Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Applying Max-Sum to Asymmetric Distributed Constraint Optimization

Roie Zivan and **Tomer Parash** and **Yarden Naveh**
Industrial Engineering and Management Department
Ben-Gurion University of the Negev
Beer-Sheva, Israel
{zivanr,parasht,navehya}@bgu.ac.il

Abstract

We study the adjustment and use of the Max-sum algorithm for solving Asymmetric Distributed Constraint Optimization Problems (ADCOPs). First, we formalize asymmetric factor-graphs and apply the different versions of Max-sum to them. Apparently, in contrast to local search algorithms, most Max-sum versions perform similarly when solving symmetric and asymmetric problems and some even perform better on asymmetric problems. Second, we prove that the convergence properties of Max-sum_ADVP (an algorithm that was previously found to outperform other Max-sum versions) and the quality of the solutions it produces are dependent on the order between nodes involved in each constraint, i.e., the *inner constraint order* (ICO). A standard ICO allows to reproduce the properties achieved for symmetric problems, and outperform previously proposed local search ADCOP algorithms. Third, we demonstrate that a non-standard ICO can be used to balance exploration and exploitation, resulting in the best performing Max-sum version on both symmetric and asymmetric standard benchmarks.

1 Introduction

Autonomous agents in a distributed environment may have different valuations of states of the world (i.e., of constraints they are involved in). Such scenarios, e.g., meeting scheduling, where different agents have different valuations for meetings they are participating in[1], cannot be represented by the standard Distributed Constraint Optimization Problems (DCOP) model in which constraints are symmetric, thus, extensions of the DCOP model were proposed for representing problems with asymmetric constraints [Maheswaran *et al.*, 2004; Petcu, 2007; Grinshpoun *et al.*, 2013].

The first attempt to capture asymmetric valuations among constrained agents was by duplicating variables involved in an asymmetric constraint and using rigid constraints to enforce equality of assignments with other agents. This scheme was termed Private Events as Variables (PEAV) in [Maheswaran *et al.*, 2004]. The PEAV formulation allows the use of complete algorithms designed for symmetric DCOPs. In contrast, PEAV does not enable the use of standard local search algorithms since every allocation that satisfies the hard equality constraints in PEAV, is a local optimum, which cannot be escaped by standard local search algorithms [Grinshpoun *et al.*, 2013]. Previous studies have shown that nodes representing hard equality constraints are redundant in factor-graphs [Penya-Alba *et al.*, 2012; Kschischang *et al.*, 2001], and thus, the PEAV representation is not compatible with algorithms that operate on factor-graphs such as Max-sum.[2]

An alternative model for representing asymmetric DCOPs (ADCOP) was proposed in [Grinshpoun *et al.*, 2013]. Agents in ADCOP hold their part of each constraint, similar to a normal form game. Standard local search algorithms that are guaranteed to converge when solving symmetric problems, are not guaranteed to converge when solving ADCOPs and evidently produce poor results. Thus, local search algorithms that guarantee convergence by exchanging constraint costs among agents (and thus, violating privacy), were proposed in [Grinshpoun *et al.*, 2013].

Max-sum is an inference (*GDL*) incomplete algorithm that has drawn much attention recently [Farinelli *et al.*, 2008; Rogers *et al.*, 2011; Zivan and Peled, 2012]. Agents in Max-sum calculate and propagate utilities (or costs) for each possible value assignment of their neighboring agents' variables.

Previous studies have revealed that when Max-sum performs on a cyclic constraint graph it does not always converge and it traverses states with low quality [Farinelli *et al.*, 2008; Zivan and Peled, 2012]. A number of studies have proposed versions of Max-sum that guarantee convergence. One required the elimination of some of the problem's constraints in order to reduce the DCOP to a tree structured problem that can be solved optimally in polynomial time [Rogers *et al.*, 2011]. A different approach had Max-sum perform on an alternating directed acyclic graph (DAG). This algorithm, (Max-sum_AD), converges after a linear number of iterations without requiring the removal of edges from the constraint graph. The alternation of the order on each edge of the

[1]For additional examples, including supply chain and smart grid applications, see [Grinshpoun *et al.*, 2013].

[2]We thank an anonymous reviewer of a previous version of this paper that indicated that this redundancy can be derived from [Penya-Alba *et al.*, 2012; Kschischang *et al.*, 2001] and allowed us to omit our own redundancy proof.

DAG after each convergence, ensures that constraints are not ignored. Adding value propagation results in an algorithm (Max-sum_ADVP) that visits (weakly) monotonic improving states after changing directions back and forth until convergence [Zivan and Peled, 2012].

