constraint Optimization (DCOP). Finding the optimal solution for a DCOP is NP-hard, so using incomplete algorithms that are faster are more desirable. Many incomplete algorithms decompose a DCOP to subgraphs to find solutions to it and maintain the partitioning of the DCOP unchanged during algorithm execution. These algorithms provide local optimal solutions. Decomposition of a DCOP has direct influence on the quality of solutions. With the popularity of incomplete algorithms, finding the best decomposition of a DCOP becomes a major issue. In this paper, we propose the first known learning algorithm by which the leader of each group optimizes its group with the purpose of increasing total utility. The leader agents learn to add/remove agents of their groups. This algorithm works dynamically to optimize the existing groups and we call it Dynamic Group Optimization algorithm (DGOPT). From quality, and convergence time point of view, DGOPT outperforms recent algorithms.

Keywords: Multi Agent Systems, Distributed Constraint Optimization, *t*-distance Optimality

1 Introduction

Multi-agent systems are a popular way to model complex interactions and coordination required to solve distributed problems. A multi-agent system is a network of agents used to perform distributed computation. Networks of agents are heterogeneous and not all agents have direct communication link to one another. Additionally, information is distributed throughout the network either due to privacy concerns or impractically of centralizing. In this network each agent is autonomous entity with local information and has ability to perform an action in cooperative situations in which agents collaborate to achieve a common goal.

Agents need to coordinate their activities to accomplish their collective goals. Distributed Constraint Optimization (DCOP) is a common formalism to represent multi-agent systems in which agents cooperate to optimize a global objective

[10, 13]. DCOP has been applied to different domains. DCOPs are able to model the task of scheduling meetings in large organizations, where privacy needs make centralized constraint optimization difficult [9]. DCOPs are also able to model the task of allocating sensor nodes to targets in sensor networks, where the limited communication and computation power of individual sensor nodes makes centralized constraint optimization difficult [11]. Finally, DCOPs are able to model the task of coordinating teams of unmanned vehicles in disaster response scenarios, where the need for rapid local responses makes centralized constraint optimization difficult [2].

There are two main categories for DCOP algorithms, complete and incomplete algorithms. Complete algorithms always find a configuration of variables that maximizes the global objective function. Adopt (Asynchronous Distributed OPTimization) [11] and DPOP (Dynamic Programming OPtimisation) [13] are two well known complete algorithms. There are lots of works which try to extend the ADOPT algorithm as a complete algorithm [4, 15]. The important point in complete algorithms is that finding DCOP solutions which maximize the global objective function is NP-hard. Some of recent works try to solve this problem [16, 17].

In contrast, incomplete algorithms find semi optimal solutions and do not guarantee to achieve global optimal solution. Algorithms such as Max-Sum [1], Distributed Arc Consistency [3] and KOPT [7] are in this category.

In the most of incomplete algorithms a network of agents is divided to groups in which a DCOP problem is solved locally [7, 10, 18]. The local attempt of agents in groups to solve DCOP leads to solving it globally, but the solution found is not the best.

KOPT and DALO algorithms are two examples of incomplete algorithms that divide the network of agents to subgroups to solve DCOP. k-optimal algorithms guarantee to provide solutions that cannot be improved by any group of k or fewer agents changing their decision. KOPT algorithm is the only incomplete algorithm which works for arbitrary k [7].

DALO is a novel asynchronous incomplete algorithm which works based on t-distance optimality [8, 18]. In DALO algorithm groups are formed based on the distance between nodes in the constraint graph instead of strict limits on group size. There are lots of incomplete algorithms to solve DCOP. The main concern in all of these algorithms is how to form groups because groups formation has direct influence on the quality of solution which is gained. We try to find better group formations through a dynamic approach. This approach works based on the contribution of each agent in the reward of the group.

The structure of the paper is as follows: In section 2, formal definitions of DCOP and t-distance optimality solutions are presented. Section 3, gives a general view of different group formations. In section 4, DALO algorithm and its main issues are described. The proposed algorithm is introduced in section 5. Detail description of dynamic group optimization is given in section 5.1. DGOPT algorithm is introduced in section 6. Experimental results of DGOPT algorithm and its comparison with DALO algorithm are depicted in section 7. Finally, conclusion and future work are presented in section 8.

2 Background

In this section, we will provide some basic definitions about DCOP and t-distance optimality.

2.1 Distributed Constraint Optimization

A DCOP is defined by a set of variables $\mathcal{V} = \{v_1, \dots, v_n\}$, a set of discrete finite domains for each v_i; $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$, and a set of constraints $\mathcal{C} = \{c_1, \dots, c_q\}$. Each variable is controlled by a separate agent capable of communicating with other agents. A joint assignment $\mathcal{A} = \{a_1, \dots, a_n\}$ specifies a value for each variable, in which a_i is the value of agent i. Each constraint includes a set of variables. A constraint defines a real-valued cost based on the values which each agent chooses for its variable. This paper holds in view binary constraints to avoid complexity; that is to say each constraint includes two variables. Thus, for each pair of variables (v_i, v_j), there is a cost function $\mathcal{F}_{ij} : \mathcal{D}_i \times \mathcal{D}_j \longrightarrow \mathfrak{R}$ which determines the value of a constraint. If there is no constraint between v_i and v_j, function $\mathcal{F}_{ij}$ will be 0. A cost function takes values of variables as an input and returns a value as a non-negative number for a given constraint. Utility of agent i for assignment $\mathcal{A}$ is:

$$\begin{aligned} &\mathcal{U}_i(\mathcal{A}) = \sum_{v_j \in \mathcal{V}} \mathcal{F}_{ij}(a_i, a_j) \\ &\text{Where } v_i \leftarrow a_i, v_j \leftarrow a_j, a_i, a_j \in \mathcal{A} \end{aligned} \tag{1}$$

