Distributed Private Constraint Optimization Problem: Cost of Privacy Loss

Marius C. Silaghi[1], Prashant Doshi[2], Toshihiro Matsui[3], Makoto Yokoo[4]

[1]Florida Tech, [2]University of Georgia, [3]Nagoya Institute of Technology, [4]Kyushu University

Abstract. We propose to merge the two – privacy and cost/utility – usual optimization criteria of Distributed Constraint Optimization Problems (DCOPs) into a unique criterion. Typically, a DCOP requests agents to agree on a tuple of assignments of values to variables such that the sum of costs defined by a set of secret weighted constraints is minimized. However, the privacy requirements on constraints is classically used to define an orthogonal optimization criteria (minimizing the number of disclosed tuples, or maximizing the entropy of the knowledge about constraints). Common complete DCOP search techniques look for a solution minimizing the cost and maintaining *some* privacy.
We start from the observation that privacy leaks are a cost. A DCOP whose secrets are labeled with the costs of the corresponding privacy leaks defines a new framework, that we will call Distributed Private Constraint Optimization (DPCOP). We propose to define the cost of an agreed tuple of assignments as the sum between the weights of the constraints for the chosen tuple, and the cost induced by secrets leaked before agreeing on the tuple. Therefore, the cost of a solution depends on the algorithm used to find it. Different *complete* algorithms will return solutions with different (privacy-loss&weight) costs, and these costs provide a metric to compare algorithms. The level of detail used for specifying costs for privacy leaks can lead to different winners among algorithms. A set of benchmarks is proposed and made available.

1 Introduction

The distributed constraint reasoning (DCR) framework addresses problems where a set of agents participate in distributed problem solving. A common assumption (used in this work) is that the agents are self-interested[1]. Another assumption is that the agents agree to cooperate for finding the values of some parameters, X, which optimize an objective function defined as a sum of a set of constraints on X. The motivation of this agreement is sometimes assumed to be enforced by mechanisms outside the DCR, while sometimes the reward (utility) of finding a solution is specified as an input of the problem. The constraints themselves are real functions. Some of the constraints may be public, and some are secrets of different participating agents. A simple example of such problem is meeting scheduling with secret constraints, where the parameters are the meeting place and time, and each agent's utility to succeed in meeting the others is specified by that agent. When the constraints are Boolean and hard (results in $\{0, \infty\}$),

[1] The main idea in this work (explicitly specifying the utility of keeping secrets) can also be used with cooperative agents, defending their privacy from outsiders.

their secrecy can be defined with the DisPrivCSPs framework, proposed in [15]. The more general case where constraints can be any real function and specify utilities is called distributed constraint optimization, (DCOP). DCOPs raise additional problems, and several approaches to addressing privacy were described recently. Here we present a solution based on the explicit evaluation of the utility of keeping the constraints secret. We propose to assume that this evaluation of utility of secrecy can be performed prior to the definition of the problem, based on concrete situations, and are an input of the corresponding agents.

The distributed private constraint satisfaction (DisPrivCSPs) framework [15] models problems with privacy requirements and enables qualitatively and quantitatively comparison of distributed CSP solvers. We introduce a similarly powerful framework for distributed constraint optimization (DCOPs), as an extension to DisPrivCSPs. Significant attention was previously given to the definition and analysis of privacy requirements in distributed constraint satisfaction problems (DisCSP) [19]. For a given agent A_i, all the solutions of a DisCSP have the same value (utility), U_i. At the basis of the DisPrivCSP theory is the observation that a rational agent A_i will drop out of search when it expects that the price of its future privacy loss is higher than U_i (intuitively leading to a negative total utility). Past privacy loss does not matter since its value was already lost. The quality of a solution to be searched for by a DisCSP algorithm is therefore defined only by the privacy criteria.

