Auton Agent Multi-Agent Sys (2006) 13:27–60
DOI 10.1007/s10458-006-5951-y

Privacy loss in distributed constraint reasoning: a quantitative framework for analysis and its applications

Rajiv T. Maheswaran · Jonathan P. Pearce · Emma Bowring · Pradeep Varakantham · Milind Tambe

Published online: 24 February 2006

Abstract It is critical that agents deployed in real-world settings, such as businesses, offices, universities and research laboratories, protect their individual users' privacy when interacting with other entities. Indeed, privacy is recognized as a key motivating factor in the design of several multiagent algorithms, such as in distributed constraint reasoning (including both algorithms for distributed constraint optimization (DCOP) and distributed constraint satisfaction (DisCSPs)), and researchers have begun to propose metrics for analysis of privacy loss in such multiagent algorithms. Unfortunately, a general quantitative framework to compare these existing metrics for privacy loss or to identify dimensions along which to construct new metrics is currently lacking. This paper presents three key contributions to address this shortcoming. First, the paper presents VPS (Valuations of Possible States), a general quantitative framework to express, analyze and compare existing metrics of privacy loss. Based on a state-space model, VPS is shown to capture various existing measures of privacy created for specific domains of DisCSPs. The utility of VPS is further illustrated through analysis of privacy loss in DCOP algorithms, when such algorithms are used by personal assistant agents to schedule meetings among users. In addition, VPS helps identify dimensions along which to classify and construct new privacy metrics and it also supports their quantitative comparison. Second, the article presents key inference rules that may be used in analysis of privacy loss in DCOP algorithms under different assumptions. Third, detailed experiments based on the VPS-driven analysis lead to the following key results: (i) decentralization by itself does not provide superior protection of privacy in DisCSP/DCOP algorithms when compared with centralization; instead, privacy protection also requires the presence of uncertainty about agents' knowledge of the constraint graph. (ii) one needs to carefully examine the metrics chosen to measure privacy loss; the qualitative properties of privacy loss and hence the conclusions that can be drawn about an algorithm can vary widely based on the metric chosen. This paper should thus serve as a call to arms for further privacy research, particularly within the DisCSP/DCOP arena.

Keywords Distributed constraint reasoning · Privacy · Distributed constraint optimization

R. T. Maheswaran (✉) · J. P. Pearce · E. Bowring · P. Varakantham · M. Tambe
Computer Science Department and Information Sciences Institute,
University of Southern California, USA
e-mail: maheswar@usc.edu

1. Introduction

Personal assistant agents are an emerging application whose integration into businesses, office environments, universities and research organizations, as well as other spheres of human activity, promises to enhance productivity by performing routine or mundane tasks and expediting coordinated activities [1,5,10,13,18]. To effectively accomplish these tasks, agents must be endowed with information about their users, that would preferably be kept private. However, in domains where humans and their agent counterparts have to collaborate with other human-agent pairs, and agents are given the autonomy to negotiate or resolve conflicts on behalf of their users, the exchange of private information is necessary to achieve a good team solution. Some of these situations include meeting scheduling, where users' valuations of certain blocks of time in a schedule, or the relative importance of different meetings, can be the information desired to be kept private [2,13,23,29]. In team task-assignment problems, the private information could be a user's capability to perform various tasks and the personal priority they assign to those tasks. Similarly, when resolving conflicts in budgets (when collaborating across different organizations), the information that needs to be kept private may be salary information. To develop trust in, and hence promote the use of, personal assistant agents, humans must believe their privacy will be sufficiently protected by the processes employed by their agents. Thus, understanding how privacy is lost in these contexts is critical for evaluating the effectiveness of strategies used to govern these interactions.

In this article, we address the problem of privacy loss in personal-assistant-agent systems and specifically in the algorithms used for coordination. Distributed constraint reasoning, in the form of distributed constraint satisfaction (DisCSP) [27,29,30] and distributed constraint optimization (DCOP) [11,13,15,17], has been offered as a key approach that addresses this problem, as it promises to provide distributed conflict resolution within a collaborating set of agents. Indeed, maintaining privacy is a fundamental motivation in *distributed* constraint reasoning [13,17,26,29]. For instance, Yokoo et al. point out one key motivation for DisCSPs: "*Furthermore, in some application problems, such as software agents, in which each agent acts as a secretary of an individual, gathering all information to one agent is undesirable or impossible for security/privacy reasons"* [29]. This initial motivation based on privacy has been amplified in recent work on DCOP. For instance, Maheswaran et al. state that DCOP is useful in domains such as meeting scheduling, where "*an organization wants to maximize the value of their employees' time while maintaining the privacy of information*" [13].