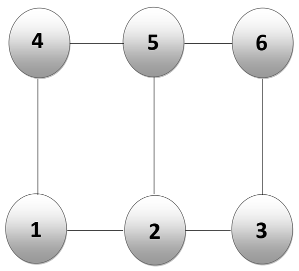
It means the utility of the i_{th} agent is the sum of the cost functions of all the constraints to which an agent belongs.

The goal is to choose values for variables such that a given objective function is maximized. The objective function is described as the sum over a set of cost functions, or valued constraints. As a result, the objective is to maximize:

$$\begin{aligned} &\mathcal{R}(\mathcal{A}) = \sum_{(v_i, v_j) \in \mathcal{V}} \mathcal{F}_{ij}(a_i, a_j) \\ &\text{Where } v_i \leftarrow a_i, v_j \leftarrow a_j, a_i, a_j \in \mathcal{A} \end{aligned} \tag{2}$$

$\mathcal{R}(\mathcal{A})$ is a solution quality for an assignment $\mathcal{A}$ [11, 12].

Figure 1 shows an example of DCOP with 6 variables and 7 constraints with the same cost function. The optimal assignment for this DCOP is $\mathcal{A} = (1, 1, 1, 1, 1, 1)$.



Vi	Vj	Fi,j
0	0	0
0	1	1
1	0	0
1	1	2

Fig. 1. An example DCOP with six binary variables. Each constraint has the same cost function.

2.2 *t*-distance Optimality

Definition 1 *For two different assignments $\mathcal{A}$ and $\mathcal{A}'$:*

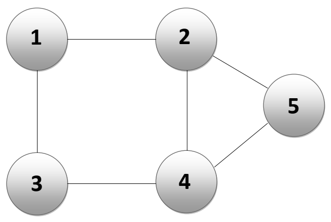
$$\mathcal{D}(\mathcal{A},\mathcal{A}') = \{v_i \in \mathcal{V} \parallel a_i \neq a'_i \ , \ v_i \leftarrow a_i \in \mathcal{A}, v_i \leftarrow a'_i \in \mathcal{A}'\} \tag{3}$$

Put simply, $\mathcal{D}$ is a deviating group between two assignments $\mathcal{A}$ and $\mathcal{A}'$.

Definition 2 *For a pair of variables v_i and v_j, let $\mathcal{T}(v_i, v_j)$ be the shortest distance between them in the constraint graph. Let $\Phi_t(v_i) = \{v_j \| \mathcal{T}(v_i, v_j) \leq t, v_i, v_j \in \mathcal{V}\}$ denote a set of variables that can be reached from v_i within t hops.*

Definition 3 *A DCOP assignment $\mathcal{A}$ is t-distance optimal if $\mathcal{R}(\mathcal{A}) \geq \mathcal{R}(\mathcal{A}')$ for all $\mathcal{A}'$, where $\mathcal{D}(\mathcal{A},\mathcal{A}') \subseteq \Phi_t(v_i)$ for some $v_i \in \mathcal{V}$ [8, 18].*

Example: Consider the graph in Figure 2. Given $t = 1$, 1-distance groups for all variables will be: $\Phi_1(v_1) = \{v_1, v_2, v_3\}$, $\Phi_1(v_2) = \{v_1, v_2, v_4, v_5\}$, $\Phi_1(v_3) = \{v_1, v_3, v_4\}$, $\Phi_1(v_4) = \{v_2, v_3, v_4, v_5\}$, $\Phi_1(v_5) = \{v_2, v_4, v_5\}$.



Vi	Vj	Fi,j
0	0	1
0	1	0
1	0	0
1	1	2

Fig. 2. An example DCOP with five binary variables. Each constraint has the same cost function.

3 Structures of Groups

In a DCOP, the problem is solved by dividing DCOP into groups, and then, all the agents in a group cooperate to maximize the objective function. In other words, they create a coalition in groups to maximize the objective function. A group structure is a partition of the overall set of agents into sub groups. DCOP division in section 2.2 with $t = 1$ for the graph in Figure 2, is a possible structure for this network of agents. The structure of groups by this division is:

$$GS = \{\{v_1, v_2, v_3\}, \{v_1, v_2, v_4, v_5\}, \{v_1, v_3, v_4\},$$
$$\{v_2, v_3, v_4, v_5\}, \{v_2, v_4, v_5\}\}$$

Obviously, by using different t for each group, other structures are gained.

Given a network of agents NET $= (\mathcal{A}g, \mathcal{F})$ with a set of agents $\mathcal{A}g$ and a set of cost functions $\mathcal{F}$, the optimal group structure GS^* is given in the following formula:

$$GS^* = \underset{GS \in all\ possible\ group\ stuctures}{\arg\max} U(GS) \quad (4)$$

Where

$$U(GS) = R_{GS}(A^*) \quad (5)$$

It indicates $U(GS)$ is the reward value for assignment $\mathcal{A}^*$ which is the best assignment reached by applying an incomplete algorithm for a DCOP.

