SSDPOP: Improving the Privacy of DCOP with Secret Sharing

Citation

Smith, Michael, Barbara Grosz, and Rachel Greenstadt. 2007. SSDPOP: Improving the Privacy of DCOP with Secret Sharing. In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems 2007, AAMAS '07: May 14 - 18, 2007, Honolulu, Hawaii, ed. IFAAMAS, 1091-1093. Red Hook, NY: Curran.

Published Version

http://doi.acm.org/10.1145/1329125.1329333

Permanent link

http://nrs.harvard.edu/urn-3:HUL.InstRepos:2562337

Terms of Use

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

SSDPOP: Improving the Privacy of DCOP with Secret Sharing

Rachel Greenstadt, Barbara Grosz and Michael D. Smith
Harvard University
<greenie,grosz,smith>@eecs.harvard.edu

ABSTRACT

Multi-agent systems designed to work collaboratively with groups of people typically require private information that people will entrust to them only if they have assurance that this information will be protected. Although Distributed Constraint Optimization (DCOP) has emerged as a prominent technique for multiagent coordination, existing algorithms for solving DCOP problems do not adeqately protect agents' privacy. This paper analyzes privacy protection and loss in existing DCOP algorithms. It presents a new algorithm, SS-DPOP, which augments a prominent DCOP algorithm (DPOP) with secret sharing techniques. This approach significantly reduces privacy loss, while preserving the structure of the DPOP algorithm and introducing only minimal computational overhead. Results show that SSDPOP reduces privacy loss by 29-88% on average over DPOP.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Algorithms, Performance, Security

Keywords

constraint reasoning, DCOP, privacy

1. INTRODUCTION

Many problem domains within multiagent systems and constraint processing require new algorithms to provide better privacy. For instance, in the domain of multiagent scheduling, agents must schedule events subject to individual constraints. These agents represent people whose preferences are personal information. Their calendars may contain the intersection of multiple professional endeavors as well as events relevant to their personal lives. They want a solution that will optimize global social welfare, but are not willing to expose their personal information to the other agents.

Resource allocation problems, which arise in many domains are another example of constraint problems requiring privacy preserving algorithms. For instance, multiple government and private groups may plan to aid in a disaster relief effort. They wish to contribute

resources optimally, but be mutually distrustful and not want the others to know their holdings and constraints.

Algorithms for distributed constraint optimization (DCOP) [5, 12] provide a useful formalism for solving these types of problems. In these algorithms, each agent passes messages as necessary to other agents to collaboratively compute the solution, rather than surrendering all data to a central server. Experimental analysis [2] has shown that this approach can improve privacy over a centralized approach, but current DCOP algorithms all leak significant amounts of private information.

A key contribution of this paper is a complete characterization of where privacy is lost in DCOP algorithms. We identify four sources of privacy loss: *initial vulnerability*, *intersection vulnerability*, *domain vulnerability* and *solution vulnerability*. For DCOP algorithms that organize agents into tree topologies, most privacy loss is a result of initial vulnerability, agents at the bottom of the tree revealing information [2].

We use this characterization to identify areas where the judicious use of cryptographic techniques can provide for greater privacy. Prior work [8, 13] has placed the entire computation in the cryptographic realm (using secure multiparty computation), adding costly overhead to constraint processing. In contrast, we protect only the most "leaky" part of the computation with cryptography.

Another contribution is a novel algorithm that uses secret sharing [7], a well-known cryptographic technique, to greatly reduce privacy loss without incurring significant runtime overhead. The algorithm, SSDPOP, augments the DPOP algorithm [6] with secret sharing in order to protect the privacy of agents at the bottom of the DCOP topology. Since secret sharing only involves addition and polynomial interpolation, it is much faster than secure multiparty computation. While we describe SSDPOP in terms of meeting scheduling, it can be applied to the full range of DCOP problems.

