Local search for Distributed Asymmetric Optimization

Alon Grubshtein[1], Roie Zivan[2],
Tal Grinshpoun[1] and Amnon Meisels[1]

[1]Dept. Of Computer Science
Ben Gurion University of the Negev
P.O.B 653
Be'er Sheva, 84105, Israel
{alongrub,grinshpo,am}@cs.bgu.ac.il

[2]Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
zivanr@cs.cmu.edu

ABSTRACT

Distributed Constraints Optimization (DCOP) is a powerful framework for representing and solving distributed combinatorial problems, where the variables of the problem are owned by different agents. DCOP algorithms search for the optimal solution, optimizing the total gain (or cost) that is composed of all gains of all agents. Local search (LS) DCOP algorithms search locally for an approximate such solution.

Many multi-agent problems include constraints that produce *different gains* (or costs) for the participating agents. *Asymmetric* gains of constrained agents cannot be naturally represented by the standard DCOP model.

The present paper proposes a general framework for Asymmetric DCOPs (ADCOPs). The new framework is described and its differences from former attempts are discussed. New local search algorithms for ADCOPs are introduced and their advantages over existing algorithms and over former representations are discussed in detail.

The new proposed algorithms for the ADCOP framework are evaluated experimentally and their performance compared to existing algorithms. Two measures of performance are used: quality of solutions and loss of privacy. The results show that the new algorithms significantly outperform existing DCOP algorithms with respect to both measures.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms

Keywords

Distributed Constraint Optimization, Asymmetric constraints

1. INTRODUCTION

Multi agent systems (MAS) often include a combinatorial problem which is distributed among the agents. Some examples of multi agent combinatorial problems are the Meeting Scheduling problem, Mobile Sensor nets and Flight Scheduling. The natural representation of such problems in terms of variables which are owned and assigned by agents and in terms of values (whether costs or utilities) for combinations of assignments to variables of different agents, has encouraged the study of Distributed Constraint Optimization Problems (DCOPs). DCOPs are a powerful framework for formulating and solving MAS combinatorial problems. In the last decade, algorithmic search techniques for DCOPs were intensively studied [13, 18, 11, 2]. Since DCOPs are NP-hard, many recent studies consider incomplete (local) search algorithms [10, 24, 16, 25, 20].

Cite as: Local search for Distributed Asymmetric Optimization, Alon Grubshtein, Roie Zivan, Tal Grinshpoun and Amnon Meisels, *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. 1015-1022

Aspects of the DCOP - MAS relationship are discussed by [8, 1]. The authors of [1] examine the analogy between the DCOP formulation and a class of games known as *"Potential Games"*. The importance of this analogy lies in the fact that every finite potential game possesses at least one pure strategy Nash Equilibrium (NE) [14].

In the world of game theory, a pure strategy NE is a stable profile of actions corresponding to the set of all participants, in which any unilateral change of action by a single participant will not yield a better *personal* gain for the participant. In the DCOP formulation, the above definition coincides with special solutions known as local optima (minima or maxima) [22, 24]. These solutions are sets of assignments to variables made by all agents, in which a single change of assignment by an agent will only reduce the *global* gain.

The source of this correspondence between NEs and local optima stems from the constraint structure of DCOPs. Each constraint C over variables of k agents is defined as a mapping from the domains of the variables to a single real value:

$$C : D_{i_1} \times D_{i_2} \times \cdots D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$$

The above definition of a constraint implies that the cost (gain) of a constraint is the same for all participating agents. When an agent lowers its cost or gain from a constraint, all of its constraint peers share a similar decrement in cost from that constraint. Thus, when considering local optima, it is clear that any change of an assignment can only reduce both global *and* personal gains.

In many real life situations a constrained agent stands to gain differently from others connected to the same constraint. In fact, this is the natural scenario in a typical MAS situation. Take the meeting scheduling problem for example. Scheduling a meeting of several agents typically results in different gains for different agents participating in a meeting.

The above observation calls for a generalization of the standard DCOP model [13, 12, 2]. A general DCOP has constraints that include asymmetric gains for the involved agents.

The present study proposes Asymmetric DCOPs (ADCOPs), a model for representing asymmetric combinatorial multi agent problems. ADCOPs naturally accommodate constraints where the participating agents have different gains or costs. A few former studies proposed to capture asymmetric gains among constraining agents by introducing additional variables for each agent. The additional variables are duplicates of the variables of constraining agents. Such a representation uses inner constraints with the duplicate variables to represent the asymmetric gains [10, 17]. The complete scheme, of duplicating all variables of constraining agents and of using rigid constraints to enforce equality of assignments with other agents was termed Private Events as Variables (PEAV) by [10].

The advantages of the proposed model in comparison to PEAV are discussed and demonstrated in Section 3. Most important, we demonstrate that PEAV generates for standard local search local optima states which may prevent local search algorithms from converging to higher quality local optima as in the proposed model.

