Utilitarian Approach to Privacy in Distributed Constraint Optimization Problems

Julien Savaux and **Julien Vion** and **Sylvain Piechowiak** and **René Mandiau**
LAMIH UMR CNRS 8201, University of Valenciennes, France

Toshihiro Matsui
Nagoya Institute of Technology, Japan

Katsutoshi Hirayama
Kobe University, Japan

Makoto Yokoo
Kyushu University, Japan

Shakre Elmane and **Marius Silaghi**
Florida Institute of Technology, USA

Abstract

Privacy has been a major motivation for distributed problem optimization. However, even though several methods have been proposed to evaluate it, none of them is widely used. The Distributed Constraint Optimization Problem (DCOP) is a fundamental model used to approach various families of distributed problems. Here we approach the problem by letting both the optimized costs found in DCOPs and the privacy requirements guide the agents' exploration of the search space. We introduce Utilitarian Distributed Constraint Optimization Problem (UDCOP) where the costs and the privacy requirements are used as parameters to a heuristic modifying the search process. Common stochastic algorithms for decentralized constraint optimization problems are evaluated here according to how well they preserve privacy.

1 Introduction

In artificial intelligence, cognition for the purposes of machines remains an open problem (Dessimoz 2016), and Distributed Constraint Optimization Problems (DCOPs) have been a standard and efficient way of modeling an agent's knowledge representation. In this work, we focus on problems modeled as such. In DCOPs, agents have to find values to a set of shared variables while optimizing a cost function. To find such assignments, agents exchange messages (frequently assumed to have unspecified privacy implications) to explore the search space until an optimal solution is found or a termination condition is met, causing privacy to be a major concern in DCOPs.

In this context, the assumption for search optimization problems is that agents designed for utility estimation are able to associate each state with a utility value (Russell et al. 2003). As such, the utility of each action is given by the difference between the utilities of initial and final states. If a user is concerned about privacy, then such a user can associate a utility value with the privacy of each piece of information in the definition of his local problem. If a user is interested in solving the problem, he must be also able to quantify the utility he draws from finding the solution. In a maximization DCOP we assume that the utility a user obtains from an assignment is represented by the values of the local constraints for that assignment. Alternatively, with a minimization DCOP, the constraints would represent the costs. These utilities can be modeled as a component in a multi-criteria DCOP (Bowring, Tambe, and Yokoo 2005).

We evaluate how much privacy is lost by the agents during the problem solving process, by the total utility of the secrecy of each information that was revealed. For example, proposing an assignment with a value assigned to a variable has a privacy cost quantifying the desire of the agent to maintain that value's existence private. We propose a DCOP framework for utility-based agents, where the cost of privacy as well as the reward of each solution is explicitly expressed. The framework is called Utilitarian Distributed Constraint Optimization Problem (UDCOP). Simple extensions to standard stochastic algorithms are studied to verify the impact of this interpretation of privacy. We then evaluate and compare several stochastic algorithms according to how well they preserve privacy. To do so, we generate distributed meeting scheduling (DMS) problems (Maheswaran et al. 2004; Gershman et al. 2008).

This paper is organized as follows: Section 2 discusses existing solvers and approaches to privacy for DCOPs. Section 3 defines the concepts involved in UDCOPs and extensions to common stochastic DCOP solvers that modify the search process to preserve privacy. Section 4 discusses the differences between our approach and existing ones. Section 5 presents experimental results. Conclusion and perspectives are presented in Section 6.

2 Background

Distributed Constraint Optimization Problems (DCOPs) have been extensively studied as a fundamental way of modeling combinatorial optimization problems in multi-agent systems. These problems have been addressed with a variety of algorithms, both stochastic and deterministic. In this paper, we focus on stochastic algorithms. Let us first review the most relevant literature concerning frameworks, stochastic algorithms, and approaches to privacy for DCOPs.

Existing Frameworks

This section presents existing frameworks used to model distributed combinatorial problems.

Distributed Constraint Optimization Problems (DCOP) is the formalism commonly used to model combinatorial problems distributed between several agents.