In this paper we contribute to the development of incomplete algorithms for solving Asymmetric DCOPs by:

1. Formalizing asymmetric factor-graphs, adjusting the different versions of Max-sum for solving them and analyzing their performance. Each constraint in an asymmetric factor-graph is represented by a number of nodes relative to the arity of the constraint. The order among them (i.e., the *inner constraint order*, ICO) was found to affect the performance of Max-sum versions in which an order is used, e.g., Max-sum_ADVP.
2. We prove that if all constraints are ordered according to a *standard* inner order (SIO), the performance of Max-sum_ADVP on ADCOPs is equivalent to its performance on standard DCOPs, thus, the convergence guarantees and the quality of the produced solutions are the same. Furthermore, it outperforms the local search algorithms designed specifically for ADCOPs in [Grinshpoun *et al.*, 2013] while preserving a higher level of privacy.
3. We propose the use of a non-standard ICO for balancing exploration and exploitation in Max-sum_ADVP, resulting in the best performing version of Max-sum for both asymmetric and standard symmetric DCOPs on a variety of standard benchmarks.

2 Background

In this section we first present formal descriptions of DCOPs and ADCOPs, then we present Max-sum and its extensions that guarantee convergence.

2.1 Distributed Constraint Optimization

Without loss of generality, in the rest of this paper we will assume that all problems are minimization problems. The inference algorithm for minimization problems is actually *Min-sum*. However, we will continue to refer to it as *Max-sum* since this name is widely accepted. Our description of a DCOP is consistent with the definitions in many DCOP studies, e.g., [Modi *et al.*, 2005; Petcu and Faltings, 2005].

A $DCOP$ is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$. $\mathcal{A}$ 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. $\mathcal{D}$ is a set of domains $\{D_1, D_2, ..., D_m\}$. Each domain D_i contains the finite set of values that can be assigned to variable x_i. An assignment of value $d \in D_i$ to x_i is denoted by an ordered pair $\langle x_i, d \rangle$. $\mathcal{R}$ 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 $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$.[3] A *binary DCOP* is a DCOP in which all constraints are binary. A *partial assignment* (PA) is a set of value assignments to variables, in which each variable appears at most once. *vars(PA)* is the set of all variables that appear 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 PA* is the sum of all applicable constraints to PA over the assignments in PA. A *complete assignment* (or a *solution*) is a partial assignment that includes all the DCOP's variables ($vars(PA) = \mathcal{X}$). An *optimal solution* is a complete assignment with minimal cost.

[3]We say that a variable is *involved* in a constraint if it is one of the variables the constraint refers to.

For simplicity we make the standard assumptions that all DCOPs are binary DCOPs in which each agent holds exactly one variable, its domain and all constraints that its variable is involved in. These assumptions are commonly made in DCOP studies, e.g., [Modi *et al.*, 2005; Petcu and Faltings, 2005; Gershman *et al.*, 2009]. Furthermore, the studies that proposed the ADCOP model also focused on binary problems where each agent holds exactly one variable [Grubshtein *et al.*, 2010; Grinshpoun *et al.*, 2013]. Thus, although we present a general description of ADCOPs next, our focus in this paper will be on binary ADCOPs.

2.2 Asymmetric DCOP

ADCOPs generalize DCOPs by explicitly defining for each combination of assignments of constrained agents, the exact cost for each participant in the constraint.

More formally, an ADCOP is defined by the following tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$, where $\mathcal{A}$, $\mathcal{X}$ and $\mathcal{D}$ are defined the same 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: $C : D_{i_1} \times D_{i_2} \times \cdots D_{i_k} \rightarrow \mathbb{R}^k_+$. Notice that here $\mathbb{R}^k_+$ is a vector that includes for each agent $1 \leq j \leq k$ its cost for each combination of value assignments, so in practice each agent $1 \leq j \leq k$ holds its part of the constraint C_j, $C_j : D_{i_1} \times D_{i_2} \times \cdots D_{i_k} \rightarrow \mathbb{R}_+$

We will make the following definition to maintain the relation between standard DCOPs and ADCOPs:

Definition 1 *DCOP P is equivalent to ADCOP P' if their sets of agents, variables and domains are equal, i.e., $A = A'$, $V = V'$ and $D = D'$, and a constraint C is included in R if and only if there exists a constraint $C' \in R'$ such that both refer to the same set of variables and $C = \sum_{0 \leq i \leq k} C'_i$.*

It is easy to see that each asymmetric problem has a single equivalent symmetric problem. However, there can be an infinite number of asymmetric problems that have the single same equivalent symmetric problem.