In following sections, we characterize privacy protection in current DCOP algorithms and the vulnerabilities future algorithms must overcome in order to better preserve privacy. We then present the SSDOP algorithm in detail and explain how it completely eliminates initial vulnerability, the most egregious of the defined vulnerabilities. Our experimental results demonstrate large privacy improvements of SSDPOP over DPOP and that the efficiency costs of SSDPOP over DPOP are small.

2. INFORMATION FLOW IN DCOP

A constraint reasoning problem is one in which a set of variables need to be assigned values, subject to various constraints. In constraint optimization, the goal is to find globally optimal values. In the meeting scheduling domain, shared constraints include information about the meetings to be scheduled and the agents participating in each meeting. This information is public, in constrast

to the individual preference constraints (i.e. utilities for scheduling meetings at particular times) which are private and need to be protected. This definition of public and private information is consistent with other studies [2, 4].

A distributed solution is appropriate for domains where privacy matters. In addition, distribution may also improve scalability, reduce communication costs, or allow increased agent autonomy.

Formally, a discrete distributed constraint optimization (DCOP) is a tuple (X, D, R) such that $X = x_1, \cdots, x_n$ is a set of variables, $D = d_1, \cdots, d_n$ is a set of finite domains of the variables, and $R = r_1, \cdots, r_m$ is a set of relations where a relation r_i denotes the utility assigned to each possible combination of values of the relevant subset of the variables in r_i. Negative utilities are costs [6].

For meeting scheduling, the meetings are the variables and the domain is the set of possible times. The constraints are the agents' valuations for sets of meetings assigned to possible times.

Agents solving a DCOP need to communicate their preference constraints on the variables to each other in order to solve the problem. In current DCOP algorithms, agents communicate directly with only their neighbors in a given topology and achieve some privacy using aggregation. Typically, a message containing constraint information is passed from agent A to agent B, then agent B does some computation and passes the result to agent C. The message from B to C contains information about both A and B's constraints. If the constraints are well aggregated, C may be unable to determine the constraints of the individual agents, affording A and B some privacy with respect to C.

However, since A went first, it could not aggregate its information with any other agent and B learns A's constraints. Protecting privacy with aggregation has a bootstrapping problem in which at least one agent's constraints are initially exposed as the distributed computation begins. We refer to this vulnerability in existing DCOP algorithms as the *initial vulnerability*. The initial vulnerability is the greatest source of privacy loss in DCOP algorithms [2]. Our SSDPOP algorithm eliminates *initial vulnerability*.

Lesser sources of privacy loss in DCOP appear more relevant once the initial vulnerability is removed. Two of these vulnerabilities occur when aggregation is insufficient to hide the private information of one or more agents. In particular, an *intersection vulnerability* occurs any time that an agent B is constrained by different sets of variables than the other agents in the aggregation, since B's partial solution will expose unaggregated variables. A *domain vulnerability* occurs when all aggregated agents' values for a particular timeslot are the same, and these values are near the edges of the domain. For example, if the domain interval is $[0, 5]$ and the aggregate value for A and B is 0, then their individual values must also be 0. SSDPOP takes measures to reduce privacy loss from intersection and domain vulnerabilities.

A final source of privacy loss in DCOP is information revealed by the solution itself: *solution vulnerability*. Though it is possible to mitigate solution vulnerability [8], we consider the solution public information and accept this privacy loss.

3. THE SSDPOP ALGORITHM

This paper focuses on extending the DPOP algorithm [6] to reduce its privacy loss. We began with DPOP for the following reasons: It is simple and efficient. In addition, DPOP has performed well in experimental privacy analysis. DPOP and Adopt were shown to be the most private algorithms, consistently outperforming both other DCOP algorithms and centralized approaches [2]. Thus, privacy improvements in DPOP are a significant advance.

To understand our modifications, it is necessary to explain the DPOP algorithm and its three phases. First, a Depth-First Search

(DFS) tree is created such that agents who share constraints are in the same branch. Each agent x identifies its parent and pseudoparents[1] in the tree. Second, UTIL messages are propagated up the tree, starting with the leaves. A UTIL message contains variables that share constraints with the sending agent's parent or pseudoparents and a table with utilities for each possible assignment of included variables. When an agent receives UTIL messages from all its children, it aggregates them into a new UTIL message, pruning away variables not shared by agents above it in the tree and adding variables that constrain its parents or pseudoparents. This phase ends when the root receives UTIL messages from all its children.