Definition 1. *A DCOP is a tuple $\langle A, V, D, C\rangle$ where:*

- $A = \langle A_1, ..., A_n\rangle$ *is a vector of* n *agents*
- $V = \langle x_1, ..., x_n\rangle$ *is a vector of* n *variables. Each agent* A_i *controls the variable* x_i*.*
- $D = \langle D_1, ..., D_n\rangle$ *is a vector of domains where* D_i *is the domain for the variable* x_i*, known only by* A_i*, and a subset of* $\{1, ..., d\}$*.*
- $C = \langle c_1, ..., c_m\rangle$ *is a vector of weighted constraints, each one defining a cost for each tuple of a relation between variables in* V*.*

The objective is to find an assignment for each variable that minimizes the total cost.

Multi-Objective DCOP A Multi-Objective Distributed Constraint Optimization Problem (MO-DCOP) is an extension of the standard mono-objective DCOPs. An MO-DCOP is defined with a set of agents A, a set of variables X, multi-objective constraints $C = \{C^1, ..., C^m\}$, i.e., a set of sets of constraint relations, and multi-objective functions $O = \{O^1, ..., O^m\}$, i.e., a set of sets of objective functions. For an objective $l(1 \leq l \leq m)$, a cost function $f^l{}_{i,j} : D_i \times D_j \rightarrow R$, and a value assignment to all variables X, let us denote:

$$R^l(X) = \sum_{(i,j)\in C^l, \{(x_i,d_i),(x_j,d_j)\}\subseteq A} f^l{}_{i,j}(d_i, d_j), \quad (1)$$

where $d_i \in D_i$ and $d_j \in D_j$

Then, the sum of the values of all cost functions for m objectives is defined by a cost vector, denoted $R(A) = (R^1(A), ..., R^m(A))$. Finding an assignment that minimizes all objective functions simultaneously is ideal. However, in general, since trade-offs exist among objectives, there does not exist such an ideal assignment. Thus, the optimal solution of an MO-DCOP is characterized by using the concept of Pareto optimality. Because of this possible trade-off between objectives, the size of the Pareto front is exponential in the number of agents, i.e., in the worst case, every possible assignment can be a Pareto solution.

Example 1. *Suppose a problem concerning scheduling a meeting between three students. They all consider to agree on a place to meet on a given time, to choose between London, Madrid and Rome. For simplicity, in the next sections, we will refer to these possible values by their identifiers: 1, 2 and 3. Student* A_1 *lives in Paris, and it will cost him* $70*,* $230 *and* $270 *to attend the meeting in London, Madrid and Rome respectively. Student* A_2 *lives in Berlin, and it will cost him* $120*,* $400 *and* $190 *to attend the meeting in London, Madrid and Rome respectively. Student* A_3 *lives in Brussels, and it will cost him* $40*,* $280 *and* $230 *to attend the meeting in London, Madrid and Rome respectively. The objective is to find the meeting location that minimizes the total cost students have to pay in order to attend.*

The privacy costs for revealing her cost for locations 1, 2, and 3 for Student A_1 *are* $\$80, \$20, \$40$*. The privacy cost for locations 1, 2 and 3 are* $\$100, \$30, \$10$ *for Student* A_2 *and* $\$80, \$30, \$10$ *for Student* A_3*. There exist various reasons for privacy. For example, students may want to keep their cost for each location private, since it can be used to infer their initial location, and they would pay an additional (privacy) price rather than revealing the said travel cost. For example, Student* A_1 *associates* $80 *privacy cost to the revelation of the travel cost of* $70 *for meeting in London.*

DCOP The DCOP framework models this problem with:

- $A = \{A_1, A_2\ A_3\}$
- $V = \{x_1, x_2, x_3\}$
- $D = \{\{1, 2, 3\}, \{1, 2, 3\}, \{1, 2, 3\}\}$
- $C = \{$
  $\{(x_1 = 1), 70\}, \{(x_1 = 2), 230\}, \{(x_1 = 3), 270\},$
  $\{(x_2 = 1), 120\}, \{(x_2 = 2), 400\}, \{(x_2 = 3), 190\},$
  $\{(x_3 = 1), 40\}, \{(x_3 = 2), 280\}, \{(x_3 = 3), 230\},$
  $\{\neg(x_1 = x_2 = x_3), \infty\}\}$

where each constraint is described with the notation $\{p, c\}$ stating that if the predicate p holds then the cost c is payed, and the notation $(x = a)$ is a predicate stating that a variable x is assigned a value a.

