2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Distributed Private Constraint Optimization

Prashant Doshi[1], Toshihiro Matsui[2], Marius Silaghi[3]*, Makoto Yokoo[4], and Markus Zanker[5]
[1]University Georgia, [2]NITECH, [3]Florida Tech, [4]Kyushu University, [5]University Klagenfurt

Abstract

We merge two popular optimization criteria of Distributed Constraint Optimization Problems (DCOPs) – reward-based utility and privacy – into a single criterion. Privacy requirements on constraints has classically motivated an optimization criterion of minimizing the number of disclosed tuples, or maximizing the entropy about constraints. Common complete DCOP search techniques seek solutions minimizing the cost and maintaining some *privacy. We start from the observation that for some problems we could provide as input a quantification of loss of privacy in terms of cost. We provide a formal way to integrate this new input parameter into the DCOP framework, discuss its implications and advantages.*

1 Introduction

The distributed constraint reasoning (DCR) framework addresses problems where a set of agents participate in distributed problem solving. The agents cooperate for finding the values of some parameters, X, which optimize an objective function defined as a sum of a set of weighted constraints on X. The agreement of cooperation among agents is sometimes assumed to be enforced by mechanisms outside the DCR, while other times the reward (utility) of finding a solution is specified as an input to the problem. The weighted constraints are real functions. Some of the constraints may be publicly known, while some are secrets of different participating agents.

We introduce a new framework called Distributed Private Constraint Optimization (DPCOP) in which two metrics of DCOPs, previously modeled as incomparable, are redefined and merged under the umbrella of utility theory, thereby yielding a unique optimization criteria, which facilitates analysis. 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. A quantification of privacy loss as cost allows for better targeted strategies, which may flexibly preserve valuable secrets and reveal less valuable ones. This contrasts with the previous approach of maximizing the entropy by guarding, possibly irrelevant, secrets.

*Contact Author: msilaghi@fit.edu

2 DPCOP Framework

In order to extend the DisPrivCSP framework to DCOPs, we start from the observation that the DCOP constraint weights, normally utilized in the objective function of the optimization, may also be considered as *a utility* – positive or negative reward (cost) – analogous to the cost induced by privacy loss. For 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 of the value of the lost privacy and the cost of the selected solution. The reward to each agent for solving the problem in this setting will remain as the previously known, possibly infinite, value analogous to DisPrivCSPs. A rational participating agent *abandons* the search if its next revelation would lead to a value of the *incremental privacy loss* which, when combined with the lowest bound on the cost of the solution, is larger than the reward for solving the problem.

For *maximization* DCOP problems that seek a solution maximizing the sum of the constraint weights (where constraint weights represent rewards), privacy loss becomes the only cost. The utility is then the difference between the reward of the solution and the value of the privacy lost during the search. A rational agent abandons the search if an *incremental privacy loss* is larger than the expected total reward of the solution.

Given a set of secrets, a leaked information about them is called a *revelation*.

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

For example, given three agents: *Alice ($\mathcal{A}$), Bob ($\mathcal{B}$), and Carol ($\mathcal{C}$)*, and the set of secrets for Carol: *Carol_rich ($\mathcal{S}_1$), Carol_corrupt ($\mathcal{S}_2$)*, a revelation of Carol is: $\{R(\mathcal{A},\mathcal{S}_1)=0.9, R(\mathcal{A},\mathcal{S}_2)=0.3, R(\mathcal{B},\mathcal{S}_1)=0.3, R(\mathcal{B},\mathcal{S}_2)=0.9\}$.