In both the former model and the ADCOP model proposed in this work, the guaranteed convergence to a local optima does not apply as for standard DCOP. Existing local search algorithms fail to converge to a local optima and as a result produce low quality solutions. A number of alternative algorithms for ADCOPs are proposed along with the trade-offs between them in terms of solution quality, privacy loss.

The rest of this paper is organized as follows. A short background to local search for DCOPs is presented in Section 2. Section 3 presents the proposed asymmetric DCOP model and discusses its importance in view of the different DCOP representations of asymmetric problems. This is followed by Section 4 which presents new algorithms for solving asymmetric problems. Section 5 includes an extensive experimental evaluation of all algorithms. Two measures are used to evaluate performance of algorithms. the first measure is the most common and natural - the quality of the produced solutions. The second measure of performance relates specifically to the asymmetry of constraints. When constraints have different gains for the constraining agents it is of much interest to measure how much of this private information is revealed during search. It turns out that the loss of privacy of constraints differs widely among the different ADCOP algorithms. The results of the experimental evaluation of Section 5 are quite conclusive. The proposed ADCOP algorithms perform better than former, standard DCOP, algorithms with respect to the above *two measures of performance*. The paper is summarized and conclusions are drawn in Section 6.

2. LOCAL SEARCH FOR DCOPS

We first present the standard DCOP model followed by the main a description of leading local search algorithms for DCOPs.

2.1 Distributed Constraint Optimization

```
DSA
value ← ChooseRandomValue()
while (no termination condition is met)
    send value to neighbors
    collect neighbors' values
    if (ReplacementDecision())
       select and assign the next value
```

Figure 1: Standard DSA.

A *DCOP* is a tuple $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} >$. $\mathcal{A}$ is a finite set of agents $A_1, A_2, ..., A_n$. $\mathcal{X}$ is a finite set of variables X_1, $X_2, ..., X_m$. Each variable is held by a single agent (an agent may hold more than one variable). $\mathcal{D}$ is a set of domains D_1, $D_2, ..., D_m$. Each domain D_i contains a finite set of values which can be assigned to variable X_i. $\mathcal{R}$ is a set of relations (constraints). Each constraint $C \in \mathcal{R}$ defines a non-negative *cost* for every possible value combination of a set of variables, and is of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary constraint* refers to exactly two variables and is of the form $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary DCOP* is a DCOP in which all constraints are binary. An *assignment* (or a label) is a pair including a variable, and a value from that variable's domain. A *partial assignment* (PA) is a set of assignments, in which each variable appears at most once. *vars(PA)* is the set of all variables that appear in PA, $vars(PA) = \{X_i \mid \exists a \in D_i \wedge (X_i, a) \in PA\}$. A constraint $C \in \mathcal{R}$ of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$ is *applicable* to PA if $X_{i_1}, X_{i_2}, \ldots, X_{i_k} \in vars(PA)$. The *cost of a partial assignment* PA is the sum of all applicable constraints to PA over the assignments in PA. A *full assignment* is a partial assignment that includes all the variables ($vars(PA) = \mathcal{X}$). A *solution* is a full assignment of minimal cost.

2.2 Local Search

The general design of local search algorithms for Distributed Constraint Optimization Problems is synchronous. In each step of the algorithm an agent sends its assignment to all its neighbors in the constraint network and receives the assignments of all its neighbors. For lack of space we present in detail only two algorithms that apply to this general framework - the *Distributed Stochastic Algorithm* (*DSA*) [24] and the *Max Gain Message* (*MGM*) algorithm [7]. [1]

In the initial step of the *DSA* algorithm agents pick some value for their variable (randomly according to [24]). Next, agents perform a sequence of steps until some termination condition is met. In each step, each agent sends its value assignment to its neighbors in the constraints graph and receives the assignments of its neighbors. The present paper follows the general definition which does not include a synchronization mechanism. If such a mechanism exists, agents in *DSA* can send value messages only in steps in which they change their assignments. After collecting the assignments of all its neighbors, each agent decides whether to keep its value assignment or to change it, by using a stochastic strategy (see [24] for details on the possible strategies and the difference in the resulting performance). A sketch of *DSA* is presented in Figure 1.

The *MGM* algorithm is a simple version of the *DBA* algo-

[1]Our description considers an improvement to be a decrease in the number of violated constraints (as in Max-CSPs).

MGM

```
1.  value ← ChooseRandomValue()
2.  while (no termination condition is met)
3.      send value to neighbors
4.      collect neighbors' values
5.      LR ← BestPossibleLocalReduction()
6.      Send LR to neighbors
7.      Collect LRs from neighbors
8.      if (LR > 0)
9.          if (LR > LRs of neighbors
                (ties broken using indexes))
10.             value ← the value that gives LR
```

Figure 2: Standard MGM.

rithm [22, 24]. In every synchronous step, each agent sends its current value assignment to its neighbors and collects their current value assignments. After receiving the assignments of all its neighbors, the agent computes the maximal improvement (e.g., reduction in cost) to its local state that can be achieved by replacing its assignment and sends this proposed reduction to its neighbors. After collecting the proposed reductions from its neighbors, each agent changes its assignment only if its proposed reduction is greater than the reductions proposed by all of its neighbors. In more advanced versions of *MGM*, agents group together in order to propose a common improvement and thus avoid local minima to which a smaller group would have converged. A sketch of the standard *MGM* algorithm is in Figure 2. After selecting a random value for its variable (line 1), the agent enters a loop where each iteration is a step of the algorithm. After sending its assignment to its neighbors and collecting their assignments (lines 3,4), the agent calculates its best weight reduction and sends it to its neighbors (lines 5,6). After receiving the possible weight reductions of all of its neighbors the agent decides whether to replace its assignment and upon a positive decision reassigns its variable (lines 7-10).

