Artificial Intelligence 266 (2019) 27–50

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Privacy preserving region optimal algorithms for symmetric and asymmetric DCOPs

Tal Grinshpoun [a,1], Tamir Tassa [b,*,1], Vadim Levit [c,a], Roie Zivan [c]

[a] Ariel University, Ariel, Israel
[b] The Open University, Ra'anana, Israel
[c] Ben-Gurion University of the Negev, Beer Sheva, Israel

ARTICLE INFO

Article history:
Received 16 July 2017
Received in revised form 6 August 2018
Accepted 7 August 2018
Available online 11 October 2018

Keywords:
DCOPs
Region Optimality
Privacy

ABSTRACT

Region-optimal algorithms are local search algorithms for solving Distributed Constraint Optimization Problems (DCOPs). In each iteration of the search in such algorithms, every agent selects a group of agents that comply with some selection criteria (each algorithm specifies different criteria). Then, the agent who selected the group, called the mediator, collects assignment information from the group and neighboring agents outside the group, in order to find an optimal set of assignments for its group's agents. A contest between mediators of adjacent groups determines which groups will replace their assignments in that iteration to the found optimal ones. In this work we present a framework called RODA (Region Optimal DCOP Algorithm) that encompasses the algorithms in the region optimality family, and in particular any method for selecting groups. We devise a secure implementation of RODA, called P-RODA, which preserves constraint privacy and partial decision privacy. Our discussion covers both symmetric and asymmetric DCOPs. The two main cryptographic means that enable this privacy preservation are secret sharing and homomorphic encryption. We estimate the computational overhead of P-RODA with respect to RODA and give an upper bound that depends on the group and domain sizes and on the graph topology, but not on the number of agents. The estimations are substantiated with experimental results, including experiments on a simulator, that compare the performance of P-RODA to that of its competing algorithm, P-Max-Sum.

1. Introduction

The Distributed Constraint Optimization Problem (DCOP) is a general model for solving distributed combinatorial problems that has a wide range of applications in artificial intelligence. Complete algorithms for DCOP-solving [1–3] are guaranteed to find the optimal solution, but because DCOPs are NP-hard, these algorithms' worst case runtime is exponential. Thus, there is a growing interest in incomplete algorithms, which may find sub-optimal solutions but run quickly enough to be applied on large-scale problems or real-time applications [4–7].

Local search algorithms, e.g., DSA [5] and DBA [8], are, in general, simple incomplete algorithms that were found empirically to produce high-quality solutions. The main disadvantage of these algorithms is that they do not offer guarantees on the quality of the solutions that they produce. Thus, in the last decade, researchers have developed solution concepts that

* Corresponding author.
E-mail address: tamirta@openu.ac.il (T. Tassa).
[1] Tal Grinshpoun and Tamir Tassa contributed equally to this paper.

https://doi.org/10.1016/j.artint.2018.08.002
0004-3702/

offer a balance in the trade-off between runtime efficiency (the time required to find a solution) and the guaranteed solution quality [9–13]. The guarantees are achieved by selecting a criterion according to which agents are grouped, and then producing solutions that are locally optimal in the following sense: there exists no better-quality solution that differs from the obtained solution only in the assignments of agents that are all contained in a single group. The criteria for selecting groups are defined by two parameters: k, an upper bound on the group's size, and t, an upper bound on the distance (i.e., the length of the shortest connecting path in the constraint graph) of agents in the group from the group's central agent, called the *mediator*. These two parameters define for each agent, serving as a mediator, a *region*, which is the collection of all maximal-sized connected groups of at most k agents that include the mediator and in which all agents are of distance at most t from the mediator [13]. Such solutions, termed *region optimal*, provide under certain conditions a guaranteed bound on the ratio between their quality and the quality of an optimal solution [12,13]. The use of a mediator in DCOP algorithms is also a main part of OptAPO [14], which is a complete algorithm. The difference is that the size of the groups of the mediators in OptAPO is not limited; the groups are generated ad-hoc as part of the algorithm and in the worst case they can include all the agents. There also exist other incomplete DCOP algorithms with different types of quality guarantees [15–17]. A detailed overview of DCOP algorithms, with special focus on those with quality guarantees, is given in Section 8.

Although the guarantees achieved by region-optimal solutions are important, they come with a cost. One cost is the increase in runtime, which is exponential in the size of the groups. The second is the loss of privacy, as region-optimal algorithms require agents to share their private information with nearby agents.

Privacy is one of the main motivations for solving constraint problems in a distributed manner. Preserving privacy is most important in distributed scenarios in which agents represent people that would not like their personal preferences and actions to be revealed, e.g., meeting scheduling [18], and smart environments (such as smart homes) [19,20]. The term privacy is quite broad, a fact that gave rise to several categorizations of the different types of privacy [21–23]. In this paper we relate to the categorization of Faltings et al. that distinguishes between agent privacy, topology privacy, constraint privacy, and decision privacy [21]. These notions of privacy are presented in Section 2.2.

Most studies that evaluated distributed constraint algorithms in terms of privacy considered complete algorithms [24–29]. Some work has focused on measuring the extent of constraint privacy loss [25,26]. Doshi et al. proposed to inject privacy-loss as a criterion to the problem solving process [27]. Some previous work was also directed towards reducing constraint privacy loss. Most efforts in the development of privacy-preserving search algorithms focused on DCSP, which is the *satisfaction* variant of DCOP. Examples include [30,24,31]. The work of Silaghi and Mitra [24] addressed both satisfaction and optimization problems. However, the proposed solution is strictly limited to small-scale problems since it depends on an exhaustive search over all possible assignments. Several privacy-preserving versions of the DPOP algorithm [2] were proposed in the past [22,32], including a more recent study by Léauté and Faltings that proposed several versions of DPOP that provide strong privacy guarantees [28]. While these versions are aimed for DCSPs, some of them may be also applicable to DCOPs. Another recent paper [29] devised a variant of SyncBB [33] that preserves topology, constraint, and decision privacy. An exception to the focus on securing complete algorithms is provided by the works of Tassa et al. [34,35] that devised a variation of an incomplete inference algorithm, Max-Sum [36], that preserves topology, constraint, and decision privacy (and even agent privacy provided that an external trusted coordinator is available).

Considering a different aspect of constraint privacy, researchers have addressed problems in which the nature of a constraint is distributed among the constrained agents. Solutions to such problems include the PEAV formulation [37] and *asymmetric* DCOPs [38]. Here, we discuss both the traditional symmetric DCOPs and asymmetric DCOPs.

Focusing on region optimality, the first attempt to produce a more secure region-optimal algorithm (actually k-optimal) was conducted by Greenstadt [39]. In that extended abstract, two methods for reducing the privacy loss in k-optimal algorithms were proposed. The first was by avoiding sending the information of more than k agents to a mediator (which is done in region-optimal algorithms that apply k-optimality). The second was that groups of agents find their optimal assignment in each iteration using a distributed algorithm instead of having the mediator find that assignment for the group. Obviously, incorporating the second improvement reduces the significance of the first one. While these ideas were clearly preliminary, we share the motivation and take herein a further step in that direction.

We describe here an algorithmic framework called RODA (Region Optimal DCOP Algorithm) that generalizes, for didactic purposes, existing local search algorithms that issue solutions with quality guarantees, e.g. the KOPT algorithm [9], or the DALO algorithm [10]. RODA is a new formalization rather than a new algorithm; it is an umbrella setup that generalizes the main existing region-optimal algorithms, and allows us to include in our study of privacy the region selection methods of all existing algorithms of the region optimality family [9,10,13].

We then proceed to present P-RODA, a privacy-preserving implementation of RODA. Hence, P-RODA includes a privacy-preserving implementation of a general region-optimal algorithm, which can implement the region traversing methods of KOPT, DALO, and any other algorithm from the region optimality family, and follow either a synchronous or an asynchronous operation mode. P-RODA is a perfect simulation of RODA, in the sense that given the same random choices (some of the algorithms that fall under the RODA framework use randomness), both RODA and P-RODA will go through the same sequence of intermediate assignments and will issue the same output after a given number of iterations. However, centralized computations in RODA that may leak private information are replaced in P-RODA with secure distributed computations that prevent such information leakage. Thus, our framework securely achieves k-size-optimality, t-distance-optimality, or any combination of the two, and finds solutions with the same guarantees on the distance from the optimum as proved in previous studies [10,12,13].

All existing region-optimal algorithms assumed the traditional DCOP setting in which all the constraints are symmetric. Transforming these solutions to deal with asymmetric constraints is rather straightforward. However, when trying to preserve constraint privacy, the move to an asymmetric setting brings with it new complexities, which we also handle.

P-RODA, in both symmetric and asymmetric settings, preserves constraint privacy and partial decision privacy. Constraint privacy is achieved by agents avoiding sharing constraint information with other agents and then finding the best improving assignment in each of the algorithm's iterations by applying a secure distributed protocol. Partial decision privacy is obtained by applying secure comparison protocols, instead of the usual practice of broadcasting the possible improvements by groups of agents.

Our empirical evaluation demonstrates the overhead required to achieve privacy in region-optimal algorithms and the size of groups for which our proposed method may be practically applied. We also conducted experiments on a simulator that revealed P-RODA's superior performance compared to P-Max-Sum [35], which is to the best of our knowledge the only existing alternative of an incomplete privacy-preserving DCOP algorithm.

The rest of the paper is organized as follows. After some preliminaries (Section 2), we describe in Section 3 the framework of Region Optimal DCOP Algorithms (RODA). Section 4 is the core of this study in which we introduce P-RODA, a privacy-preserving implementation of RODA. In that section we concentrate on symmetric DCOPs. Then, in Section 5 we consider asymmetric DCOPs and describe the necessary modifications in P-RODA that are needed for handling such problems. In Section 6 we discuss the theoretic properties of P-RODA, and in particular its privacy guarantees. In Section 7 we analyze the efficiency of P-RODA and experimentally evaluate its performance on a simulator. Finally, after reviewing some relevant work on DCOP algorithms in Section 8, we conclude in Section 9.

A preliminary version of this paper was published at IJCAI 2016 conference [40]. The present journal version extends the preliminary version by including a privacy-preserving implementation of RODA in asymmetric settings, what called for a thorough modification of the algorithms in the symmetric setting and introduction of new sub-protocols and techniques (Section 5). In addition, we present in the current version a much extended discussion, including complete proofs (Section 6). Finally, we provide in Section 7 an extended efficiency analysis and experimental evaluation, for both the symmetric and asymmetric settings, including experiments on a simulator that compare the performance of P-RODA to that of the P-Max-Sum algorithm.

2. Preliminaries

Herein we provide the necessary preliminaries. Section 2.1 includes the terminology and notations of the DCOP model, in both the symmetric and asymmetric settings. In Section 2.2 we present the privacy notions that we adopt herein.

2.1. Distributed Constraint Optimization

A Distributed Constraint Optimization Problem (DCOP) [33] 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 $X_1, X_2, \ldots, X_n$, and $\mathcal{D}$ is a set of finite domains $D_1, D_2 \ldots, D_n$. Each constraint $C \in \mathcal{R}$ defines a non-negative *cost* for every possible value combination of a set of variables, and is of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary constraint* refers to exactly two variables and is of the form $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$.[2] A *binary DCOP* is a DCOP in which all constraints are binary. Each variable X_i takes values in the domain D_i, and it is held by a single agent.

As done in most DCOP studies, we make the standard assumptions that each agent holds exactly one variable and that all constraints are binary. However, we note that there exist problems that include k-ary constraints with $k > 2$ and with more than one variable per agent. There exist few studies that take such scenarios into consideration, [41–44], but the vast majority of the literature on DCOPs concentrate on the case of binary constraints and one-to-one mappings between agents and variables.

In traditional symmetric DCOPs, a constraint $C_{i,j} \in \mathcal{R}$ defines a non-negative cost for every possible value combination of X_i and X_j, for some $1 \leq i < j \leq n$. Namely, for each of the $\binom{n}{2}$ possible pairs of agents, $1 \leq i < j \leq n$, $C_{i,j}$ is a nonnegative real function over $D_i \times D_j$, where, for each pair of values $(a_i, a_j) \in D_i \times D_j$, $C_{i,j}(a_i, a_j)$ is the cost incurred for both A_i and A_j if $X_i \leftarrow a_i$ and $X_j \leftarrow a_j$. On the other hand, in asymmetric DCOPs [38] the cost may be different for the two agents involved. Hence, in such asymmetric settings, $\mathcal{R}$ includes a constraint $C_{i,j}$ for each of the $n^2 - n$ ordered pairs of agents, $1 \leq i \neq j \leq n$, where $C_{i,j}(a_i, a_j)$ equals the cost incurred by A_i if $X_i \leftarrow a_i$ and $X_j \leftarrow a_j$, while $C_{j,i}(a_j, a_i)$ equals the cost incurred by A_j for that very same combination of assignments. In the bulk of this study we focus on symmetric DCOPs; later on we extend our discussion to include also asymmetric DCOPs. (We note that the case of unary constraints is covered by including in $\mathcal{R}$ also the "diagonal" terms, namely $C_{i,i}(a_i, a_i)$, which can be shortened to $C_i(a_i)$; those are the costs incurred by A_i alone if $X_i \leftarrow a_i$. The extension of our discussion to include also such unary constraints is straightforward and can be completed by the reader.)

[2] We say that a variable is *involved* in a constraint if it is one of the variables to which the constraint refers.

A *value assignment* is a pair including a variable and a value from that variable's domain. A *complete assignment* consists of value assignments to all variables in $\mathcal{X}$. The objective is to find a complete assignment of minimal cost.[3]

The *constraint graph* is a graph whose nodes are the n agents (or variables) where two nodes are connected by an edge if the two corresponding variables are constrained. We shall refer to such pairs of agents or variables as *neighbors*. For every agent A_i, $1 \leq i \leq n$, we let $N_t(A_i)$ denote the set of all agents whose distance to A_i in the constraint graph is at most t. The special case of the distance-1 neighborhood will be denoted $N(A_i) := N_1(A_i)$.

2.2. Privacy notions

Faltings et al. have distinguished between four notions of private information in DCSPs [21]. Those notions are also relevant in the context of DCOPs, whence we adopt them herein:

- *Agent privacy* – hiding from each agent the identity or even the existence of other agents with which it is not constrained.
- *Topology privacy* – hiding from each agent the topological structures in the constraint graph in which it is not involved. Namely, each agent should be aware of the nodes of variables with which its variable is constrained, and the edges that connect its node to those nodes, but nothing else on the graph structure.
- *Constraint privacy* – hiding from each agent the constraints in which it is not involved. Namely, agent A_k should not know anything about $C_{i,j}(\cdot,\cdot)$ if $k \notin \{i, j\}$.
- *Decision privacy* – hiding from each agent the final assignments to variables other than its own.

The above definition of constraint privacy relates to the traditional symmetric DCOP model. However, when the problem consists of asymmetric constraints, these constraints should be kept private even from other agents that are involved in them. Namely, agent A_j should not know anything about $C_{i,j}(\cdot,\cdot)$ if $A_i \neq A_j$. Grinshpoun termed this notion of privacy as *internal* constraint privacy [23].