This emphasis on privacy has led to significant increasing interest in DisCSP and DCOP for its applications in software personal assistant domains [1,2,10,13,18,26]. One approach to provide privacy in DisCSPs has been to use cryptographic techniques [31], but the required use of multiple external servers may not always be desirable, available or justifiable for its benefit (see Section 6 for further discussions). Instead, a second approach has attracted significant attention, where researchers have begun providing metrics for quantifying the privacy loss in DisCSP algorithms [8,16,25]. If we can guarantee a limited privacy loss in specific DisCSP algorithms in the first place, then additional cryptographic techniques are unnecessary. Unfortunately, these privacy approaches are based on DisCSPs and they are not immediately portable to DCOPs which optimize rather than satisfy. More importantly, there is a lack of a principled quantitative framework that would allow us to express and construct different metrics for measuring privacy or to understand the relationship among these metrics. It is also difficult to identify dimensions along which to derive new metrics in a principled fashion, whether in the context of DCOPs or DisCSPs.

This article provides three key contributions to address the above shortcomings. First, we propose Valuation of Possible States (VPS), a unifying quantitative framework to express privacy loss in multiagent settings. Quantification of privacy loss in VPS is based on other agents' estimates about an agent's possible states before and after a protocol is engaged. In particular, within VPS, privacy is interpreted as a valuation on the other agents' estimates about the possible states that one lives in. VPS is a general framework, which enables existing metrics to be re-cast within the framework for cross-metric comparison. VPS also helps us identify dimensions along which to construct and classify new privacy metrics. Second, we develop techniques to analyze and compare privacy loss in DCOP algorithms; in particular, when using approaches ranging from decentralization (SynchBB [11], partial centralization (OptAPO [15]), as well as centralization. This involves constructing principled sets of inference procedures under various assumptions of knowledge by the agents. Third, we generate and investigate several distributed meeting-scheduling scenarios modeled as DCOPs, where we are able to perform a cross-metric comparison of privacy loss in these three approaches. We detail extensive experimental results presented on two sets of assumptions about real-world meeting scheduling scenarios, one in which agents possess knowledge of the distributed constraint graph and another one which introduces uncertainty in this knowledge. Key implications of our experiments are as follows: (i) Decentralized approaches for constraint optimization do not automatically outperform centralized approaches with respect to privacy loss, a result that consistently holds over many metrics and scenarios; in our experiments, privacy protection is shown to have improved in the presence of uncertainty about agents' knowledge of the constraint graph. (ii) The qualitative properties of privacy loss can vary widely depending on the metric chosen. For example, privacy loss may increase or decrease as a function of the length of the schedule depending on which metric one chooses. More significantly, they can rank the effectiveness of privacy protection due to various algorithmic approaches differently based on metric. Thus, one must carefully justify any metric used.

The rest of this paper is organized as follows. Section 2 outlines the VPS framework and illustrates its ability to unify the expression of existing metrics in a common language. Section 3 describes a distributed meeting scheduling problem model and discusses how it can be solved as a DCOP. In Section 4 we introduce several different metrics for privacy loss applicable for the meeting scheduling problems expressed as DCOPs. We also describe how privacy loss occurs in a variety of methods for solving DCOPs, including inference procedures for distributed approaches. Section 5 presents the experimental domains and results when applying inference and different metrics. In Section 6, we discuss related work, and in Section 7, we present some concluding thoughts.

2. Valuations of possible states

Given a setting where a group of agents, each representing a single user, must collaborate to achieve some task, each agent must be endowed with some private information about its user to ensure that it accurately represents their status, capabilities or preferences in the joint task. The goal is to understand privacy loss in collaboration where multiagent negotiation protocols necessarily lead to revelation of private information. In this section, we describe the VPS framework which provides a foundation for expressing various instantiations of privacy metrics and demonstrate its ability to unify by capturing existing metrics proposed by the agents research community within the same framework. Privacy generally represents the notion of minimizing the information about some aspect of an entity in others' beliefs about

that entity. In this paper, we will use the term *agents* to refer to such entities with private information engaged in a collaborative effort, though *people* or *users* can be equivalently substituted. We model the intuitive notion of privacy of an observed agent as a function over a probability distribution over a state space, where the distribution constitutes observer agents' models of the observed agent's private information. We begin with an example which we will refer to throughout the exposition.