Finding the best structure is impossible because the number of structures that can be created in a graph is exponential.

4 DALO Algorithm and Issues

DALO algorithm, as an incomplete algorithm, was introduced by Yin [18]. It is an asynchronous algorithm for DCOP based on t-distance optimality.

DALO algorithm has three phases. In phase one, each agent sends a message containing all its constraints to agents in a distance of t hops. Then, it broadcasts its initial value to a distance of $t + 1$ hops in a separate message. In phase two, based on the information gathered in the previous phase, all the leaders compute a new optimal assignment using a centralized variable elimination algorithm in parallel. In phase three, if the new assignment improves the utility of a group, the group leader attempts to set the new assignment. There might be conflicts among overlapping groups while all leaders try to set their assignments. The conflicts are resolved by an asynchronous locking and commitment protocol.

4.1 DALO Issues

Although DALO is an effective algorithm to solve DCOP problems, it suffers from some drawbacks. 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 [18]. Using distance as a criterion to create groups may produce groups with large number of nodes; especially, when there are hub nodes with many connections or subgraphs which are densely connected.

As it is explained in DALO algorithm in phase two, a complete algorithm is used to solve DCOP. All group leaders compute new optimal assignments for their groups in parallel. A leader node uses a centralized variable elimination algorithm to find the best assignment for the local group. Variable elimination algorithms are complete algorithms with exponential computational complexity in the number of agents. By increasing t, the number of agents in a group will increase and using a complete DCOP solver will not be tolerable from size and space point of view. To solve this problem, instead of using a centralized variable elimination algorithm, we use a genetic approach in phase two which is discussed in [14].

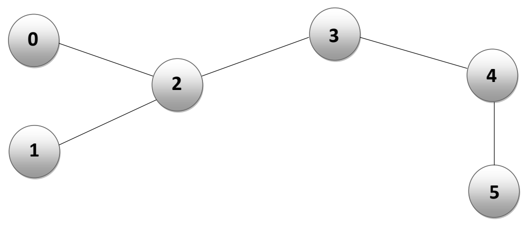
One of the significant problems in DALO algorithm and some other incomplete algorithms is how to form groups to reach the highest utility. Groups formation has direct influence on the quality of solutions for a given DCOP. Using some formations, the algorithm cannot improve the quality through increasing the number of rounds [7]. On the other hand, by changing the group formation new values may be set and the quality may improve. This problem stems from the conflicts among groups. The presence of some agents in some groups does not let a group improve its local solution, since these agents are common agents among different groups and some of them do not commit to the assignment of many groups to which they belong.

Changing groups leads to solution variation. Finding the best group formation, which the best solution could gained from, is very difficult and in some cases is impossible. With the purpose of finding the best group formation all possible formation should be considered and after comparing the results the best one is chosen. As it is clear, it is impossible in networks with large number of agents.

This paper introduces a distributed approach to improve groups formation. This method works based on the contribution of each agent in the reward of the group. To shed light on the problem an example is given in the next section.

Example Consider the graph in Figure 3. The cost function for each constraint is given. Groups are formed using $t = 1$. Active agents are shown in bold and passive agents are shown in italic. Here after we use the terms active and passive agent more. Hence, it is worthwhile to define them here. Active agents are those that can change their value to the value which leaders send to them. In contrast, passive agent are those agents in the boundary of group whose values do not change by the leader of group and their values remain constant during algorithm execution.

$\mathcal{G}_0 = \{\mathbf{0}, \mathbf{2}, \mathit{1}, \mathit{3}\}, \mathcal{G}_1 = \{\mathbf{1}, \mathbf{2}, \mathit{0}, \mathit{3}\}, \mathcal{G}_2 = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathit{4}\}, \mathcal{G}_3 = \{\mathbf{2}, \mathbf{3}, \mathbf{4}, \mathit{0}, \mathit{1}, \mathit{5}\}, \mathcal{G}_4 = \{\mathbf{3}, \mathbf{4}, \mathbf{5}, \mathit{2}\}, \mathcal{G}_5 = \{\mathbf{4}, \mathbf{5}, \mathit{3}\}$



0	2	F
0	0	10
0	1	0
1	0	0
1	1	0

2	3	F
0	0	2
0	1	0
1	0	2
1	1	0

1	2	F
0	0	0
0	1	8
1	0	0
1	1	0

4	5	F
0	0	8
0	1	0
1	0	0
1	1	0

3	4	F
0	0	0
0	1	0
1	0	8
1	1	0

Fig. 3. DCOP examples

All leaders $\mathcal{L} = \{0, 1, 2, 3, 4, 5\}$ compute the best assignment for their group members by starting from initial value 1 for all agents. Among these leaders, leader 4 can set its assignment. Since, leader 4 sets its assignment, the other leaders cannot set their assignments. Therefore, DCOP assignment will be $A = (1, 1, 1, 1, 0, 0)$. The utility of DCOP will be $U(A) = 16$.

Agent 3 is the common agent in groups $\mathcal{G}_2$, $\mathcal{G}_4$ and also is the active agent in these two groups. In computing the best assignment the value given to agent 3 by leader 2 is 0 but the value given by leader 4 to the same agent is 1. Based on the rule in DALO algorithm, an agent is committed to the group which has the highest utility. Hence, based on cost functions in Figure 3, agent 3 chooses the value given by leader 4.