Out of the four privacy notions of Faltings et al. [21], constraint privacy, has drawn the most attention in past research, see e.g. [24,25,27]. In fact, most of the privacy types in the categorization of Greenstadt et al. [22] are actually variations of constraint privacy.

3. Region optimal DCOP algorithms

Region optimality is a concept that is based on assigning agents to groups, such that each group of agents has one agent that acts as its mediator (the meaning of being a mediator is described below). For a given agent A_h, its region R_h is the collection of all groups of which A_h can serve as the mediator. Groups, and consequently regions, are commonly defined by two parameters, k and t:

Definition 1 *(Group).* A group for which agent A_h can act as a mediator is a subset of agents $B \subset \mathcal{A}$ such that: (a) $A_h \in B$, (b) $|B| \leq k$, (c) $B \subseteq N_t(A_h)$, (d) the constraint graph induced by B is connected, and (e) there exists no proper superset of B that complies with all previous conditions.

Definition 2 *(Region).* The region R_h of agent A_h is the collection of all possible groups for which A_h can act as the mediator.

Definition 3 *(Region-optimal solution).* Given a DCOP with n agents and the entire set of corresponding regions $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$, a region-optimal solution to this DCOP is a complete assignment whose cost cannot be reduced by changing the value assignments only to agents that are all included in a single group that belongs to some region in $\mathcal{R}$.

Example. Consider a DCOP with $n = 5$ agents and the constraint graph as shown in Fig. 1. Given the parameters $k = 3$ and $t = 2$, the region $R_1 \in \mathcal{R}$ of agent A_1 includes three groups as follows,

$$R_1 = \{\{A_1, A_2, A_3\}, \{A_1, A_2, A_4\}, \{A_1, A_2, A_5\}\}.$$

Let $\mathbf{a} := (a_1, a_2, a_3, a_4, a_5)$ be a solution to the DCOP. If it is region optimal with respect to R_1, it means that:

1. There exists no tuple $(b_1, b_2, b_3) \in D_1 \times D_2 \times D_3$ so that $(b_1, b_2, b_3, a_4, a_5)$ has a smaller cost than $\mathbf{a}$.
2. There exists no tuple $(b_1, b_2, b_4) \in D_1 \times D_2 \times D_4$ so that $(b_1, b_2, a_3, b_4, a_5)$ has a smaller cost than $\mathbf{a}$.
3. There exists no tuple $(b_1, b_2, b_5) \in D_1 \times D_2 \times D_5$ so that $(b_1, b_2, a_3, a_4, b_5)$ has a smaller cost than $\mathbf{a}$.

The region optimality of $\mathbf{a}$ also applies to each of the groups in the other regions R_j, $j = 2, 3, 4, 5$.

[3] We follow the standard definition of DCOPs as minimization problems. Nevertheless, all methods and results that we present here apply to maximization problems as well, if we replace the constraint functions $C_{i,j}(\cdot,\cdot)$ with $C - C_{i,j}(\cdot,\cdot)$, where $C := \max_{i,j} \max_{a_i \in D_i, a_j \in D_j} C_{i,j}(a_i, a_j)$.

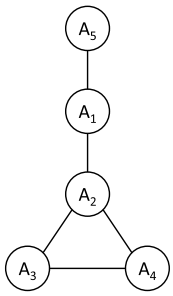

Fig. 1. An example of a DCOP constraint graph.

Protocol 1 describes RODA, an algorithmic framework that generalizes existing region-optimal algorithms [9,10,13]. We proceed to explain it in detail. Hereinafter, when we speak of a general agent we denote it by A_i; however, when we speak of that agent as a mediator we denote it by A_h, $1 \leq i, h \leq n$.

RODA starts with an initialization phase (Step 1), in which each agent A_h gathers information from agents in its t-distance neighborhood $N_t(A_h)$; the collected information is needed for A_h to determine its region R_h (Definition 2), and later on to function as the mediator. That information includes: (a) the domains of variables controlled by agents in that neighborhood; and (b) the constraints that agents in that neighborhood have with all their neighbors (these constraints can involve agents of distance at most $t+1$ from A_h).

After the initialization phase ends, the local search phase starts. First, each agent selects an initial assignment to its variable (Step 2). Then, the algorithm performs a fixed number L of improvement iterations. In the ℓth iteration (Steps 3–10), some of the agents will update their current assignment from $a_i^{\ell-1}$ to a_i^{ℓ} towards reducing the overall cost. After all agents initiate the next assignment to be like the current one (Step 4), every agent A_h selects a group $\mathcal{A}_h^{\ell}$ from its region R_h (Step 5); the selection can be random or one that follows some systematic rule. In the next steps A_h will act as the mediator of the selected group.

In Step 6, each mediator A_h gathers the current assignments of all agents in $\mathcal{A}_h^{\ell}$ as well as of their direct neighbors (some of whom could be outside $\mathcal{A}_h^{\ell}$). With that information, A_h finds the locally best assignments β for the agents in its group $\mathcal{A}_h^{\ell}$ (Step 7). Then, A_h computes the improvement Δ_h^{ℓ} in the overall cost, in case the current assignments in its group are replaced with the found locally best ones (Step 8). It broadcasts its findings (β and Δ_h^{ℓ}) to its group (Step 9). Later on we explain in detail how A_h computes β for its group and the corresponding Δ_h^{ℓ}.

Next, a contest takes place between neighboring groups (groups that include agents that are neighbors).[4] Every group which is a local winner (namely, the improvement that it offers is greater than the improvement offered by any of its neighboring groups, where ties are broken by the groups' indices) updates its current assignments to the found local optimal ones β (Step 10). After conducting the preset number of iterations L, the agents set their variables to the last assignment found (Step 11).

Existing region-optimal algorithms [9,10,13] differ mainly in two aspects: the criteria that define the regions, and the process of converging to region-optimal solutions. Regarding the first aspect, the KOPT algorithm [9] refers only to the k-size criterion, the DALO algorithm [10] was designed to work with either k-size or t-distance criteria, whereas Vinyals et al. combine these criteria by introducing the notion of $\mathcal{C}$-optimality in which both parameters are considered [13].

Note that, for the sake of simplicity of presentation, our description of RODA (and P-RODA) adopts the synchronous operation of KOPT [9]. The adaptation to an asynchronous operation as in DALO [10] can be achieved by adding a lock-commit mechanism. This mechanism validates for a group that is about to replace its assignment that all nodes in the group and their neighbors are committed to the assignment replacement of this specific group, and will refuse to commit to an assignment replacement of any other group until the replacement they have committed to is complete.

We now revisit Steps 7 and 8 and provide the necessary technical details. Let:

- $D_h^{\ell} := \prod_{A_i \in \mathcal{A}_h^{\ell}} D_i$ be the Cartesian product of the domains of all variables of $\mathcal{A}_h^{\ell}$, and $N_h^{\ell} := \prod_{A_j \in (\bigcup_{A_i \in \mathcal{A}_h^{\ell}} N(A_i)) \setminus \mathcal{A}_h^{\ell}} D_j$ be the Cartesian product of the domains of all variables that correspond to agents outside $\mathcal{A}_h^{\ell}$ that have neighbors in $\mathcal{A}_h^{\ell}$.
- $\beta_h^{\ell-1} \in D_h^{\ell}$ be the current partial assignment to the variables controlled by $\mathcal{A}_h^{\ell}$, and $\alpha_h^{\ell-1} \in N_h^{\ell}$ be the partial assignment that was received by the agents in $\mathcal{A}_h^{\ell}$ from their neighbors outside $\mathcal{A}_h^{\ell}$. ($\beta_h^{\ell-1}$ and $\alpha_h^{\ell-1}$ consist of the assignments that A_h retrieves in Step 6.) Then, A_h finds a tuple $\beta \in D_h^{\ell}$ that minimizes the local cost. In the case of a symmetric DCOP, the local cost is given by

[4] It is possible that two groups will be the same, namely that two mediators A_h and A_k chose in the ℓth iteration equal groups, $\mathcal{A}_h^{\ell} = \mathcal{A}_k^{\ell}$. Those two groups are of course neighboring, but as their equality implies that $\Delta_h^{\ell} = \Delta_k^{\ell}$, no contest takes place between them. In fact, redundant groups that are contained in other groups or are identical to them are eliminated in DALO [10].

$$\hat{C}(\beta, \alpha_h^{\ell-1}) := \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \cap \mathcal{A}_h^\ell, j>i} C_{i,j}(\beta|_{D_i}, \beta|_{D_j}) + \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} C_{i,j}(\beta|_{D_i}, \alpha_h^{\ell-1}|_{D_j}); \quad (1)$$

here, the notation $\beta|_{D_i}$ denotes the D_i-entry of the tuple β. The first term on the right-hand side of Eq. (1) is the sum of constraints that involve two agents inside the group $\mathcal{A}_h^\ell$. The second term is the sum of constraints that involve one agent in $\mathcal{A}_h^\ell$ and one external agent. In case of a tie, A_h selects β randomly from all tuples in D_h^ℓ that minimize $\hat{C}(\beta, \alpha_h^{\ell-1})$. In the case of an asymmetric DCOP, the local cost is given by

$$\hat{C}(\beta, \alpha_h^{\ell-1}) := \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \cap \mathcal{A}_h^\ell, j \neq i} C_{i,j}(\beta|_{D_i}, \beta|_{D_j}) + \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} \left(C_{i,j}(\beta|_{D_i}, \alpha_h^{\ell-1}|_{D_j}) + C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta|_{D_i}) \right). \quad (2)$$

Note that the first summation on the right-hand side of Eq. (2) includes all *ordered* pairs of agents inside the group, while in Eq. (1) it was sufficient to sum only over all *non-ordered* pairs, due to the symmetry of the constraints there. Similarly, the second summation on the right-hand side of Eq. (2) includes two terms – one for the cost of the internal agent and one for the cost of its external neighbor.

Finally, given the found optimal tuple β, A_h computes

$$\Delta_h^\ell := \hat{C}(\beta_h^{\ell-1}, \alpha_h^{\ell-1}) - \hat{C}(\beta, \alpha_h^{\ell-1}) \geq 0, \quad (3)$$

which is the improvement in the local cost for $\mathcal{A}_h^\ell$ if the current local partial assignment $\beta_h^{\ell-1}$ is replaced with β.

Protocol 1 RODA.

1: The agents exchange local information (see details in the text).
2: A_i randomly selects $a_i^0 \in D_i$, $1 \leq i \leq n$.
3: **for** $\ell = 1, \ldots, L$ **do**
4: A_i sets $a_i^\ell = a_i^{\ell-1}$, $1 \leq i \leq n$.
5: A_h selects $\mathcal{A}_h^\ell \in R_h$, $1 \leq h \leq n$.
6: A_h receives $a_j^{\ell-1}$ from A_j for all $A_j \in \bigcup_{A_i \in \mathcal{A}_h^\ell} N(A_i)$.
7: A_h finds a locally optimal tuple of assignments, $\beta \in D_h^\ell := \prod_{A_i \in \mathcal{A}_h^\ell} D_i$, for the variables controlled by its group $\mathcal{A}_h^\ell$.
8: A_h computes Δ_h^ℓ, the cost improvement if all agents in $\mathcal{A}_h^\ell$ update their assignment to the one given by β.
9: A_h informs all agents in $\mathcal{A}_h^\ell$ about the found β and Δ_h^ℓ.
10: For every $1 \leq h \leq n$: If $\mathcal{A}_h^\ell$ wins the contest against $\mathcal{A}_{h'}^\ell$ for all groups $\mathcal{A}_{h'}^\ell$ that are neighboring to $\mathcal{A}_h^\ell$, then $\forall A_i \in \mathcal{A}_h^\ell$, A_i sets a_i^ℓ according to β. (Winning occurs if $\Delta_h^\ell > \Delta_{h'}^\ell$ or if $\Delta_h^\ell = \Delta_{h'}^\ell$ and $h < h'$.)
11: A_i sets $X_i := a_i^L$, $1 \leq i \leq n$.

3.1. Convergence guarantees

Although RODA is an abstract generalization of existing region-optimal algorithms (e.g., KOPT and DALO), we provide here an intuitive explanation for how it guarantees convergence to region-optimal solutions, based on the proofs in [10,12,13].

In every iteration of the algorithm, each mediator proposes an improved assignment, if exists, which is optimal with respect to the current state for a group in its region. Since no two neighboring groups get to update their assignments in the same iteration, each assignment change by a group improves the global cost of the solution to the entire problem [12]. Obviously, the number of such improvements is finite. Thus, as long as after each improvement all groups that apply to the restrictions that define the region are considered (i.e., each of the region's groups gets the chance to replace their assignment if a better assignment to that group exists), the algorithm is guaranteed to converge to a region-optimal solution. Hence, the way in which the mediator selects a group from its region in each iteration constitutes an important aspect of the algorithm that determines its convergence to a region-optimal solution. This selection method must allow all groups to be considered. In KOPT this selection is random, thus, the guarantee to consider all groups is in infinite time [12], while in DALO the selection is systematic whence in the $\max_{1 \leq h \leq n} |R_h|$ iterations following each assignment replacement all groups will be considered.

4. Private RODA

In this section we explain how to execute the RODA algorithm in a privacy-preserving manner. To that end, we follow the algorithmic flow of Protocol 1 and explain, for each step in that protocol, how to execute it in a privacy-preserving manner. We make the standard assumption (e.g. [28,29,35]) that the agents are semi-honest, which means that they respect the protocol and do not form coalitions; but at the same time they examine the messages that they receive during the execution of the protocol and try to extract from them inferences on private information of other agents.

For the sake of simplicity, we focus here on the case of symmetric DCOPs. Later on, in Section 5, we describe the needed modifications in the protocol that we present herein for the sake of asymmetric DCOPs.

The basic RODA algorithm is inconsistent with privacy requirements. Indeed, the information gathering that is carried out in Step 1 violates agent, topology and constraint privacy. Such information gathering is essential in RODA in order to allow each mediator to carry out the central computations in Steps 7 and 8. The main effort in transforming RODA into a private algorithm, to which we refer hereinafter by P-RODA, lies in converting the centralized computations that are carried out by the mediator into distributed computations that are performed by all agents in the group using secure multi-party protocols. The replacement of those centralized computations in RODA with secure multi-party protocols in P-RODA allows us to reduce the information exchange in Step 1 (and also the one that is performed in Step 6). Specifically, in RODA, in order to allow the mediator to perform the centralized computations for its group, it had to receive in the initialization phase (Step 1) and then in each iteration (Step 6) private information from agents that are at distance at most $t+1$ from it. As such exchange of data is no longer needed in P-RODA, we replace it with a much reduced exchange of data. In Section 4.2 we describe the privacy-preserving modifications to Protocol 1 that are done in P-RODA. Before that, we describe in Section 4.1 the main cryptographic tool that P-RODA utilizes.

4.1. Cryptographic assumptions

A cipher is called public-key if its encryption function $E(\cdot)$ depends on one key, $\mathbf{k}_e$, which is publicly known, while the corresponding decryption function $E^{-1}(\cdot)$ depends on a private key $\mathbf{k}_d$ that is known only to the owner of the cipher, and $\mathbf{k}_d$'s derivation from $\mathbf{k}_e$ is computationally hard.