2.3 Max-sum

[4]Max-sum operates on a *factor-graph*, which is a bipartite graph in which the nodes represent variables and constraints [Kschischang *et al.*, 2001]. Each variable-node representing a variable of the original DCOP is connected to all function-nodes that represent constraints, which it is involved

[4]For lack of space we describe the algorithm and its extensions briefly and refer the reader to more detailed descriptions in [Farinelli *et al.*, 2008; Rogers *et al.*, 2011; Zivan and Peled, 2012].

in. Similarly, a function-node is connected to all variable-nodes that represent variables in the original DCOP that are involved in the constraint it represents. Variable-nodes and function-nodes are considered "agents" in Max-sum, i.e., they can send and receive messages, and perform computation.

A message sent to or from variable-node x (for simplicity, we use the same notation for a variable and the variable-node representing it) is a vector of size $|D_x|$ including a cost for each value in D_x. In the first iteration all messages include vectors of zeros. A message sent from a variable-node x to a function-node f is formalized as follows: $Q^i_{x\to f} = \sum_{f'\in F_x, f'\neq f} R^{i-1}_{f'\to x} - \alpha$, where $Q^i_{x\to f}$ is the message variable-node x intends to send to function-node f in iteration i, F_x is the set of function-node neighbors of variable-node x and $R^{i-1}_{f'\to x}$ is the message sent to variable-node x by function-node f' in iteration $i-1$. α is a constant that is reduced from all costs included in the message (i.e., for each $d \in D_x$) in order to prevent the costs carried by messages throughout the algorithm run from growing arbitrarily.

A message sent from a function-node f to a variable-node x in iteration i includes for each value $d \in D_x$: $min_{PA_{-x}} cost(\langle x, d\rangle, PA_{-x})$, where PA_{-x} is a possible combination of value assignments to variables involved in f not including x. The term $cost(\langle x, d\rangle, PA_{-x})$ represents the cost of a partial assignment $a = \{\langle x, d\rangle, PA_{-x}\}$, which is: $f(a) + \sum_{x'\in X_f, x'\neq x, \langle x', d'\rangle \in a} Q^{i-1}_{x'\to f}.d'$, where $f(a)$ is the original cost in the constraint represented by f for the partial assignment a, X_f is the set of variable-node neighbors of f, and $Q^{i-1}_{x'\to f}.d'$ is the cost that was received in the message sent from variable-node x' in iteration $i-1$, for the value d' that is assigned to x' in a. x selects its value assignment $\hat{d} \in D_x$ following iteration k as follows: $\hat{d} = \arg\min_{d\in D_x} \sum_{f\in F_x} R^k_{f\to x}.d$.

Since Max-sum is not guaranteed to converge when solving cyclic factor-graphs, versions of the algorithm that guarantee convergence were proposed. Bounded Max-sum eliminates edges of a cyclic factor-graph until it transforms to a tree structured factor-graph [Rogers *et al.*, 2011]. Then, standard Max-sum is used to find the optimal solution for the tree structured factor-graph. The aggregated maximal costs of the edges removed, bound the distance of the solution cost from the optimum. Recent studies proposed methods for selecting the removed edges, which tighten the bound [Rollon and Larrosa, 2012; 2014].

An alternative method for guaranteeing convergence was proposed in [Zivan and Peled, 2012]. In order to avoid cycles, the algorithm is performed on a DAG, determined by an order on all nodes in the factor-graph (e.g., according to agents' indices). Each node sends messages only to neighboring nodes that follow it in the selected order. After a linear number of iterations in which the algorithm is guaranteed to converge, the order is reversed and messages are sent in the opposite direction. The order is alternated following each convergence (the end of each *phase*) until termination.

Selecting an order includes a decision on where to place each (binary) function-node, with respect to its neighboring variable-nodes, i.e., determine the *inner constraint order*

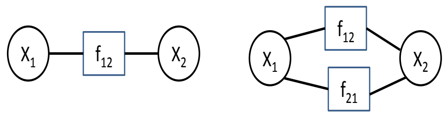

Figure 1: A factor-graph with a DCOP constraint (on the left) and one with an ADCOP constraint (on the right)

(ICO). We refer to an ICO that has the function-node ordered after one neighboring variable-node and before the other as a *standard inner order* (SIO). A *standard order* on all nodes of the factor-graph is an order in which all ICOs are standard.

In [Zivan and Peled, 2012] only standard orders were used, since any other alternative,i.e., placing the function-node before or after its two neighboring variable-nodes, results in a redundant execution where either the function-node or the variable nodes do not receive messages.[5] Therefore, we assume that all orders used by Max-sum_AD and Max-sum_ADVP, when solving symmetric DCOPs, are standard.