Among groups $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2$, without considering other groups of graph, group $\mathcal{G}_2$ can set its assignment, but other groups cannot because all the common agents in these groups commit to the values given by leader 2. On the other side, due to the presence of agent 3 which commits to the value of leader 4, leader 2 cannot set its assignment. Consequently, the presence of agent 3 makes groups assignment stay unchanged. It is needless to say quality does not enhance as result of this presence.

According to the description above among all groups, group $\mathcal{G}_4$ changes its assignment and all other groups stay in their initial values. Consider agent 3 is removed from group $\mathcal{G}_2$. This change having been incorporated, group $\mathcal{G}_2$ and group $\mathcal{G}_4$ set their assignments simultaneously which leads to utility enhancement.

This example clarifies the main problem in algorithms that use a fix group formation in which the utility of the solution does not increases without changing group formation. Worded differently, it shows that the presence of some active/passive agents creates problems in setting new assignments and adding or removing them to or from some groups solves the problem.

For the above reasons, we focus on the impact of each agent in each group. We use a novel algorithm which tries to estimate the impact of agents in groups. This algorithm is an efficient and distributed method to change groups dynamically. The algorithm considers the impact of active agents in a group as well as passive agents. Moreover, each group leader keeps only the local group information to run the algorithm which makes the communication bandwidth and storage requirement low.

5 Embedding Group Optimization in Solution Procedure

The solution procedure starts from a random initial assignment and monotonically improves the solution quality. To have our discussion simple we divide the procedure into four phases: initialization, groups optimization, computing the best assignment and implementing assignments.

- **Initialization:** At first, every agent sends a message containing all its constraints to all agents in distance of t hops from it. Then, it chooses an initial value from its domain and broadcasts it to agents in distance of $t + 1$ hops. In this phase, a leader starts to construct its group. Given t, all agents whose distance of center node are lower than t will be a member of the group. The additional hop in sending a message is for boundary nodes. The nodes in the boundary of a group are considered static in computing the best assignment.
- **Groups optimization:** In phase two, some leaders are selected and optimize their groups to facilitate the achievement of a better solution for a DCOP problem. Detailed description of optimization algorithm is given in section 5.1.
- **Computing the best assignment:** In this phase, all the leaders compute a new optimal assignment using a centralized variable elimination algorithm in parallel. A leader agent finds a new value for active agents in a group considering the fact that passive agents stay unchanged.
- **Implementing assignments:** Each agent belongs to different groups and receives various assignments from different leaders, and lastly every agent commits to a group with the highest utility. To resolve the conflict among overlapping groups, a method is used for resolving conflict described in DALO algorithm [18].

Phase one is done just once. Since the computation complexity of phase two is high, this phase is executed after each m rounds. The value of parameter m depends on the size of graph and is specified through experiment. Optimizing group is our main contribution in this paper and so we provide our deep discussion about this issue in the next subsection. Phase three and four are executed in each round in all groups in parallel. The algorithm stops running if it converges to a value.

5.1 Optimizing Groups

As it is mentioned before, we try to find a group formation by which solutions with higher quality are gained. To this end, a leader tries to change its group by adding /removing some agents to/from its group. When a leader decides to add a passive agent to the group, it changes it to an active one. In contrast, to remove an agent a leader makes an active agent passive.

How to add or remove agents is the major problem. A criterion should be introduced to use in adding or removing the agents. We utilize a marginal contribution concept to change groups.

Definition 4 *Let $\mu_i(\mathcal{G})$ be the marginal contribution of $\mathcal{A}gent_i$ to the group $\mathcal{G}$ which is computed by adding or removing it from $\mathcal{G}$.*

$$\mu_i(\mathcal{G}) = R(A') - R(A) \tag{6}$$

A' is the best assignment of group $\mathcal{G} \subseteq all\ agent \cup \mathcal{A}gent_i$ and A is the best assignment of group $\mathcal{G} \subseteq all\ agent \backslash \mathcal{A}gent_i$.

We use this concept in group formation. In each group of a DCOP, utility of a group before and after removing (adding) an agent is computed and if the absence (presence) of an agent increases the local utility, we try to change the group by removing (adding) an agent.

There are three main issues in groups optimization. The first one is that the decision about any changes cannot be made through group information and decision about the change should be made using the information of the whole DCOP, but we do not have the global information of the graph in each group. The second issue is about the method of group alternation as adding and removing an agent causes some other agents to join or leave the group. Changing all groups is not efficient which forces us to choose some leaders to change their groups. This is the source for the third issue. All these problems and our proposed solutions are discussed in the following sections in more detail. From now on, we call the group which we try to change *the target group.*

Local View of the Leader Agent Changing a group has an effect on the whole DCOP, because by removing (adding) agents the assignment for the members changes. As a matter of fact, the new assignment of this group has direct influence on other groups. The new values may or may not enable some other groups set their assignments. For that reason, a leader agent should be aware of the status of other groups and whether or not they set their new assignment through the new change. A group leader can firmly claim that the change has positive effect if it has global information about the graph. It is obvious that the leader does not have such information.

