Privacy Preserving DCOP Solving by Mediation

Pablo Kogan[1,3], Tamir Tassa[1][0000−0001−9681−8824], and Tal Grinshpoun[2,3][0000−0002−4106−3169]

[1] Department of Mathematics and Computer Science, The Open University of Israel
[2] Department of Industrial Engineering and Management, Ariel University, Israel
[3] Ariel Cyber Innovation Center, Ariel University, Israel
pablokogan@pm.me, tamirta@openu.ac.il, talgr@ariel.ac.il

Abstract. In this study we propose a new paradigm for solving DCOPs, whereby the agents delegate the computational task to a set of external mediators who perform the computations for them in an oblivious manner, without getting access neither to the problem inputs nor to its outputs. Specifically, we propose MD-Max-Sum, a mediated implementation of the Max-Sum algorithm. MD-Max-Sum offers topology, constraint, and decision privacy, as well as partial agent privacy. Moreover, MD-Max-Sum is collusion-secure, as long as the set of mediators has an honest majority. We evaluate the performance of MD-Max-Sum on different benchmarks. In particular, we compare its performance to PC-SyncBB, the only privacy-preserving DCOP algorithm to date that is collusion-secure, and show the significant advantages of MD-Max-Sum in terms of runtime.

Keywords: DCOP · Max-Sum · Privacy · Multiparty computation · Mediated computing

1 Introduction

A Distributed Constraint Optimization Problem (DCOP) [5,2] is a commonly accepted and practical mathematical framework for solving coordination challenges in multi-agent systems. A DCOP consists of a set of variables that are controlled by several independent agents. Some subsets of variables may be dependent in the sense that when they are assigned values from their respective domains, different combinations of those values may incur costs. The goal is to assign values to all variables so that the sum of all incurred costs would be minimal. One of the main motivations for solving constraint optimization problems in a distributed manner is to preserve the privacy of the interacting agents. Hence, many *privacy-preserving* DCOP algorithms were proposed over the past two decades (see the review of related work in Section 2).

All existing DCOP algorithms (privacy-preserving or not) are carried out by the agents themselves. However, some of those algorithms, and in particular the privacy-preserving ones, require significant computing resources. In addition, all DCOP algorithms assume that the agents are connected through a communication network. We propose here a new paradigm in DCOPs: solving them in the

so-called mediated model; i.e., there are external servers to whom the agents send the problem inputs in some protected manner. The external servers, whom we call *mediators*, simulate a DCOP algorithm on those inputs. At its completion, they send to the agents messages from which the agents extract the outputs, i.e., the variable assignments. Throughout this process, the mediators remain oblivious to the problem inputs, to the content of the protected messages that they exchange, and to the outputs.

Performing the DCOP algorithm in such a mediated manner offers several significant advantages: (a) it protects the privacy of the agents and their sensitive private data; (b) the agents do not need to establish a communication network amongst them; and (c) it delegates the computational workload from the agents, who may have limited computational resources, to dedicated servers.

In this work we demonstrate the power of mediated computing by presenting MD-Max-Sum, a mediated execution of the Max-Sum algorithm [1]. In MD-Max-Sum, the agents send to the mediators secret shares [13] in their private inputs. The mediators proceed to execute the Max-Sum computations on the shares of the problem inputs, while remaining oblivious to the value of the underlying inputs. At the end, the mediators send to the agents messages from which the agents may infer the assignments to their variables. If the mediators have an honest majority (namely, the number of colluding mediators is smaller than the number of mediators outside the coalition), then MD-Max-Sum provides topology, constraint, and decision privacy, as well as partial agent privacy (see [9] for the definition of those notions). Those security guarantees hold against any collusion among the set of agents.

2 Related work

Léauté and Faltings [9] devised three secure versions of the complete DCOP-solving algorithm DPOP [12], each with different runtimes and privacy guarantees. In their study they suggested four notions of privacy for the DCOP framework, to which we adhere in this study: agent, topology, constraint, and decision privacy. Grinshpoun and Tassa [3] presented a privacy-preserving version of another complete algorithm – SyncBB [5]. The resulting method, P-SyncBB, preserves topology, constraint, and decision privacy.