A public-key cipher is called (additively) *homomorphic* if its plaintext domain, $\mathcal{G}_P$, is an additive commutative group, its ciphertext domain, $\mathcal{G}_C$, is a multiplicative commutative group, and for every two plaintexts, $x_1, x_2 \in \mathcal{G}_P$, $E(x_1 + x_2) = E(x_1) \cdot E(x_2)$.

When the encryption function is randomized (in the sense that $E(x)$ depends on x as well as on a random string), E is called *probabilistic*. Hence, a probabilistic encryption function is a one-to-many mapping (every plaintext $x \in \mathcal{G}_P$ has many encryptions $y = E(x) \in \mathcal{G}_C$), while the corresponding decryption function is a many-to-one mapping (all possible encryptions $y \in \mathcal{G}_C$ of the same plaintext $x \in \mathcal{G}_P$ are mapped by $E^{-1}(\cdot)$ to the same x). When encryption is applied on a small dictionary of possible plaintexts, it is essential to use a probabilistic encryption in order to prevent a known-plaintext attack. In addition, probabilistic encryption does not allow to infer from $E(x)$ and $E(y)$ whether $x = y$.

The semantically secure Paillier cipher [45] is a public-key cipher that is both homomorphic and probabilistic. Its plaintext and ciphertext domains are $\mathcal{G}_P = \mathbb{Z}_\nu$ and $\mathcal{G}_C = \mathbb{Z}_{\nu^2}$, respectively, where the so-called modulus ν is a product of two large primes, p and q.

In what follows we assume that every mediator A_h creates for itself a key pair in such a public-key homomorphic and probabilistic cipher (e.g. Paillier) and that it notifies all agents in its group of the public key in that cipher. We shall denote the encryption function in that cipher by $E_h(\cdot) : \mathcal{G}_P \to \mathcal{G}_C$ and the corresponding decryption by $E_h^{-1}(\cdot) : \mathcal{G}_C \to \mathcal{G}_P$.

4.2. Privacy-preserving implementations of sensitive steps in RODA

Herein we provide explanations only on those steps in Protocol 1 that have to be modified in P-RODA; steps that we do not mention below remain the same.

• **Step 1 (Initialization).** This phase is performed in P-RODA in a reduced manner. Every agent A_h, $1 \leq h \leq n$, learns only topological information in the form of the restriction of the constraint graph to $N_t(A_h)$. This is sufficient for A_h to determine its region R_h. All the other information that A_h collects in this phase in RODA from its neighbors – the domains of variables for all agents in $N_t(A_h)$, and the constraint information relating to each pair of agents of which at least one agent is from $N_t(A_h)$ – is not needed in P-RODA. As explained earlier, the excessive exchange of information is not needed in P-RODA because the centralized computations that A_h did as a mediator in RODA are replaced by secure multi-party protocols.

• **Step 6 (Assignment propagation).** In RODA the mediator needed to perform centralized computations, and it therefore collected in this step assignment data from all agents in its group and from their neighbors. Contrary to that, the secure multi-party computations that are performed in P-RODA only require agents to collect assignment data from direct neighbors that are outside the group. Consequently, every agent A_i sends its current assignment $a_i^{\ell-1}$ only to its direct neighbors $(N(A_i) \setminus \{A_i\})$.

• **Step 7 (Locally optimal assignments).** This step includes the computations that are hardest to perform securely; hence, we postpone the discussion of their private execution to Section 4.3. The sub-protocol described in Section 4.3 ends with

all agents in the group learning the currently optimal tuple β; therefore, that protocol accomplishes Step 7 and part of Step 9 in RODA (the part in which all agents in the group are notified of the currently optimal β). In P-RODA, as opposed to RODA, it is essential to have all agents in the group know β before proceeding to computing Δ_h^ℓ in Step 8, because that computation is done in P-RODA by a distributed protocol involving all agents.

• **Steps 8–9 (Local improvements).** While in RODA it was the mediator who computed Δ_h^ℓ, and then informed all agents in its group about the result, in P-RODA all agents in the group compute it jointly and privately using a secure summation protocol. Consider the local cost $\hat{C}(\beta, \alpha_h^{\ell-1})$, as defined in Eq. (1); it can be broken into the following sum,

$$\hat{C}(\beta, \alpha_h^{\ell-1}) = \sum_{A_i \in \mathcal{A}_h^\ell} \gamma_i(\beta, \alpha_h^{\ell-1}), \tag{4}$$

where

$$\gamma_i(\beta, \alpha_h^{\ell-1}) := \sum_{A_j \in N(A_i) \cap \mathcal{A}_h^\ell, j>i} C_{i,j}(\beta|_{D_i}, \beta|_{D_j}) + \sum_{A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} C_{i,j}(\beta|_{D_i}, \alpha_h^{\ell-1}|_{D_j}). \tag{5}$$

Now, since by Eq. (3)

$$\Delta_h^\ell := \hat{C}(\beta_h^{\ell-1}, \alpha_h^{\ell-1}) - \hat{C}(\beta, \alpha_h^{\ell-1}),$$

we infer from Eq. (4) that

$$\Delta_h^\ell = \sum_{A_i \in \mathcal{A}_h^\ell} \Delta_h^\ell(i),$$

where

$$\Delta_h^\ell(i) := \gamma_i(\beta_h^{\ell-1}, \alpha_h^{\ell-1}) - \gamma_i(\beta, \alpha_h^{\ell-1})$$

is a value known to $A_i \in \mathcal{A}_h^\ell$; indeed, A_i knows β from Step 7 and it also knows the relevant components of $\beta_h^{\ell-1}$ and $\alpha_h^{\ell-1}$ from the reduced Step 6 in P-RODA. (Note that in order to compute $\Delta_h^\ell(i)$ it is sufficient to know the components of $\beta_h^{\ell-1}$ and $\alpha_h^{\ell-1}$ that relate to direct neighbors of A_i.) Therefore, Δ_h^ℓ can be computed by all agents in $\mathcal{A}_h^\ell$ using a secure summation protocol, such as the information-theoretic secure protocol of Benaloh [46].

Secure summation protocols usually reveal the final sum to at least one of the agents. Alas, revealing the final sum to an agent might hinder constraint privacy. Indeed, the final sum equals a linear combination of constraint values. An agent that knows all of the assignments to all variables in that linear combination will be able to obtain a linear equation in unknown constraint values. By collecting several such linear equations, that agent might be able to infer information on constraint values that should be kept secret.

To prevent such a potential violation of constraint privacy, we perform the secure summation protocol with a small "twist" so that instead of issuing the final sum Δ_h^ℓ, it issues two shares, denoted s_h and s'_h, that distribute uniformly at random on $\mathbb{Z}_S$ for some large integer S, and $\Delta_h^\ell = s_h + s'_h \mod S$. (The secure protocol of Benaloh [46] can be easily modified that way; we omit further details as they are straightforward.) The mediator will get s_h while all other members of its group will get s'_h. Using such secret sharing implies that, under the semi-honesty assumption, none of the agents learns *any* information on Δ_h^ℓ.

For the sake of the next step, it is necessary to note that all mediators must select the same large integer S. That selection can be made once at the outset of the protocol.

• **Step 10 (Contest).** In RODA, the difference Δ_h^ℓ was known to the mediator; hence, it was easy for mediators of neighboring groups to compare those values in order to determine which of the groups are winners in this iteration (and, consequently, get to update their local assignments). In P-RODA, on the other hand, the value of Δ_h^ℓ is not known to any single agent in the group $\mathcal{A}_h^\ell$; as explained above, Step 7 ends with the mediator holding a random value $s_h \in \mathbb{Z}_S$, while every other agent in the group gets to hold $s'_h = \Delta_h^\ell - s_h \mod S$. Therefore, we proceed to describe the manner in which the contest of Step 10 can be carried out in P-RODA.

First, the agents in each group $\mathcal{A}_h^\ell$ find the set of all neighboring groups. Specifically, let $A_i \in \mathcal{A}_h^\ell$ be an agent in the group. A_i requests from each of its neighbors, $A_j \in N(A_i)$, the list L_j^ℓ of groups to which the latter agent A_j belongs in the current ℓth iteration. Then A_i unifies those lists into $L_{h,i}^\ell := \bigcup_{A_j \in N(A_i)} L_j^\ell$. The latter list, $L_{h,i}^\ell$, holds the indices of all groups that are neighbors to $\mathcal{A}_h^\ell$ through A_i. Finally, all agents in the group compute $\mathbf{L}_h^\ell := \bigcup_{A_i \in \mathcal{A}_h^\ell} L_{h,i}^\ell$.

Consequently, each group $\mathcal{A}_h^\ell$ engages in a contest vis-a-vis every group $\mathcal{A}_m^\ell \in \mathbf{L}_h^\ell$ with $m > h$. (Recall that, as explained in Section 3, a contest is held only between neighboring groups that are different. If $\mathcal{A}_m^\ell \in \mathbf{L}_h^\ell$ but $\mathcal{A}_m^\ell = \mathcal{A}_h^\ell$ then the contest is not carried out since it is pointless.) The contest is performed by carrying out the following steps:

- A_h and A_m select "deputies" in their respective groups, $A'_h \in \mathcal{A}^\ell_h \setminus \{A_h\}$ and $A'_m \in \mathcal{A}^\ell_m \setminus \{A_m\}$. The selection is made so that at least one of the four agents A_h, A'_h, A_m, A'_m will not be in $\mathcal{A}^\ell_h \cap \mathcal{A}^\ell_m$. It is possible to do so since a contest is held only between two groups that are different. Let us assume for now that A'_m is an agent that does not belong to both $\mathcal{A}^\ell_h$ and $\mathcal{A}^\ell_m$; later on we explain how to modify the steps that we describe below if the above assumption does not hold.
- We recall that A_h and A'_h hold two shares, s_h and s'_h respectively, so that $\Delta^\ell_h = s_h + s'_h \mod S$; similarly, A_m and A'_m hold two shares, s_m and s'_m respectively, so that $\Delta^\ell_m = s_m + s'_m \mod S$. The modulus S is the same for both groups, as we pointed out earlier.
- A_h sends its share in Δ^ℓ_h, s_h, to A_m.
- A_m sends $(s_h - s_m) \mod S$ to A'_h.
- A'_h sends $(s_h - s_m + s'_h) \mod S$ to A'_m.
- A'_m computes $(s_h - s_m + s'_h - s'_m) \mod S$. This value equals $\left(\Delta^\ell_h - \Delta^\ell_m\right) \mod S$.
- A'_m infers from the last difference whether $\Delta^\ell_h \geq \Delta^\ell_m$ or not. (To allow such inference, the modulus S must be sufficiently large, and in particular greater than twice the upper bound on $\Delta^\ell_h, \Delta^\ell_m$. With such a selection of S, it is easy to see that $\Delta^\ell_h \geq \Delta^\ell_m$ if and only if $\left(\Delta^\ell_h - \Delta^\ell_m\right) \mod S$ is smaller than $S/2$.) If $\Delta^\ell_h \geq \Delta^\ell_m$, A'_m informs A_h who proceeds to inform all agents in $\mathcal{A}^\ell_h$ that their group won. Otherwise, A'_m informs all agents in $\mathcal{A}^\ell_m$ that their group won. (Notice that, as in RODA, in case of a tie, the group with the lower index wins.)

If $A'_m \in \mathcal{A}^\ell_h \cap \mathcal{A}^\ell_m$ then at least one of the other agents A_h, A'_h, A_m must be outside of $\mathcal{A}^\ell_h \cap \mathcal{A}^\ell_m$. In that case we can re-order the sequence of information exchanges described above so that the final difference $\left(\Delta^\ell_h - \Delta^\ell_m\right) \mod S$ is recovered by that agent. Namely, we can always guarantee that the final difference is revealed to an agent that does not belong to both groups. This condition is required for the sake of privacy preservation (see Theorem 4).

4.3. Private computation of the best partial assignment for a group of agents

Here we discuss the privacy-preserving execution of Step 7 in which the agents within each group need to solve a local DCOP. One possibility to execute this step in a privacy-preserving manner would be to invoke an existing privacy-preserving complete algorithm for that purpose: Grinshpoun and Tassa [47,29] proposed P-SyncBB, a privacy-preserving version of the SyncBB algorithm, while Léauté and Faltings [28] proposed three privacy-preserving versions of DPOP – P-DPOP$^{(+)}$, P$^{3/2}$-DPOP$^{(+)}$, and P^2-DPOP$^{(+)}$.

One may implement any of those above mentioned algorithms in the framework of P-RODA for a privacy-preserving execution of Step 7 (after a simple modification in order to take into account also constraints vis-a-vis agents outside the group). However, all these algorithms have their shortcomings: P-SyncBB incurs a large number of communication rounds and messages; P-DPOP$^{(+)}$ and P$^{3/2}$-DPOP$^{(+)}$ ensure only partial constraint privacy; and P^2-DPOP$^{(+)}$ is very inefficient (in terms of runtime). Hence, we proceed to describe here a novel protocol for the secure solution of the local DCOP within a group. The protocol can be executed efficiently only when the Cartesian product of the domains of the group's variables, D^ℓ_h, is not too large. Thus, we suggest to use it whenever feasible, otherwise one should use one of the above mentioned algorithms.

4.3.1. A statement of the computational problem

Given $\alpha^{\ell-1}_h \in N^\ell_h$, the partial assignment that was received by the agents in $\mathcal{A}^\ell_h$ from their neighbors outside $\mathcal{A}^\ell_h$ (Step 6 in P-RODA), the agents in $\mathcal{A}^\ell_h$ wish to compute a tuple

$$\beta \in D^\ell_h = \prod_{A_i \in \mathcal{A}^\ell_h} D_i \tag{6}$$

that minimizes the local cost $\hat{C}(\beta, \alpha^{\ell-1}_h)$, Eq. (1), in a privacy-preserving manner. (Note that each agent in $\mathcal{A}^\ell_h$ only knows those components in $\alpha^{\ell-1}_h$ that correspond to its direct neighbors outside the group.)

4.3.2. Arranging the partial costs in a tensor

For simplicity, let us assume that $\mathcal{A}^\ell_h = \{A_1, A_2, \ldots, A_k\}$. Assume further that each of the domains D_j, $1 \leq j \leq n$, is publicly ordered: $D_j = \{v^j_0, v^j_1, \ldots, v^j_{|D_j|-1}\}$. Then each $A_i \in \mathcal{A}^\ell_h$ can construct a private k-dimensional tensor G_i such that for any multi-index $[i_1, \ldots, i_k]$, where $0 \leq i_j \leq |D_j| - 1$, $1 \leq j \leq k$, the corresponding entry in the tensor is

$$G_i[i_1, i_2, \ldots, i_k] := \gamma_i(\beta, \alpha^{\ell-1}_h)\,,$$

where $\beta = (v^1_{i_1}, \ldots, v^k_{i_k})$, and γ_i is as defined in Eq. (5). According to Eq. (4), the sum of these tensors is a tensor G in which

$$G[i_1, i_2, \ldots, i_k] := \sum_{A_i \in \mathcal{A}^\ell_h} G_i[i_1, i_2, \ldots, i_k] = \hat{C}(\beta, \alpha^{\ell-1}_h)$$

for $\beta = (v^1_{i_1}, \ldots, v^k_{i_k})$. To summarize: each of the agents $A_i \in \mathcal{A}^\ell_h$ holds a private tensor G_i. They jointly wish to compute the multi-index of the entry in $G = \sum_{i=1}^k G_i$ which is minimal. That multi-index reveals the required β. The protocol presented next performs that.