While Max-sum_AD guarantees convergence in each phase, agents may propagate inconsistent costs, calculated while assuming different value assignments for the same variable [Zivan and Peled, 2012]. This can be overcome by using value propagation, as used in complete inference algorithms for avoiding ties [Petcu and Faltings, 2005; Vinyals *et al.*, 2011]. In the resulting algorithm, Max-sum_AD with value propagation (Max-sum_ADVP), variable-nodes include in their messages their selected value assignments. Function-nodes select costs while considering only the value assignments received from their variable-node neighbors.

For value propagation to be most effective, it is started only after two phases of Max-sum_AD, after all the problem's constraints were considered [Zivan and Peled, 2012].

3 Asymmetric Factor Graphs

An asymmetric factor-graph has each (binary) constraint represented by two function-nodes, one for each part of the constraint held by one of the involved agents. Each function-node is connected to both variable-nodes representing the variables involved in the constraint.

Figure 1 presents two factor-graphs including a single constraint between variables x_1 and x_2. On the left, the factor-graph represents a symmetric DCOP, therefore it includes a single function-node. The factor-graph on the right represents an ADCOP, hence, it includes two function-nodes, one held by each agent representing its part of the constraint.

Thus, a factor-graph representing a binary ADCOP has four nodes involved in each constraint, two function-nodes and two variable-nodes. Therefore, for Max-sum versions that order the nodes in the factor-graph, e.g., Max-sum_AD, these four nodes need to be ordered. Figure 2 presents three options to order the four nodes involved in a binary constraint.[6] All orders are considered from left to right. The top left order is the *standard inner order* (SIO), similar to the standard order used for symmetric DCOPs, in which both

[5]For asymmetric problems, other alternatives will be discussed.

[6]Again, we ignore orders where both variable-nodes come before or after both function-nodes (as discussed for symmetric problems).

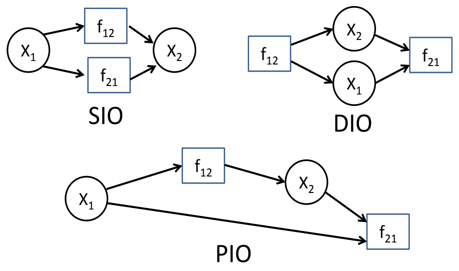

Figure 2: Different ICOs for a binary constraint.

function-nodes come after one variable-node and before the other. The top right ICO has the two variable-nodes ordered after one function-node and before the other. We call it a *diverse inner order* (DIO). The last order on the bottom has one of the variable-nodes ordered first, a function-node following it, then another variable-node and finally the second function-node. This ICO is termed *partial inner order* (PIO). Notice, that when a PIO is reversed we have the function-node first, then a variable-node, another function-node and finally the second variable-node. Since orders are reversed in alternating phases, we will refer to both as PIOs. We will demonstrate that the selection of the ICOs has a major effect on the performance of the algorithms using them.

4 Properties

Definition 2 *An algorithm execution on a problem is a list of sets of messages $M_1, M_2, ..., M_k$, where k is the number of iterations the algorithm is executed for and for each $1 \leq i \leq k$ the set M_i includes all messages sent in iteration i.*[7]

Let E and E' be executions of algorithms solving equivalent problems.

Definition 3 *E in iteration i is equivalent to E' in iteration j if and only if there exists a set of n constant scalars $\{\beta_1, ..., \beta_n\}$ such that for every variable-node x_v, $1 \leq v \leq n$, $c_v = c'_v + \beta_v$, where c_v is a vector that includes the sum of messages sent to x_v in iteration i of E and c'_v is a vector including the sum of messages sent to x_v in iteration j of E'.*

The following lemma will help us establish the equivalence in the execution of Max-sum_ADVP on symmetric and asymmetric problems when SIO is used in both.
Let $C \in R$ be a binary constraint between variables x and x' in DCOP P and C_x and $C_{x'}$ the two parts of C in the equivalent ADCOP P'. Let f, f_x and $f_{x'}$ be the function-nodes representing C, C_x and $C_{x'}$ in the two factor-graphs respectively. Consider two executions E and E' of Max-sum_ADVP, E solving P and E' solving P', both using SIO for constraint C and both running for k iterations.