978-0-7695-3496-1/08 $25.00 © 2008 IEEE
DOI 10.1109/WIIAT.2008.426

Note that this definition of revelation is more general than the version used by DisPrivCSPs, as it allows us to model statistical privacy losses [15, 7]. An equivalent definition adapted to multi-valued secrets [7] is:

$$R(A,S) : (A \times S) \rightarrow (D \rightarrow [0,1]),$$

which maps each peer agent and secret to a probability distribution learned by that agent for each value (from D) of that secret. An example with $\mathcal{S}_1$ in {not_rich (v_1), millionaire (v_2), billionaire ($v3$)} and $\mathcal{S}_2$ Boolean is {$R(\mathcal{A},\mathcal{S}_1)$=[P($v_1$)=0.2, P($v_2$)=0.7, P($v_3$)=0.1], $R(\mathcal{A},\mathcal{S}_2)$=[P(t)=0.2, P(f)=0.8], $R(\mathcal{B},\mathcal{S}_1)$=[P($v_1$)=0.4, P($v_2$)=0.2, P($v_3$)=0.4], $R(\mathcal{B},\mathcal{S}_2)$=[P(t)=0.4, P(f)=0.6]}

A simpler version used in our benchmarks is:

$$R(A,S) : A \rightarrow \mathcal{PS}(S),$$

which maps each peer agent to the element of the power-set of the set of secrets, $\mathcal{PS}(S)$, that he learns, thereby accounting only for completely revealed secrets. An example is: {[$\mathcal{A}\rightarrow \{\mathcal{S}_1\}$];[$\mathcal{B}\rightarrow \{\mathcal{S}_1, \mathcal{S}_2\}$]}.

Definition 2 (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 $\mathcal{C}_i$ 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., $P_i(r) : R(A, \mathcal{C}_i) \rightarrow \mathbb{R}_+$. U is a set $U_1, ..., U_K$. U_i is the reward received by A_i if a solution is found (used for deciding to abandon the search).*

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

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

where $\Pi_i(\varepsilon)$ is the revelation $R(\mathcal{C}_i, A)$ during the process leading to the agreement on the assignments ε.

The set of rewards U can be used to qualitatively compare DCOP solvers based on which solves more problems without any agent abandoning the process [13]. Formally, the agent, A_i, abandons the search if:

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

where r is the revelation by A_i up to this moment, r^* is the revelation after the next planned sequence of actions, and W is a lower bound on the cost of the expected solution.

Many approaches consider a simplified version of DCOP (equivalent in expressive power) where each agent *owns* some variables, and agents enforce only those constraints with variables assigned by previous agents [19]. In this case, each weighted constraint is defined as a function, $\phi_i^j : X_i \rightarrow \mathbb{R}_+$ where $X_i \subseteq \{x_1, ..., x_i\}$.

Maximization DPCOPs are defined without the element U. The solution is:

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

A rational 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.

Example As a complete example of a (minimization) DPCOP, consider the case of the agents of two consultants, *Alice (A_1) and Bob (A_2)*, trying to schedule a meeting, at one out of two possible dates, T_1 and T_2. At this meeting the consultants plan to exchange experience estimated by Alice at \$200, and estimated by Bob at \$300. The costs to travel on the two dates are known only to the corresponding agent: \$600 and \$800 for Alice, \$600 and \$300 for Bob. These costs reflect both the ticket price, as well as the value of the wage the consultants would receive on those dates if they would work. These wages are secret (as each consultant does not want the other to know the value of his contract). Alice is ready to pay \$30 to hide the real amount he is paid for each of the task scheduled on date T_1, respectively \$50 for the task on day T_2. Bob would pay \$40 and \$20 for the corresponding privacy.

This can be represented as a DPCOP with a single variable, x_1, specifying the time. A_1 has a constraint ϕ_1^1 specifying its cost to travel on each of the two dates: {$\phi_1^1(T_1)$=\$600,$\phi_1^1(T_2)$=\$800}. The costs for A_2 are specified by ϕ_2^1: {$\phi_2^1(T_1)$=\$600,$\phi_2^1(T_2)$=\$300}. P_1 is {$P_1(A_2,\{\phi_1^1(T_1)\})$=\$30, $P_1(A_2,\{\phi_1^1(T_1)\})$=\$50}. The privacy cost functions are assumed additive, i.e., $P_1(A_2,\{\phi_1^1(T_1), \phi_1^1(T_2)\})$=\$80. P_2 is {$P_2(A_1,\{\phi_2^1(T_1)\})$=\$40, $P_2(A_1,\{\phi_2^1(T_1)\})$=\$20}. This implies that $P_2(A_1,\{\phi_2^1(T_1), \phi_1^1(T_2)\})$=\$60.