4.3.3. The protocol

The protocol goes as follows:

1. The mediator A_h selects a deputy $A'_h \in \mathcal{A}^\ell_h$.
2. Each agent A_i, $1 \leq i \leq k$, sends to A'_h the component-wise encryption of its private tensor $E_h(G_i)$, where E_h is as described in Section 4.1.
3. A'_h computes the component-wise product $\prod_{i=1}^k E_h(G_i)$. Owing to the homomorphic property, A'_h gets as a result the encrypted tensor $E_h(\sum_{i=1}^k G_i) = E_h(G)$.
4. A'_h selects a random $c \in \mathcal{G}_C$ (namely, a random ciphertext, see Section 4.1) and computes $c \cdot E_h(G)$. Since $c = E_h(r)$ for some random $r \in \mathcal{G}_P$, it holds that $c \cdot E_h(G) = E_h(r) \cdot E_h(G) = E_h(G + r)$ (the multiplication with the scalar c and the addition with the scalar r are component-wise).
5. A'_h selects a secret and random permutation π on $D^\ell_h = \prod_{i=1}^k D_i$ and then sends to the mediator A_h the permuted tensor $\pi(E_h(G + r)) = E_h(\pi(G + r))$.
6. A_h decrypts and recovers the tensor $\pi(G + r)$.
7. A_h informs A'_h of the position in $\pi(G + r)$ of the minimal entry.
8. A'_h uses π^{-1} to recover the original multi-index $[i_1', i_2', \ldots, i_k']$ of the minimal entry, which reveals the new $\beta = (v^1_{i_1'}, \ldots, v^k_{i_k'})$. A'_h informs all group members about β.

A note about finding the minimal entry: We view the entries of G (and $\pi(G)$) as integers, as they describe local costs of tuples in D^ℓ_h. We wish to find a minimal entry in $\pi(G)$. However, the entries in $\pi(G + r) = \pi(G) + r$ are the result of addition modulo ν, and not an addition of integers. (Recall that the plaintext domain in Paillier cipher is $\mathcal{G}_P = \mathbb{Z}_\nu$, see Section 4.1.) So such an addition may create a wrap around which, usually, will prevent A_h from finding a minimal entry in $\pi(G)$. However, as ν is typically a very large integer, and in particular larger than twice the maximal entry in $\pi(G)$, A_h can infer from $\pi(G) + r$ the location of a minimal entry in $\pi(G)$, when those are viewed as integers. Indeed, A_h can sort the entries in $\pi(G) + r$ in a monotonically non-decreasing order, $0 \leq g_1 \leq \cdots \leq g_M < \nu$, where M denotes here the number of entries in G. Then it may look for the maximal difference modulo ν between two successive values in that sequence; it is easy to see that those differences are $\{g_2 - g_1, g_3 - g_2, \ldots, g_M - g_{M-1}, g_1 - g_M + \nu\}$. If the largest difference is $g_1 - g_M + \nu$ then there is no wrap around and then g_1 is the minimal entry; if, however, $g_{i+1} - g_i$ is the largest difference for some $1 \leq i \leq M - 1$ then g_{i+1} is the minimal entry.

4.3.4. The case of more than one optimal tuple

If there exists exactly one tuple that minimizes the local cost for the group, then all manners of computing that tuple will find the same tuple. Namely, a centralized computation as done in the non-privacy-preserving RODA, a distributed and privacy-preserving manner as described in the first paragraph of Section 4.3 (say, by P-SyncBB), or the privacy-preserving protocol that is presented in Section 4.3.3 – all will issue the same output tuple β. In case there is more than one optimal tuple, RODA does not specify which one to choose, but all of its provable properties remain indifferent to the way in which the selection is made in such cases.

One possible approach to specify the selection of an optimal tuple in case there is more than one optimal tuple is as follows. One may define upfront an order on all variables and an order on each of the variable domains; such orders will induce a lexicographical order on the set of all possible tuples for any given group. Then, one may adopt the strategy of always selecting the optimal tuple which is minimal with respect to that lexicographical order. In order to support that strategy in the protocol in Section 4.3.3, then in Step 7 in that protocol, instead of A_h informing A'_h of the position in $\pi(G + r)$ of one minimal entry, A_h will inform A'_h of the position of all minimal entries. Then, in Step 8, A'_h will choose from among those entries the one which is minimal by the lexicographical order.

4.4. A concluding remark about synchronicity

We chose to present a synchronous version of RODA and P-RODA. However, P-RODA can be easily implemented also in an asynchronous mode, like DALO. Instead of performing comparisons between neighboring groups in order to decide who "wins", as we do in the synchronous case, in the asynchronous case it is needed to successfully lock all agents in the group in order to allow it to update its assignments. To do that, the mediator can send a locking request to all agents in its group and only if it gets an approval from all of them it can broadcast to all agents a message that allows them to update their assignment to the newly found one. Such communication can be carried out in the clear. However, for increased privacy it is possible to perform the poll regarding the locking status in a private manner: each agent in the group will hold a flag indicating whether it is still unlocked. The mediator can initiate a protocol, such as in [48], for privately computing the AND of those bits. If the result shows that all group agents are still unlocked, they all proceed to update their assignment.

5. Private RODA for asymmetric DCOPs

Asymmetric DCOP [38] is a model that generalizes the standard symmetric DCOP model by allowing different agents that share a constraint to have different valuations regarding that constraint. This asymmetry in the valuation of a shared constraint (interaction) is standard in game theory and in many multi-agent problems. Several researchers suggested to model various multi-agent scenarios as DCOPs with asymmetric constraints. These include distributed meeting scheduling [37], distributed resource allocation [49], charging of electric vehicles [50], negotiation of combinatorial problems [51], and scheduling of devices in smart homes [52]. Asymmetric DCOPs have also been used in game theory for finding appropriate incentive mechanisms, such as taxation [53] and side payments [54].

In the context of P-RODA, the case of asymmetric DCOPs calls for a special attention because the cost of a tuple of group assignments, $\beta \in D_h^\ell$, is given by Eq. (2) instead of Eq. (1). Let us recall Eq. (1) which spells out the cost of a tuple β in the case of symmetric constraints:

$$\hat{C}(\beta, \alpha_h^{\ell-1}) := \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \cap \mathcal{A}_h^\ell, j>i} C_{i,j}(\beta|_{D_i}, \beta|_{D_j}) + \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} C_{i,j}(\beta|_{D_i}, \alpha_h^{\ell-1}|_{D_j}) .$$

The above cost $\hat{C}(\beta, \alpha_h^{\ell-1})$ could be computed in the symmetric case solely by the agents in the group, since each term in that sum is known to at least one agent in the group: each term in the first summation on the right-hand side above is known to both A_i and A_j, who are neighboring agents within the group; each term in the second summation is known to A_i, because the cost $C_{i,j}$ is symmetric and thus it is known by both agents that participate in the constraint.

Now let us recall Eq. (2) for the asymmetric setting:

$$\hat{C}(\beta, \alpha_h^{\ell-1}) := \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \cap \mathcal{A}_h^\ell, j \neq i} C_{i,j}(\beta|_{D_i}, \beta|_{D_j}) + \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} \left(C_{i,j}(\beta|_{D_i}, \alpha_h^{\ell-1}|_{D_j}) + C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta|_{D_i}) \right) .$$

Each term in the first summation on the right-hand side above is known to exactly one agent in the group: $C_{i,j}(\beta|_{D_i}, \beta|_{D_j})$ is the cost for A_i and thus it is known to that agent; both agents A_i and A_j are within the group, so there is no problem to perform this summation in the asymmetric setting. However, in the second summation, while the first term is known to A_i, who belongs to the group, the second term is known only to $A_j \in N(A_i) \setminus \mathcal{A}_h^\ell$ that does not belong to the group. Hence, the evaluation of such costs and the determination of an optimal group assignment β should be revisited.

The only steps in asymmetric P-RODA that we should revisit are Steps 7–10. All remaining steps are indifferent to the asymmetry of the constraints and hence can be left as they were in P-RODA as we described it in the previous section for the symmetric case. In what follows we describe the needed modifications.

5.1. Modifying Step 7: finding a tuple of locally optimal assignments

As in Section 4.3.2, we assume for the sake of simplicity that $\mathcal{A}_h^\ell = \{A_1, A_2, \ldots, A_k\}$, and that each of the domains D_j, $1 \leq j \leq n$, is publicly ordered: $D_j = \{v_0^j, v_1^j, \ldots, v_{|D_j|-1}^j\}$. Then each $A_i \in \mathcal{A}_h^\ell$ can construct a private k-dimensional tensor G_i such that for any multi-index $[i_1, \ldots, i_k]$, where $0 \leq i_j \leq |D_j| - 1$, $1 \leq j \leq k$, the corresponding entry in the tensor is

$$G_i[i_1, i_2, \ldots, i_k] = \gamma_i(\beta, \alpha_h^{\ell-1}) ,$$

where $\beta = (v_{i_1}^1, \ldots, v_{i_k}^k)$, and γ_i is

$$\gamma_i(\beta, \alpha_h^{\ell-1}) := \sum_{A_j \in N(A_i) \cap \mathcal{A}_h^\ell, j \neq i} C_{i,j}(\beta|_{D_i}, \beta|_{D_j}) + \sum_{A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} C_{i,j}(\beta|_{D_i}, \alpha_h^{\ell-1}|_{D_j}) . \tag{7}$$

Now, consider the tensor $G[i_1, i_2, \ldots, i_k] := \sum_{A_i \in \mathcal{A}_h^\ell} G_i[i_1, i_2, \ldots, i_k]$ that represents the sum of the private tensors. In view of Eq. (2) we get

$$G[i_1, i_2, \ldots, i_k] = \hat{C}(\beta, \alpha_h^{\ell-1}) - \sum_{A_i \in \mathcal{A}_h^\ell, A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta|_{D_i}) \tag{8}$$

for $\beta = (v_{i_1}^1, \ldots, v_{i_k}^k)$. The sum on the right-hand side of Eq. (8) is comprised of all terms that are needed for the computation of the local cost $\hat{C}(\beta, \alpha_h^{\ell-1})$ of a candidate tuple β, which are not held by the group members.

In view of the above, the protocol proceeds as follows:

1. The mediator A_h selects a deputy $A'_h \in \mathcal{A}_h^\ell$.
2. Each agent A_i, $1 \le i \le k$, informs each of its external neighbors, $A_j \in N(A_i) \setminus \mathcal{A}_h^\ell$, of the encryption key in E_h, where E_h is as described in Section 4.1.
3. A_i receives from each $A_j \in N(A_i) \setminus \mathcal{A}_h^\ell$ the values $E_h(C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, v))$ for all $v \in D_i$.
4. A_i sends to A'_h the values

$$\mathcal{E}_i[i_1, i_2, \ldots, i_k] := E_h(G_i[i_1, i_2, \ldots, i_k]) \cdot \prod_{A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} E_h(C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta|_{D_i}))$$

   where $\beta = (v^1_{i_1}, \ldots, v^k_{i_k})$, for all tuples $[i_1, i_2, \ldots, i_k]$.
5. A'_h computes $\mathcal{E}[i_1, i_2, \ldots, i_k] := \prod_{i=1}^k \mathcal{E}_i[i_1, i_2, \ldots, i_k]$ for all tuples $[i_1, i_2, \ldots, i_k]$. Owing to the homomorphic property and in view of Eq. (8), $\mathcal{E}[i_1, i_2, \ldots, i_k] = E_h(\hat{C}(\beta, \alpha_h^{\ell-1}))$ for $\beta = (v^1_{i_1}, \ldots, v^k_{i_k})$.
6. A'_h selects a random $c \in \mathcal{G}_C$ (see Section 4.1) and computes $\hat{\mathcal{E}}[i_1, i_2, \ldots, i_k] := c \cdot \mathcal{E}[i_1, i_2, \ldots, i_k]$ for all tuples $[i_1, i_2, \ldots, i_k]$.
7. A'_h selects a secret and random permutation π on $D_h^\ell = \prod_{i=1}^k D_i$ and then sends to the mediator A_h the values $\hat{\mathcal{E}}[i_1, i_2, \ldots, i_k]$ under the chosen permutation.
8. A_h decrypts and recovers the permuted tensor of shifted costs $\hat{C}(\beta, \alpha_h^{\ell-1}) + r$, where $r = E_h^{-1}(c)$.
9. A_h informs A'_h of the position in the recovered tensor of the minimal entry.
10. A'_h uses π^{-1} to recover the original multi-index $[i_1{}', i_2{}', \ldots, i_k{}']$ of the minimal entry, which reveals the sought-after $\beta = (v^1_{i_1{}'}, \ldots, v^k_{i_k{}'})$. A'_h informs all group members about β.

5.2. Modifying Steps 8–9: computing the local improvements

Once the currently optimal tuple $\beta \in D_h^\ell$ for the group $\mathcal{A}_h^\ell$ was found (as described in Section 5.1), it is needed to compute the corresponding local improvement,

$$\Delta_h^\ell := \hat{C}(\beta_h^{\ell-1}, \alpha_h^{\ell-1}) - \hat{C}(\beta, \alpha_h^{\ell-1}) \ge 0\,,$$

where $\hat{C}(\beta, \alpha_h^{\ell-1})$ is given by Eq. (2). Let us denote

$$\Delta_h^\ell(i) := \gamma_i(\beta_h^{\ell-1}, \alpha_h^{\ell-1}) - \gamma_i(\beta, \alpha_h^{\ell-1})\,, \tag{9}$$

where γ_i is as in Eq. (7). In addition, let us denote

$$\delta_h^\ell(i) := \sum_{A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} (C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta_h^{\ell-1}|_{D_i}) - C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta|_{D_i}))\,. \tag{10}$$

Then

$$\Delta_h^\ell = \sum_{A_i \in \mathcal{A}_h^\ell} (\Delta_h^\ell(i) + \delta_h^\ell(i))\,. \tag{11}$$

The first term in the summation on the right-hand side of Eq. (11) is known to A_i – one of the agents in the group. However, the second term on the right-hand side of Eq. (11) is not known to A_i, as it depends on constraints of A_i's neighbors outside the group.