In the third and final phase, the root agent determines the optimal values for its variables and sends the result to its children in a VALUE message. Each agent determines its optimal value based on the computation it performs and the VALUE message sent by its parent. The agent then sends VALUE messages to its children.

3.1 SSDPOP: DPOP with Secret Sharing

Most of the privacy loss in DPOP can be attributed to initial vulnerability. Every leaf in the DFS tree suffers from initial vulnerability. SSDPOP modifies DPOP to protect these leaves. Without loss of generality, we will assume the tree is a simple chain, and an agent B is the bottom of the chain. SSDPOP uses secret sharing [7] to aggregate the results of a single meeting M, without revealing the individual valuations that went into that meeting. The aggregate values for meeting M are then passed to agent B. Agent B aggregates this information with his own valuations and sends the aggregate up the chain. The initial vulnerability is eliminated.

The SSDPOP algorithm has four phases:

(1) Topology building: SSDPOP assigns agents to positions in the chain. One meeting M is chosen for secret sharing.

(2) Secret sharing: The agents in M use secret sharing to send aggregate values for M to the bottom agent B without revealing their individual valuations. B forms a UTIL message combining its valuations with the aggregate valuations for M.

(3) UTIL phase: Proceeds just as in DPOP, except that other agents in the chain act as if meeting M does not exist.

(4) VALUE phase: Proceeds as in DPOP, except that agent B will eventually receive an assignment for all meetings except M. When B receives the VALUE message from its parent, it computes the optimal value for M and sends that value up to the relevant agents.

The following two sections describe the first two phases in greater detail. Phases 3 and 4 are straightforward.

3.2 Topology Building

This phase chooses a meeting M for secret sharing and a bottom agent B. There are three constraints on this choice.

The first constraint is that M cannot be a two-agent meeting where one of the participants is B. In such a situation, B could simply subtract out his own valuations to derive those of the other agent. In other words, we must avoid the straightforward intersection vulnerability. The second constraint, also to avoid an intersection vulnerability, is that all B's meetings must contain at least one agent who also is attending meeting M. This constraint is satisfied when meeting M includes B.

The last (soft) constraint is to choose the largest possible meeting M to avoid the domain vulnerability. If all agents in M have the same valuation for a timeslot and that valuation is extreme, this information will be revealed by the aggregate of the meeting to agent B. This problem is less likely to occur for large meetings.

3.3 Secret Sharing

---
[1] Ancestors of x in the tree who share constraints with x.

Secret Sharing [7] is used to communicate aggregate valuations for meeting M to agent B. Secret sharing is a method to divide a secret among a group of agents, each of which is allocated a *share* of the secret.

In the Shamir scheme, secret sharing uses the fact that n (x, y) coordinate pairs define a polynomial of degree $n - 1$. The secret is the y-intercept of the polynomial and the other coefficients are determined randomly. The dealer gives each of the agents a point on the curve. When n agents pool their shares, they can use polynomial interpolation to determine the polynomial, and thus the secret. However, if $n - 1$ agents pool shares, they learn absolutely nothing.

Two other properties of secret sharing are useful for our purposes: (1) Additive Homomorphism: If t_{s_1} is a share of secret s_1 and t_{s_2} is a share of secret of s_2, then $t_{s_1} + t_{s_2} = t_{s_1+s_2}$, a share of $s_1 + s_2$. (2) Efficiency: Unlike cryptography requiring exponentiation, secret sharing is efficient, requiring only polynomial evaluation and interpolation.

In the secret-sharing step of SSDPOP, meeting M has n member agents, $x_1 \cdots x_n$. Each member generates n shares of the secret and sends share i to agent x_i. Each member then adds the shares it has received, $s_1 + \cdots + s_n$, and sends the sum to the bottom agent B. Due to the additive homomorphism, B can construct aggregate values for meeting M, but none of the individual values. B then aggregates its values with the those for M in a UTIL message.