Distributed constraint optimization problems (DCOPs) are an extension of DisCSP where constraints have different costs. Some attention was already given to privacy in distributed constraint optimization [7, 5]. In prior work, DCOPs with privacy requirements are treated as a multi-criteria optimization where the constraint weights are of a different nature from privacy (often perceived from an information theoretic perspective). The changes needed for generalizing the DisPrivCSP framework have not yet been analyzed. We start by noting that its idea does not immediately apply to DCOPs, since in DCOPs the value of the optimal solution is not known in advance (otherwise the problem would become a constraint satisfaction problem). While DisPrivCSPs are optimization problems for the privacy criterion, DCOPs were so far seen as multi-criteria optimization problems along two incomparable metrics (privacy/entropy and cost/utility).

We propose a new framework called Distributed Private Constraint Optimization (DPCOP) where the two – previously incomparable – metrics of DCOPs are redefined and merged under the utility theory, yielding a unique and easy to analyze optimization criteria. We explicitly model the loss of privacy as a cost and we assume that this cost is provided as a part of the input specification. When the costs of revealing each secret can be obtained like this, they allow for much better targeted strategies, keeping the most valuable secrets and revealing less valuable secrets (rather than just maximizing the entropy by guarding many irrelevant secrets). A set of benchmarks are proposed for the new framework (see [2]), and baseline algorithms are proposed and analyzed experimentally.

The DPCOP framework also defines a new hybrid between the DCOP and the multi-agent planning research areas, since a DPCOP solver becomes a planner (where the actions taken during search have an impact on the solution).

2 Related Work

Privacy has been a fundamental motivation for distributed constraint optimization, since the early beginning of the field [19]. However, due to wide disagreement on how to formalize privacy requirements, proposed techniques are often evaluated not from the privacy perspective but solely from the perspective of efficiency and cost. The first quantitative measurement of privacy loss was based on simply counting the number of disclosed tuple values [3]. The distributed private constraint satisfaction problems introduced in [15] label each secret with a number corresponding to the cost induced by its privacy loss. The privacy loss incurred during a computation is given by the sum of the the privacy values of each leaked secret. Other approaches to quantifying privacy loss are based on information theory, maximizing entropy. The privacy loss is therefore expressed in bits of information, or some related units [7]. All these approaches lack a clear way to trade off privacy for solution quality, resulting in a difficult multi-criteria optimization.

2.1 Distributed Private CSPs

With DisPrivCSPs [15], each secret (cost/weight of a constraint or combination thereof) is associated with a *privacy value*. The privacy value of a secret specifies the incremental loss of utility due to the revelation of that secret. Note that DisPrivCSPs could only model additive privacy loss, restriction also removed in the definition proposed here. The *reward for solving the problem* is given as a constant. The weights (satisfying/unsatisfying) of a constraint have no direct relation to the utility. Agents in DisPrivCSPs abandon the search when the utility loss due to predictable privacy loss (next incremental privacy loss) is higher than the reward for finding a solution. A qualitative comparison of algorithms was possible based on the ability to solve problems without abandoning the search.

2.2 Baseline Algorithms

The simplest distributed algorithm for solving distributed CSPs is the one proposed in [3]. In this algorithm, an agent proposes a value for the variables (a solution) at a time, and the other agents answer with messages specifying whether their constraints are satisfied by that assignment.

There are other algorithms for solving DCOPs such as ADOPT [9], DPOP [11], and DisAO [8]. There also exist DCOP optimization techniques using cryptographic protocols [16, 5], and which offer significantly high levels of privacy guarantees.

3 DPCOP Framework

While prior work treated distributed constraint optimization problems (DCOPs) with privacy requirements as a multi-criteria optimization, where the constraint weights are measured in utility and privacy is measured in information bits or related metrics, here we propose to measure privacy in the same type of utility as the constraint weights.