Since complete algorithms are not scalable, research efforts were invested also in designing privacy-preserving implementations of incomplete algorithms. Tassa et al. [15] developed the P-Max-Sum algorithm, which runs Max-Sum with cryptographic enhancements in order to preserve privacy. P-Max-Sum provides topology, constraint, and decision privacy, and it may be extended to provide also agent privacy. Grinshpoun et al. [4] devised another incomplete privacy-preserving algorithm, P-RODA, which is based on region optimality [6,7]. P-RODA provides constraint privacy and partial decision privacy.

All of the above mentioned privacy-preserving DCOP algorithms assume *solitary* conduct of the agents. However, if two or more agents collude and combine the information that they have, they may extract valuable information about

other agents. To address the risk of collusion, Tassa et al. [14] introduced PC-SyncBB, the first privacy-preserving DCOP algorithm that is *collusion-secure*. It is secure under the assumption that the agents have an honest majority. PC-SyncBB does extensive usage of Secure Multiparty Computation (MPC) [16] in order to obliviously compare between costs of partial assignments.

In this work we propose MD-Max-Sum, the first *incomplete* privacy-preserving DCOP algorithm that is *collusion-secure*.

3 DCOP definitions and the Max-Sum algorithm

A Distributed Constraint Optimization Problem (DCOP) [5] is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$ where $\mathcal{A}$ is a set of agents $A_1, A_2, \ldots, A_N$, $\mathcal{X}$ is a set of variables $X_1, X_2, \ldots, X_N$, $\mathcal{D}$ is a set of finite domains $D_1, D_2, \ldots, D_N$, and $\mathcal{R}$ is a set of relations (constraints). Each variable X_n, $n \in [N] := \{1, 2, \ldots, N\}$, takes values in the domain D_n, and it is held by the agent A_n.[4] Each constraint $C \in \mathcal{R}$ defines a non-negative cost for every possible value combination of some subset of variables, and is of the form $C : D_{n_1} \times \cdots \times D_{n_k} \to [0, q]$, for some $1 \leq n_1 < \cdots < n_k \leq N$, and a publicly known maximal constraint cost q.

An *assignment* is a pair including a variable and a value from that variable's domain. The goal of the agents is to find assignments to their variables so that the sum of all costs that those assignments incur would be minimal.

We consider here a binary version of DCOPs, in which every $C \in \mathcal{R}$ constraints exactly two variables and takes the form $C_{n,m} : D_n \times D_m \to [0, q]$, where $1 \leq n < m \leq N$. Such an assumption is customary in DCOP literature, see e.g. [11,12]. As the domains are finite, they me be ordered. Hence, the binary constraint $C_{n,m}$ between X_n and X_m may be described by a matrix, which we also denote by $C_{n,m}$, of dimensions $|D_n| \times |D_m|$; in that matrix, $C_{n,m}(i, j)$ equals the cost that corresponds to the assignment of the ith value in D_n to X_n and the jth value in D_m to X_m, where $1 \leq i \leq |D_n|$ and $1 \leq j \leq |D_m|$.

The constraint graph $G = (V, E)$ is an undirected graph over the set of variables $V = \mathcal{X}$, where an edge in E connects two variables if and only if they are constrained. If we define for every pair of variables $(X_n, X_m) \notin E$ a constraint matrix $C_{n,m}$ which is the zero matrix of dimensions $|D_n| \times |D_m|$, then the set of all such matrices,

$$\mathcal{C} := \{C_{n,m} : 1 \leq n < m \leq N\}, \tag{1}$$

encompasses all topology and constraint information.

Every DCOP is also associated with a so-called *factor graph*. It is a bipartite graph $G' = (V', E')$ that is defined as follows. The set V' has two types of nodes: *variable nodes*, $X_1, \ldots, X_N$, and *function nodes*, X_e, for each $e = (X_n, X_m) \in E$. The edge set E' has an edge connecting X_n with X_e if and only if e is an edge in G that is adjacent to X_n.

[4] We make the standard assumption that the number of variables equals the number of agents, and that each variable is held by a distinct single agent, see e.g. [11,12].