Lemma 1 *If in some iteration $i < k$ of E and in some iteration $j < k$ of E', x' is ordered after x and the same value d is selected for x, then there exists a constant scalar β such that: $R^{i+1}_{f \to x'} + \beta = R^{j+1}_{f_x \to x'} + R^{j+1}_{f_{x'} \to x'}$.*

Proof: Let $cost(d)$ be the cost for d sent by x to function-node f in iteration i of E and $cost(d)'$ be the cost for d sent by x to function-nodes f_x and $f_{x'}$ in iteration j of E'. In each of the messages sent in iterations $i+1$ and $j+1$ in E and E' repectively, from each of the function-nodes f, f_x and $f_{x'}$ to x', the costs for each $d' \in D_{x'}$ include the sum of $cost(d)$ (or $cost(d)'$ respectively) and the cost of the constraint for the assignment $\{\langle x, d\rangle, \langle x', d'\rangle\}$. By definition, the cost in C is equal to the sum of costs in C_x and $C_{x'}$ for this assignment. Thus, for each value $d' \in D_{x'}$:
$[R^{j+1}_{f_x \to x'}.d' + R^{j+1}_{f_{x'} \to x'}.d'] - R^{i+1}_{f \to x'}.d' = 2cost(d)' - cost(d)$.
Notice that since d was selected in the previous iterations in both executions (i in E and j in E'), $cost(d)$ and $cost(d)'$ are constants in iterations $i+1$ in E and $j+1$ in E', i.e., $\beta = 2cost(d)' - cost(d)$. □

The above lemma allows us to generalize the equivalence of the executions of Max-sum_ADVP on symmetric and asymmetric problems. Let DCOP P and ADCOP P' be equivalent and let o be an order on all variables in P. Let E and E' be executions of Max-sum_ADVP on P and P' respectively, both using o to order variable-nodes and SIO on all constraints, both running for k iterations.

Lemma 2 *If in some iteration $i < k$ of E and in some iteration $j < k$ in E', both executions are using order o (and not its reverse) and in both executions, for each variable x, the same value $d \in D_x$ is selected, then iteration $i+1$ in E and iteration $j+1$ in E' are equivalent.*

Proof: Immediate from Lemma 1. Since each variable in iterations $i+1$ in E and $j+1$ in E' receives messages from function-nodes representing the same constraints, it is clear that there exists a set of constant scalars $\{\beta_1, ..., \beta_n\}$ such that for each variable-node x_v, $1 \leq v \leq n$, the sum of messages it receives in iteration $i+1$ of E from its function-node neighbors plus β_v is equal to the sum of messages that x_v receives in iteration $j+1$ of E' from its function-node neighbors (β_v is the sum of differences between costs received from function-nodes of each constraint as described in Lemma 1).[8] □

Let E and E' be executions running Max-sum_ADVP on DCOP P and an equivalent ADCOP P' respectively. Assume both are using the same order o on all variable-nodes in P (and P') and SIO on all constraints. Furthermore, assume they both have the same phase length and start VP after the same number of iterations.

Proposition 1 *If in the first iteration of both E and E' in which VP is used, the same value assignments are selected by all variable-nodes, then all following iterations of E and E' are equivalent.*

Proof: Immediate from Lemma 2. According to the lemma, in the second iteration the sum of costs the variable-nodes receive are equal in both executions, hence, the value assignments selected in the second VP iteration will be the same in both executions. Obviously, this will be true in all the following iterations as well. □

The importance of the proposition is that, in contrast to other algorithms, Max-sum_ADVP can maintain its convergence properties and quality of solutions in the ADCOP

[7]All algorithms we consider are deterministic, thus, executions of the same algorithm on the same problem are identical.

[8]By sum of messages we mean the vector of costs and not the value selection added for VP of course.

model, as long as we apply the ordering restrictions and start from the same initial assignment. We demonstrate empirically that even if we do not impose the selection of the same value assignments, the performance of Max-sum_ADVP when solving DCOPs and its performance when solving ADCOPs are similar. We note that Proposition 1 applies for standard Max-sum with value propagation (without ordering restrictions of course) as well. For lack of space we chose to focus on the proof for Max-sum_ADVP because of its convergence properties and because it produces results with much higher quality (see Section 5).

On the other hand, execution of Max-sum_ADVP solving an ADCOP that is not using SIO, is not guaranteed to (and probably will not) be equivalent to its execution when solving an equivalent DCOP. For example, consider the setup described above in Lemma 1. Assume DIO is used and both variable-nodes x and x' are ordered after f_x and before $f_{x'}$. x and x' do not receive messages from $f_{x'}$ throughout the phase and therefore it is unlikely that the messages the variables receive from f in E will be equal to the messages they receive from f_x in E'.

If PIO is used then one variable-node receives messages only from one function-node. Obviously, the messages it receives will be different from the messages sent by the function-node in the symmetric case since only part of the constraint is considered in each phase.