In order to compute the required local improvement, Δ_h^ℓ, we proceed as follows:

1. The mediator A_h selects a deputy $A'_h \in \mathcal{A}_h^\ell$.
2. A'_h selects a random $s'_h \in \mathbb{Z}_\nu$ (see Section 4.1) and notifies all agents in $\mathcal{A}_h^\ell \setminus \{A_h\}$ of s'_h.
3. Each agent A_i, $1 \le i \le k$, informs each of its external neighbors, $A_j \in N(A_i) \setminus \mathcal{A}_h^\ell$, of the encryption key in E_h.
4. A_i receives from each $A_j \in N(A_i) \setminus \mathcal{A}_h^\ell$ the values $E_h(C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, v))$ for all $v \in D_i$.
5. A_i sends to A'_h the value

$$\mathcal{E}_i := E_h(\Delta_h^\ell(i)) \cdot \prod_{A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} E_h(C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta_h^{\ell-1}|_{D_i})) \cdot \prod_{A_j \in N(A_i) \setminus \mathcal{A}_h^\ell} \left(E_h(C_{j,i}(\alpha_h^{\ell-1}|_{D_j}, \beta|_{D_i}))\right)^{-1}\,;$$

here, $\Delta_h^\ell(i)$ is as given in Eq. (9), while the encrypted values in the product defining $\mathcal{E}_i$ above are extracted by A_i from the encrypted values that it received from its neighbors in Line 4 above. The homomorphism of E_h and Eq. (10) imply that

$$\mathcal{E}_i = E_h(\Delta_h^\ell(i) + \delta_h^\ell(i)) \,. \tag{12}$$

6. A'_h computes $\mathcal{E} := \prod_{A_i \in \mathcal{A}_h^\ell} \mathcal{E}_i$. The homomorphism of E_h and Eqs. (11) and (12) imply that $\mathcal{E} = E_h(\Delta_h^\ell)$.
7. A'_h sends to A_h the value $\hat{\mathcal{E}} := \mathcal{E} \cdot E_h(-s'_h) = E_h(\Delta_h^\ell) \cdot E_h(-s'_h) = E_h(\Delta_h^\ell - s'_h)$.
8. A_h decrypts $\hat{\mathcal{E}}$ and recovers the value $\Delta_h^\ell - s'_h$ (where the subtraction is in $\mathbb{Z}_\nu$). This value serves as agent's A_h share s_h.

(Lines 3 and 4 above coincide with Lines 2 and 3 in the implementation of Step 7, as we described in Section 5.1. They need not be repeated but we include them here for the sake of presenting a complete description of the actions that need to be carried out in order to implement Steps 8–9 in the asymmetric setting.)

The above protocol ends with the following outcome: the local improvement Δ_h^ℓ for the group $\mathcal{A}_h^\ell$ is split into two additive shares, s_h and s'_h, modulo ν, where ν is the modulus of the cipher E_h. Specifically, $s_h + s'_h = \Delta_h^\ell$ in $\mathbb{Z}_\nu$. Furthermore, s_h is known only to A_h while s'_h is known to all other agents in the group.

This outcome is different from the corresponding outcome of this stage in P-RODA for symmetric DCOPs. In the symmetric case, all additive shares for all values Δ_h^ℓ, $1 \le h \le n$, were in $\mathbb{Z}_S$ for the same large integer S. However, in the case which we discuss herein, the value of Δ_h^ℓ is split into two additive shares in $\mathbb{Z}_\nu$, where ν is different for each group. Hence, we need to revisit also the contest stage (Step 10) and discuss its execution in that modified setting.

5.3. Modifying Step 10: the contest between neighboring groups

The implementation of Step 10 in the asymmetric setting remains mostly as described in Section 4 for symmetric DCOPs. The only difference that occurs in the asymmetric setting is that now, each group $\mathcal{A}_h^\ell$ has the value of improvement Δ_h^ℓ shared between its members modulo ν, which is the modulus of the encryption function E_h that was selected by that group's mediator (see Section 5.2). Namely, the mediator A_h holds a share s_h that distributes uniformly at random over $\mathbb{Z}_\nu$, while all other members of the group hold the complement share $s'_h = \Delta_h^\ell - s_h \mod \nu$. In particular, each group has its own modulus ν. This is unlike the symmetric setting, where all groups use the same modulus S for sharing Δ_h^ℓ. The fact that in the symmetric setting the domain for all secret sharing in all groups is the same – $\mathbb{Z}_S$, is essential in holding the contest.

Hence, we proceed to explain below how all groups can translate their uniformly random shares modulo ν to uniformly random shares modulo S, where S is some common value, known to all groups. We assume that all agents agree upfront to use the Paillier cryptosystem with moduli ν that consist of q bits (say, $q = 1024$). Under that assumption, they may set $S = 2^q$; note that such a value of S is greater than all moduli ν. Once this ν-to-S translation procedure is carried out by all groups, they can proceed to perform the contest as in the symmetric setting.

The ν-to-S translation procedure is based on the following observation.

Lemma 1. *Assume that $a \ll \nu < S$. Assume further that x and y are two random additive shares in a modulo ν, namely, that x distributes uniformly at random in $\mathbb{Z}_\nu$ and $x + y = a \mod \nu$. Let $\tilde{x}$ be chosen uniformly at random from $\mathbb{Z}_S$ and define*

$$\tilde{y} = y - (\tilde{x} - x) + \delta \mod S \,, \tag{13}$$

where $\delta = 0$ if $x + y < \nu$ and $\delta = S - \nu$ otherwise. Then

$$\tilde{x} + \tilde{y} = a \mod S \,. \tag{14}$$

Proof. Assume first that $x + y < \nu$. Then $x + y = a$ in $\mathbb{Z}$ (and not just in $\mathbb{Z}_\nu$). In that case $\delta = 0$ and, consequently by Eq. (13),

$$\tilde{x} + \tilde{y} = \tilde{x} + y - (\tilde{x} - x) = x + y \mod S \,. \tag{15}$$

But since $x + y = a$ in $\mathbb{Z}$ and $a < S$, we have that $x + y = a$ also in $\mathbb{Z}_S$. Hence, Eq. (14) follows in that case from Eq. (15). Next, let us assume that $x + y \ge \nu$. In that case, since $x + y = a \mod \nu$, we infer that $x + y - \nu = a$ in $\mathbb{Z}$. Therefore, as $\delta = S - \nu$ in that case, we get by Eq. (13) that

$$\tilde{x} + \tilde{y} = \tilde{x} + y - (\tilde{x} - x) + (S - \nu) = x + y + (S - \nu) = a + S = a \mod S \,,$$

thus proving Eq. (14) also in this case. □

Lemma 1 allows the agents in each group to translate their original random shares in Δ_h^ℓ modulo ν to new fully random shares in Δ_h^ℓ modulo S. Namely, starting from s_h (held by A_h) and s'_h (held by $\mathcal{A}_h^\ell \setminus \{A_h\}$) that satisfy $s_h + s'_h = \Delta_h^\ell \mod \nu$

and are both uniformly distributed in $\mathbb{Z}_\nu$, they will compute $\tilde{s}_h$ (to be held by A_h) and $\tilde{s}'_h$ (to be held by $\mathcal{A}_h^\ell \setminus \{A_h\}$) that satisfy $\tilde{s}_h + \tilde{s}'_h = \Delta_h^\ell \mod S$ and are both uniformly distributed in $\mathbb{Z}_S$. To that end, they perform the following steps:

1. They determine whether $s_h + s'_h < \nu$ or not, and set δ to be 0 or $S - \nu$ accordingly.
2. A_h chooses $\tilde{s}_h$ uniformly at random from $\mathbb{Z}_S$ and notifies all other members of its group of the value $w := \tilde{s}_h - s_h \mod S$. Since w distributes uniformly at random on $\mathbb{Z}_S$ it conveys no information on s_h.
3. Every other agent in the group sets $\tilde{s}'_h := s'_h - w + \delta \mod S$. Lemma 1 ensures that $\tilde{s}_h$ (held by A_h) and $\tilde{s}'_h$ (held by all other agents in $\mathcal{A}_h^\ell$) are uniformly distributed in $\mathbb{Z}_S$ and $\tilde{s}_h + \tilde{s}'_h = \Delta_h^\ell \mod S$.

It remains only to explain how the agents in the group carry out the verification of the inequality in Line 1 above. Namely, how can they check whether

$$s_h + s'_h < \nu \tag{16}$$

without the mediator revealing to the other agents the value of s_h (since then they would have been able to recover Δ_h^ℓ, which surrenders information on private constraints) and without the other agents revealing to the mediator the value of s'_h. The verification of inequality (16) is based on the observation that if it holds, then it holds with a very large margin. Indeed, if $s_h + s'_h < \nu$ then $s_h + s'_h = \Delta_h^\ell$ (where the sum is in the usual sense of integers, and not just modulo ν). Since

$$\Delta_h^\ell \leq B := \sum_{i,j} \sum_{a_i \in D_i, a_j \in D_j} C_{i,j}(a_i, a_j)$$

and $B \ll \nu$ (because ν is typically a very large integer, usually of 1024 bits), then inequality (16) holds if and only if

$$s_h + s'_h \leq B\,. \tag{17}$$

Hence, the mediator A_h can select uniformly at random a value $r \in [0, \nu - B)$ and send to any other member in its group, say $A_j \in \mathcal{A}_h^\ell \setminus \{A_h\}$, the value $t := s_h + r$. Next, A_j checks if $t + s'_h < \nu$. If so, then inequality (16) holds (and then δ is set to zero); otherwise, inequality (16) does not hold (and then δ is set to $S - \nu$). We note that the value $t + s'_h$ equals either $\Delta_h^\ell + r$ or $\Delta_h^\ell + r + \nu$. In either case, A_j can infer the sum $\Delta_h^\ell + r$. Given the large interval from which the masking addend r is selected uniformly at random, that sum reveals no information on Δ_h^ℓ with overwhelming probability of $1 - O(\nu^{-1})$, as shown in Lemma 4 in [29].

6. Properties of P-RODA

We discuss here the properties of the P-RODA algorithm. In Section 6.1 we relate to the solutions issued by P-RODA and their consequent guarantees. Then, in Section 6.2 we discuss the privacy features of P-RODA.

6.1. P-RODA simulates RODA

An important observation about the relation between RODA and P-RODA is the following. Both RODA and P-RODA are randomized algorithms, as they involve random decision makings. P-RODA simulates RODA in the sense that to every execution of P-RODA there exists an execution of RODA that issues the very same intermediate and final tuples of full assignments. Hence, all properties of RODA are supported also by P-RODA.

Theorem 2. *P-RODA perfectly simulates RODA in the following sense. Assume that the two algorithms are executed on the same DCOP, and that:*

(a) both start with the same random initial assignment $(a_1^0, \ldots, a_n^0)$*;*

(b) in each iteration every mediator selects the same group from its region in P-RODA as it does in RODA, and

(c) in the case of having more than one optimal tuple for a group, both algorithms choose the optimal tuple which is minimal with respect to some predefined lexicographical order.

Then the sequence of intermediate assignments $(a_1^\ell, \ldots, a_n^\ell)$, $\ell \geq 1$, *that the two algorithms produce will be the same.*

Proof. P-RODA acts exactly like RODA, with one difference: several of the computations in RODA, that are executed in a centralized manner by the mediator, are replaced by secure multi-party protocols that are executed by all agents in the group. To recap, we had three such protocols: the computation of the currently optimal tuple β for the group (for secure implementation of Step 7), the secure summation protocol to reveal the local improvement (Step‘8), and the contest (Step 10). Since each of those protocols is designed so that it outputs the same output as the centralized computation (where in Step 7 this is guaranteed by assumption (c) above), then all computation results will be the same in both RODA's and P-RODA's runs. Consequently, so will be the final output in each iteration, which is the updated full assignment $(a_1^\ell, \ldots, a_n^\ell)$. □

The descriptions of RODA and P-RODA do not relate to the method in which groups are selected in each iteration (Step 5). This was done on purpose in order to keep the transition from RODA to P-RODA as general as possible, and specifically to generalize existing region-optimal algorithms including their group selection method. The convergence guarantees of each region-optimal algorithm are direct consequences of the chosen group selection method (see Section 3.1). Theorem 2 ensures that if the group selection method of any specific instance of RODA (e.g., KOPT [12], DALO [10]) is maintained in the transition to P-RODA, then both RODA and P-RODA reach the same solution.

Corollary 3. *All convergence guarantees for RODA, as implied by the proofs in [10,12,13], equally apply to P-RODA.*

6.2. Privacy

In this section we discuss the privacy of P-RODA with respect to the common notions of privacy in this field [21,28].

Theorem 4. *P-RODA maintains constraint privacy in the sense that no agent may use its view during P-RODA's execution in order to infer binary constraints of other agents.*

Proof. There are three places in P-RODA where information that relates to constraints is exchanged between agents.

(a) There is an exchange of information regarding constraint values in the secure implementation of Step 7, as described in Sections 4.3 (symmetric DCOPs) and 5.1 (asymmetric DCOPs). In the symmetric setting, the only agents that are exposed to information on constraints of other agents are the mediator A_h and its deputy A'_h. In the asymmetric setting, also all agents in the group receive constraint information from their neighbors outside the group. Let us start by considering the agents who are not the mediator: they receive information on constraints of other agents, but that information is encrypted by E_h. Since only the mediator holds the corresponding decryption key then, assuming that breaking E_h is intractable, none of those agents can infer information on constraints of other agents. In addition, since E_h is probabilistic, they cannot even apply a chosen plaintext attack in order to guess the protected plaintext values. (If the encryption had not been probabilistic, then any given plaintext value would have always been encrypted into the same ciphertext, what would have enabled an agent that receives an encrypted value, $y := E_h(x)$, to guess x, encrypt it and check whether the observed y equals $E_h(x)$.)

As for A_h, it learns the entries of the tensor G under two protective measures: each entry is shifted by the random r, and the entries are permuted. The random shift prevents A_h from learning the actual value of every given entry, but it does allow A_h to infer differences between entries in G. Such differences are linear combinations in unknown constraint values. However, owing to the secret random permutation π, A_h cannot relate the entries in G to tuples $\beta \in D_h^\ell$, whence it cannot use the obtained difference values to derive equations in the unknown constraints.

(b) In the secure implementation of Steps 8–9 in both the symmetric and asymmetric settings, the mediator A_h and the other members of $\mathcal{A}_h^\ell$ respectively learn the shares s_h and s'_h in $\Delta_h^\ell = \hat{C}(\beta_h^\ell, \alpha_h^{\ell-1}) - \hat{C}(\beta, \alpha_h^{\ell-1})$, see Eq. (3). They learn nothing further (as implied by the perfect security of the secure summation protocol in the symmetric setting, and the intractability of breaking E_h in the asymmetric setting). However, as each of these shares is completely random it contains no information. (Recall that the agents are semi-honest and do not collude. If collusion between agents had been allowed, then it would have been necessary to modify this computation so that it becomes immune against coalitions of some limited size. Such enhancements are beyond the scope of the present study.)

(c) During the secure execution of the contest (Step 10), one of the four agents A_h, A'_h, A_m, A'_m learns the difference $\left(\Delta_h^\ell - \Delta_m^\ell\right) \mod S$. As noted earlier, we make sure that the agent who receives that difference is not a member of both groups $\mathcal{A}_h^\ell$ and $\mathcal{A}_m^\ell$. Let us denote that agent by A and the group to which it does not belong by $\mathcal{A}^\ell$. (For example, $A = A'_h$ and $\mathcal{A}^\ell = \mathcal{A}_m^\ell$.) Since A does not know the optimal tuple of assignments (which we denoted by β) for the other group $\mathcal{A}^\ell$, the value of the difference cannot be used to infer information on binary costs of other agents. □

Theorem 5. *P-RODA respects partial decision privacy: an agent A_i can infer the final decision of A_j if and only if there exists a group $\mathcal{A}_h^L$ in the last Lth iteration such that both A_i and A_j were members of that group, and that group won the contest against its neighboring groups.*

Proof. Assume first that the latter condition holds, namely, that in the last iteration A_i and A_j were members of the same group, and that the group won the contest against its neighboring groups. Then all members of that group will update their assignment according to the tuple β that they jointly found in Step 7. The values of β become known to all group members, and therefore A_i and A_j will learn the final decision of each other.