The Max-Sum algorithm [1] is an iterative message-passing algorithm that operates on the factor graph G'. In every iteration a message is sent from each node in V' to each one of its adjacent nodes. In the kth iteration, each variable node X_n sends to each adjacent function node X_e a message denoted $Q^k_{n\to e}$, while the message in the opposite direction is denoted $R^k_{e\to n}$; both messages are vectors of dimension $|D_n|$. In the $k = 0$ iteration all messages are zero. After completing the kth iteration, the messages in the next iteration will be as follows. Fixing a variable node X_n and letting V_n be the set of function nodes adjacent to X_n in G', then for each $X_e \in V_n$, X_n will send to X_e the vector

$$Q^{k+1}_{n\to e} := \sum_{X_f \in V_n \setminus \{X_e\}} R^k_{f\to n}\,. \tag{2}$$

As for messages sent from function nodes, if X_e is a function node that connects the two variable nodes X_n and X_m then the message sent from X_e to X_n is

$$R^{k+1}_{e\to n}(x) := \min_{y\in D_m} \left[C_{n,m}(x,y) + Q^k_{m\to e}(y)\right]\,, \quad \forall x \in D_n\,; \tag{3}$$

the message that X_e sends to X_m is constructed similarly. Finally, after completing a preset number K of iterations, each variable node X_n computes $\overline{R}_n := \sum_{X_e\in V_n} R^K_{e\to n}$ and then selects a value $x \in D_n$ for which $\overline{R}_n(x)$ is minimal.

4 Mediated Max-Sum

In order to implement Max-Sum in a manner that preserves the privacy of the agents even when some of them collude, we propose herein MD-Max-Sum – an implementation of Max-Sum in the mediated model.

Let $\mathbf{M} = \{M_1, \ldots, M_L\}$ be an external committee of so-called *mediators*. The agents in $\mathcal{A}$ will share their DCOP private inputs, namely, the topology and constraint information, with the mediators using a t-out-of-L threshold secret sharing scheme [13], where $t := \lfloor (L+1)/2 \rfloor$. Specifically, the agents distribute to the mediators t-out-of-L shares in each of the entries in each of the matrices in $\mathcal{C}$, see Eq. (1); the underlying secret sharing field will be denoted henceforth by $\mathbb{Z}_p$. The agents trust the mediators to have an honest majority, in the sense that if some of the mediators decide to collude in order to reconstruct the shared private data, the number of colluding mediators would be smaller than the number of mediators outside the coalition. Under that assumption, the mediators cannot recover the private inputs that were shared with them, since at least t mediators have to collude in order to be able to reconstruct the shared secrets, and $t = \lfloor (L+1)/2 \rfloor \geq L - t$.

After the agents had completed sharing all their private inputs with the mediators, they go to rest and the mediators start emulating the performance of the entire Max-Sum algorithm by implementing secure multiparty computation (MPC) on the shared data. The main challenge in this regard is to design an implementation of Max-Sum that operates on *shared* data, namely, in a manner that is oblivious to the underlying topology and constraint values.

When the mediators complete their emulation of Max-Sum, say by running an agreed preset number of iterations, K, they send to each of the agents a message from which that agent can infer the assignment of its variable in the solution that the algorithm had found.

In order to hide the constraint graph topology from the mediators, MD-Max-Sum operates on an augmented version $G_+ = (V, E_+)$ of the constraint graph $G = (V, E)$. G_+ is a complete graph in the sense that E_+ includes all $\binom{N}{2}$ pairs of nodes/variables from $V = \mathcal{X}$, where all edges $(X_n, X_m) \in E_+ \setminus E$ are associated with a zero constraint matrix, $C_{n,m}$, as described earlier in Section 3. In the full version of this paper [8] we show that the addition of such so-called *phantom edges* does not change the algorithm's outputs. However, with such added phantom edges, the mediators, who only get shares in the constraint matrices, cannot distinguish between a zero matrix and a non-zero matrix and, consequently, they cannot tell which of the edges in the augmented graph are phantom ones, so the graph topology is preserved.

4.1 The MD-Max-Sum algorithm

We assume that all agents know the total number of agents N, and the identifying index $n \in [N]$ of each agent. In addition, the sizes of all domains, $|D_n|$, $n \in [N]$, are also publicly known.