Privacy

A basic motivation for solving ADCOPs is to maintain constraint privacy [Grinshpoun *et al.*, 2013]. For lack of space we omit most of the details of the formal analysis and empirical evaluation we performed, which revealed that all Max-sum versions we applied to ADCOP, preserve a higher level of privacy than the ADCOP local search algorithms proposed in [Grinshpoun *et al.*, 2013],[9] and provide only an intuitive explanation for this and the highlights of the comparison.

Let $C_j^{i,j}$ be the part of the constraint between A_i and A_j held by A_j and represented by function-node $f_{j,i}$. $C_j^{i,j}$ is a table that includes a cost for each combination of value assignments to x_i and x_j. The entropy for each such constraint is calculated by the ratio between the number of possible tables considering revealed information and the total number of possible tables [Greenstadt *et al.*, 2006; Grinshpoun *et al.*, 2013]. The local search ADCOP algorithms exchange entries of these tables. In Max-sum on the other hand, only in the first two iterations, the costs sent from function-nodes to variable-nodes are equal to entries in constraint tables and not to sums of multiple entries (as in complete inference algorithms [Greenstadt *et al.*, 2007]). Moreover, the cost $R^1_{f_{j,i} \rightarrow x_i}.d$ reveals some entry in a row of table $C_j^{i,j}$ and not a specific entry. Thus, a single entry exchanged in local search, reduces the entropy approximately as much as Max-sum reduces the entropy in its entire run. Our experiments[10] indicate that the average number of entries that each agent revealed to at least one of its neighbors, when performing the most private ADCOP local search algorithm, was 3.3 in uniform random problems, 4.74 in scale free networks and 12.4 in meeting scheduling problems, where privacy matters most.

[9]Since we start VP only after a large number of iterations, the additional privacy loss in versions that include VP is negligible.

[10]Experiments setup details are specified in Section 5.

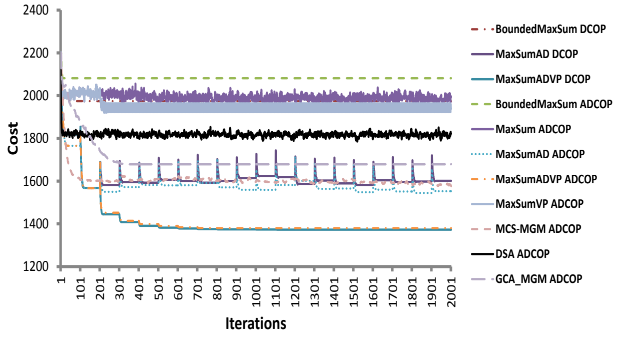

Figure 3: Solution cost for low density uniform random problems ($p_1 = 0.2$)

5 Experimental Evaluation

We compared the performance of Max-sum versions when solving ADCOPs in comparison to their performance when solving symmetric DCOPs and to the most successful local search algorithms, previously proposed for ADCOPs. Moreover, we demonstrate the advantage of using a non-standard ICO (PIO) when solving ADCOPs with Max-sum_ADVP.

The first set of experiments was performed on different benchmarks that are commonly used for evaluating DCOP algorithms: uniform random, scale free networks and meeting scheduling. In each of these we first generated ADCOP instances and then, their equivalent DCOPs.

The experiments on uniform problems included ADCOPs with 50 agents, each agent holding exactly one variable with ten values in its domain. Constraints were generated randomly in each experiment by selecting the probability p_1 for a constraint among any pair of agents/variables. Each cost held by an agent for a pair of assignments of values to a constrained pair of variables (its own variable and another) was selected uniformly between 1 and 10. The equivalent DCOPs were generated by summing the two parts of each binary constraint into a single symmetric constraint.

We compared the different versions of Max-sum: Bounded Max-sum, standard Max-sum, Max-sum_AD, Max-sum_VP[11] and Max-sum_ADVP. Each algorithm solved the same 50 ADCOPs and their equivalent DCOPs. The results presented are an average over these 50 runs. We also ran statistical significance tests between all results produced. Max-sum_AD and Max-sum_ADVP used a lexicographic order on the variable-nodes and SIO on all constraints. The phase lengths in each of them was 100 iterations (the longest possible path in the DAG, in case the graph has a chain structure). In all the versions of Max-sum we used random personal preferences for breaking ties as suggested in [Farinelli *et al.*, 2008]. For each of the algorithms we averaged over the sum of costs of constraints included in the assignment it would have selected, in each iteration.

[11]This version adds value propagation to standard Max-sum after 200 iterations (just like Max-sum_ADVP).