4. RESULTS

In order to evaluate the privacy loss of SSDPOP as it compares to DPOP, we ran experiments over seven scenarios used in previous studies [2, 4]. We found that SSDPOP reduced average privacy loss by 29%-76% over DPOP using the VPS metric EntropyTS [4]. The benefits of using SSDPOP appeared even more pronounced for the $D|A$ metric [3], with an average reduction of 56%-88%. Where privacy loss did occur in SSDPOP, it was due to domain vulnerabilities in the secret shared meeting M.

The overhead incurred by SSDPOP is acceptable in comparison to running DPOP in a chain. Three factors cause performance differences between DPOP and SSDPOP: (1) Selection of the meeting M and the bottom agent B. This takes $O(NE)$ in the worst case where N is the number of agents and E is the number of events to be scheduled. (2) Addition of the secret sharing step, which uses operations that take $O(|M|log^2|M|)$ time. (3) Elimination of the selected meeting M from the optimizing computation, which actually reduced run time. Costs from (1) and (2) can be incurred as preprocessing steps before optimizing.

5. RELATED WORK

Initial privacy work [1,9–11] established the importance of a rigorous understanding of privacy for constraint satisfaction problems. The VPS framework built on this early work and illustrated how key metrics of privacy from earlier work could be captured and cross-compared [4]. However, the VPS framework showed privacy loss to be greater in some DCOP algorithms than in a centralized one. Further work using these metrics showed that the DCOP and ADOPT algorithms improved privacy over centralized approaches [2]. We make use of $D|A$, a metric for privacy in DCOP that focuses on the aggregation of certain data, to produce qualitative results [3]. Yokoo et al [13] discuss a secure DisCSP algorithm that uses cryptographic techniques to ensure strict privacy.

6. CONCLUSION AND FUTURE WORK

The main contributions of this paper are a more complete analysis of information flow in DCOP algorithms, describing four vulnerabilities to privacy loss, and a novel algorithm which significantly reduces privacy loss. The SSDPOP algorithm augments DPOP with secret sharing techniques. SSDPOP significantly reduces privacy loss, by eliminating initial vulnerability, while minimizing domain and intersection vulnerabilities. Results show that SSDPOP reduces privacy loss by 29-88% on average over DPOP according to the VPS metrics EntropyTS and $D|A$. SSDPOP requires only two additional rounds of communication and a preprocessed, polynomial-time computation.

SSDPOP can be extended to protect the leaves in tree topologies. Extending to trees will cause more opportunity for privacy loss from domain vulnerability, but such losses may be acceptable when efficiency concerns prevail.

7. ACKNOWLEDGMENTS

8. REFERENCES

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

[2] R. Greenstadt, J. Pearce, and M. Tambe. An analysis of privacy loss in distributed constraint optimization. In *AAAI*, 2006.

[3] R. Greenstadt and M. Smith. Collaborative scheduling: Threats and promises. In *Workshop on Economics and Information Security*, 2006.

[4] R. Maheswaran, J. Pearce, E. Bowring, P. Varakantham, and M. Tambe. Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications. *JAAMAS*, 2006.

[5] J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 161:149–180, 2005.

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

[7] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11), 1979.

[8] M. Silaghi. Meeting scheduling guaranteeing n/2-privacy and resistant to statistical analysis. In *3rd IC on Web Intelligence*, 2004.

[9] M. Silaghi and B. Faltings. A comparison of distributed constraint satisfaction approaches with respect to privacy. In *Workshop on Distributed Constraint Reasoning*, 2002.

[10] M. Silaghi and D. Mitra. Distributed constraint satisfaction and optimization with privacy enforcement. In *IAT*, 2004.

[11] R. Wallace and E. Freuder. Constraint-based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving. *AIJ*, 161(1-2):209–227, 2005.

[12] M. Yokoo, E. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5), 1998.

[13] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *CP 2002*, Berlin, 2002.