4.1.1 Distributing to the mediators shares in the problem inputs In this preliminary stage, the agents share with the mediators the problem inputs, which, as explained in Section 3, are encoded through the set of matrices $\mathcal{C}$, see Eq. (1). To do so, each agent A_n, $1 \leq n \leq N-1$, shares with the mediators $\mathbf{M}$ the constraint matrices $C_{n,m} \in \mathcal{C}$ for all $n < m \leq N$, where, as explained in Section 3, the matrix $C_{n,m}$ is of dimensions $|D_n| \times |D_m|$ and it either spells out the constraint values between those two agents, or, if they are not constrained, it is the zero matrix. The matrices are shared by performing an independent t-out-of-L secret sharing for each entry in each of those matrices, where $t = \lfloor (L+1)/2 \rfloor$. Letting $n < m \in [N]$ be indices of two agents, and $\ell \in [L]$ be an index of a mediator, we denote the share of the cost $C_{n,m}(i,j)$ that the mediator M_ℓ receives by $C^\ell_{n,m}(i,j)$. The entire matrix of shares that M_ℓ receives is denoted

$$C^\ell_{n,m} = \left(C^\ell_{n,m}(i,j) : 1 \leq i \leq |D_n|, 1 \leq j \leq |D_m|\right) .$$

After each mediator M_ℓ, $\ell \in [L]$, got its share matrix $C^\ell_{n,m}$ for all $1 \leq n < m \leq N$, they have all problem inputs and they may now begin an MPC emulation of Max-Sum over those inputs.

We assume that the mediators have an *honest majority*. Hence, as we use t-out-of-L secret sharing with $t = \lfloor (L+1)/2 \rfloor$, the mediators cannot learn any information on the content of the constraint matrices. Therefore, not only the constraints themselves are kept secret, also the topology is kept secret, since the mediators cannot tell from their shares whether $C_{n,m}$ is the zero matrix or not.

While Max-Sum, as well as P-Max-Sum, operate on the exact factor graph G', the mediated algorithm MD-Max-Sum operates on an *augmented factor graph*, denoted $G'_+ = (V'_+, E'_+)$, in which every two variable nodes are connected through a function node, even if some of those function nodes stand for a zero/phantom constraint (which was introduced only for the purpose of hiding the real topology of G from the mediators).

After the agents finish distributing to the mediators shares in the problem inputs, they go to rest and let the mediators do the work. The mediators start an emulation of each of the iterations in Max-Sum. They do so by producing proper shares in the true messages that would have been sent along each edge of the factor graph, if the agents had run the Max-Sum algorithm by themselves.

We proceed to explain in the next sections the details of the MD-Max-Sum implementation. Specifically, we need to explain how in each iteration of the algorithm, the mediators create shares in the messages that the corresponding Max-Sum algorithm would have generated. In doing so, we focus on an arbitrary pair of neighboring nodes in the augmented factor graph: a variable node X_n, $n \in [N]$, and a function node, X_e, where $e = (X_n, X_m)$ and $m \in [N] \setminus \{n\}$.

4.1.2 Producing shares in the messages of the initial iteration In iteration 0, all of the L mediators have to emulate zero messages between X_n and X_e,

$$Q^0_{n\to e} = (0, \ldots, 0) \in \mathbb{Z}_p^{|D_n|}\,, \quad R^0_{e\to n} = (0, \ldots, 0) \in \mathbb{Z}_p^{|D_n|}\,. \tag{4}$$

To do so, each mediator M_ℓ, $\ell \in [L]$, creates for himself corresponding zero share vectors as follows:

$$Q^{0,\ell}_{n\to e} = (0, \ldots, 0) \in \mathbb{Z}_p^{|D_n|}\,, \quad R^{0,\ell}_{e\to n} = (0, \ldots, 0) \in \mathbb{Z}_p^{|D_n|}\,. \tag{5}$$

Note that no interaction between the mediators is needed at this stage, and that the L vector shares of the Q-messages in Eq. (5) are t-out-of-L vector shares in the zero Q-messages in Eq. (4), and likewise for the R-messages.