Assume next that the condition does not hold. Namely, either A_i and A_j do not belong to the same group in the last iteration, or they do belong to the same group, but that group did not win the contest. In the first case, A_i and A_j remain oblivious of the corresponding locally optimal tuples that were found in the respective groups of each other and whether those groups won or not. Similarly for the second case, even if A_i and A_j do belong to some group $\mathcal{A}_h^L$ in the last Lth iteration, each one of them could be a member of other groups as well. In that case if A_j also belongs to $\mathcal{A}_{h'}^L$, while $A_i \notin \mathcal{A}_{h'}^L$, and $\mathcal{A}_{h'}^L$ wins the contest, then A_i remains oblivious of the final decision that such a win dictates for A_j, as well as of the mere fact that such a win occurred.

Hence, the only setting in which A_i learns the final decision of another agent A_j is when they both belong to a winning group in the last iteration. □

We conclude with a note about the remaining privacy notions of Faltings et al. [21] – agent privacy and topology privacy (see Section 2.2). It seems that RODA is inconsistent with either of those two privacy notions in the sense that an algorithm that achieves one of those privacy goals will essentially differ in its operation from RODA. Indeed, for any $t > 1$, each mediator has to gather around it a group of agents in a distance up to t from it. Clearly, such a group reveals to the mediator the existence of agents that are not direct neighbors. Moreover, the group must be connected, so that even if the group is selected by a trusted third party, the mediator learns topological properties of the graph, which topology privacy forbids.

7. Efficiency analysis

We analyze here the computational overhead of P-RODA with respect to the baseline RODA. We count only encryption and decryption operations, since the other performed operations (e.g. random numbers' generation, additions, or multiplications) have computational costs that are orders of magnitude smaller than those of the cryptographic operations.

Let us consider first the symmetric setting. Fix an agent A_i, $1 \leq i \leq n$, and denote by p_i^ℓ the number of groups to which it belongs in the ℓth iteration, and by $h_{i,j}^\ell$, $1 \leq j \leq p_i^\ell$, the indices of those groups. Then the excessive computational cost for A_i is to perform $\sum_{j=1}^{p_i^\ell} |D_{h_{i,j}^\ell}^\ell|$ encryptions, where $D_{h_{i,j}^\ell}^\ell$ (see Eq. (6)) is the Cartesian product of the domains of variables that are controlled by the agents in the group $\mathcal{A}_{h_{i,j}^\ell}^\ell$. Indeed, in order to find the best tuple β for the group $\mathcal{A}_{h_{i,j}^\ell}^\ell$, A_i needs to encrypt the entries of its private tensor that consists of $|D_{h_{i,j}^\ell}^\ell|$ entries (see Line 2 in the protocol in Section 4.3.3). In addition, as A_i is the mediator of its own group, $\mathcal{A}_i^\ell$, it needs to perform also $|D_i^\ell|$ decryptions (see Line 6 in the protocol in Section 4.3.3).

Proposition 6. *The computational overhead in each iteration of P-RODA, in the symmetric setting, is bounded by $d^k n_t C_E + d^k C_D$, where $d := \max_{1 \leq i \leq n} |D_i|$ is the maximal domain size, $n_t := \max_{1 \leq i \leq n} |N_t(A_i)|$ is the maximal size of a t-neighborhood, and C_E and C_D are the costs of encryption and decryption, respectively. In particular, the computational overhead does not directly depend on the number of agents n.*

Proof. Since an agent may be selected for groups only of mediators within distance t from it, we infer that $|p_i^\ell| \leq n_t$. In addition, since every group has at most k agents, we get (by Eq. (6)) that

$$|D_{h_{i,j}^\ell}^\ell| \leq d^k, \quad 1 \leq i \leq n, 1 \leq j \leq p_i^\ell .$$

Therefore,

$$\sum_{j=1}^{p_i^\ell} |D_{h_{i,j}^\ell}^\ell| \leq d^k n_t .$$

Hence, the computational overhead on A_i due to encryptions is bounded by $d^k n_t C_E$. As for decryptions, since $|D_i^\ell| \leq d^k$, the computational overhead on A_i due to decryptions is bounded by $d^k C_D$. Hence, the overall computational overhead on A_i is bounded by $d^k n_t C_E + d^k C_D$. □

Let us now turn our attention to the asymmetric setting. Again, we fix an agent A_i, $1 \leq i \leq n$, and use the same notations p_i^ℓ and $h_{i,j}^\ell$, $1 \leq j \leq p_i^\ell$, as above. We proceed to examine the computational overhead due to the modified sub-protocols for implementing Step 7 (as described in Section 5.1), Steps 8–9 (Section 5.2) and Step 10 (Section 5.3).

- **Step 7.** Here, every agent has to perform the computations that were done also in the symmetric settings; those computations include the encryption of its private tensor $E_h(G_i[i_1, i_2, \ldots, i_k])$, for each group to which it belongs in that iteration (Line 4 in the protocol in Section 5.1), and the decryption of the tensor for the group that it mediates (Line 8 in Section 5.1). As we saw earlier, those computations incur a computational overhead that is bounded by $d^k n_t C_E + d^k C_D$. In addition, each agent has to reply to requests from its neighbors (see Line 3 in Section 5.1). As the number of neighbors of every given agent is at most n_1, and the reply to each neighbor entails at most d encryptions, we conclude that the total overhead due to Step 7 in the asymmetric setting is bounded by $(d^k n_t + d n_1) C_E + d^k C_D$.
- **Steps 8–9.** The additional computational overhead due to the sub-protocol that implements those steps in the asymmetric setting is as follows, for every given agent A_i: one encryption (computing $E_h(\Delta_h^\ell(i))$) for each group to which A_i belongs in that iteration (Line 5 in Section 5.2); one encryption (computing $E_h(-s'_h)$) for each group in which A_i is the selected deputy (Line 7 in Section 5.2); and one decryption for the group that A_i mediates (Line 8 in Section 5.2). Hence,

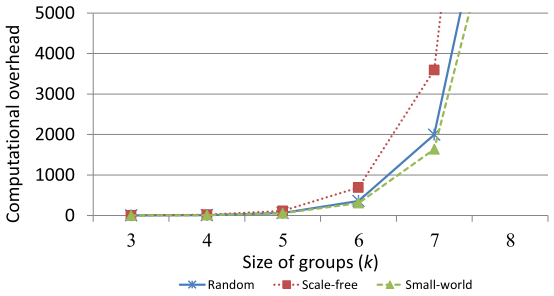

Fig. 2. Computational overhead (minutes) in symmetric DCOPs as a function of k.

the computational overhead here is bounded by $2n_tC_E + C_D$. (Note that Line 4 in Section 5.2 also entails encryptions, but as noted earlier, Line 4 in that sub-protocol is the same as Line 3 in the sub-protocol in Section 5.1 and, therefore, is not carried out again.)

• **Step 10.** The computations in this sub-protocol do not depend on high-cost operations such as encryption or decryption.

In view of the above, we arrive at the following conclusion.

Proposition 7. *The computational overhead in each iteration of P-RODA, in the asymmetric setting, is bounded by* $((d^k + 2)n_t + dn_1)C_E + (d^k + 1)C_D$, *where* d, n_t, C_E *and* C_D *are as in Proposition 6. Here too, the computational overhead does not directly depend on the number of agents* n.

7.1. Computational overhead experiments

To measure the actual time it will take P-RODA to run we followed the *simulated time* approach [55] by measuring the time of atomic operations performed in the algorithm and then counting the non-concurrent times these operations are performed. An advantage of this approach is that it reports results that are implementation-independent and (almost) environment-independent (except for the runtimes of the atomic operations in the given environment). We measured the runtimes of encryption and decryption by averaging multiple runs of the common Java implementation of the Paillier cryptosystem[5] on a hardware comprised of an Intel i7-4600U processor and 16 GB memory. Our tests show that encryption takes at most $C_E = 2$ ms, while decryption takes at most $C_D = 3$ ms.

According to Proposition 6 the factors that impact the computational overhead are k, d, and the topology of the t-distance neighborhoods. As the dependence on d is polynomial while the dependence on k is exponential we focus here on the latter dependence. In order to understand the effect of the neighborhood's topology, we consider three classic types of networks – random networks (Erdős–Rényi model [56]), scale-free networks (Barabási–Albert model [57]), and small-world networks (Watts–Strogatz model [58]).

We start with symmetric DCOPs. Fig. 2 depicts the computational overhead as a function of k in the three network types, when running the algorithm for $L = 50$ iterations. In this setup we fix the number of agents to $n = 100$, the constraint density to $p = 0.1$, the domain sizes to $d = 5$, and the upper bound of the distance to $t = 1$, and vary $k = 3, \ldots, 8$. Fig. 2 shows the exponential dependency on k, whence for high values of k one may need to switch in Step 7 to another complete DCOP private algorithm such as P-SyncBB. Another interesting phenomenon is the higher computational overhead in scale-free networks. This is not surprising, since scale-free networks consist of several highly-connected hubs, and that directly affects n_t. Nevertheless, when using higher distance values ($t \geq 3$), scale-free networks exhibit similar performance to that of the other network types. Varying the distance t has almost no effect on the performance of random and small-world networks, and therefore these results are omitted.

Although not directly affecting the computational overhead, an increase in the number of agents n might potentially affect the topology of neighborhoods, and lead to an indirect effect on the computational overhead. We therefore use the same setting of the previous experiment, but now we fix $k = 3$ and vary $n = 100, \ldots, 1000$. Fig. 3 confirms that the problem's size has almost no effect on the computational overhead, except for the case of scale-free networks in which n_t does increase when n increases and, consequently, we witness a moderate increase in the computational overhead.

The computational overhead of asymmetric problems is very similar to that of the symmetric ones, at least when plotted on a graph. In fact, the graph representing the computational overhead in asymmetric DCOPs as a function of k turned out almost identical to its symmetric counterpart (Fig. 2), due to the exponential effect of k that does not change in the asymmetric setting; thus, that graph was omitted. Fig. 4 depicts the computational overhead in asymmetric DCOPs as a function of the number of agents n. In order to faithfully compare between symmetric and asymmetric problems we used the same set of underlying networks as in the respective symmetric experiment (Fig. 3). As can be clearly seen by comparing Figs. 3 and 4, the excessive computational overhead due to asymmetric constraints is marginal.

[5] http://www.csee.umbc.edu/~kunliu1/research/Paillier.html.

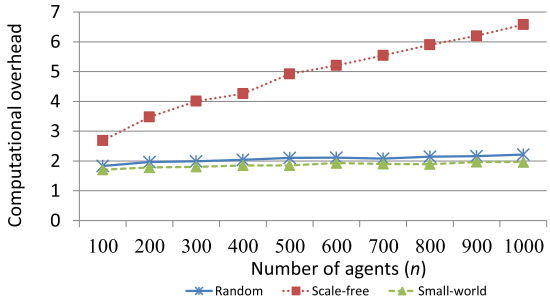

Fig. 3. Computational overhead (minutes) in symmetric DCOPs as a function of n.

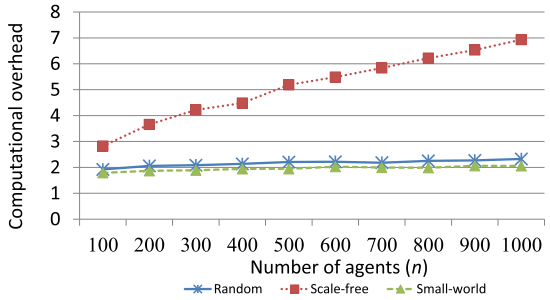

Fig. 4. Computational overhead (minutes) in asymmetric DCOPs as a function of n.

7.2. Experiments on simulator

We implemented full-scale versions of both RODA and P-RODA algorithms on the AgentZero simulator [59]. Running experiments on a simulator has several inherent shortcomings. First of all, a simulator runs on a single computer, as opposed to a real distributed system composed of several computers. Hence, the parallelism of different agents (threads in the simulator) is restricted to the number of cores in the computer's hardware. This is usually not a fundamental restriction for the evaluation of complete DCOP algorithms, since such algorithms are usually inherently restricted to small problems. This is also not a major restriction for the evaluation of incomplete DCOP algorithms, since usually these algorithms are very efficient runtime-wise and therefore it is of no interest to evaluate their runtime performance. Nonetheless, runtime performance becomes an issue when privacy-preservation techniques are applied, even in the case of incomplete algorithms.

Another disadvantage of experiments on a simulator is that the results depend highly on overheads of the specific simulator (e.g., message passing, message handling, threading, idle detection), as well as on implementation issues of the specific algorithm, and the hardware on which the simulator is executed.

Having said that, experiments on a full-scale implementation can help one to confirm the applicability of a given approach, to corroborate trends that were shown theoretically, and to perform comparisons with other algorithms.

Our experiments on the simulator compare P-RODA to RODA, in order to corroborate the computational overhead of P-RODA. In addition, we compare P-RODA to P-Max-Sum [35], which is to the best of our knowledge the only existing alternative of an incomplete privacy-preserving DCOP algorithm. In the experiments we considered three different group sizes $k \in \{3, 4, 5\}$ and a maximal distance from the mediator of $t = 2$ for both P-RODA and RODA. For simplicity of reference, those algorithms are respectively termed P-RODA-k and RODA-k. In each experiment we varied either the constraint densities (p) or the domain sizes (d). Each data point in each experiment represents the average over 50 independently generated problems.

Our simulator experiments consist of two sets. In one set of experiments we evaluated the runtime of each iteration of the algorithm. In an additional set of experiments we evaluated the quality of solutions that are obtained in a predefined time frame.

In the first experiment, which focused on the runtime of a single iteration with varying constraint densities, we followed the respective setting of the P-Max-Sum experiments [35, Fig. 7], which consists of symmetric unstructured random problems with $n = 24$ agents, domain size $d = 5$, and varying constraint densities, $p = 0.1, \ldots, 0.9$.

Fig. 5 focuses on the comparison between P-RODA and RODA. It is evident that the group size k considerably affects the runtime; this corroborates the exponential dependency on k that was stated in Proposition 6 and was also witnessed in the computational overhead experiment in Fig. 2. It is interesting to see that the group size has virtually no effect on the runtime of RODA. This can be explained by the fact that small problems of up to 5 agents are very quickly solved in a non-private manner. It is also clear that the constraint density has very negligible effect on both RODA and P-RODA, which is indeed supported by the computational overhead analysis of Proposition 6. The exception is in the results for very

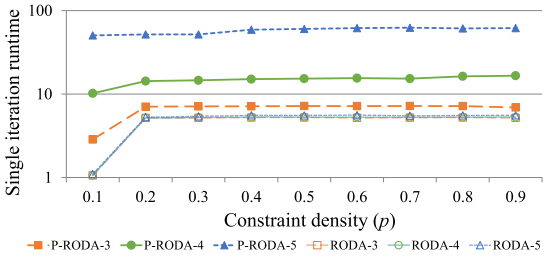

Fig. 5. Runtime (seconds) of a single iteration as a function of p – P-RODA vs. RODA.