One could attempt to model the privacy requirements by aggregating the solution quality, called $solutionCost$ and the $privacyCosts$ into a unique cost. However, this is not possible. Indeed, in a DCOP, agents explore the search space to find a better solution, and only pay the corresponding solution cost when the search is over and the solution is accepted. This means that the solution cost decreases with time. However, privacy costs are cumulative and are paid during the search process itself (each time a solution is proposed), no matter what solution is accepted at the end of the computation. This means that the total privacy loss increases with time. Aggregating the solution costs and privacy costs or using a multi-criteria DCOP would not consider the privacy cost of the solutions that are proposed but not kept as final. Also, a given solution may imply different privacy losses depending on the algorithm used to reach it.

Stochastic Algorithms

The main stochastic algorithms for solving DCOPs are the distributed breakout algorithm and the distributed stochastic algorithm. In these algorithms, a flawed solution violating some constraints is revised until all constraints are satisfied.

Distributed Breakout (DBO) (Yokoo and Hirayama 1996) is an iterative improvement algorithm, originally proposed for DCOPs with hard constraints (distributed constraint satisfaction problems). In DBO, a weight starting at 1 is defined for each pair of assignments that does not satisfy some constraints. The evaluation of a given solution is the summation of the weights of all constraints for the involved assignment. With hard constraints, the summation is equal

to the number of constraint violations. In the breakout algorithm, an assignment is changed to decrease the solution value. If the evaluation of the solution cannot be decreased by changing the value of any variable, the current state may be a local minimum. When trapped in a local minimum, the breakout algorithm increases the weights of constraint violation pairs in the current state by 1 so that the evaluation of the current state becomes higher than the neighboring states. Thus the algorithm can escape from a local minimum.

Distributed Stochastic Algorithms (DSA) (Zhang, Wang, and Wittenburg 2002) make agents start by randomly selecting an initial value. Agents then enter a loop, where they first send their new assigned values (if changed) to their neighbors and collect any new values assigned by those neighbors. Agents select the next candidate value based on the values received from other agents, and usually, based also on maximizing some utility function. There exists several variations of the DSA algorithm with slightly different properties.

Privacy

Privacy is an important problem in a lot of applications, intrinsic to the main motivation, in addition to the usual efficiency/optimality trade-offs. The cost of privacy lost in the process of reaching a solution needs to be considered. For example, in distributed scheduling problems, problems of confidentiality happen when information is exchanged between agents. Such coordinated decisions are in conflict with the need to keep constraints private (Faltings, Léauté, and Petcu 2008).

Sample Cryptographic Technique As an example of cryptographic technique, the approach described in (Yokoo, Suzuki, and Hirayama 2002), achieves a high level of privacy using encryption, giving more importance to privacy than to the efficiency of the resolution. It consists of using randomizable public key encryption scheme. In this algorithm, three servers receive encrypted information from agents and cooperate to find an encrypted solution. Relevant parts of the solution are then sent to each agent. While ensuring privacy (Hirt, Maurer, and Przydatek 2000), cryptographic techniques are usually slower, and sometimes require the use of external servers or computationally intensive secure function evaluation techniques that may not always be available or justifiable for their benefits. DCOP problems have also been addressed with fully cryptographic protocols (Silaghi, Faltings, and Petcu 2006).

DPOP with Secret Sharing Distributed Pseudo-tree Optimization Procedure (DPOP) (Petcu and Faltings 2005) consists in creating a Depth-First Search (DFS) tree, where agents sharing constraints are on the same branch, Propagating UTIL messages up the tree, starting with the leaves, and determining the optimal values for variables by the root agent. DPOP with Secret Sharing SSDPOP (Greenstadt, Grosz, and Smith 2007) modifies DPOP to protect leaves in the DFS tree. The tree can be viewed as a simple chain. SSDPOP uses secret sharing (Shamir 1979) to aggregate the results of a given solution, without revealing the corresponding individual valuations. The aggregated values for this solution are then passed to the agent at the bottom of the chain, who aggregates this information with his own valuations and sends the aggregate up the chain.