4.1.3 Producing shares in *Q*-messages In iteration $k+1$, the mediators have to emulate the message $Q^{k+1}_{n\to e}$ from the variable node X_n to the adjacent function node, X_e, where $e = (X_n, X_m)$. In view of Eq. (2), and as the mediators already have t-out-of-L shares in R-messages of the kth iteration, such a computation can be done, without interaction between the mediators, by computing $Q^{k+1,\ell}_{n\to e} := \sum_{X_f} R^{k,\ell}_{f\to n}$, where the sum is over all $N-2$ function nodes, X_f, between X_n and X_i for any $i \in [N] \setminus \{n, m\}$.

4.1.4 Producing shares in *R*-messages Here, we concentrate on the more involved task of computing t-out-of-L shares in the R-messages, $R^{k+1}_{e\to n}$, from the function node, X_e, where $e = (X_n, X_m)$, to the variable node X_n. We rewrite Eq. (3) in the following manner,

$$R^{k+1}_{e\to n}(x) := \min_{y\in D_m} B^k_{n,m}(x, y)\,, \quad x \in D_n\,, \tag{6}$$

where $B^k_{n,m}(x,y)$ denotes the sum

$$B^k_{n,m}(x,y) := C_{n,m}(x,y) + Q^k_{m\to e}(y)\,. \tag{7}$$

The L mediators hold t-out-of-L shares in $C_{n,m}(x,y)$ for all $(x,y) \in D_n \times D_m$ (denoted $C^\ell_{n,m}(x,y)$, $\ell \in [L]$), since such shares were generated and distributed to them by the agents in the preliminary stage. Moreover, the mediators had computed in the kth iteration t-out-of-L shares in $Q^k_{m\to e}(y)$ for all $y \in D_m$, where M_ℓ's shares are denoted $Q^{k,\ell}_{m\to e}(y)$. Hence, $C^\ell_{n,m}(x,y) + Q^{k,\ell}_{m\to e}(y)$, which we denote by $B^{k,\ell}_{n,m}(x,y)$, are t-out-of-L shares in $B^k_{n,m}(x,y)$, as implied by Eq. (7) and the linearity of secret sharing. Hence, the main computational challenge is to compute t-out-of-L shares in the left-hand side of Eq. (6) from the shares that the mediators hold in each of the terms on the right-hand side of Eq. (6). This task is non-trivial because the minimum function is non-linear.

Protocol 1, which we describe below, is simultaneously executed by each of the L mediators. It is executed for each pair of a function node in the augmented factor graph, X_e, where $e = (X_n, X_m)$, and one of its two adjacent variable nodes, X_n. At the completion of that protocol, each mediator M_ℓ holds a share $R^{k+1,\ell}_{e\to n}$ in $R^{k+1}_{e\to n}$. That protocol will be executed in every iteration $N(N-1)$ times, as there are $\binom{N}{2} = N(N-1)/2$ function nodes in the augmented factor graph G'_+, and each one of them has two adjacent variable nodes.

Protocol 1: Computing shares in an R-message from the function node X_e, where $e = (X_n, X_m)$, to the variable node X_n.

Input: Mediator M_ℓ, $\ell \in [L]$, holds a t-out-of-L share, $B^{k,\ell}_{n,m}(x,y)$, in $B^k_{n,m}(x,y)$, for every $x \in D_n$ and $y \in D_m = \{y_1, \ldots, y_{|D_m|}\}$.

1 **forall** $x \in D_n$ **do**
2 | M_ℓ sets $\beta^\ell_{n,m}(x) \leftarrow B^{k,\ell}_{n,m}(x, y_1)$
3 | **forall** $j = 2, \ldots, |D_m|$ **do**
4 | | **if COMPARE**$(\{B^{k,\ell}_{n,m}(x,y_j)\}_{\ell\in[L]}, \{\beta^\ell_{n,m}(x)\}_{\ell\in[L]}) =$ **true then**
5 | | | M_ℓ sets $\beta^\ell_{n,m}(x) \leftarrow B^{k,\ell}_{n,m}(x,y_j)$
6 | M_ℓ sets $R^{k+1,\ell}_{e\to n}(x) \leftarrow \beta^\ell_{n,m}(x)$

Output: Mediator M_ℓ, $\ell \in [L]$, gets a t-out-of-L share $R^{k+1,\ell}_{e\to n}(x)$ in $R^{k+1}_{e\to n}(x)$.