Figure 3 presents the costs of the solutions found by the five algorithms when solving problems of relatively low density ($p_1 = 0.2$). The results presented roughly divide the Max-sum versions to three sets that produced results with small differences between them. The standard Max-sum versions, with and without value propagation and Bounded Max-sum produce similar results with high costs.[12] The only exception is Bounded Max-sum solving ADCOPs that produces results with significantly higher costs. The Max-sum_VP version seems to repeatedly shift between alternating solutions, both when solving symmetric and asymmetric problems. While the differences between its results and the results of standard Max-sum are small, they were found to be significantly better. The Max-sum_AD algorithm, when solving both symmetric and asymmetric problems, performs significantly better than standard Max-sum and Max-sum_VP. Max-sum_ADVP for both types of problems performs best and converges after a small number of phases. These results are consistent with the results presented in [Zivan and Peled, 2012] for symmetric problems. It is apparent that most versions of the algorithm perform similarly on DCOPs and ADCOPs. The exceptions are Bounded Max-sum and Max-sum_AD. Bounded Max-sum produces better results when solving symmetric problems, since the asymmetric problem includes more edges that need to be removed, thus, more information is lost. The surprising results are produced by Max-sum_AD where in most phases the algorithm converges to significantly better results when solving asymmetric problems. The small differences between the performance of value propagating versions on symmetric and asymmetric problems are caused since in the first 200 iterations (two phases in the case of Max-sum_ADVP) values are not propagated.

In order to give a perspective on how the Max-sum versions perform compared to algorithms that were previously proposed for ADCOPs we included in the graph the results of MCS-MGM and GCA-MGM. The first is the incomplete algorithm that was found to perform best in [Grinshpoun *et al.*, 2013] and the second is a slightly different version that guarantees convergence. We also demonstrate how a general DCOP algorithm (DSA [Zhang *et al.*, 2005]) performs when solving ADCOPs. Obviously Max-sum_ADVP has a significant advantage over these algorithms.

Similar results were obtained for uniform problems with higher density ($p_1 = 0.6$), scale free networks and meeting scheduling problems (omitted for lack of space). Next, we investigated the effect of the selection of ICOs on the performance of the algorithms that use alternating DAGs.

Figure 4 presents the results when solving random DCOPs with low density. We ran the algorithms for 5000 iterations each, to allow all algorithms to converge (if possible). The DIO version of both algorithms failed to converge and produced identical poor results (thus, only one line is visible). This is probably because in DIO each function node either does not receive messages or does not send messages throughout an entire phase. Max-sum_AD solving

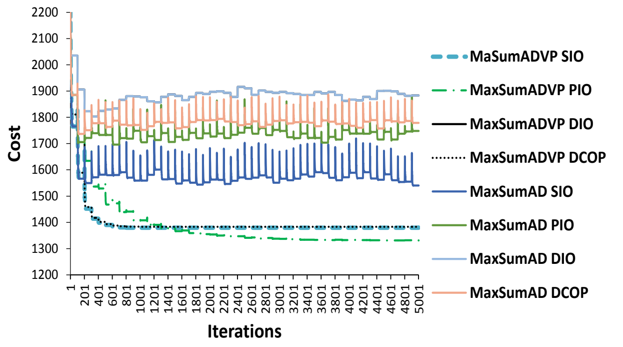

Figure 4: Solution cost of Max-sum versions using different ICOs when solving problems with low density ($p_1 = 0.2$)

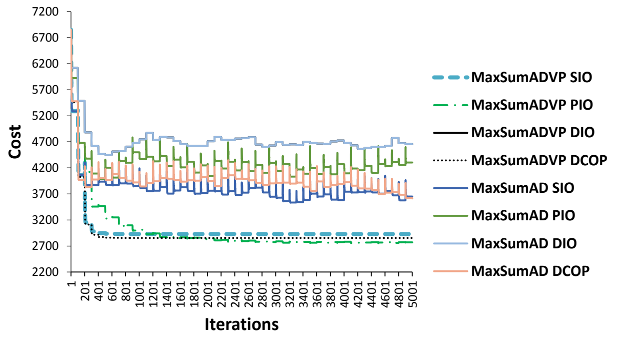

Figure 5: Solution cost of Max-sum versions using different ICOs when solving scale free networks

ADCOPs with SIO performs significantly better than Max-sum_AD solving the equivalent DCOPs. In the case of Max-sum_ADVP, as analyzed in Section 3, the ADCOP version with SIO performs similar to the DCOP version. On the other hand, the version using PIO converges to better solutions. We note that when using PIO, Max-sum_ADVP converges slower, so in case time is critical, SIO is the preferred option. However, if the algorithm is run for more than 1500 iterations, Max-sum_ADVP solving an ADCOP finds better solutions than its DCOP version.[13] Similar results were obtained for denser random problems (omitted for lack of space).