Based on the description above, due to the connections among groups and propagation of the change in the whole DCOP, to get the best decision, a leader agent should be informed of their adjacent leaders and decide what happens in the whole DCOP after the new change. But as we know, it is impossible to solve a problem distributively.

We solve the problem by creating a local view for each leader. The local view of the leader of target group is a subgraph consists of the target group and all its adjacent groups. In the process of decision making about the new change in the target group, the leader uses local view. There is no need to know all the members of the adjacent groups. Because the leader of the target group should only be in contact with the leaders of the adjacent groups. Finding the leader of any of adjacent groups is very easy because these leaders are the active agents of the target group.

When a leader changes its group, it computes a new assignment for the new group and sends the new values to all group members. All agents in this group receive new assignment and decide about commitment to the group again. Then, all the groups in the local view of the leader of the target group use the new assignment and decide about implementation of their assignments. Adjacent groups do not compute new assignments and they just receive a message from agents which commit to other groups. The leader of the target group sends a message to its adjacent leaders to be aware of the utility of the adjacent groups. All adjacent leaders send back their utilities to the central leader.

Add/Remove Agents of a Group A leader has information about its members including active and passive agents. Because of the limited information of the leader node, the agent we try to add to the group should be chosen from passive agents of the target group. All agents with direct link to the added agent and not belonging to the group will be considered as passive agents of the new group.

In removing an agent from a group, we choose an active agent and make it a passive one, but all passive agents which are connected to the removed agent should be removed from the group as well. We cannot remove all of these agents because there are some agents among these agents which are connected to the group via other active agents. Therefore, in removing an agent from a group, agents connected to a group just by the removed agent will be removed and all other agents will remain by means of other active agents in the group.

Choosing Groups to be Optimized One of the main issues in this new approach is to choose groups to be optimized. We can apply the new approach to all groups, but it has some problems. The first is that using the new approach in solving a DCOP, the computational cost increases and consequently optimizing all groups is not tolerable from computational point of view. The second problem is related to coordinating the decision of all leaders in changing their groups. Since leaders do not have global information of a DCOP, their decision about the changes in their groups may have conflicts and optimizing all groups will be useless.
In line with aforementioned description, some leaders should be chosen to optimize their groups. Choosing groups optimizing of which provides us with the

best result is impossible. We use a simple and cost effective method to select groups. This approach is derived from a partial approach introduced in [6]. Using this approach all leaders should try to change their groups, at least once, during algorithm execution. As a matter of fact, the changes in a group formation influences the whole DCOP and increases the computational cost. Hence, optimizing groups is done after each m rounds.

Consider there are n agents in a network. Consequently, there are n groups in a network. We define the index set $\mathcal{L} = \{1, 2, \dots, n\}$. The index set $\mathcal{L}$ is divided into h subsets $\mathcal{S} = \{S_1, \dots, S_h\}$. Each subset contains leaders ID which should optimize its groups. After each m rounds, a subset, S_i, is selected and leaders in this subset try to optimize their groups. The main problem is to assign leaders to subsets. Finding the best division is not computable within limited time.

We use a simple approach called sequential approach. In sequential approach, in the first round, S_1 is selected, in round m subset S_2, and in round mh subset S_h is selected; in round $m(h+1)$, subset S_1 is selected again. So, after mh rounds, all subsets will be selected only once.

To specify groups to be optimized, we use a simple rule. In round r, a leader with ID ℓ checks if:

$$\ell \,\%\, h = r \,\%\, h \ , \ where\ r \,\%\, m = 0 \tag{7}$$

Then, this leader tries to optimize its group. The number of groups which should be optimized and the rounds in which we optimize groups are found through experiment. There is no specific rule to do so.

6 DGOPT Algorithm

In this section, we explain DGOPT in more detail. As it is obvious DGOPT adds a new step to DALO. For the sake of simplicity we just emphasize on the new step when we refer to DGOPT. We divide the algorithm into 3 steps to have more concentration on our explanation. This algorithm is applied to groups which are selected based on descriptions in the previous section. The following algorithm represents the process of removing an agent. The process of adding is much the same way.

– **Local information gathering:** At first, the leader node finds its adjacent leaders which are the active agents of its group. The leader of target group stores the utility of its adjacent groups which are computed before optimizing its group. Since the leaders in the local view are the active agents of the target groups, obtaining information about their groups is not very time-consuming. Next few lines yield the justification of why this process is not time-consuming. To commit to the new assignment, some messages pass among leader and its active agents. We can include the utility of adjacent groups in the messages by which active agents inform the leader whether or not they have committed to the new assignment. Therefore, there is no need to send and receive more messages and we can include the information for decision making in messages which are exchanged among leader and active agents to set the new assignment. Based on this information, we sum up the utility of the target group and the groups in its local view in line 6 to have the utility of groups in local view before changing the target group.