The external loop in the protocol (lines 1-6) is over all values x in the domain D_n, i.e., over all entries in the vector message $R^{k+1}_{e\to n}$. For each such $x \in D_n$, the mediators have to find the minimum among $\{B^k_{n,m}(x,y) : y \in D_m\}$, where each of the values in that set is shared by a t-out-of-L scheme among them. The t-out-of-L shares of the minimum will be stored in $\beta^\ell_{n,m}(x)$, $\ell \in [L]$. First (line 2), each mediator initiates its β-shares with the shares corresponding to $B^k_{n,m}(x,y_1)$. Then (lines 3-5), for each y_j, $j = 2, \ldots, |D_m|$, the mediators compare $B^k_{n,m}(x,y_j)$ to the current minimum, in which they have t-out-of-L shares in $\beta^\ell_{n,m}(x)$, $\ell \in [L]$. The comparisons are performed in a secure manner by invoking a distributed sub-protocol that all mediators jointly execute, as will be explained below. If $B^k_{n,m}(x,y_j)$ is smaller than the current minimum, then each mediator updates its share of the minimum (line 5).

In order to perform comparisons between values that are known to the mediators only through t-out-of-L shares, without recovering those values and perform the comparison over those recovered values, Protocol 1 calls upon an MPC sub-protocol called **COMPARE** (line 4), which all the mediators run together in a distributed manner. That sub-protocol assumes that the mediators have t-out-of-L shares in two values $x, y \in \mathbb{Z}_p$; it returns **true** if $x < y$ (when x and y are interpreted as integers) and **false** otherwise. This sub-protocol is perfectly secure in the sense that it reveals to the mediators nothing about the two compared values beyond the final output bit which indicates which of the two is smaller. (A full description of **COMPARE** is provided in [8].)

Finally (line 6), each mediator stores in $R^{k+1,\ell}_{e\to n}(x)$ its share in the minimum that was found above, $\beta^{\ell}_{n,m}(x)$.

4.1.5 Termination After completing a preset number of K iterations, the final assignment to X_n, $n \in [N]$, is determined by the minimal entry in $\overline{R}_n = \sum_{X_e \in V_n} R^K_{e\to n}$. To find that assignment, each agent A_n, $n \in [N]$, selects a subset of t mediators and asks them for their shares in the vector $\overline{R}_n$. Using those vector shares, A_n can recover each of the $|D_n|$ entries in $\overline{R}_n$. Afterwards, A_n finds the minimal entry in $\overline{R}_n$ and then assigns the corresponding value to X_n.

4.2 Correctness and privacy

The MD-Max-Sum algorithm is correct and privacy-preserving, as stated in Theorems 1 and 2. (The proofs are given in the full version of this paper [8]).

Theorem 1. *When* MD-Max-Sum *and* Max-Sum *are executed the same number of iterations K on the same input problem, they will issue the same assignments to all variables.*

Theorem 2. MD-Max-Sum *provides topology, constraint, and decision privacy, as long as the mediators have an honest majority.*

5 Experimental evaluation

We implemented and executed the MD-Max-Sum algorithm on the AgentZero simulator [10], running on AWS C5a instances comprised of a 2nd generation AMD EPYC™ 7R32 processor and 64 GB memory, except for the call to the **COMPARE** sub-protocol that was executed over LAN with EC2 machines of type c5.large in Amazon's North Virginia data center.

We compared MD-Max-Sum with the baseline algorithm Max-Sum (no privacy) and P-Max-Sum [15] (provides privacy, but not against coalitions). As shown in [15], and stated above in Theorem 1, both of those privacy-preserving implementations of Max-Sum simulate perfectly the basic Max-Sum. We used in all experiments $K = 10$ iterations in each of these algorithms, similarly to [15]. MD-Max-Sum was executed with $L = 5$ mediators. In addition, we included