Privacy-Preserving Synchronous Branch and Bound Synchronous Branch and Bound (SyncBB) (Hirayama and Yokoo 1997) was one of the first distributed algorithms for solving DCOPs. Privacy-Preserving Synchronous Branch and Bound (P-SyncBB) (Grinshpoun and Tassa 2014) is a privacy-preserving version of SyncBB for solving DCOPs while respecting constraint privacy. P-SyncBB preserves the private constraint information by computing the costs of CPAs (current partial assignments) and comparing them to the current upper bound, using secure multi-party protocols.

3 Utilitarian Approach

In this section we present our utilitarian frameworks and algorithms for privacy in DCOPs. A similar approach was used to extend Distributed Satisfaction Problems in (Savaux et al. 2016).

Utilitarian DCOP We propose to ground the theory of DCOP in the well-principled theory of utility-based agentry. We introduce the Utilitarian Distributed Constraint Optimization Problem (UDCOP). Unlike previous DCOP frameworks, besides results, we are also interested in the search process, as it incurs privacy leak.

Definition 2. *A UDCOP is a tuple $\langle A, V, D, C, U\rangle$ where:*

- *$\langle A, V, D, C\rangle$ is a DCOP*
- *U is a vector of privacy costs for each agent, each one defining the set of costs in showing the loses of an agent for the revelation of the values in his variable.*

The state *of agent A_i includes the subset of D_i that it has revealed, as well as its cost. The problem is to search for an assignment of the variables such that the total utility is maximized, meaning that the cost is minimized.*

Example 2. *The DCOP in the Example 1 is extended to a UDCOP by specifying the additional parameter U:*

$U = \langle\{u_{1,1} = 80, u_{1,2} = 20, u_{1,3} = 40\},$
$\{u_{2,1} = 100, u_{2,2} = 30, u_{2,3} = 10\},$
$\{u_{3,1} = 80, u_{3,2} = 30, u_{3,3} = 10\}\rangle$

where $u_{i,j}$ is the privacy cost for A_1 to reveal the assignment $(x_i = j)$.

Now we discuss how the standard DBO and DSA algorithms are adjusted to UDCOPs. The state of an agent includes the `agentView`. After each state change, each agent computes the estimated utility of the state reached by each possible action, and selects randomly one of the actions leading to the state with the maximum expected utility. In our algorithms, agents use the risk of one of their assignments not being a part of the final solution, to estimate the utilities. Each risk of rejection is estimated by Equation 2:

$$agreementProb = \frac{1}{|D_i|} \tag{2}$$

Before proposing a new value, agents estimate the utility that will be reached in the next state. This value is the summation of the costs of revealed `agentViews` (weighted by their probability of being the final solution) in the state reached, and of the corresponding privacy costs. The agent proposes the next value only if this $estimatedCost$ is lower than the estimation of the current state.

The Distributed Breakout with Utility (DBOU) algorithm is obtained from DBO by adding 9 lines to the original algorithm (lines 2 to 10 in Algorithm1 displayed beside text). Line 2, the maximal improvement is initialized at 0. Line 3, the next value is initialized at the current value. Line 4, the possible next value is set to the value that gives the maximal improvement. Line 5, the set of revealed values is the union of the already revealed values and the new value Line 6, we estimate the cost reached after the next value is proposed. Line 7, the cost of the current state is estimated. Line 8, if the next cost is lower than the current cost, the maximal improvement and next value are updated.

Similarly the algorithm Distributed Stochastic Algorithm with Utility (DSAU) is obtained from DSA by adding the lines 6 to 10 in Algorithm 2. In both these extensions, other lines remain the same as in the initial algorithms.