Algorithm 1

```
(* Remove an Agent From a Group *)
1.  Target group $\mathcal{G}$;
2.  Create Local view();
3.  Local Information Gathering();
4.  $R_1 = 0$;
5.  for $i \leftarrow 1$ to Number of Groups in Local View
6.      do $R_1$=$R_1$+utility(group($i$));
7.  for $i \leftarrow 1$ to some randomly chosen active agents
8.      do $R_2 = 0$;
9.          Temporary Remove Agent(agent(i));
10.         Compute Utility($\mathcal{G}$);
11.         for $j \leftarrow 1$ to Number of Groups in Local View
12.             do $R_2$=$R_2$+utility(group($j$));
13.  if $(R_2 - R_1) > 0$
14.     then Remove Agent Permanently();
15.  return $\mathcal{G}$;
```

- **Changing group temporarily and computing the new assignment:** The leader removes an active agent temporarily. The leader agent makes the active agent a passive one and removes all agents connected to the group by this agent. If the agent is connected to the group by other active agents, we do not remove it. After removing an agent, the leader node re-computes the best assignment in the new group. All leaders of adjacent groups just decide about the implementation of their assignments by the new change and they send the utility of their groups to the leader of the target group. We emphasize that by the above justification the adjacent leaders are not in need of sending new messages to inform the leader of the target group of their utilities. The leader agent sums up its new utility and its adjacent leaders' utilities again in line 12.
- **Computing marginal contribution:** In this step, marginal contribution of the removed agent is computed. The positive marginal contribution of the removed agent shows that the new formation increases the utility, but if the marginal contribution is negative, the change is not promising. If the change is promising the group is changed permanently in line 14.

The first phase is executed just once. The other phases are repeated for all selected active agents.

7 Experimental Results

In this section we put forward some experimentations to show the efficiency of our DGOPT algorithm. We setup our experiments on some graphs with different densities and the same size. Before presenting our results we introduce our three evaluation metrics which the algorithms are compared based on them, namely, quality of solution, number of rounds and (Gain,#Locked Variables).

- **Quality of Solution:** The primary aim of this paper is to construct groups to reach the solution with higher quality. The quality of solution is computed according to equation 2. Based on this definition, a solution with higher reward is more qualified [5, 7, 12].
- **Number of Rounds:** A round is one unit of algorithm progress in which all agents perform any required computation. After some rounds, the solution reached by algorithm does not change. In this case, the algorithm converges to the best possible solution. In the evaluation, we consider the number of rounds required to converge. This metric is a convenient, standardized metric for estimating the performance of a DCOP algorithm [5].
- **(Gain,#Locked Variables):** Versus to above mentioned metrics which are used to evaluate the performance of DCOP algorithms in the whole DCOP, tuple $(Gain, \#Locked\,Variables)$ analyzes the performance of the algorithm on local groups. The gain is the quality of group and the $\#Locked\,Variables$ is the number of variables that are locked to set the new assignment.

7.1 Results

The result that we are reporting is based on some random graphs with four different densities $\mathcal{D} = \{0.2, 0.4, , 0.6, 0.8\}$. All graphs used in our experiment have the size 50 with different densities and structures. Variables have a binary domain and rewards are integers drawn from $[1, 500]$. In the experiment we generated 20 random graphs with different structures while kept the size and density the same. The solution quality shown in the following figures are the average quality that gained from these graphs.

Both algorithms start from a same random initial assignment. The stopping criterion is also defined in a same for both algorithms. The algorithm stops running whenever all groups do not tend to change their assignments because there is no new assignment to increase the utility of groups.

We set parameter t to 2, h to 3, and m to 5 respectively. Determination of the exact values of h and m is made just by experiment and we set the parameters to the values which have the more desirable results. More discussion related to determination of h and m can be found in [6].

In our first experiment we compare the solution quality of our DGOPT algorithm and DALO. Obviously, algorithm that achieves a final solution of higher quality in a lower number of rounds is more desirable. Figures 4 through 7 show that the solution quality increases by DGOPT algorithm in comparison with DALO. For instance in Figure 4 the final solution quality for graphs with

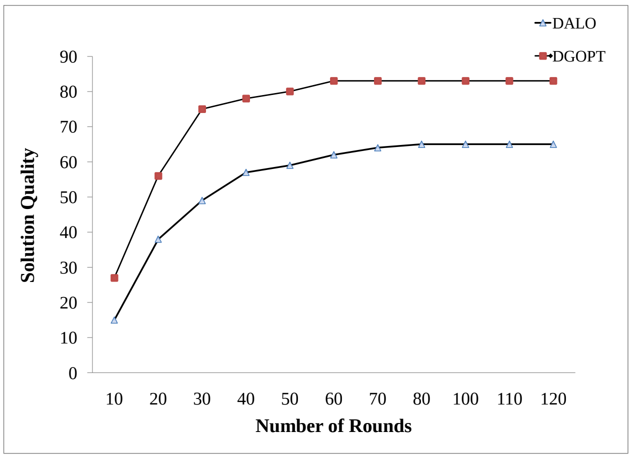


Fig. 4. Solution quality: DGOPT vs DALO for graphs with density 0.2

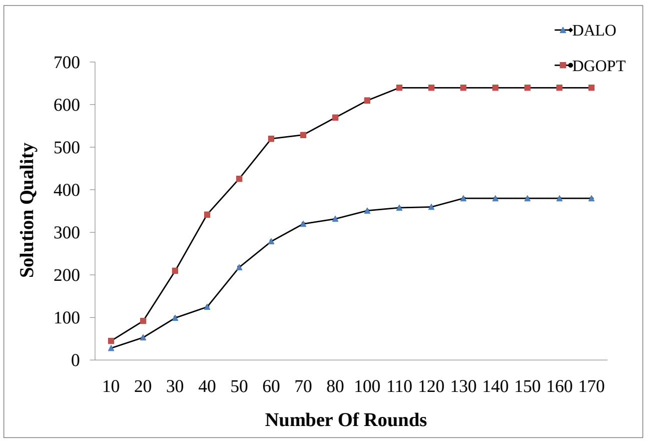


Fig. 5. Solution quality: DGOPT vs DALO for graphs with density 0.4

$\mathcal{D} = 0.2$ using DALO is 65, but using DGOPT the quality in the same graphs is 82. Moreover, after group alteration through DGOPT, there will be a boost in the solution quality; these increases end in the algorithm convergence to a higher solution quality in lower number of rounds in comparison with DALO. As an example, consider the diagrams in Figure 6, DGOPT converges after 150 rounds and DALO converges after 195 rounds.