The rewards for meeting are U_1=\$200, and U_2 = \$300.

3 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, 4], 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 [5]), 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.

procedure *leader* **do**
  foreach *next tuple ε with better local weight than currently best tuple* **do**
    decide next_leader // only N-leaders version;
    send ask(ε,next_leader);
    set next leader // only N-leaders version;
    wait answers;
    update identity of best tuple;

procedure *slaves* **do**
  when *ask (ε, next_leader)* **do**
    compute local cost for ε;
    send answer(ε, cost) to leader;
    recompute privacy_loss;
    leader := next_leader // only N-leaders version;
    if *(leader = myself)* **then**
      change to leader mode // N-leaders version;

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

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]).

4 RPS Stable Matchings Benchmarks

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*.

Stable matching Distributed stable matching consist in matching m participants of a type to m participants of another type.

We developed a generator for DPCOP models of stable matching between m agents of a type and m agents of another type. The first m agents are of one type, and the last m agents are of the second type. To model secret preferences we introduce a variable $P^i_{j,k}$ for each pair j, k, $0 \leq j < k \leq m$ [16, 12]. In a version with 2 preferences, variable $P^i_{j,k}$ has one of two values (0 meaning that A_i prefers A_{2m-j} to A_{2m-k}, and 1 meaning that it does not prefer A_{2m-j} to A_{2m-k}. Each agent A_i receives $m*(m-1)/2$ private unary constraints on the variables $P^i_{j,k}$. In a version with 3 preferences, a private variable has one of three values: 0 meaning that A_i prefers A_{2m-j} to A_{2m-k}, 1 meaning that it equally prefers A_{2m-j} and A_{2m-k}, and 2 stands for the remaining situation.

In general someones preferences may not be transitive (as in rock/paper/scissors mating patterns [17]). For such problems it is possible that $P^i_{j,k}$ and $P^i_{k,t}$ are both 0, but $P^i_{j,t}$ is 1. We call this version RPS Stable Matching Problems, and remark that instances of such problems may not have any stable solution.

The definition of the problem is based on m additional variables, x_i, each of them with m values. The value of x_i give the index of the participant that is matched with A_i in a stable solution. The conditions of stability and the fact that each agent can be matched with exactly one agent of the other type, are specified using a set of public quaternary constraints. A quaternary constraint is created between each quadruplet x_i, x_j, $P^i_{u,v}$, and $P^u_{j,i}$. This constraint specifies that:

"If A_i is matched with A_{2m-u} and A_j is matched with A_{2m-v}, then u must be different from v; and if A_i prefers A_{2m-u} to A_{2m-v}, then A_{2m-v} prefers A_j to A_i."

In the generated DPCOP models, the private constraints are hard constraints (with weights in $\{0, \infty\}$), to anchor in reality the evaluation of a solution. The public constraints are soft constraints, each unstable matching having cost 1. The fact that each agent is matched with exactly one other agent of the opposite type remains a hard constraint. The rewards generated for reaching a solution with minimization DPCOPs are infinite.

A pseudocode of the RPS-SM problem generator is given in Algorithm 2.

print the number of variables;
print the variables with their domains;
print total number of constraints;
foreach *participant* A_i **do**
  foreach *participant pair* (A_k, A_j) *of opposite type* **do**
    print i's unary secret constraint on $P^i_{k,j}$;
foreach *guadruple:* (x_i, x_j, $P^i_{u,v}$, $P^u_{j,i}$) **do**
  print the public constraint x_i, x_j, $P^i_{u,v}$, and $P^u_{j,i}$. This constraint specifies that, if A_i is matched with A_{2m-u} and A_j is matched with A_{2m-v}, then u must be different from v; and if A_i prefers A_{2m-u} to A_{2m-v}, then A_{2m-v} prefers A_j to A_i.