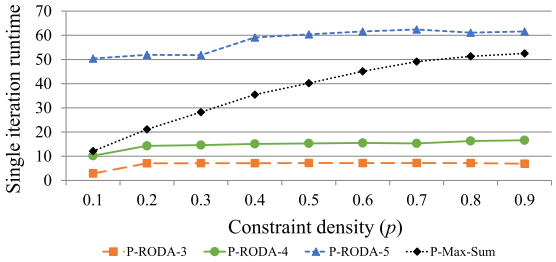

Fig. 6. Runtime (seconds) of a single iteration as a function of p – P-RODA vs. P-Max-Sum.

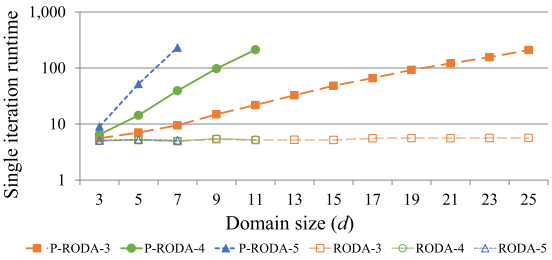

Fig. 7. Runtime (seconds) of a single iteration as a function of d – P-RODA vs. RODA.

sparse problems with $p = 0.1$; the sparsity of the constraint graph naturally leads to less neighboring groups, which in turn drastically decreases the extent of the contest step (Step 10). Although the contests do not involve high-cost operations such as encryption or decryption, they do consist of rather extensive message passing, which their handling entails some computational effort in the AgentZero simulator. Indeed, both RODA and P-RODA sustain the overhead of the contests, but this overhead decreases as the computational efforts of the other steps become higher; actually, in P-RODA-5 this overhead completely disappears. However, there is a slight increase in the runtime of P-RODA-5 between $p = 0.3$ and $p = 0.4$, probably because in the lower densities of the constraint graph not all groups included all $k = 5$ possible agents due to the limitation of the maximal distance $t = 2$. The latter increase can be better viewed in linear scale (see Fig. 6).

Fig. 6 shows the results of the same experiment but now focuses on the comparison between the different incomplete DCOP algorithms, i.e. P-RODA with group sizes $k \in \{3, 4, 5\}$ and P-Max-Sum. It is clear that the constraint density has considerable effect on the runtime of P-Max-Sum, as opposed to P-RODA. Nonetheless, even in sparse problems both P-RODA-3 and P-RODA-4 complete an iteration faster than P-Max-Sum. This is especially interesting given the fact that an iteration in P-RODA includes many more steps, and hence communication cycles, than a P-Max-Sum iteration. Indeed, when switching from Max-Sum to P-Max-Sum, the computational overhead increases by more than 4 orders of magnitude, as was shown in [35, Figs. 7 and 8], whereas the respective increase in computational overhead when switching from RODA to P-RODA is much smaller (Fig. 5).

In the second experiment we tested the dependence of the runtime of a single iteration on the variable domain sizes. As in the previous experiment, we adopted the respective setting of the P-Max-Sum experiments [35, Fig. 8], which consists of symmetric unstructured random problems with $n = 24$ agents, constraint density $p = 0.2$, and varying domain sizes $d = 3, \ldots, 25$.

Fig. 7 reports those runtimes for P-RODA-k and RODA-k, for $k \in \{3, 4, 5\}$. In this experiment we used a cutoff time of 5 minutes. As a result, no runtimes are shown for P-RODA-k and RODA-k in cases in which P-RODA-k did not complete a single iteration for a given density value within 5 minutes. As can be seen, the domain size has significant effect on the runtime performance of P-RODA, as opposed to RODA; this is consistent with the analysis of Proposition 6.

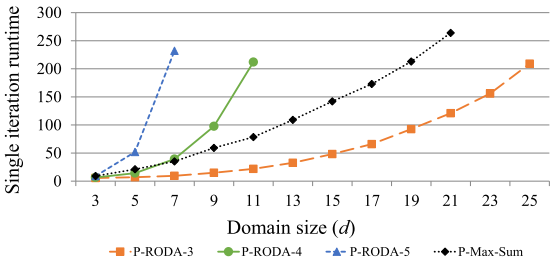

Fig. 8. Runtime (seconds) of a single iteration as a function of d – P-RODA vs. P-Max-Sum.

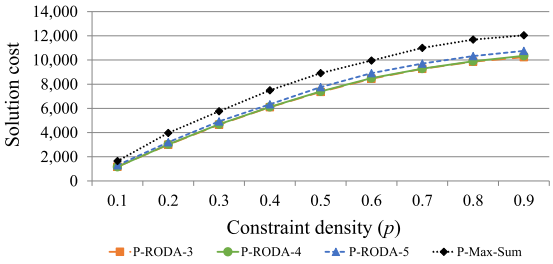

Fig. 9. Solution quality as a function of p.

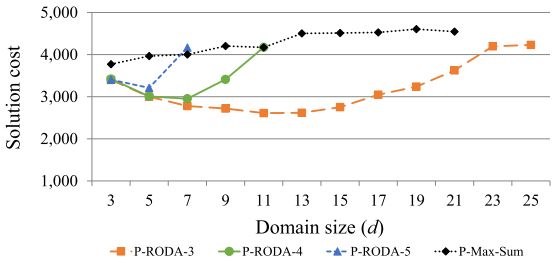

Fig. 10. Solution quality as a function of d.

Fig. 8 reports the corresponding runtime comparison between P-RODA and P-Max-Sum. Here too, we adopted the same 5 minute cutoff practice as in Fig. 7. The effect of the domain size on the runtime performance of P-RODA is evident, and when $d \geq 7$ P-Max-Sum completes an iteration faster than P-RODA-4. Yet, an iteration of P-RODA-3 is still faster than an iteration of P-Max-Sum even when large domains are considered. In fact, P-Max-Sum could not complete a single iteration within the 5 minute cutoff time for $d \geq 23$.

In the next set of experiments we evaluated the quality of solutions that are obtained by the competing incomplete privacy-preserving DCOP algorithms. In these experiments we evaluated the quality of solutions by setting a time threshold of 5 minutes for each instance and letting each of the evaluated algorithms complete as many iterations as possible within this time limit.

Figs. 9 and 10 present the quality of solutions that were obtained on the respective settings as in the runtime experiments of Figs. 6 and 8. As is evident from those figures, P-RODA-3 finds better solutions (lower cost) than P-Max-Sum in all the evaluated settings. We have noticed while running the experiments that even for the exact same number of iterations P-RODA-3 finds better solutions than P-Max-Sum; due to the fact that P-RODA-3 iterations are faster than P-Max-Sum iterations (see Figs. 6 and 8), the achieved solutions of P-RODA-3 given the 5 minute cutoff time are of substantially better quality than those of P-Max-Sum. In the experiment that varies the constraint densities (Fig. 9) even P-RODA-4 and P-RODA-5 outperform P-Max-Sum in terms of solution quality.

An interesting phenomenon is that P-RODA-3 finds higher quality solutions than P-RODA-4 and P-RODA-5 in almost all settings. In fact, our experiments corroborate the expected assertion that the higher the groups size k is, the better the results are after each iteration. However, since P-RODA-3 is able to perform substantially more iterations than P-RODA-4 and P-RODA-5 in the given time frame, it eventually reaches solutions of higher quality. For $0.2 \leq p \leq 0.9$, P-RODA-3 performed 41–43 iterations in each instance, P-RODA-4 performed 18–21 iterations, and P-RODA-5 performed 4–5 iterations. However, for $p = 0.1$, P-RODA-3 performed 104 iterations, P-RODA-4 performed 29 iterations, and P-RODA-5 performed 5 iterations. The latter setting was the only one in which P-RODA-4 found higher quality solutions than P-RODA-3, since it was able to perform sufficiently many iterations in that setting. Following the iteration runtime trends of P-Max-Sum (Fig. 6), the

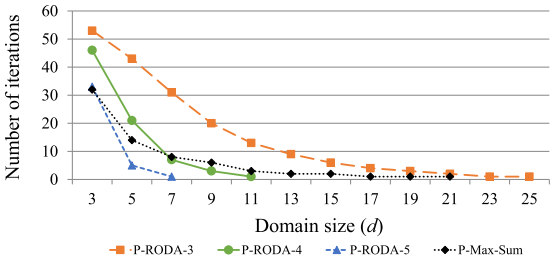

Fig. 11. Number of iterations in 5 minutes as a function of d.

number of iterations it managed to perform in the given time frame constantly reduced from 24 iterations for $p = 0.1$ down to 5 iterations for $p = 0.9$.

Turning to the experiment that varies the domain sizes (Fig. 10), it is important to note that larger domains actually lead to solutions of lower cost, since there are more possibilities to find a combination of low cost in each constraint. Indeed, as the domain sizes grow, the solution cost reduces, but only until some point where the solution cost starts growing. This happens because the algorithms complete less iterations in the given time frame, and therefore do not have enough time to reach these lower-cost solutions. In order to better understand this trend we present in Fig. 11 the number of iterations that were performed by each algorithm in the experiment of Fig. 10.

It is important to mention that state-of-the-art complete privacy-preserving algorithms, such as P-DPOP$^{(+)}$ [28] and P-SyncBB [29], are restricted to much smaller problems than those of the above experiments, cf. [29, Figs. 7 and 8], which makes P-RODA the best alternative to date for solving large-scale DCOPs in a privacy-preserving manner.

8. Related work on DCOP algorithms

A number of complete algorithms were proposed in the last two decades for solving DCOPs. The simplest algorithm among these is the Synchronous Branch and Bound (SyncBB) algorithm [1], which is a distributed version of the well-known centralized Branch and Bound algorithm. Another algorithm that uses a Branch and Bound scheme is Asynchronous Forward Bounding (AFB) [3], in which agents perform sequential value assignments that are propagated for bound checking and early detection of a need to backtrack. A number of complete algorithms use a pseudo-tree, which is derived from the structure of the constraints network, in order to improve the process of acquiring a solution. ADOPT [1] and BnB-ADOPT [60] are two such asynchronous search algorithms, in which assignments are passed down the pseudo-tree. Agents compute upper and lower bounds for possible assignments and send costs, which are eventually accumulated by the root agent, up to their parents in the pseudo-tree.

DPOP [2] is another algorithm that exploits a pseudo-tree. In DPOP, each agent receives from the agents that are its children in the pseudo-tree all the combinations of partial solutions in their sub-tree and their corresponding costs. The agent calculates and generates all the possible partial solutions, which include the partial solutions it received from its children and its own assignments, and sends the resulting combinations up the pseudo-tree. Once the root agent receives all the information from its children, it identifies the value assignment that is a part of the optimal solution and propagates it down the pseudo-tree to the rest of the agents, allowing them to find their own value assignment and thus, produce the optimal solution. DPOP is a GDL algorithm [61], i.e., it stems from the general distributive law that allows accumulation of costs and utility calculation operations performed by different agents in a distributed system for a mutual decision on the optimal solution. A number of more recent studies investigate this paradigm and propose improvements to the DPOP algorithm [62,63].

A very different approach was implemented in the OptAPO algorithm [64,65], in which the agents are partitioned into groups that attempt to find consistent partial solutions. Each such group is headed by a mediator that collects data from all the group members and attempts to solve the local subproblem that is induced by the group. The partition mechanism is dynamic during search and enables the groups to grow whenever there remain conflicts that cannot be resolved by the mediators in the current state of the groups. The formation of groups and the use of a mediator for solving the induced subproblem of each group resembles the mode of operation in region-optimal algorithms. However, unlike region-optimal algorithms, the groups in OptAPO constantly grow and in the worst case a group can include all the agents. The latter feature ensures the completeness of OptAPO, as opposed to region-optimal algorithms, which are incomplete.

All of the algorithms mentioned above are complete. While this is an advantage in the sense that they guarantee to report the optimal solution, this is also a drawback since DCOPs are NP-hard; thus, in order to validate that an acquired solution is optimal they must traverse the entire search space in the worst case. This drawback limits the use of these algorithms to relatively small problems.

One common approach towards incomplete methods for solving DCOPs is *distributed local search*. The general design of most local search algorithms for DCOPs is synchronous [8,5,66]. In each step of the algorithm an agent sends its assignment to all its neighbors in the constraint network and receives the assignments of all its neighbors. The algorithms differ in the method they use in order to decide when to replace the value assignment of their variables and to which new values [5,8].

A different approach towards incomplete distributed problem solving is implemented by the Max-Sum algorithm. Max-Sum [36] is a GDL algorithm that operates on a *factor graph*, which is a bipartite graph in which the nodes represent variables and constraints. Although Max-Sum is completely exploitive, it is not guaranteed to converge in problems with factor graphs that include cycles [36]. On such problems it performs implicit exploration. A number of studies addressed the inconvergence of the algorithm on graphs that include multiple cycles. A first attempt yielded the Bounded Max-Sum algorithm [15], in which the factor graph is reduced to a tree-structured graph. Then, the tree-structured graph is solved optimally using standard Max-Sum. By accounting for the maximal cost (or utility) of the edges removed from the factor graph during the reduction phase, the algorithm can provide a bound on the distance of the quality of the obtained solution from the quality of the optimal solution. A second attempt proposed to shift the factor graph into a directed acyclic graph (DAG) on which Max-Sum is guaranteed to converge, but not necessarily to the optimal solution. After convergence, the directions of all edges in the DAG are inverted so that all constraints are taken under consideration [67]. Recently, we have proposed privacy-preserving versions of Max-Sum and its successors [35]. While both Bounded Max-Sum and region-optimal algorithms provide a quality bound with respect to the optimal solution, the quality of the bounds that can be achieved by region optimality are dependent on the size of the groups. Thus, region optimality allows a selection of parameters that can balance the tradeoff between the runtime and the quality of the issued solution. Bounded Max-Sum does not provide such a tradeoff. Moreover, on dense problems, most of the edges of the factor graph must be removed during the reduction to a tree-structured graph in Bounded Max-Sum, and the bound it can provide is very loose. If we consider the quality of the solutions, even 1-opt MGM produces much better results than Bounded Max-Sum on standard DCOP benchmarks [67].

Approximate DPOP (ADPOP) [16] is an incomplete version of DPOP, parameterized by an upper limit *maxDims* on the number of dimensions of cost/utility tables communicated by agents. When an agent needs to compute in DPOP a table with more than *maxDims* dimensions, it instead computes in ADPOP upper and lower bound projections to tables with *maxDims* dimensions. This can be done by selecting the set of dimensions to be included in the lower-dimensional table, then independently projecting tables received from children and constraints involving this constraint (but not its children) to tables that only involve the retained constraints. These are then combined as in DPOP to compute the bound projections to be sent to the agent's parent. In the worst case, computing the projection for each table received from a child requires finding the maximum and minimum over joint assignments to $maxDims - 1$ dimensions. Since this must be done for each of the joint assignments in the *maxDims* dimensions of the generated table and each child table, computing a table in ADPOP has complexity $\Theta\left(N \cdot D^{2 \cdot maxDims}\right)$ where N is the number of neighbors and D is the size of the domains.