The results also show that the DGOPT algorithm is even more efficient on dense graphs. For example, the maximum difference in solution quality for graphs with $\mathcal{D} = 0.2$ is almost 30, but the maximum difference for graphs with $\mathcal{D} = 0.4$

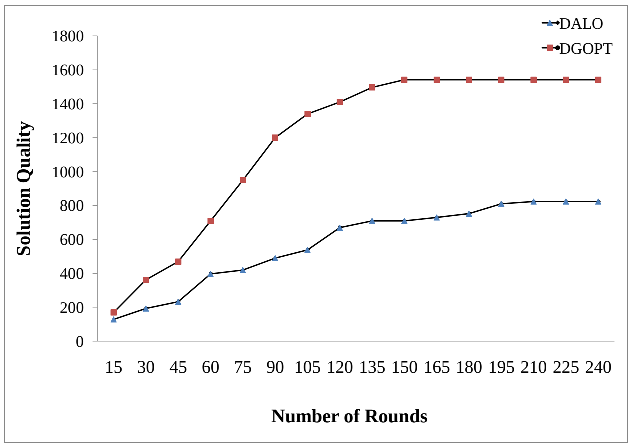


Fig. 6. Solution quality: DGOPT vs DALO for graphs with density 0.6

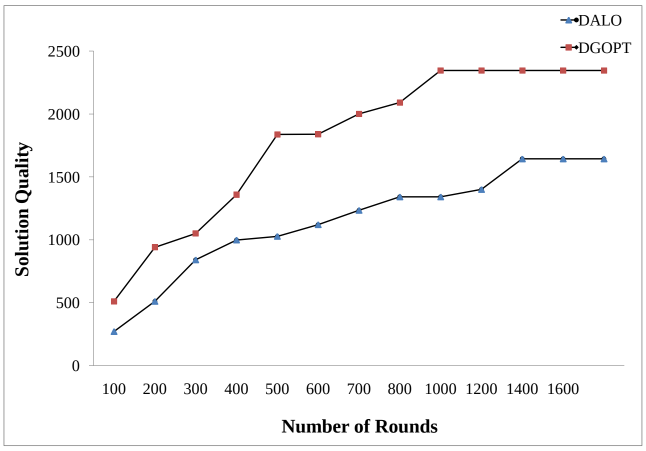


Fig. 7. Solution quality: DGOPT vs DALO for graphs with density 0.8

is almost 300. It is clear that the groups in dense graphs have more number of agents in comparison with sparse graphs. Accordingly the overlap among groups increases and there will be more number of agents which are common among groups. In this case, there will be more number of agents which do not allow a leader to set its assignment by committing to other groups. Overally, the results in our experiment show that the quality of solution increases 43% and the number of round decreases 21%, on average. It can be concluded that using DGOPT algorithm, solutions with higher quality are gained in a lower number of rounds.