In order to extend the DisPrivCSP framework to DCOPs, we start from the observation that the DCOP constraint weights, normally used in the objective function of the optimization, can also be considered to be a positive (or negative) utility – or cost – of the same nature as the cost induced by privacy loss. As such, in DCOPs *minimizing* the sum of the constraint weights (i.e., where weights represent a cost, with negative utility), the total cost is given by the sum between the value of the total lost privacy and the cost of the selected solution. The reward for each agent of solving the problem will still be considered in this setting to be a previously known (possible infinite) value, like with DisPrivCSPs. A rational participating agent is expected to *abandon* the search if its next revelation would lead to a value for the *incremental privacy loss* which, together with a lowest bound on the cost of the solution, becomes larger than the reward for solving the problem.

For *maximization* DCOP problems, namely problems seeking a solution maximizing the sum of the constraint weights (i.e., where constraint weights represent rewards), privacy loss becomes the only cost. The utility is defined by the difference between the reward of the solution and the value of the privacy lost during the search. A rational agent will therefore abandon the search problem if its next (or expected) disclosures leads to an *incremental privacy loss* that is larger than the expected total reward of the solution.

In order to formally define the framework described so far, we first formalize the concept of privacy leaks in a way general enough to model non-additive functions. Given a set of secrets, a leaked information about some of these secrets will be called *revelation*.

Definition 1 (Revelation). *Given a set of secrets S and a set of agents A, the set of possible revelations $R(S, A)$ is a function $R(S, A) : A \rightarrow (S \rightarrow [0, 1])$ which maps each peer agent to a functional relation specifying the probability learned by that agent for each secret.*

Note that this definition of revelation is more general than the version used by DisPrivCSPs, as here it can model statistical privacy losses. While the above definition has a rich modeling power, one can assume that sometimes users may find it difficult to provide the data related to all possible revelations defined in this way. We therefore also consider a simplified version that requires less data (but is somewhat less general):

Definition 2 (simplified revelation). *Given a set of secrets S and a set of agents A, the set of possible revelations $R(S, A)$ is the function, $R(S, A) : A \rightarrow \mathcal{PS}(S)$, which maps each peer agent to the element of the power-set of the set of secrets, $\mathcal{PS}(S)$, that he learns.*

The simplified revelation definition assumes that privacy is lost only when a secret is completely revealed. It does not account for secrets about which other probabilistic information is made available. Now we can formally define the DPCOPs.

Definition 3 (DPCOP). *A (minimization) Distributed Private Constraint Optimization Problem (DPCOP) is defined by a tuple (A, X, D, C, P, U). A is a set of agents $\{A_1, ..., A_K\}$. X is a set of variables $\{x_1, ..., x_n\}$, and D is a set of domains*

$\{D_1, ..., D_n\}$ such that each variable x_i may take values only from the domain D_i. The variables are subject to a set C of sets of weighted constraints $\{C_0, C_1, ..., C_K\}$, where $C_i = \{\phi_i^1, ..., \phi_i^{c_i}\}$ holds the secret weighted constraints of agent A_i, and C_0 holds the public constraints. Each weighted constraint is defined as a function $\phi_i : X_i \rightarrow \mathbb{R}_+$ where $X_i \subseteq X$. The value of such a function in an input point is called constraint entry*, and each C_i can be seen as a set of such constraint entries.*

P is a set of privacy loss cost functions $\{P_1, ..., P_K\}$, one for each agent. P_i defines the cost inflicted to A_i by each revelation r of its secrets, i.e., $P_i(r) : R(C_i, A) \rightarrow \mathbb{R}_+$.

A solution is an agreement between agents in A on a tuple τ^* of assignments of values to variables that minimizes the total cost:

$$\tau^* = argmin_\tau \sum_i (\sum_j \phi_i^j(\tau)) + P_i(\Pi_i(\tau))$$

where $\Pi_i(\tau)$ is the revelation in $R(C_i, A)$ performed during the process leading to the agreement on the assignments τ.

U is a set of rewards $U_1, ..., U_K$, one for each agents, that the corresponding agent receives if a solution is found, and that agents use for deciding whether to abandon a search given their foreseen incremental privacy loss.