Example 3. *Continuing with Example 2, at the beginning of the computation with the DSAU solver, the participants select a random value. The resulting* `agentView` *of each agent is* $x_1 = 1, x_2 = 1, x_3 = 3$*. The utilities of the reached state are:*
$c_{1,1} + u_{1,1} = 70 + 80 = 150$,
$c_{2,1} + u_{2,1} = 120 + 100 = 220$,
$c_{3,3} + u_{3,3} = 230 + 10 = 240$
for Students A_1*,* A_2*, and* A_3 *respectively. The participants then inform each others of their value. They then consider changing their value to a new randomly selected one. The considered* `agentView` *is* $x_1 = 2, x_2 = 3, x_3 = 1$*. If the participants change their value, the utilities of the reached states would be:*
$(c_{1,1} + c_{1,2})/2 + u_{1,1} + u_{1,2} = 250$,
$(c_{2,1} + c_{2,3})/2 + u_{2,1} + u_{2,3} = 265$,
$(c_{3,3} + c_{3,1})/2 + u_{3,3} + u_{3,1} = 225$,
for Students A_1*,* A_2*, and* A_3 *respectively. Student* A_1 *and Student* A_2 *do not propose the new value as it would increase their utility. However, Student* A_3 *chooses to change its value from* 2 *to* 1 *which lowers its utility from 240 to 225. In the next step, the* `agentView` *is* $x_1 = 1, x_2 = 1, x_3 = 1$*. Participants then do not change their value anymore, as all other options would not decrease the utility. At the final step, the previous* `agentView` *is therefore the optimal solution. With DSAU, the reached utilities are* $70 + 80 = 150, 120 + 100 = 220, 40 + 10 + 80 = 130$ *for Student* A_1*, Student* A_2*, and Student* A_3 *respectively. With standard DSA, the final utilities are:*
$(c_{1,1} + u_{1,1} + u_{1,2} + u_{1,3}) = 230$,
$(c_{2,1} + u_{2,1} + u_{2,2} + u_{2,3}) = 260$,
$(c_{3,1} + u_{3,2} + u_{3,1} + u_{3,3}) = 160$,
for Students A_1*,* A_2*, and* A_3 *respectively. Therefore, using DSAU instead of DSA reduces the utility by* $80, 40, 30$*.*

Algorithm 1: sendImprove_DBOU

Input: $\varnothing$
Output: $\varnothing$
1 eval ← evaluation value of currentValue;
2 myImprove ← 0 ; newValue ← currentValue;
3 possibleValue ← value giving maximal improvement;
4 possibleRevealedConstraints ← revealedConstraints ∪ constraints containing possibleValue ;
5 nextCost ← estimateCost($utilities$, $domain$, $nextRevealedValues$);
6 currentCost ← estimateCost($utilities$, $domain$, $revealedValues$);
7 **if** *(nextCost < currentCost)* **then**
8 myImprove ← possible max improvement;
9 newValue ← value giving maximal improvement;
10 **if** *(eval == 0)* **then** consistent ← true;
11 **else** consistent ← false; terminationCounter ← 0;
12 **if** *(myImprove > 0)* **then**
13 canMove ← true; quasiLocalMin ← false;
14 **else** canMove ← false; quasiLocalMin ← true;
15 send (improve, x_i, myImprove, eval, terminationCounter) to neighbors;

Algorithm 2: DSAU

Input: $\varnothing$
Output: $\varnothing$
1 Randomly choose a value;
2 **while** *(no termination condition is met)* **do**
3 **if** *(a new value is assigned)* **then**
4 send the value to neighbors;
5 collect neighbors' new values, if any;
6 temp ← randomly chosen value;
7 add constraints with temp to revealedConstraints;
8 costTemp ← estimateCost($utilities$, $domain$, $nextRevealedValues$);
9 cost ← estimateCost($utilities$, $domain$, $revealedValues$);
10 **if** *($costTemp < cost$)* **then** assign temp ;

Algorithm 3: estimateCost

Input: $agreementProb$, D, $probD$
Output: $cost$
1 **if** *(only one value is left in the domain)* **then**
2 **return** marginalCost(value) × $probD$;
3 **else**
4 $v \leftarrow D[0]$; $costRound \leftarrow$ estimateCost ($agreementProb$, $\{v\}$, $probD$);
5 costTemp ← estimateCost $(1 - agreementProb$, $D \setminus \{v\}$, $(1 - agreementProb) \times probD)$;
6 estimatedCost ← $costRound + costTemp$;
7 **return** estimatedCost;

4 Discussion

To further clarify why Multi-Objective DCOPs (MO-DCOPs) cannot integrate our concept of privacy as one of the criteria they aggregate, we give an example of what would be achieved with MO-DCOPs, as opposite to the results using the proposed UDCOPs. In the following example we show a comparative trace based on one of the potential techniques in MO-DCOPs, to show why MO-DCOPs cannot aggregate privacy lost during execution in the same way as UDCOP. In this example, the privacy value of each assignment and its constraint cost are two elements of an ordered pair defining the weight of the MO-DCOP. For illustration, in this example pairs of weights are compared lexicographically with the privacy having priority.