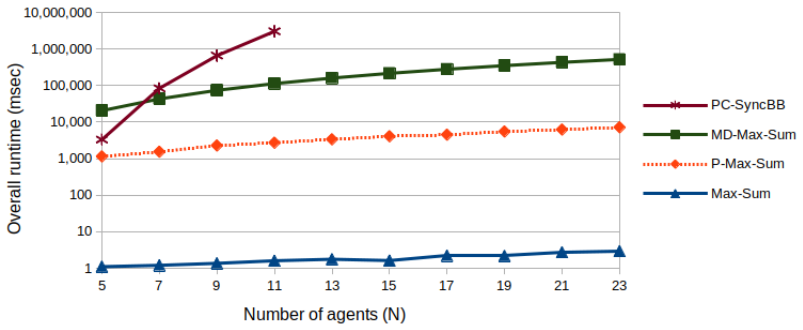

Fig. 1. Unstructured random graphs ($p_1 = 0.3$, $|D_n| = 5$), varying N

in our experiments the PC-SyncBB algorithm [14], which is the only other DCOP-solving algorithm that is privacy-preserving and collusion-secure. Recall that unlike MD-Max-Sum, PC-SyncBB is a complete algorithm; hence, it outputs the optimal solution but it is expected to be more time consuming.

The first experiment, shown in Figure 1, was conducted on unstructured random graphs with constraint density $p_1 = 0.3$ and domains of size $|D_n| = 5$, $n \in [N]$. We varied the number of agents N to observe the scalability of the algorithms. The cut-off time for a single execution was set to 30 minutes.

The performance gap between Max-Sum and P-Max-Sum demonstrates the price of privacy. The gap between P-Max-Sum and MD-Max-Sum demonstrates the price of collusion security. Those two gaps remain constant. Conversely, the gap between MD-Max-Sum and PC-SyncBB, which demonstrates the price of completeness, is constantly growing. For small problems with $N \leq 7$, PC-SyncBB is competitive with MD-Max-Sum and even with P-Max-Sum. However, as the number of agents increases, we can see that the performance of PC-SyncBB becomes much more time-consuming than MD-Max-Sum's. This advantage of MD-Max-Sum over PC-SyncBB is explained as follows: a significant portion of the runtime of both algorithms is in performing secure comparisons between secret values. In PC-SyncBB, that MPC sub-protocol is carried out by all agents; in MD-Max-Sum, on the other hand, it is carried out by the mediators. The runtime of this computation depends on the number of interacting parties, see [14, Table 1]. Hence, while the time spent in PC-SyncBB on secure comparisons increases with N, in MD-Max-Sum it is independent of N. This mitigation of the dependency of the runtime on N demonstrates the strength of the *mediated model*. (Of course, the runtime of MD-Max-Sum does depend on N through other computations, outside the secure comparisons in **COMPARE**, since N affects the size of the graph.)

We also evaluated the algorithms on 3-color graph coloring problems, similar to the setting described by Zivan et al. [17]. In this setting, for every $1 \leq n < m \leq N$, $C_{n,m}(x, y) = q$ if $x = y$ and $C_{n,m}(x, y) = 0$ if $x \neq y$, for some positive constant q. Figure 2 presents the runtime of the algorithms on 3-color graph problems with $p_1 = 0.4$ and shows similar scalability properties to the previous

experiments. The small domain size, $|D_n| = 3$, enables us to experiment with problems of larger sizes. For this experiment, we started with $N = 5$ and moved up to $N = 75$ agents in steps of 10. While all other algorithms remain within the cut-off limit of 30 minutes per single execution, the runtime of PC-SyncBB exceeded the cut-off limit already for $N = 20$. Hence, we include in Figure 2 the runtime of PC-SyncBB for $N = 19$, which was the highest number of agents that could be processed within 30 minutes. The trends are similar to those in the former experiment.

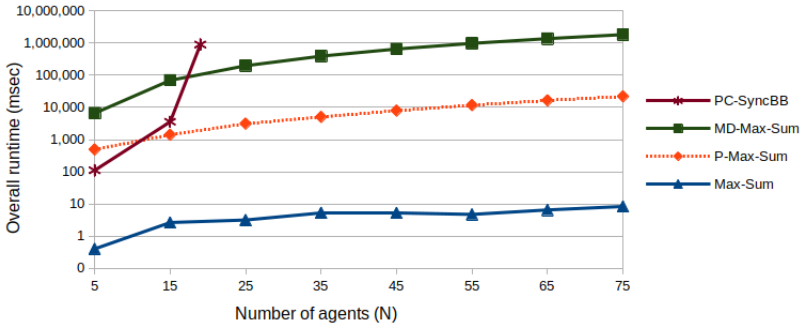

Fig. 2. 3-color graph coloring problems ($p_1 = 0.4$), varying N

6 Conclusion

In this work we introduced MD-Max-Sum, the first incomplete privacy-preserving DCOP algorithm that is also collusion-secure. It is an implementation of Max-Sum in the mediated model of computation. It preserves topology, constraint, decision, and partial agent privacy. We analyzed the security and correctness of the algorithm and, using experimentation, demonstrated its characteristics, its advantages over the only other collusion-secure DCOP algorithm, PC-SyncBB, and its viability.