Okimoto, et al. [17] proposed an incomplete algorithm with similarities to both Bounded Max-Sum and ADPOP. Like ADPOP, it uses a pseudo-tree, although one based on a variable ordering rather than depth-first search. It adds constraints to make the dependencies in the pseudo-tree explicit (i.e., the variable dimensions that would have to be included in tables sent in ADPOP). Similar to Bounded Max-Sum, constraints are then eliminated to achieve the desired induced width while providing bounds on solution quality based on the eliminated constraints. The reduced problem is then solved using a complete algorithm. This algorithm, like region-optimal algorithms, offers a balance between runtime and optimality guarantees. Nevertheless, unlike Bounded Max-Sum, a private version of this algorithm has not been proposed yet.

9. Conclusion

Region optimality is an important concept that offers a balance between runtime efficiency and guarantees on the solution quality. Privacy loss is a major drawback of region-optimal algorithms, yet, to the best of our knowledge, no secure protocol for finding region-optimal solutions has been proposed to date. In this paper we proposed P-RODA, an algorithmic framework that converges to region-optimal solutions while preserving constraint privacy and partial decision privacy, for both symmetric and asymmetric DCOPs. The underlying algorithm RODA generalizes the algorithms in the region optimality family, such as KOPT [9] and DALO [10]. Privacy is achieved chiefly by using secret sharing and homomorphic encryption. P-RODA is scalable with respect to the number of agents n, but it is more suitable for regions with a small group size k. Experiments performed on a simulator showed that P-RODA is currently the best alternative for solving large-scale DCOPs in a privacy-preserving manner.

As future work we aim at improving the privacy guarantees of P-RODA. One such improvement could be the obtaining of full decision privacy. Currently, P-RODA reveals to every agent the final decision of all direct neighboring agents that happened to be with that agent in the same winning group in the last iteration. It is possible to achieve *full* (rather than partial) decision privacy by enhancing the secure implementation of Step 7 in P-RODA so that each agent learns only its own component in the optimal local assignment β rather than the entire tuple β, as is the case now. Similarly, it might be possible to achieve assignment privacy by replacing the transfer of current assignment data in Step 6 of P-RODA with secure multi-party protocols. Clearly, such enhancements of privacy come with a price tag, as they entail greater computational and communication costs.

Another possible improvement could be achieving partial agent privacy. By that we mean that agents will be allowed to learn about the existence of non-neighboring agents (as explained in Section 6.2, such a disclosure of information is inherent in RODA), but not their identity. Léauté and Faltings proposed the use of codenames, instead of clear agent identifiers, as a means to achieve agent privacy in the privacy-preserving versions of DPOP [28]. However, such techniques for hiding agent identities are not sufficient here, since the main difficulty in attempting to achieve such agent privacy is that it is necessary to limit the communication only to messages between neighboring agents. Such a restriction makes even the most simple

primitive that we utilize, secure multi-party addition, more challenging. Namely, given a connected group of agents, where each one holds a private integer, we need a secure protocol for adding those private inputs, where only neighboring agents can exchange messages. All existing protocols for secure addition do not work under such a restriction. Designing new secure addition protocols that comply with that restriction is an independent research problem for which we are not aware of any related prior art.

It is interesting to see that within our 5-minute time limit, P-RODA-3 found higher quality solutions than P-RODA-4 and P-RODA-5. From these results we conclude that when dealing with privacy-preserving algorithms that inherently require more runtime, the number of iterations is an important factor, to be considered alongside the quality of each iteration. Following this comprehension, we believe that devising privacy-preserving versions of simple local search algorithm, such as DSA and DBA, is an important direction for future research.

References

[1] P.J. Modi, W. Shen, M. Tambe, M. Yokoo, ADOPT: asynchronous distributed constraints optimization with quality guarantees, Artif. Intell. 161 (2005) 149–180.
[2] A. Petcu, B. Faltings, A scalable method for multiagent constraint optimization, in: IJCAI, 2005, pp. 266–271.
[3] A. Gershman, A. Meisels, R. Zivan, Asynchronous forward bounding for distributed COPs, J. Artif. Intell. Res. 34 (2009) 61–88.
[4] R.T. Maheswaran, J.P. Pearce, M. Tambe, Distributed algorithms for DCOP: a graphical-game-based approach, in: PDCS, 2004, pp. 432–439.
[5] W. Zhang, Z. Xing, G. Wang, L. Wittenburg, Distributed stochastic search and distributed breakout: properties, comparison and applications to constraints optimization problems in sensor networks, Artif. Intell. 161 (2005) 55–88.
[6] W.T.L. Teacy, A. Farinelli, N.J. Grabham, P. Padhy, A. Rogers, N.R. Jennings, Max-Sum decentralised coordination for sensor systems, in: AAMAS, 2008, pp. 1697–1698.
[7] R. Zivan, S. Okamoto, H. Peled, Explorative anytime local search for distributed constraint optimization, Artif. Intell. 212 (2014) 1–26.
[8] K. Hirayama, M. Yokoo, The distributed breakout algorithms, Artif. Intell. 161 (1–2) (2005) 89–116.
[9] H. Katagishi, J.P. Pearce, Kopt: distributed DCOP algorithm for arbitrary k-optima with monotonically increasing utility, in: DCR, 2007.
[10] C. Kiekintveld, Z. Yin, A. Kumar, M. Tambe, Asynchronous algorithms for approximate distributed constraint optimization with quality bounds, in: AAMAS, 2010, pp. 133–140.
[11] R.T. Maheswaran, J.P. Pearce, M. Tambe, A family of graphical-game-based algorithms for distributed constraint optimization problems, in: Coordination of Large-Scale Multiagent Systems, Springer-Verlag, 2006, pp. 127–146.
[12] J.P. Pearce, M. Tambe, Quality guarantees on k-optimal solutions for distributed constraint optimization problems, in: IJCAI, 2007, pp. 1446–1451.
[13] M. Vinyals, E.A. Shieh, J. Cerquides, J.A. Rodríguez-Aguilar, Z. Yin, M. Tambe, E. Bowring, Quality guarantees for region optimal DCOP algorithms, in: AAMAS, 2011, pp. 133–140.
[14] R. Mailler, V. Lesser, Asynchronous partial overlay: a new algorithm for solving distributed constraint satisfaction problems, J. Artif. Intell. Res. 25 (2006) 529–576.
[15] A. Rogers, A. Farinelli, R. Stranders, N.R. Jennings, Bounded approximate decentralised coordination via the Max-Sum algorithm, Artif. Intell. 175 (2011) 730–759.
[16] A. Petcu, B. Faltings, Approximations in distributed optimization, in: CP, 2005, pp. 802–806.
[17] T. Okimoto, Y. Joe, A. Iwasaki, M. Yokoo, B. Faltings, Pseudo-tree-based incomplete algorithm for distributed constraint optimization with quality bounds, in: CP, 2011, pp. 660–674.
[18] A. Gershman, A. Grubshtein, A. Meisels, L. Rokach, R. Zivan, Scheduling meetings by agents, in: Proc. PATAT-08, Montreal, 2008.
[19] P. Rust, G. Picard, F. Ramparany, Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems, in: IJCAI, 2016, pp. 468–474.
[20] F. Fioretto, W. Yeoh, E. Pontelli, A multiagent system approach to scheduling devices in smart homes, in: AAMAS, 2017, pp. 981–989.
[21] B. Faltings, T. Léauté, A. Petcu, Privacy guarantees through distributed constraint satisfaction, in: WI-IAT, 2008, pp. 350–358.
[22] R. Greenstadt, B. Grosz, M.D. Smith, SSDPOP: improving the privacy of DCOP with secret sharing, in: AAMAS, 2007, p. 171.
[23] T. Grinshpoun, When you say (DCOP) privacy, what do you mean? in: ICAART, 2012, pp. 380–386.
[24] M.C. Silaghi, D. Mitra, Distributed constraint satisfaction and optimization with privacy enforcement, in: IAT, 2004, pp. 531–535.
[25] R.T. Maheswaran, J.P. Pearce, E. Bowring, P. Varakantham, M. Tambe, Privacy loss in distributed constraint reasoning: a quantitative framework for analysis and its applications, Auton. Agents Multi-Agent Syst. 13 (2006) 27–60.
[26] R. Greenstadt, J. Pearce, M. Tambe, Analysis of privacy loss in distributed constraint optimization, in: AAAI, 2006, pp. 647–653.
[27] P. Doshi, T. Matsui, M.C. Silaghi, M. Yokoo, M. Zanker, Distributed private constraint optimization, in: WI-IAT, 2008, pp. 277–281.
[28] T. Léauté, B. Faltings, Protecting privacy through distributed computation in multi-agent decision making, J. Artif. Intell. Res. 47 (2013) 649–695.
[29] T. Grinshpoun, T. Tassa, P-SyncBB: a privacy preserving branch and bound DCOP algorithm, J. Artif. Intell. Res. 57 (2016) 621–660.
[30] K. Nissim, R. Zivan, Secure discsp protocols – from centralized towards distributed solutions, in: DCR Workshops, 2005.
[31] M. Yokoo, K. Suzuki, K. Hirayama, Secure distributed constraints satisfaction: reaching agreement without revealing private information, Artif. Intell. 161 (2005) 229–246.
[32] M.C. Silaghi, B. Faltings, A. Petcu, Secure combinatorial optimization simulating DFS tree-based variable elimination, in: ISAIM, 2006.
[33] K. Hirayama, M. Yokoo, Distributed partial constraint satisfaction problem, in: CP, 1997, pp. 222–236.
[34] T. Tassa, R. Zivan, T. Grinshpoun, Max-Sum goes private, in: IJCAI, 2015, pp. 425–431.
[35] T. Tassa, T. Grinshpoun, R. Zivan, Privacy preserving implementation of the Max-Sum algorithm and its variants, J. Artif. Intell. Res. 59 (2017) 311–349.
[36] A. Farinelli, A. Rogers, A. Petcu, N.R. Jennings, Decentralised coordination of low-power embedded devices using the Max-Sum algorithm, in: AAMAS, 2008, pp. 639–646.
[37] R.T. Maheswaran, M. Tambe, E. Bowring, J.P. Pearce, P. Varakantham, Taking DCOP to the real world: efficient complete solutions for distributed multi-event scheduling, in: AAMAS, 2004, pp. 310–317.
[38] T. Grinshpoun, A. Grubshtein, R. Zivan, A. Netzer, A. Meisels, Asymmetric distributed constraint optimization problems, J. Artif. Intell. Res. 47 (2013) 613–647.
[39] R. Greenstadt, An overview of privacy improvements to k-optimal DCOP algorithms, in: AAMAS, 2009, pp. 1279–1280.
[40] T. Tassa, R. Zivan, T. Grinshpoun, Preserving privacy in region optimal DCOP algorithms, in: IJCAI, 2016, pp. 496–502.
[41] R. Zivan, H. Yedidsion, S. Okamoto, R. Glinton, K.P. Sycara, Distributed constraint optimization for teams of mobile sensing agents, Auton. Agents Multi-Agent Syst. 29 (2015) 495–536.
[42] A. Farinelli, A. Rogers, N.R. Jennings, Agent-based decentralised coordination for sensor networks using the max-sum algorithm, Auton. Agents Multi-Agent Syst. 28 (2014) 337–380.

[43] T. Grinshpoun, Clustering variables by their agents, in: WI-IAT, 2015, pp. 250–256.
[44] F. Fioretto, W. Yeoh, E. Pontelli, Multi-variable agents decomposition for DCOPs, in: AAAI, 2016, pp. 2480–2486.
[45] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Eurocrypt, 1999, pp. 223–238.
[46] J.C. Benaloh, Secret sharing homomorphisms: keeping shares of a secret secret, in: Crypto, 1986, pp. 251–260.
[47] T. Grinshpoun, T. Tassa, A privacy-preserving algorithm for distributed constraint optimization, in: AAMAS, 2014, pp. 909–916.
[48] T. Tassa, E. Gudes, Secure distributed computation of anonymized views of shared databases, ACM Trans. Database Syst. 37 (2012), Article 11.
[49] A. Netzer, A. Meisels, R. Zivan, Distributed envy minimization for resource allocation, Auton. Agents Multi-Agent Syst. 30 (2) (2016) 364–402.
[50] B. Lutati, V. Levit, T. Grinshpoun, A. Meisels, Congestion games for v2g-enabled ev charging, in: AAAI, 2014, pp. 1440–1446.
[51] L. Duan, M.K. Doğru, U. Özen, J.C. Beck, A negotiation framework for linked combinatorial optimization problems, Auton. Agents Multi-Agent Syst. 25 (1) (2012) 158–182.
[52] F. Fioretto, W. Yeoh, E. Pontelli, A multiagent system approach to scheduling devices in smart homes, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS '17, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2017, pp. 981–989.
[53] V. Levit, T. Grinshpoun, A. Meisels, A.L. Bazzan, Taxation search in boolean games, in: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 183–190.
[54] Z. Komarovsky, V. Levit, T. Grinshpoun, A. Meisels, Efficient equilibria in a public goods game, in: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 2, IEEE, 2015, pp. 214–219.
[55] E. Sultanik, R.N. Lass, W.C. Regli, DCOPolis: a framework for simulating and deploying distributed constraint reasoning algorithms, in: AAMAS (demos), 2008, pp. 1667–1668.
[56] P. Erdős, A. Rényi, On random graphs, Publ. Math. (Debr.) 6 (1959) 290–297.
[57] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.
[58] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442.
[59] B. Lutati, I. Gontmakher, M. Lando, A. Netzer, A. Meisels, A. Grubshtein, AgentZero: a framework for simulating and evaluating multi-agent algorithms, in: Agent-Oriented Software Engineering – Reflections on Architectures, Methodologies, Languages, and Frameworks, 2014, pp. 309–327.
[60] W. Yeoh, A. Felner, S. Koenig, BnB-ADOPT: an asynchronous branch-and-bound DCOP algorithm, J. Artif. Intell. Res. 38 (2010) 85–133.
[61] S.M. Aji, R.J. McEliece, The generalized distributive law, IEEE Trans. Inf. Theory 46 (2000) 325–343.
[62] I. Brito, P. Meseguer, Improving DPOP with function filtering, in: AAMAS, 2010, pp. 141–148.
[63] M. Vinyals, M. Pujol, J.A. Rodríguez-Aguilar, J. Cerquides, Divide-and-coordinate: DCOPs by agreement, in: AAMAS, 2010, pp. 149–156.
[64] R. Mailler, V.R. Lesser, Solving distributed constraint optimization problems using cooperative mediation, in: AAMAS, 2004, pp. 438–445.
[65] T. Grinshpoun, A. Meisels, Completeness and performance of the APO algorithm, J. Artif. Intell. Res. 33 (2008) 223–258.
[66] M. Jain, M.E. Taylor, M. Tambe, M. Yokoo, DCOPs meet the real world: exploring unknown reward matrices with applications to mobile sensor networks, in: IJCAI, 2009, pp. 181–186.
[67] R. Zivan, T. Parash, L. Cohen, H. Peled, S. Okamoto, Balancing exploration and exploitation in incomplete min/max-sum inference for distributed constraint optimization, Auton. Agents Multi-Agent Syst. 31 (2017) 1165–1207.