Algorithm 2: Random RPS Stable Matching generator

Privacy Leaks For problems with $m = 2$ the number of private constraints per agent is 1, and therefore secrets are lost each time that they are used for answering to a leader with a cost (weight).

For problems with more participants, one positive (0 weight) answer in any of the two baseline techniques will reveal all three secrets involved, but a infinite weight answer contains an aggregated information about a set of secrets. It reveals one secret only if the remaining secrets aggregated with it are revealed by finite weight answers, and requires additional data storage for reconstructing shadow COPs [18].

The inference technique is shown in Algorithm 3.

shadow constraint ← unknown;
list of ∞ cost answers ← $\emptyset$;
when *finite cost answer* **do**
  set values in shadow COP to 0;
  remove assignments of newly learned unary constraints from ∞ cost answers;
  revisit ∞ cost answers becoming unary;
when ∞ *cost answer* **do**
  remove variables of known constraints from answer;
  if *answer less than* k*-ary* **then**
    add answer to list of ∞ cost answers;
  revisit ∞ cost answers becoming unary;
procedure *revisit* ∞ *cost answers becoming unary* **do**
  set shadow tuples in unary ∞ cost answers to infinity;
  remove the cost answer from the list of answers;

Algorithm 3: SMI: Stable Matching Inference of secrets

This algorithm creates in each agent a shadow of each secret unary constraint of the other agents. This shadow is filled with each new information received from the corresponding agent. Each time a finite weight is received from an agent for a given assignments tuple, the projection of the tuple on the variables of the secret unary constraints of that agent are marked as having cost 0. Whenever an infinite cost is received for a tuple, the set of projections of the tuple on the variables in the secret unary constraints of the sender is enqueued in the list of ∞ cost answers (if the number of unknown secrets involved is smaller than a bound k, used to bound the space complexity). Each time that a set from this list contains a known infinite cost element, it is removed. Any 0-cost element is removed from its set. When a set from this list contains a single unknown element (is unary), we infer that this element has infinite weight.

For non-RPS Stable Matching problems, where preferences present transitivity properties, one can also apply Floyd-Warshall to compute the transitive closure of these preferences, recovering additional secrets.

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
BL	RPSX	4	3	443	116.7	0.7
BL	RPSX	6	2	1502	$9.6*10^5$	5822

Table 1. Stable matching versions.

Privacy loss avoidance Inferred shadows for the secret constraints of other participants are used by the leader to predict the answers of other agents and to even skip asking the question if the prediction proves that the current tuple is suboptimal (e.g., one of the other agents have preferences that make a matching unstable).

Experimental Results For stable matching problems, results averaged over 25 instances are given in Table 1. The cases are: RPS stable matching with soft constraints (RPSX), RPS stable matching with hard constraints (RPSM), classic stable matching with transitive preferences (STM). Results are given for the baseline algorithm with one leader (BL), and for the (HP) cryptographic implementation of the [12] technique. The cryptographic algorithm leaks only the secrets implied by the fact that the solution is stable (if an agent A prefers another than its match in the solution, that other did not prefer A to his match).

Some cryptographic solvers are guaranteed to find optimal solutions for DPCOPs, at the expense of efficiency [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 [4].

5 Related Work

Privacy has been a fundamental concern in distributed constraint optimization, since the beginning of the field [19, 9]. An early quantitative measurement of privacy loss was based on simply counting the number of disclosed tuple values [2, 3]. Some attention has been given to privacy in distributed constraint optimization (DCOP) [7]. However, this work considers all secrets to be equally important, and does not use prior knowledge about the utility of secrecy. Problems with privacy requirements are treated as multi-criteria optimization problems, where the constraint weights are of a different nature from privacy (e.g., perceived from an information theoretic perspective). Some common algorithms for solving DCOPs are ADOPT [10], DPOP [11], and DisAO [8]. There also exist DCOP optimization techniques using cryptographic protocols [4, 14] that preserve privacy.