A very different approach towards local search is implemented in the *Max-Sum* algorithm [19]. *Max-Sum* is an asynchronous algorithm in which agents exchange messages which accumulate the sum of costs for possible assignments. The algorithm begins by agents sending to their neighbors messages which include the cost for each of the receiving agents' possible assignments. An agent that receives a message, sends messages to each of its other neighbors in which the cost of the possible assignments of the neighbor is added to the best cost according to the message received. When the algorithm is terminated (after a predefined condition such as the number of messages sent by each agent) each agent selects the value assignment with the lowest cost.

The *Max-Sum* algorithm was found very successful in producing higher quality solutions than standard local search algorithms for DCOPs [19]. Furthermore, one of its main advantages for standard DCOPs is that for a tree structure constraints graph it converges to an optimal solution in polynomial time. This advantage was used to produce guarantees to the quality solution in the general case in [19].

3. ASYMMETRIC DCOPS (ADCOP)

ADCOPs generalize DCOPs in the following manner: instead of assuming equal gains for constrained agents, the ADCOP constraint explicitly defines the exact gain of each participant. That is, domain values are mapped to a tuple of costs, one for each constrained agent.

More formally, an ADCOP is defined by the following tuple $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} >$, where $\mathcal{A}$, $\mathcal{X}$ and $\mathcal{D}$ are defined in exactly the same manner as in DCOPs. Each constraint $C \in \mathcal{R}$ of an asymmetric DCOP defines a set of non-negative *costs* for every possible value combination of a set of variables, and takes the following form:

$$C : D_{i_1} \times D_{i_2} \times \cdots D_{i_k} \rightarrow \prod_{j=1}^{k} \mathbb{R}^+ \cup \{0\}$$

As usual, a *binary constraint* refers to exactly two variables and for ADCOPs takes the form $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \times \mathbb{R}^+$.

This definition of a constraint is natural to general MAS problems, and requires little manipulation when formulating a problem as an ADCOP. The upper part of Figure 3, along with Figure 4 illustrate an example ADCOP that represents a MAS problem. Each agent holds either the left or right hand side of the values depicted in the bi-matrix of Figure 4. The constraint between the two interacting agents maps assignment pairs to *value pairs*. For example: if agent A_1 assigns b, and agent A_2 assigns x, A_1's cost will be 7, and A_2's cost will be 2.

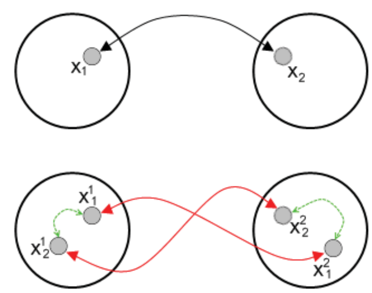

Figure 3: An ADCOP representation of a MAS problem (above), and its translation to a PEAV DCOP(below)

Let us present in short some alternatives for representing asymmetric constraints by standard DCOPs.

- **Disclosure of constraint gains:** The simplest approach to solving MAS problems involving asymmetric gains by DCOPs is through the disclosure of constraint gains. Since the DCOP constraint structure needs only information on the joint gain of an assignment, one may decide to disclose information of all constraint peers and assign the aggregated value to the constraint. However, revealing this information is essentially similar to revealing preferences which are often private [4, 9, 23].

- **Use of unary constraints** A possible technique for representing preferences and affecting different gains is through the introduction of unary constraints. Constraints are added to each variable participating in a

$A_1 \backslash A_2$	x	y
a	$3,4$	$6,1$
b	$7,2$	$5,8$

Figure 4: A 2 agent interaction. Agent 1's action is either a or b and its costs are the left hand values. Agent 2's actions are x or y and its costs are on the right side

constraint and the additional costs generate asymmetry. However, this approach fails to properly capture constraints where the personal valuation of a state by an agent is dependent upon assignments by other agents.

- **Private Events As Variables (PEAV)** The PEAV model [10, 17] was the first model capable of capturing asymmetric gains. In PEAV each agent holds in addition to its own standard variable representing its private state, one mirror variable per each of its neighbors in the constraint network (i.e., for each agent constrained with it). Consistency with the neighbors' state variables is imposed by a set of hard equality constraints. In order to avoid ideally hard constraints, the PEAV model proposes costs for unequal assignment pairs which are large enough to exceed that of a predefined calculated upper bound [9].