Table 1: Comparative trace of two rounds with UDCOP DSAU vs. MO-DCOP DSA with lexicographical comparison on vectors, privacy costs (privacyC) before solution costs (solutionC). Candidate values are marked with * if they are better than old values, and will be adopted.

Framework	UDCOP			MO-DCOP		
Agent	A_1	A_2	A_3	A_1	A_2	A_3
current state						
$value_0$	1	1	3	1	1	3
solutionC	70	120	230	70	120	230
privacyC	80	100	10	80	100	10
situation	150	220	240	[80,70]	[100,120]	[10,230]
believed next state						
considered	2	3	1	2	3	1
solutionC	150	155	135	230	190	40
privacyC	100	110	90	20	10	80
situation	250	265	225*	[20,230]*	[10,190]*	[80,40]
achieved next state						
$value_1$	1	1	1	2	3	3
solutionC	70	120	40	230	190	230
privacyC	80	100	90	100	110	10
situation	150	220	130	[100,230]	[110,190]	[10,230]

Example 4. *Suppose we now want to model the Example 2 with a MO-DCOP. As also illustrated in the trace in Table 1, at the beginning of the computation with the DSA solver, the participants select a random value. The resulting* `agentView` *is* $x_1 = 1, x_2 = 1, x_3 = 3$. *The participants then inform each others of their value. They then consider changing their value to a new randomly selected one. The considered* `agentView` *is* $x_1 = 2, x_2 = 3, x_3 = 1$. *Like with UDCOPs, Student* A_1 *does not propose the new value as it would increase their cost, and Student* A_3 *chooses to change its variable's value from* 2 *to* 1. *However, with MO-DCOPs Student* A_2 *changes its value to* 3, *which is not the case with UDCOPs, which implies privacy loss. The* `agentView` *is now* $x_1 = 1, x_2 = 3, x_3 = 1$. *As we see, with the MO-DCOP model, Student* A_2 *reveals more values and loses more privacy (a difference of (110-100)=10 units of privacy) than with UDCOPs.*

5 Experimental Results

We evaluate our framework and algorithms on randomly generated instances of *distributed meeting scheduling problems (DMS)*. Previous work (Wallace and Freuder 2005) in distributed constraint satisfaction problems has already addressed the question of privacy in distributed meeting scheduling by considering as private the information on whether an agent can attend a meeting. They evaluate the privacy loss brought by an action as the difference between the cardinalities of the final set and of the initial set of possible availabilities for a participant.

The problems are parametrized as follows: 10 agents, 10 possible values, the cost for the constraints is a random number between 0 and 9, and the cost of a revelation is a random number between 0 and 9. Each set of experiments is an average estimation over 50 instances with the different algorithms (DBO, DBOU, DSA, DSAU). The experiments are carried out on a computer under Windows 7, using a 1 core 2.16GHz CPU and 4 GByte of RAM. The implementation is done in Java, using the JADE framework.

Figure 1 shows the total amount of privacy lost by all agents, averaged over 50 problems, function of the density of unary constraints (proportion of unary constraints assignments with a non null cost). We observe that both of our extensions (DBOU and DSAU) reduce the loss of privacy compared to the initial algorithms (DBO and DSA, respectively). Also, DSA/DSAU imply more privacy loss than DBO/DBOU, likely due to the high number of message exchanged involved in DSA/DSAU.