The set of rewards U can be used to qualitatively compare DCOP solvers, as to which solver can solve more problems than another solver without any agent abandoning the process. Such a hierarchy of solvers was built for DisPrivCSPs in [15]. Formally, the agent A_i abandons the search if:

$$P_i(r*) - P_i(r) + W \geq U_i$$

where r is the revelation performed by A_i up to this moment, $r*$ is the revelation after the next planned sequence of actions, and W is a low bound on the quality (cost) of the expected solution.

The *privacy-loss cost functions* P_i are a new concept. These functions are part of a problem model (just like utilities of auction outcomes, used to infer bids in Vickrey auctions). Just as utilities are an agent's input for auctions, a privacy-loss cost function is an agent's input for DPCOPs. An agent can infer a privacy-loss cost function by simulating how much utility it may lose when each revelation is performed.

The above DPCOP definition is for the general case where the constraint of an agent may involve all variables. Many approaches consider a simplified version (equivalent in expressive power) where each agent *owns* some variables, and agents enforce only constraints with variables assigned by previous agents. We provide next the corresponding DPCOP simplification, allowing for most existing DCOP algorithms.

Definition 4 (simplified DPCOP). *A (minimization) distributed private constraint optimization problem is defined by a tuple (A, X, D, C, P, U). A is a set of agents $\{A_1, ..., A_n\}$. X is a set of variables $\{x_1, ..., x_n\}$, and D is a set of domains $\{D_1, ..., D_n\}$ such that each variable x_i may take values only from the domain D_i. The variables are subject to a set C of weighted constraints sets $\{C_0, C_1, ..., C_n\}$, where $C_i = \{\phi_i^1, ..., \phi_i^{c_i}\}$ holds the secret weighted constraints of agent A_i, and C_0 holds*

public constraints. Each weighted constraint is defined as a function $\phi_i^j : X_i \rightarrow \mathbb{R}_+$ *where* $X_i \subseteq \{x_1, ..., x_i\}$.

P *is a set of privacy loss cost functions* $\{P_1, ..., P_n\}$*, one for each agent.* P_i *defines the cost inflicted by the revelation of any subset of secret elements of* $P_i : R(C_i, A) \rightarrow \mathbb{R}_+$.

A solution is an agreement between the agents in A *on a tuple* τ^* *of assignments of values to variables that minimizes the total cost:*

$$\tau^* = argmin_\tau \sum_i (\sum_j \phi_i^j(\tau)) + P_i(\Pi_i(\tau))$$

where $\Pi_i(\tau)$ *is the revelation in* $R(C_i, A)$ *performed during the process of agreeing on the assignments* τ.

U *is a set of rewards* $U_1, ..., U_K$*, one for each agents, that the corresponding agent receives if a solution is found.*

Maximization Maximization DPCOPs are defined similarly, but without the element U, and redefining the solution as:

$$\tau^* = argmax_\tau \sum_i (\sum_j \phi_i^j(\tau)) - P_i(\Pi_i(\tau)).$$

An agent abandons the maximization search if:

$$W - (P_i(r*) - P_i(r)) \leq 0$$

where r is the revelation performed by A_i up to this moment, $r*$ is the revelation after the next planned sequence of actions, and W is an upper bound on the quality of the expected solution.

Simplified cost functions While (in general) privacy-loss cost functions are not additive, we expect that additive randomly generated benchmarks have the simplicity that can help in the theoretical understanding of the new framework. In a simplified version, the value of privacy leaks towards a peer agent can also be considered independent of privacy leaks towards other peers (assumption not applicable to all problems). For *additive* privacy cost functions, an array of privacy costs can simply be attached to each constraint tuple.

An important case of *non-additive* privacy-loss cost function is where the cost of a leak is independent of the agent (revelation to an agent being considered to be a revelation to all agents), while being additive along the dimension of the secrets. Such a privacy cost function can be represented by a single cost associated with each constraint tuple.

4 Comparison with previous frameworks

The closest previous framework is the Distributed Private CSPs (DisPrivCSPs) that we introduced in [15], which deals with distributed constraint satisfaction problems

(DisCSPs). DisPrivCSPs also have costs for privacy loss, but that cost is not integrated in any way with the cost of the agreement tuples.