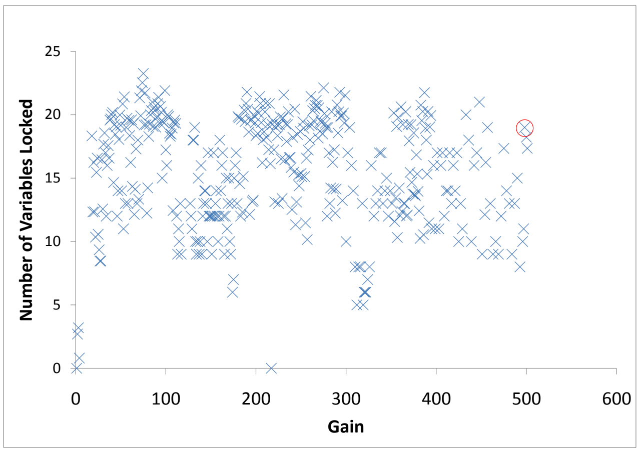


Fig. 8. (Gain,#Locked Variables) for DALO

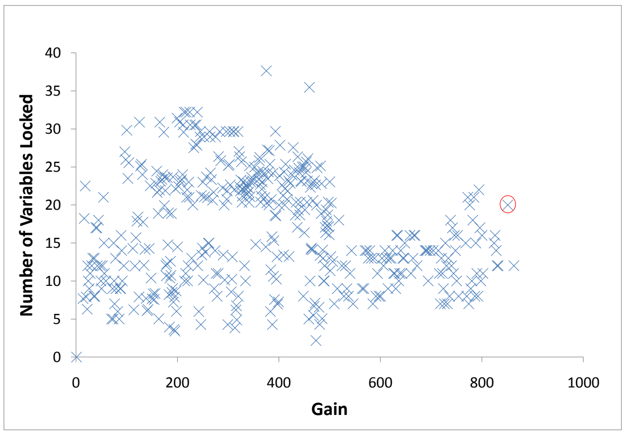


Fig. 9. (Gain,#Locked Variables) DGOPT

To further understand and compare the performance of DGOPT and DALO, we provide an analysis on local group changes. In each group, the leader locks some of the variables and if all group members commit to the new assignment, it will be set. By setting the new assignment, the utility of group, which we call it gain, will change. The $(Gain, \#Locked\ Variables)$ pair is used as a metric to compare DCOP algorithms in [18]. It is a proper metric to compare the effect of different group formations in solving DCOPs. The more the number of locks, the more the number of conflicts. Hence, groups with lower number of locks and larger gain are more preferred.

To compare the algorithms, we depict the result for graphs with size 50 and density 0.4. As it is declared in Figure 8 DALO never achieves a gain larger than 500 and barely locks more than 20 variables. On the other hand, DGOPT achieves gain 800 by locking more number of variables. For example as it is specified in the Figures 8, 9 by locking 20 variables DALO achieves gain 500 while DGOPT can achieve gain 850.

In Figure 8, the congestion is on the value 18 which indicates that most of groups locked 18 variables. On the contrary, as it is depicted in Figure 9, the congestion is on the value 23. The difference in the number of locked variables is not very much, but the quality improvement is considerable. Hence, by slight increase in the number of variables better solutions are gained. Our experimentations show hat DGOPT outperforms DALO both in term of solution quality and the number of rounds that this quality is achieved.

8 Conclusion

As it is explained in this paper some group formations are not very efficient to solve DCOP problems and lead to solution with lower quality. In this paper, we proposed a distributed dynamic algorithm to optimize groups in a DCOP. The purpose of this algorithm is to find better group formations to reach higher solution quality. This algorithm achieves solutions with higher quality in low number of rounds. Moreover, by slight increase in the number of locked variables in groups, solution quality increases considerably. The proposed algorithm can be applied to other incomplete DCOPs by slight modification. We are planning to extend the algorithm by using agents previous interactions to improve groups.

References

1. Aji and R. McEliece. The generalized distributive law. ieee transactions on information theory. *IEEE Transactions on Information Theory*, pages 325–343, 2000.
2. A. Chapman, R. A. Micillo, R. Kota, and N. Jennings. Decentralised dynamic task allocation: A practical game-theoretic approach. *the Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09)*, pages 915–922, May 2009.
3. M. Cooper, S. de Givry, and T. Schiex. Optimal soft arc consistency. In *the Proceedings of the 20th Internation Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 68–73, 2007.
4. J. Davin and P. Modi. Hierarchical variable ordering for multiagent agreement problems. In *the Proceedings of AAMAS*, pages 1433–1435, 2006.
5. J. Davin and P. J. Modi. Impact of problem centralization in distributed constraint optimization algorithms. In *the Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1057–1063, July 2005.

6. E.Bigdeli, M.Rahmaninia, and M.Afsharchi. Pkopt: Faster k-optimal solution for dcop by improving group selection strategy. In *the proceeding 22th international conference on tools with Artificial Intelligence (ICTAI).* October, July 2010.
7. H. Katagishi and J. P. Pearce. Distributed dcop algorithm for arbitrary k-optima with monotonically increasing utility. In *CP Workshop on Distributed Constraint Reasoning*, September 2007.
8. C. Kiekintveld, Z. Yin, A. Kumar, and M. Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *the Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, May 2010.
9. R. Maheswaran, E. Bowring, J. Pearce, P. Varakantham, and M.Tambe. Taking dcop to the real world: efficient complete solutions for distributed multi-event scheduling. In *the Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2004)*, pages 310–317, 2004.
10. R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 438–445. IEEE Computer Society, 2004.
11. P. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt:asynchronous distributed constraint optimization with quality guarantees. *ARTIFICIAL INTELLIGENCE*, 16(1-2):149–180, 2005.
12. J. P. Pearce, M. Tambe, and R. T. Maheswaran. Solving multiagent networks using distributed constraint optimization. *AI Magazine*, 28(3):47–66, September 2008.
13. A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *the Proceedings of the International Joint Conference on Artificial Intelligence*, pages 266–271, aug 2005.
14. M. Rahmaninia, E. Bigdeli, and M. Afsharchi. Solving dstributed constraint optmazation problem: An evalutionary approach. In *the Proceedings of International Conference on Agents and Artificial Intelligence (ICAART)*, January 2011.
15. M. Silaghi and M. Yokoo. Dynamic dfs tree in adopt-ing. In *the Proceedings of AAAI*, pages 763–769, 2007.
16. W. Yeoh, A. Felner, and S. Koenig. Bnb-adopt: An asynchronous branch-and-bound dcop algorithm. In *the Proceedings of AAMAS*, pages 591–598, 2008.
17. W. Yeoh, X. Sun, and S. Koenig. Trading off solution quality for faster computation in dcop search algorithms. In *the Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 354–360, 2009.
18. Z. Yin, C. Kiekintveld, A. Kumar, and M. Tambe. Local optimal solutions for dcop: New criteria, bound, and algorithm. In *AAMAS 2009 Workshop on Optimization in Multi-Agent Systems*, May 2009.