  The resulting representation of an assymetric MAS problem in a PEAV DCOP is much larger in terms of variables and constraints than an ADCOP. This is clear in the example in the lower part of Figure 3. On the other hand, one may argue that formulating a problem as a DCOP allows one to employ all existing DCOP search algorithm instead of devising specialized protocols. This argument, which applies for complete algorithms, is not always true for local search algorithms.

Consider the 2 agent problem described in the bi-matrix in Figure 4, in which agents try to minimize the total sum of costs. This interaction corresponds to the constraints graph in Figure 3.

Figure 5 presents the PEAV representation of the same problem. The ADCOP formulation of this problems consists of a single constraint containing the entire table of Figure 4, while the PEAV formulation splits this table in two (each side corresponds to one of the agent's intra-agent constraints), and adds the consistency constraints. The equality constraint for the problem at hand incurs a cost of 50 when a pair of unequal assignments is made.

Introducing new variables and constraints slightly changes the nature of the interaction itself. The growth of each agent's assignment space results in the PEAV interaction described with the aid of "complex" variables in the table depicted in Figure 5.

The table in Figure 5 demonstrates an important aspect of the new interaction - each one of the "consistent" states is now also a local minimum. As a result, this revised interaction now poses difficulties to many local search techniques for DCOPs:

$A_1 \backslash A_2$	$\langle x,a'\rangle$	$\langle y,a'\rangle$	$\langle x,b'\rangle$	$\langle y,b'\rangle$
$\langle a,x'\rangle$	7	54	55	111
$\langle a,y'\rangle$	60	7	108	64
$\langle b,x'\rangle$	61	108	9	65
$\langle b,y'\rangle$	109	56	57	13

Figure 5: The underlying search space of the generated PEAV formulation

- **DSA** [24] - Consider for example an initial random assignment: $A_1 : \langle b,x'\rangle, A_2 : \langle y,b'\rangle$. If a single agent decides to update its assignment (with probability p), the system converges to a local minimum in which no agent will attempt to change its value. Note that the end value in such a case is either one of the worst two solutions.

- **MGM** [7] - *MGM*, being a "hill climber" algorithm by nature, will present similar behavior. However, unlike DSA the converged solution of *MGM* will be the best of two possible ones.

The PEAV formulation of an asymmetric cost problem (e.g., Figure 4) modifies the original nature of the problem in two important aspects:

1. By generating new local minima, and thus, implicitly, new NEs. The new local minima can easily be observed in Figure 5, and when considering and analyzing the personal cost of each agent from this interaction one sees that these correspond to NEs. This implies that the PEAV formulation of a given problem produces new stable points (local optima)! In the case of the example in Figure 4, four new NEs are generated where there orginally were none.

2. Hard equality constraints between "mirror" variables prevent a single agent from performing any assignment replacement with positive reduction after the convergence to the first valid state.

In view of these problems, it is clear that the PEAV formulation loses much of its appeal when considering large asymmetric problems. Next, we turn to ADCOPs.

4. ADCOP ALGORITHMS

Let us start the discussion of local search algorithms for ADCOPs by demonstrating the shortcomings of existing local search methods.

Consider again the 2 agent problem described in Figure 4. Assuming each agent is only aware of the left (agent A_1) or right (A_2) hand value in the matrix, standard DCOP LS algorithms such as *DSA* and *MGM* can be applied to this problem. In *DSA*, for example, agents only consider their personal gain, or improvement, and as a result change values according to their local state. A similar situation exists with respect to *MGM*. However, the maximum change that is reported by agents running MGM does not necessarily imply an improvement to neighbors as well.

The asymmetric structure of constraints alters the algorithms' behavior. For example, while *DSA* and *MGM* converge to local optima on standard DCOPs, this is not true for ADCOPs. In local search agents continueously attempt to change their assignment if an improving assignment exists. When no such assignment is found by any of the agents the state of the system as a whole is said to be stable. This state is not necessarily a local optimum when asymmetric gains are considered. A change of an assignment by an agent may increase its own local cost, but due to asymmetry this change can also result in an overall lower cost to the system as a whole! On the other hand, such stable solutions comply with the definition of Nash Equilibria - no unilateral change by any single agent can improve its state. For similar reasons, *MGM* in ADCOPs looses its important monotonicity property [7]. Agents sending their maximal possible improvement to the current state to their neighbors can actually consider a change that would cause a deterioration of the state of their neighbors and of the global state.

Nash Equilibrium does not necessarily coincide with the optimum of a global objective function. In the well known example of the prisoners' dilemma, when maximizing the gain of participants, the globally worst solution is the only NE. It is important to note that pure strategy NE do not exist in every asymmetric problem and even in the presence of NEs, it is possible that neither *DSA* nor *MGM* will converge to it. Thus, the convergence prediction for DCOPs made by [1] does not apply in the case of ADCOP. It seems that despite their applicability to ADCOPs, some LS algorithms are expected to provide low quality results for ADCOPs. It is important to notice that the PEAV model does not solve the problem for *DSA* and *MGM*. In the case of *DSA*, an agent only considers its neighbors' current assignments. In the case of *MGM*, every change to a variable that would generate inequality would not be considered as a maximal reduction. Thus, both algorithms are expected to perform exactly the same in the PEAV model.