DisCSPs can be modeled as a special case of DCOPs, namely when the constraints are functions with results only in $\{0, \infty\}$, rather than in $\mathbb{R}_+$. This is because:

$$\sum_i (\sum_j \phi_i^j(\tau))$$

has the same value for all the satisfying tuples of the DisPrivCSP.

Previous research related to privacy in DCOPs has already found inspiration in DisPrivCSPs [7], and can be seen as straightforward applications of DisPrivCSPs to DCOPs. DPCOPs are a less straightforward extension of DisPrivCSPs. We think that the main innovation in DPCOPs versus a straightforward DisPrivCSPs usage with DCOPs is:

- DPCOPs unify the metric for cost of privacy loss with the metric used for specifying weights of constraints (in DisPrivCSPs they were incomparable metrics).
- The revelation is more general in DPCOPs, allowing for statistical and non-additive privacy loss functions.

Among smaller differences, while with DisPrivCSPs an agent A_i will abandon the search when incremental costs are higher than U_i, with maximization DPCOPs there may be no known finite limit on the reward of the agent. Also, for DisPrivCSPs we provided only theoretical and qualitative comparison of techniques, while with DPCOPs we provide benchmarks, random problem generators, and experimental analysis of techniques.

5 Baseline DPCOP Solvers

Any of the existing DCOP techniques can be used to solve DPCOPs. Techniques using cryptographic methods, such as the ones in [16, 5], can guarantee optimality with minimal privacy leak. Other techniques may offer more efficiency at the expense of optimality. We evaluate simple algorithms for solving DPCOPs. Probably the simplest technique consists of an agent consecutively asking each publicly possible tuple one after another, while the other agents answer with their costs. This is an adaptation to optimization of the technique proposed in [3]. The agent asking the questions in this **1-leader** version is called *the leader*. In the **N-leaders** variant, the search space is distributed between agents (related to [6]), and each agent asks costs for his part. The baseline version we evaluate in the N-leaders version is even simpler, with agents acting in turn rather than simultaneously, each question also delegates the leader for the next question. At the end, the agents publish the best tuples for their sub-parts, and the best overall tuple is selected.

Leaders may propose tuples that are suboptimal (with worse local cost than their currently best tuple), lying to increase privacy (lying occurs also in [1]).

```
procedure leader do
    foreach next tuple τ with better local weight than currently best tuple do
        decide next_leader // only N-leaders version;
        send ask(τ,next_leader);
        set next leader // only N-leaders version;
        wait answers;
        update identity of best tuple;
    end
end do.
procedure slaves do
    when ask (τ, next_leader) do
        compute local cost for τ;
        send answer(τ, cost) to leader;
        recompute privacy_loss;
        leader := next_leader // only N-leaders version;
        if (leader = myself) then
            change to leader mode // N-leaders version;
        end
    end do.
end do.
```

Algorithm 1: Baseline (1-leader and N-leaders versions)

6 Non-cooperative multi-agent problem solving

The DPCOP framework typically models semi-cooperative paradigms. Some other agent paradigms that are also useful are:

- Cooperative in which agents are willing to readily exchange any data that can improve their global performance (e.g., robots or humans in a team exploring Mars [13]).
- Non-cooperative and self-interested in which the agents are enemies and do not coordinate with each other and do not agree on any protocol (e.g., reciprocally spying robots or humans in a war [10]).
- Semi-cooperative and self-interested in which the agents in their own interest agree to follow social protocols in order to coexist (bidders in an auction [18]).

The two self-interested situations are extreme and often, in practice, the self-interested situations are in-between. For example, enemies in a cold war are interested in destroying each other, but (since a blunt nuclear war would destroy all parties) agree to coordinate (unreliably) to minimize the probability of being destroyed. This is representative of the situation of communication between the two actors in the well known Prisoner's Dilema game, each trying to convince the other to stay silent, while trying to catch an opportunity to betray.