A closely related framework is the Distributed Private Constraint Satisfaction Problems (DisPrivCSPs) [13], which focuses on distributed constraint satisfaction problems. DisPrivCSPs label each secret with a number corresponding to its importance, provided as an input to the problem. DisPrivCSPs also utilize cost of privacy loss, but not integrated with the cost of the agreement tuples.

The innovation in DPCOPs as compared to previous research in DCOPs is:

- DPCOPs unify the metric for cost of privacy loss with the metric used for specifying weights of constraints. In DisPrivCSPs they were modeled as incomparable and treated distinctly.
- As in [7], revelation in DPCOPs allows for statistical and non-additive privacy loss costs. Unlike [7], DPCOPs also model varying importance of secrets.

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 finite limit on the reward of the agent. While integration of efficiency criteria into the utility framework was already addressed in the general agent framework in [6], its specialization to DPCOP may raise additional interesting issues in future research.

6 Conclusion

We present a new framework of DPCOP where privacy loss and weight of constraints in DCOPs are measured with the same unit (utility) and integrated into a single optimization criteria. DPCOPs allow us to model problems with complex privacy requirements. We note that existing DCOP solvers apply to DPCOPs, and we provide some benchmarks.

References

[1] I. Brito and P. Meseguer. Distributed forward checking may lie for privacy. In *CP DCR Workshop*, 2007.

[2] M. Franzin, F. Rossi, F. E.C., and R. Wallace. Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. *Computational Intelligence*, 20(2), 2004.

[3] E. Freuder, M. Minca, and R. Wallace. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. In *Proc. IJCAI DCR*, pages 63–72, 2001.

[4] R. Greenstadt, B. Grosz, and M. D. Smith. SSDPOP: Improving the privacy of PDCOP with secret sharing. 2007.

[5] Y. Hamadi. Interleaved backtracking in distributed constraint networks. In *ICTAI*, pages 33–41, 2001.

[6] S. Kraus and J. Wilkenfeld. The function of time in cooperative negotiations. In *AAAI*, pages 179–184, 1991.

[7] R. T. Maheswaran, J. P. Pearce, E. Bowring, P. Varakantham, and M. 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] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS*, pages 438–445, 2004.

[9] P. Meseguer and M. Jiménez. Distributed forward checking. In *CP'2000 Distributed Constraint Satisfaction Workshop*, 2000.

[10] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *AIJ*, 161, 2005.

[11] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, 2005.

[12] M.-C. Silaghi. Incentive auctions and stable marriages problems solved with privacy of human preferences. Technical Report TR-FIT-11/2004, Florida Institute of Technology, Melbourne, FL, July 2004.

[13] M.-C. Silaghi and B. Faltings. A comparison of DisCSP algorithms with respect to privacy. In *AAMAS-DCR*, 2002.

[14] M.-C. Silaghi, B. Faltings, and A. Petcu. Secure combinatorial optimization using DFS-based variable elimination. In *Symposium on AI and Maths*, January 2006.

[15] M.-C. Silaghi and V. Rajeshirke. The effect of policies for selecting the solution of a DisCSP on privacy loss. In *AAMAS*, pages 1396–1397, 2004.

[16] M.-C. Silaghi, M. Zanker, and R. Bartak. Desk-mates (stable matching) with privacy of preferences, and a new distributed CSP framework. In *Proc. of CP'2004 Immediate Applications of Constraint Programming Workshop*, 2004.

[17] B. Sinervo and C. M. Livley. The rock-paper-scissors game and the evolution of alternative male strategies. Nature 340:240-243, 1996.

[18] R. Wallace and M.-C. Silaghi. Using privacy loss to guide decisions in distributed CSP search. In *FLAIRS'04*, 2004.

[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.