Max-Sum is another local search algorithm that also turns out to be problematic. When solving the small example in Figure 4, each agent will generate a single message and send it to its neighbor. In both models, this message will include the costs for every possible assignment of its neighbors. Since the problem has no cycles, this exchange of messages will be the only exchange of messages and agents would select their assignments accordingly. However, in the presented example, A_1 will select b while A_2 will select y, resulting in the worst possible solution (with highest cost for both). This is in contrast to using *Max-Sum* in the symmetric DCOP model where when the problem contains no cycles, the algorithm is guaranteed to converge to the optimal solution [19]. Note that the same problem would occur in the PEAV model since *Max-Sum* messages do not include the selected assignments, only costs. Thus, each agent will select the assignment according to its neighbors' valuation.

4.1 Local Search with asymmetric gains

The present study proposes several local search algorithms specifically designed for ADCOPs. These algorithms, described next, attempt to incorporate information from the agent's local neighborhood and utilize it to locate high quality solutions.

Asymmetric Coordinated Local Search (ACLS)

The *ACLS* algorithm presented in Figure 6, attempts to combine information from each agents' surrounding in order to produce a global evaluation.

```
ACLS
1.  value ← ChooseRandomValue()
2.  while (no termination condition is met)
3.      send value to neighbors
4.      collect neighbors' values
5.      IMP_SET ← LocalReductions()
6.      PV ← RandomSelectProposedValue(IMP_SET)
7.      send PV to neighbors
8.      collect PVs from neighbors
9.      foreach (neighbor n)
10.         send constraint cost with n's PV
11.     collect all constraints costs
12.     cost ← sumOfAllConstraintsCosts × C
13.     if (cost < currentState)
14.         assign with probability p
            value ← PV
```

Figure 6: ACLS.

```
MCS-MGM
value ← ChooseRandomValue()
while (no termination condition is met)
    send value to neighbors
    collect neighbors' values
    foreach (neighbor n)
        Δ ← increase due to n's new value
        if (Δ > n's last known LR)
            send constraint cost with n's new value
            change constraint cost with n's new value to 0
    collect neighbors' constraint updates
    update constraint with each neighbors
    LR ← BestPossibleLocalReduction()
    Send LR to neighbors
    Collect LRs from neighbors
    if (LR > 0)
        if (LR > LRs of neighbors
                (ties broken using indexes))
            value ← the value that gives LR
```

Figure 7: MCS-MGM.

It proceeds in synchronous steps and continues running (after a random initial assignment) until a termination condition is met. At each step, an *ACLS* agent begins by sending its current assignment to its neighbors and collecting assignments from them. It then collects all assignment which can improve its local state (line 5). Based on this improving set a proposed assignment PV is randomly picked according to the distribution of gains from each proposal (line 6). This proposal is sent to all neighbors and the neighbors proposals are collected (lines 7-8). An agent receiving a proposal responds with the value of its side of the constraint, resulting from its current assignment and the proposed assignment (lines 9-10). When all such *impact* messages arrive, the agent assesses the potential gain or loss from the assignment (lines 11-12). *ACLS* agents use a special coordination value, C, representing the amount of cooperation with their neighborhood. That is, when this constant is zero, all *impact* messages are ignored and *ACLS* produces results similar to those of *DSA* (albeit with a high overhead of network load and privacy degradation). An *ACLS* agent concludes each round by committing to a change with probability p (lines 13-14).

Minimal Constraint Sharing MGM (MCS-MGM)

Similarly to *ACLS*, the *MCS-MGM* algorithm presented in Figure 7 also attempts to employ knowledge of its local neighborhood to achieve a better gain to its surrondings.

The *MCS-MGM* algorithm also proceeds in synchronous steps and terminates according to a pre-defined condition. Each step consists of three different interaction phases. An agent begins by exchanging assignments with its neighbors (lines 3-4). It then evaluates the impact of its neighbor's assignment change on its own local state. If the neighbor's assignment change degrades the current state by more than that neighbor's last known best local reduction, the constraint is passed on to the neighbor. That is, the agent sends to its neighbor its side of the constraint with the neighbor's new value, and assigns a cost of zero instead (lines 5-9). The updated constraints are gathered by agents and the local sub-problem is slightly modified. Using the new information, the agent seeks the best local reduction and sends this information to its peers. As in *MGM*, the agents declaring the highest local reductions, change their values (lines 12-17).