The DPCOP framework assumes that the agents are involved in some important amount of coordination, and desire to maximize the sum of independently declared utility functions (with trustworthiness guaranteed by other externel mechanisms, such as Clarke tax).

Privacy as utility in non-cooperative self-interested paradigms However, our idea of modeling privacy as utility also applies to non-cooperative self-interested situations. While DPCOP solving is a kind of multi-agent planning (MAP), non-cooperative situations are more commonly addressed in general approaches to MAP, such as partial order planning or POMDPs [12, 4]. While current MAP approaches enforce privacy using entropy maximization [10], our idea of modeling privacy loss with utility (cost) can be utilized to get simpler and more intuitive frameworks. The effort then shifts toward specifying the privacy loss functions.

7 Evaluation

It is easy to learn a secret weight of a constraint entry for an agent when a message sent by this agent is based solely on the weight of that secret constraint entry. If an agent controls a single secret constraint, each message that the agent sends in response to a leader's challenge reveals a secret weight. If an agent holds several secret constraints, a message is an aggregation of secrets from those constraints, and learning the component secrets is sometimes possible, but more computationally involved (solving the corresponding systems of equations, when they are determined). First we perform an experimental study for the simpler case where *each agent enforces a single private constraint*.

Random problem generator An important component of constraint-based frameworks consists in the development of random problem generators. Our DPCOP generator (made available on the DPCOP site [2]) is based on a typical CSP generator parametrized by density and number of values per variable. The weights we generate for constraint tuples are finite, generated according to a distribution D_w, defined by the uniform distribution $U(0, B_w)$ between 0 and an upper bound B_w. Each constraint is tagged with the ID of an owner agent (i for agent A_i). For additive privacy functions, each secret constraint tuple is tagged with a vector of privacy loss costs, one for each non-owner agent. The privacy loss cost is drawn from another distribution, D_p, also chosen as a uniform distribution $U(0, B_p)$ between 0 and a bound B_p. For the version with costs of privacy leaks independent of the target agent, the array of costs per constraint entry has length 1. The rewards generated for reaching a solution with minimization DPCOPs are infinite.

DPCOP files Each randomly generated DPCOP, as well as any non-random benchmark we generate is stored in a file in the following format:

```
<nb variables>
<var_name1> <dom_size> <val_1> ... <val_d>
<var_name2> <dom_size> <val_1> ... <val_d>
...
<nb constraints>
<arity1>
<owner>
```

```
<size privacy-vector/tuple>
<var1_name>
...
<weight1> [<privacy vector>] ...
```

An example file (with additive privacy costs) is:

```
2 # nb agents
2 # nb variables
 x0 3 0 1 2
 x1 3 0 1 2

2 # nb constraints

1 # arity first constraint
0 # owner agent (-1 means public)
2 # length privacy-leaks vector
x0 # variable_name
3 [ 0 4  ] 0 [ 0 1  ] 3 [ 0 1  ]

2 # arity second constraint
1 # owner agent (-1 means public)
2 # length privacy-leaks vector
x1 # var1
x0 # var2
3 [ 3 0  ] 4 [ 0 0  ] 1 [ 3 0  ]
3 [ 4 0  ] 2 [ 1 0  ] 3 [ 1 0  ]
1 [ 3 0  ] 1 [ 3 0  ] 1 [ 3 0  ]
```

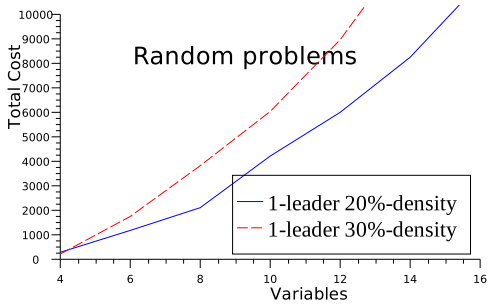

Fig. 1. Total cost (privacy+weight) with 1 leader.