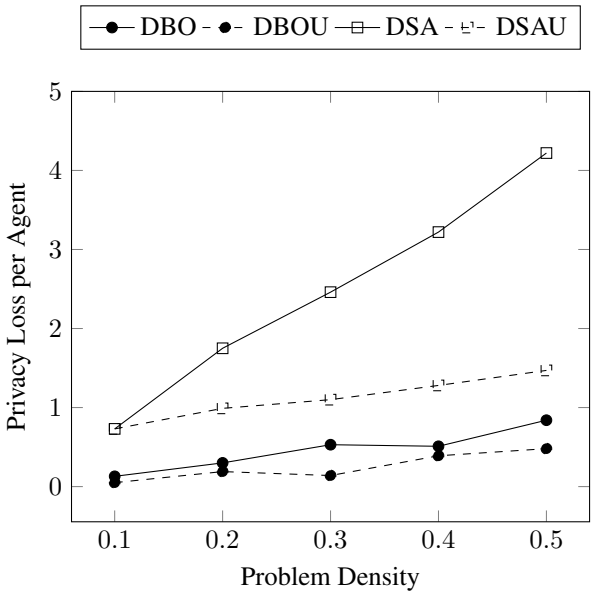

Figure 1: Evaluation of privacy loss in DCOPs

6 Conclusion

While privacy has been addressed in distributed constraint optimization problems, none of the existing techniques is widely used, likely due to the difficulty in modeling common problems. As privacy cannot be interpreted as a criteria of a standard DCOP, we propose in this article a framework called Utilitarian Distributed Constraint Optimization Problem (UDCOP). It models the privacy loss for the revelation of an agent's costs for violating constraints. We present algorithms that let agents use information about privacy to modify their behavior and guide their search process, by proposing values that reduce the amount of privacy loss. We then show how adapted stochastic algorithms (DBOU and DSAU) behave and compare them with standard techniques on distributed meeting scheduling problems. The experiments show that explicit modeling and reasoning with the utility of privacy allows for significant savings in privacy.

References

Bowring, E.; Tambe, M.; and Yokoo, M. 2005. Distributed multi-criteria coordination: Privacy vs. efficiency. In *Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-2005)-Workshop on Distributed Constraint Reasoning-DCR05, Edinburgh, Scotland*. Citeseer.

Dessimoz, J.-D. 2016. Cognition, cognitics, and team action—overview, foundations, and five theses for a better world. *Robotics and Autonomous Systems* 85:73–82.

Faltings, B.; Léauté, T.; and Petcu, A. 2008. Privacy guarantees through distributed constraint satisfaction. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 2, 350–358. IEEE.

Gershman, A.; Grubshtein, A.; Meisels, A.; Rokach, L.; and Zivan, R. 2008. Scheduling meetings by agents. In *Proc. 7th Intern. Conf. on Pract. & Theo. Automated Timetabling (PATAT 2008), Montreal (August 2008)*.

Greenstadt, R.; Grosz, B.; and Smith, M. D. 2007. Ssdpop: improving the privacy of dcop with secret sharing. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 171. ACM.

Grinshpoun, T., and Tassa, T. 2014. A privacy-preserving algorithm for distributed constraint optimization. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 909–916. International Foundation for Autonomous Agents and Multiagent Systems.

Hirayama, K., and Yokoo, M. 1997. Distributed partial constraint satisfaction problem. In *International Conference on Principles and Practice of Constraint Programming*, 222–236. Springer.

Hirt, M.; Maurer, U.; and Przydatek, B. 2000. Efficient secure multi-party computation. In *International Conference on the Theory and Application of Cryptology and Information Security*, 143–161. Springer.

Maheswaran, R. T.; Tambe, M.; Bowring, E.; Pearce, J. P.; and Varakantham, P. 2004. Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 310–317. IEEE Computer Society.

Petcu, A., and Faltings, B. 2005. A scalable method for multiagent constraint optimization. Technical report.

Russell, S. J.; Norvig, P.; Canny, J. F.; Malik, J. M.; and Edwards, D. D. 2003. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River.

Savaux, J.; Vion, J.; Piechowiak, S.; Mandiau, R.; Matsui, T.; Hirayama, K.; Yokoo, M.; Elmane, S.; and Silaghi, M. 2016. Discsps with privacy recast as planning problems for self-interested agents. 359–366.

Shamir, A. 1979. How to share a secret. *Communications of the ACM* 22(11):612–613.

Silaghi, M.-C.; Faltings, B.; and Petcu, A. 2006. Secure multiparty constraint optimization simulating dfs tree-based variable elimination.

Wallace, R. J., and Freuder, E. C. 2005. Constraint-based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving. *Artificial Intelligence* 161(1):209–227.

Yokoo, M., and Hirayama, K. 1996. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 401–408.

Yokoo, M.; Suzuki, K.; and Hirayama, K. 2002. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *International Conference on Principles and Practice of Constraint Programming*, 387–401. Springer.

Zhang, W.; Wang, G.; and Wittenburg, L. 2002. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *Proceedings of AAAI Workshop on Probabilistic Approaches in Search*.