A small adjustment to the *MCS-MGM* algorithm can guarantee its convergence to a local optima (note that it converges to a local optima and not to a NE). Line 7 of the algorithm is replaced by:

7. **if** $(\Delta > 0)$

We call the resulting algorithm *Guaranteed Convergence Asymmetric MGM* (*GCA-MGM*). *GCA-MGM* is expected to preserve less privacy than *MCS-MGM* since it has a weaker condition for exchanging constraints among agents but it guarantees convergence:

THEOREM 1. *GCA-MGM is guaranteed to converge to a local optima in a finite number of steps.*

Proof: Assume *GCA-MGM* does not converge. Thus, the agents repeatedly change their assignments. Each change that causes an increase for some agent triggers a constraint exchange and therefore the next time this assignment change is performed, it will not cause an increase (i.e. cannot occur more than once). Thus, the number of increases in cost is bounded by the number of constraints which is finite. After all possible increments have caused an exchange of constraints, the convergence is guaranteed as for standard *MGM* [10, 16]. □

Our experiments demonstrate that although MCS-MGM does not guarantee convergence, both versions converge very fast and this fast convergence has a strong impact on privacy loss during the search process as will be presented in the following section.

5. EXPERIMENTAL EVALUATION

The experimental evaluation uses two different domains. The first set of experiments includes Max-ADCSPs problems, which are an asymmetric variation of Max-DCSPs. Max-DCSPs are a subclass of DCOPs in which all constraint costs (weights) are one, and all agents search for the minimal cost assignment [13, 6]. The asymmetric version of such problems, Max-ADCSPs, includes *asymmetric constraints* with a cost of one. The second set of experiments uses random graphical games [5, 15, 7]. In these problems, each constraint between two agents represents a local randomly generated game. In these local interactions, each constrained agent is assigned a cost in the range [0..9] for each joint action (assignment pair) of the two constraining agents. The goal of the agents is to reach a globally minimal cost assignment.

Five different constraint graphs were considered in the Max-ADCSP setup, and ten other graphs in the random graphical games. Both involved 100 agents each with an average of 10 neighbors and a domain of size 10 values. In the case of Max-ADCSPs, a random 20% of each agent's values resulted in a "broken constraint" (of cost 1), with each one of its neighbors (note that this translates to a 36% chance that an assignment pair will have a cost higher than zero). In the case of the random games, each constraint matrix included 50% zeroes, while the rest of the assignment pair values had random values in the range [0..9].

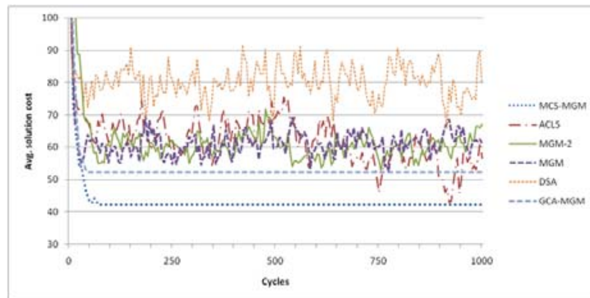

Figure 8: Solution quality - Max-ADCSP

A large number of algorithms were examined in both setups. These include *DSA*, *MGM*, *MGM-2*, *Max-Sum*, *ACLS*, *MCS-MGM* and *GCA-MGM* (with different parameters). These algorithms were executed for a maximum of 1000 cycles, where a cycle includes all actions between two consecutives *value* messages sent by the same agent (*Max-Sum* cycles include both P and Q type messages).

Figure 8 presents the average solution quality for each one of the algorithms as a function of cycles. *MCS-MGM* produced the highest quality results and *GCA-MGM* produced solutions of slightly lower quality. After 1000 cycles the three best algorithms were *MCS-MGM, GCA-MGM* and *ACLS* (average costs of 42.2, 52.2, 56.6 respectively). The highest costs (e.g., worst solutions) were reported by *Max-Sum* with average costs of 190.6 after 1000 cycles (these results fall beyond the scope of the graph in Figure 8). *Max-Sum* converged to its final solution in less than 5 cycles, and did not further explore the search space. Both *MCS-MGM* and *GCA-MGM* converged fast and within a surprisingly low number of cycles - 49.2 (*MCS-MGM*) and 36 (*GCA-MGM*) cycles on average.

Also of interest is the relation between *MGM* and *MGM-2*. While [7] reports higher quality solutions for *MGM-2* over *MGM* on standard DCOPs, our results indicate that in the asymmetric case, *MGM* and *MGM-2* provide roughly the same quality of results (with a slight advantage to *MGM*'s final solution - a cost of 67 for *MGM-2* and a cost of 61.4 for *MGM*). In *MGM-2*, an agent optimizing for itself and another agent can cause neighboring agents of both an increase in their valuation of the proposed alternative state. As a result, agents optimizing for different pairs can generate loops of assignment changes just as described for *MGM*. Thus, the increase in the size of the group of agents considered by the optimizing agent is not sufficient to insure convergence. A similar phenomenon where in the presence of uncertainty *MGM-2* fails to provide higher quality solutions than *MGM* was reported by [21].