Experimental Results Result quality, averaged over random 25 minimization DPCOPs at each size with the baseline 1-leader algorithm are reported in Figure 1, as total cost

Algo	DPCOP	Size	Pref	Cost	Cycles	Time
HP	STM	4	2	3	14	1.47
BL	STM	4	2	257	9.6	0.85
BL	RPSM	4	2	249	8.8	0.56
BL	RPSX	4	2	310	27.6	0.82

Table 1. Benchmarks.

(to be subtracted from the result reward). Results for the N-leaders variant were also obtained on the same problems and found to lead to 10% higher cost for small problems, 4-6 variables, while being similar at larger problems (which is explained by the random nature of the problems).

We also show evaluations based on benchmark problems [2]. The results are given in Table 1. RPSX and STM based on soft constraints, and RPSM based on hard constraints, Results are given for the baseline algorithm with one leader (BL), and for cryptographic algorithm (HP) in [14]. The cryptographic algorithm leaks only the secrets implied by the fact that the solution satisfies the public constraints

Some cryptographic solvers are guaranteed to find optimal solutions for DPCOPs, at the expense of efficiency [17, 16]. Assuming that no two agents exchange information about peers, there exist partially cryptographic solvers that are quite efficient but may rarely leak information due to solution vulnerabilities [5].

8 Conclusion

This is the first approach where privacy loss and weight of constraints in DCOPs are measured with the same unit (utility) and integrated into a unique optimization criteria.

We define the framework of Distributed Private Constraint Optimization to model problems with complex privacy requirements. We also provide a generator for random DPCOPs with additive or agent-independent privacy functions.

Baseline algorithms are evaluated for problems in the new framework. All existing DCOP solvers apply to DPCOPs. Some cryptographic solvers provide the optimal solution, at the expense of efficiency.

References

1. I. Brito and P. Meseguer. Distributed forward checking may lie for privacy. In *CP DCR Workshop*, 2007.
2. DPCOP. Distributed private optimization problems. `http://www.cs.fit.edu/~msilaghi/DPCOP`, 2008.
3. E.C. Freuder, M. Minca, and R.J. Wallace. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. In *Proc. IJCAI DCR*, pages 63–72, 2001.
4. Piotr Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research (JAIR)*, 24:49–79, 2005.
5. Rachel Greenstadt, Barbara Grosz, and Michael D. Smith. SSDPOP: Improving the privacy of PDCOP with secret sharing. 2007.

6. Youssef Hamadi. Interleaved backtracking in distributed constraint networks. In *ICTAI*, pages 33–41, 2001.
7. Rajiv T. Maheswaran, Jonathan P. Pearce, Emma Bowring, Pradeep Varakantham, and Milind Tambe. Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications. *Journal of Autonomous Agents and Multiagent Systems (JAAMAS)*, 2006.
8. Roger Mailler and Victor Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS*, pages 438–445, 2004.
9. Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *AIJ*, 161, 2005.
10. Praveen Paruchuri, Milind Tambe, Fernando Ordó nez, and Sarit Kraus. Security in multiagent systems by policy randomization. In *AAMAS*, 2006.
11. Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, 2005.
12. S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 2nd edition, 2002.
13. M. Sierhuis, J. M. Bradshaw, A. Acquisti, R. v. Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice. In *The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2003.
14. M.-C. Silaghi. Hiding absence of solution for a discsp. In *FLAIRS'05*, 2005.
15. M.-C. Silaghi and B. Faltings. A comparison of DisCSP algorithms with respect to privacy. In *AAMAS-DCR*, 2002.
16. M.-C. Silaghi, B. Faltings, and A. Petcu. Secure combinatorial optimization using DFS-based variable elimination. In *Symposium on AI and Maths*, January 2006.
17. M.-C. Silaghi and D. Mitra. Distributed constraint satisfaction and optimization with privacy enforcement. In *3rd IC on Intelligent Agent Technology*, pages 531–535, 2004.
18. W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
19. M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE TKDE*, 10(5):673–685, 1998.