Aside from the performance gains achieved by utilizing an incomplete algorithm (as opposed to PC-SyncBB that is based on a complete algorithm), the transition to the mediated model offers other significant benefits: MD-Max-Sum is privacy-preserving and is immune to any coalition among the agents, under the assumption of an honest majority within the mediators; the agents do not need to communicate with each other, a significant advantage in settings where the agents do not have an efficient way to communicate among themselves; the agents, that may run on computationally-bounded devices, can outsource costly and cryptographically-complex computations to dedicated servers; and, finally, MD-Max-Sum is more robust than all previous DCOP algorithms since if an agent goes offline (e.g., due to a technical failure) after secret sharing its private data to the mediators, the algorithm can still be executed and issue the correct outputs to all agents.

We believe that the mediated model of computation could be successfully implemented for other DCOP algorithms as well as for various problems of fed-

erated learning, in order to achieve enhanced privacy guarantees, and to reap the advantages of the mediated model of computation as we have identified herein.

Acknowledgments This work was partially supported by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office.

References

1. Farinelli, A., Rogers, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: AAMAS. pp. 639–646 (2008)
2. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research **61**, 623–698 (2018)
3. Grinshpoun, T., Tassa, T.: P-SyncBB: A privacy preserving branch and bound DCOP algorithm. Journal of Artificial Intelligence Research **57**, 621–660 (2016)
4. Grinshpoun, T., Tassa, T., Levit, V., Zivan, R.: Privacy preserving region optimal algorithms for symmetric and asymmetric DCOPs. Artificial Intelligence **266**, 27–50 (2019)
5. Hirayama, K., Yokoo, M.: Distributed partial constraint satisfaction problem. In: CP. pp. 222–236 (1997)
6. Katagishi, H., Pearce, J.P.: Kopt: Distributed DCOP algorithm for arbitrary k-optima with monotonically increasing utility. In: DCR (2007)
7. Kiekintveld, C., Yin, Z., Kumar, A., Tambe, M.: Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In: AAMAS. pp. 133–140 (2010)
8. Kogan, P.: Privacy Preserving Solution of DCOPs by Mediation. Master's thesis, supervised by Tassa, T. and Grinshpoun, T., The Open University of Israel, https://www.openu.ac.il/Lists/MediaServer_Documents/PersonalSites/TamirTassa/MD_Max_Sum.pdf (2022)
9. Léauté, T., Faltings, B.: Protecting privacy through distributed computation in multi-agent decision making. Journal of Artificial Intelligence Research **47**, 649–695 (2013)
10. Lutati, B., Gontmakher, I., Lando, M., Netzer, A., Meisels, A., Grubshtein, A.: AgentZero: A framework for simulating and evaluating multi-agent algorithms. In: Agent-Oriented Software Engineering. pp. 309–327 (2014)
11. Modi, P.J., Shen, W.M., Tambe, M., Yokoo, M.: ADOPT: asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence **161**, 149–180 (2005)
12. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: IJCAI. pp. 266–271 (2005)
13. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
14. Tassa, T., Grinshpoun, T., Yanai, A.: PC-SyncBB: A privacy preserving collusion secure DCOP algorithm. Artificial Intelligence **297**, 103501 (2021)
15. Tassa, T., Grinshpoun, T., Zivan, R.: Privacy preserving implementation of the Max-Sum algorithm and its variants. Journal of Artificial Intelligence Research **59**, 311–349 (2017)
16. Yao, A.C.: Protocols for secure computation. In: FOCS. pp. 160–164 (1982)
17. Zivan, R., Okamoto, S., Peled, H.: Explorative anytime local search for distributed constraint optimization. Artificial Intelligence **212**, 1–26 (2014)