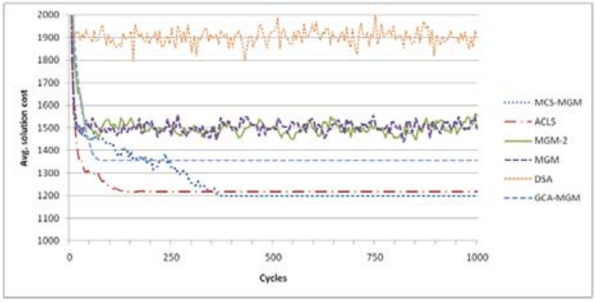

Figure 9: Solution quality - random graphical games

The above results for Max-ADCSPs are for the large part consistent with the results of the second setup - random graphical games (Figure 9). As before, the highest quality solutions were produced by *MCS-MGM, ACLS* and *GCA-MGM* (an average cost of 1198.4, 1217.6 and 1355.5 respectively). However, in the random games setup ACLS converge to a solution faster than *MCS-MGM* (60.3 cycles for *GCA-MGM*, 124.5 for *ACLS* and 268 for *MCS-MGM*), and to a better solution than that of *GCA-MGM*. In this setting *Max-Sum* failed to converge. Its solution cost moved between 2300 and 2500 (again, beyond the scope of the graph in Figure 9).

The above results indicate that the three proposed algorithms ,*MCS-MGM, GCA-MGM* and *ACLS*, are better suited for the asymmetric case. Clearly, the cooperation of agents running these algorithms allows them to achieve better solutions. Such cooperation, however, requires some revelation of private information. Thus, it is important to asses the privacy loss resulting from the coordination of agents (in contrast to standard local search (1-opt) algorithms which are preserving high level of privacy [3]). To measure the overall loss of privacy in our system of agents one needs to aggregate the number of revealed constraint parts by each agent [4, 3].

In *ACLS*, a fraction of the constraint is revealed in line 10 (Figure 6), while *MCS-MGM* and *GCA-MGM* reveal constraint information in lines 8 and 9 (Figure 7). Another algorithm that attempts to coordinate joint moves is *MGM-2* [7], in which *offerer* agents propose several improving assignments along with their costs to one of their peers, which respond with the lowest improving cost incurred on them and thus revealing a much larger fraction of the constraint in every such interaction.

Figures 10 and 11 present the privacy loss measurements. Agents running *MGM-2* reveal most of their problem structure, while the other algorithms maintain a substantially higher degree of constraint privacy. In the case of Max-ADCSPs (Figure 10), *MCS-MGM* reveals 0.244% of constraint information, *GCA-MGM* reveals 0.32%, *ACLS* reveals 57.25% and *MGM-2* reveals 82.06% of the private constraint information. Privacy loss measurements for the random graphical games are presented in Figure 11. Privacy loss is 1.28%, 1.58%, 5.819% and 95.19% respectively. Note that *ACLS*'s exhibits a great improvement in privacy preservation. This improvement is due to the fact that in the graphical games problems setup, *ACLS* agents managed to converge to a solution within a short number of cycles. In fact, the slight change in privacy loss for *MCS-MGM* and *GCA-MGM* may also be explained by the longer time to convergence. We conclude that quick convergence to a solution may have a substantial impact on privacy loss.

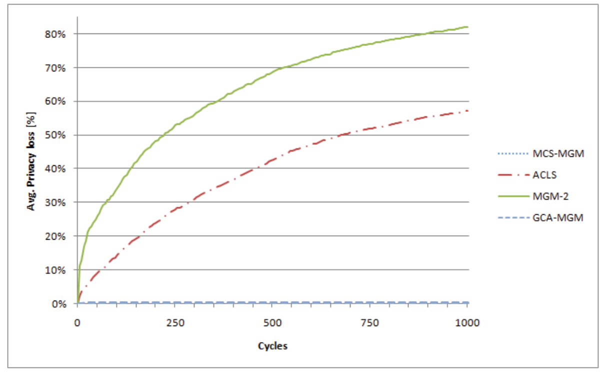

Figure 10: Privacy loss - Max-ADCSP

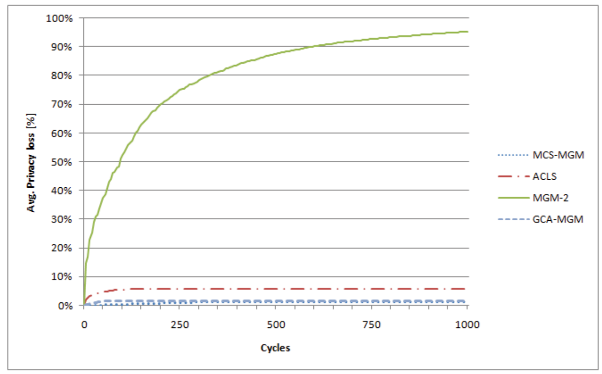

Figure 11: Privacy loss - random graphical games

6. CONCLUSIONS

Many problems which are distributed by nature include agents which have different valuations of the possible states of the world. For distributed constraints optimization problems agents can have different costs assigned to valued constraints. The present paper describes the shortcomings of representing asymmetric interactions by the standard DCOP model. An alternative model - Asymmetric Distributed Constraint Optimization (ADCOP) is proposed. The proposed ADCOP represent private gains without revealing private information a priori. Instead, agents reveal only the information which is necessary during the distributed search for a solution. This is in contrast to alternative DCOP formulations which either centralize constraints (reveal all information and may result in a heavy network load), or change the problem into a more complex structure (PEAV).