The results for scale free nets [Barabási, 2003] are presented in Figure 5. The results are similar, but the advantages of the ADCOP versions that were successful in the uniform random benchmarks over the DCOP versions are less apparent. Experiments on meeting scheduling problems produced similar results and were omitted for lack of space.

We note that the communication cost in the ADCOP versions of Max-sum is exactly two times the cost in the DCOP versions, i.e., the size of messages remains linear.

6 Conclusion

We adjusted different versions of Max-sum to asymmetric problems by formulating asymmetric factor-graphs. Our empirical results demonstrate that, in contrast to standard local search algorithms, which are not effective for asymmetric problems, most versions of Max-sum, when solving

[12]In order to avoid density we omitted the DCOP results of Max-sum and Max-sum_VP, which are similar to the ADCOP results produced by these algorithms.

[13]If time is indeed critical, the exact diameter of the graph can be calculated using a BFS algorithm and the phases can be shorten.

ADCOPs, produce similar results to the results they produce when solving standard DCOPs. In fact, versions of Max-sum with value propagation (as we proved for Max-sum_ADVP) are guaranteed under some restrictions when solving asymmetric problems, to produce the same results as when they solve symmetric problems, e.g., in the case of Max-sum_ADVP, to converge to qualitative solutions. The Max-sum versions were compared to a standard local search algorithm (DSA) and to local search algorithms that were designed for ADCOPs and overcome the weaknesses of standard local search by allowing agents to exchange constraints. Max-sum_ADVP was found to have a significant advantage over these algorithms when solving ADCOPs.

We further demonstrated that the results of versions of Max-sum that use an order on the nodes of the factor-graph, are dependent on the selection of the ICOs. While the use of a standard ICO guarantees for Max-sum_ADVP the same properties it has when solving symmetric problems, by using a non-standard ICO, a balance between exploration and exploitation is achieved such that on the benchmarks tested, the results it produces are even better than the results it produces when solving symmetric problems.

References

[Barabási, 2003] A. L. Barabási. *Scale Free Networks*. Scientific America, 2003.

[Farinelli *et al.*, 2008] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the seven'th international joint conference on Autonomous agents and multiagent systems (AAMAS) 2008*, pages 639–646, 2008.

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

[Greenstadt *et al.*, 2006] 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.

[Greenstadt *et al.*, 2007] R. Greenstadt, B. J. Grosz, and M. D. Smith. SSDPOP: improving the privacy of DCOP with secret sharing. In *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007*, page 171, 2007.

[Grinshpoun *et al.*, 2013] T. Grinshpoun, A. Grubshtein, R. Zivan, A. Netzer, and A. Meisels. Asymmetric distributed constraint optimization problems. *Journal of Artificial Intelligence Research (JAIR)*, 47:613–647, 2013.

[Grubshtein *et al.*, 2010] A. Grubshtein, R. Zivan, T. Grinshpon, and A. Meisels. Local search for distributed asymmetric optimization. In *Proc. of the 9th conference on Autonomous Agents and Multi Agent Systems (AAMAS 2010)*, pages 1015–1022, May 2010.

[Kschischang *et al.*, 2001] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 47:2:181–208, Febuary 2001.

[Maheswaran *et al.*, 2004] 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. AAMAS-04*, pages 310–317, New York, NY, 2004.

[Modi *et al.*, 2005] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraints optimizationwith quality guarantees. *Artificial Intelligence*, 161:1-2:149–180, January 2005.

[Penya-Alba *et al.*, 2012] T. Penya-Alba, J. Cerquides, J. A. Rodríguez-Aguilar, and M. Vinyals. Scalable decentralized supply chain formation through binarized belief propagation. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, pages 1275–1276, 2012.

[Petcu and Faltings, 2005] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 266–271, 2005.

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

[Rogers *et al.*, 2011] A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artif. Intell.*, 175(2):730–759, 2011.

[Rollon and Larrosa, 2012] E. Rollon and J. Larrosa. Improved bounded max-sum for distributed constraint optimization. In *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 624–632, 2012.

[Rollon and Larrosa, 2014] E. Rollon and J. Larrosa. Decomposing utility functions in bounded max-sum for distributed constraint optimization. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, pages 646–654, 2014.

[Vinyals *et al.*, 2011] M. Vinyals, J. A. Rodríguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems*, 22(3):439–464, 2011.

[Zhang *et al.*, 2005] 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.

[Zivan and Peled, 2012] R. Zivan and H. Peled. Max/min-sum distributed constraint optimization through value propagation on an alternating DAG. In *AAMAS*, pages 265–272, 2012.