The algorithmic impact of introducing a new framework was discussed, as well as the applicability of existing DCOP algorithms. Three novel algorithms for ADCOPs were proposed: *ACLS, MCS-MGM* and *GCA-MGM*. In these algorithms agents cooperate and perform search in their local neighborhood, instead of maximizing their own gain. A convergence proof for *GCA-MGM*, which is of great importance in terms of privacy preservation, was presented.

Extensive empirical evaluation of the proposed ADCOP algorithms was performed on two large scale multi agents setups. These demonstrated that the new algorithms consistently find higher quality solutions, and do so with a high degree of privacy preservation. It turns out that the convergence property limits strongly the amount of privacy loss.

7. ACKNOWLEDGMENTS

This work was supported by the Lynn and William Frankel Center for Computer Sciences, and by the Paul Ivanier Center for Robotics and Production Management at BGU.

8. REFERENCES

[1] A. C. Chapman, A. Rogers, and N. R. Jennings. A parameterisation of algorithms for distributed constraint optimisation via potential games. In *Proc. AAMAS Workshop on Distributed Constraint Reasoning (DCR-08)*, pages 99–113, Estoril, Portugal, 2008.

[2] A. Gershman, A. Meisels, and R. Zivan. Asynchronous forward bounding. *J. of Artificial Intelligence Research*, 34:25–46, 2009.

[3] R. Greenstadt. An analysis of privacy loss in k-optimal algorithms. In *Workshop on Distributed Constraints Reasoning (DCR08), at AAMAS 2008*, Estoril, Portugal, May 2008.

[4] R. Greenstadt, J. Pearce, and M. Tambe. Analysis of privacy loss in distributed constraint optimization. In *AAAI-06*, pages 647–653, Boston, MA, USA, July 2006.

[5] M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 253–260, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[6] J. Larrosa and T. Schiex. Solving weighted csp by maintaining arc consistency. *Artificial Intelligence*, 159:1–26, 2004.

[7] R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *Proc. Parallel and Distributed Computing Systems PDCS)*, pages 432–439, September 2004.

[8] R. T. Maheswaran, J. P. Pearce, and M. Tambe. A family of graphical-game-based algorithms for distributed constraint optimization problems. In *Coordination of Large-Scale Multiagent Systems*, pages 127–146. Springer-Verlag, 2006.

[9] R. T. Maheswaran, J. P. Pearce, P. Varakantham, E. Bowring, and M. Tambe. Valuations of possible states (VPS): a quantitative framework for analysis of privacy loss among collaborative personal assistant agents. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1030–1037. ACM, 2005.

[10] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proc. 3rd Intern. Joint Conf. on Autonomous Agents & Multi-Agent Systems (AAMAS-04)*, pages 310–317, NY, New York, 2004.

[11] R. Mailler and V. R. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS-04*, pages 438–445, 2004.

[12] A. Meisels. *Distributed Search by Constrained Agents: Algorithms, Performance, Communication*. Springer Verlag, 2007.

[13] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: asynchronous distributed constraints optimization with quality guarantees. *Artificial Intelligence*, 161:1-2:149–180, January 2005.

[14] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[15] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[16] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI*, pages 1446–1451, 2007.

[17] A. Petcu. *A class of algorithms for distributed constraint optimization*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2007.

[18] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proc. IJCAI-05*, pages 266–271, Edinburgh, Scotland, UK, 2005.

[19] A. R. R. Stranders, A. Farinelli and N. R. Jennings. Decentralised coordination of continuously valued control parameters using the max-sum algorithm. In *Proc. 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Budapest, Hungary, May, 10Ű15, 2009.

[20] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of continuously valued control parameters using the max-sum algorithm. In *AAMAS (1)*, pages 601–608, 2009.

[21] M. E. Taylor, M. Jain, P. Tandon, and M. Tambe. Using dcops to balance exploration and exploitation in time-critical domains. In *Proc. Workshop on Distributed Constraints Reasoning (DCR09), at IJCAI-09*, Pasadena, CA, USA, July 2009.

[22] M. Yokoo. Algorithms for distributed constraint satisfaction problems: A review. *Autonomous Agents & Multi-Agent Sys.*, 3:198–212, 2000.

[23] M. Yokoo, K.Suzuki, and K. Hirayama. Secure distributed constraints satisfaction: Reaching agreement without revealing private information. In *Proc. CP-2002*, pages 387–401, Ithaca, NY, USA, September 2002.

[24] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. *Artificial Intelligence*, 161:1-2:55–88, January 2005.

[25] R. Zivan. Anytime local search for distributed constraint optimization. In *AAAI-2008*, Chicago, USA, 2008.