Example 1 *Meeting Scheduling.* Consider a scenario where three agents (A, B, C) have to negotiate a meeting for either 9:00 AM or 4:00 PM. Each agent has a preference or availability denoted by 0 or 1 for each time. Before negotiation, all agents will have some beliefs about the preferences of other agents. After negotiation, agents will alter these beliefs due to inferences they make from the messages received. The privacy loss due to the negotiation is generally the difference, according to some measure, between the initial and final beliefs.[1]

2.1. Framework

To express VPS more formally, let the private information of the ith agent be modeled as a state $s_i \in S_i$, where S_i is a set of possible states that the ith agent may occupy. For simplicity, we assume that $\{S_i\}_{i=1}^{N}$ are discrete sets, though these ideas can be extended to continuous sets. Here N is the number of agents who are indexed by the set $\mathcal{N} := \{1, \ldots, N\}$. In Example 1, each agent is in one of four states from the state space $S_A = S_B = S_C = \{[0\,0], [0\,1], [1\,0], [1\,1]\}$, where the elements of the vector denote the preference for 9:00 AM and 4:00 PM, respectively. Then,

$$\mathbb{S}_{-j} := S_1 \times S_2 \times \cdots \times S_{j-1} \times S_{j+1} \times \cdots \times S_{N-1} \times S_N,$$

is the set of all possible states of all agents except the jth agent. The jth agent knows that the other agents' private information is captured by an element of the set S_{-j}. In Example 1, agent A models agents B and C as an element of the set $\mathbb{S}_{-A} = S_B \times S_C$ where $S_B = S_C = \{[0\,0], [0\,1], [1\,0], [1\,1]\}$. Since an agent does not know the private information of other agents exactly, we can model the jth agent's belief as a probability distribution over the possible states of all other agents denoted as $\mathbb{P}^j(\mathbb{S}_{-j})$. Given that we have discrete sets, we will have a probability mass function,

$$\mathbb{P}^j(\mathbb{S}_{-j}) = [P^j(\tilde{s}_1) \cdots P^j(\tilde{s}_k) \cdots P^j(\tilde{s}_{K_1})],$$

where $\tilde{s}_k \in \mathbb{S}_{-j}$ is a possible state of all other agents. There are $K_1 = \Pi_{z \neq j}|S_z|$ possible states and since the vector is a probability mass function, we have the conditions $P^j(\tilde{s}_k) \geq 0$, $\sum_{\tilde{s} \in \mathbb{S}_{-j}} P^j(\tilde{s}) = 1$. In Example 1, agent A's knowledge of the other agents would be a probability vector of length $K_1 = 16$, i.e. $\mathbb{P}^A(S_{-A}) = [P^A(\tilde{s}_1) \;\cdots\; P^A(\tilde{s}_{16})]$. The states $\{\tilde{s}_1, \ldots, \tilde{s}_{16}\}$ map isomorphically to $\{[0\,0\,0\,0], \ldots, [1\,1\,1\,1]\}$ which captures the possible states of agents B and C. The states of other agents are represented jointly because information about multiple agents can be coupled in a single message.

Thus, an agent's knowledge of other agents can be represented by a joint probability mass function over the product set of possible states of all other agents. The jth (observer) agent's knowledge of a particular agent, say the ith (observed) agent, is then the marginal probability

[1] The term negotiation in this example and throughout the rest of this paper alludes to the interaction that occurs among agents in the DisCSP/DCOP algorithms that are analyzed. However, the VPS framework is not limited to DisCSP/DCOP algorithms.

of this distribution with respect to i, as follows:

$$\mathbb{P}_i^j(S_i) = [P_i^j(s_1) \cdots P_i^j(s_i) \cdots P_i^j(s_{K_2})], \quad s_i \in S_i, \qquad K_2 = |S_i|, \tag{1}$$
$$P_i^j(s_i) = \sum_{\tilde{s} \in \mathbb{S}_{-j} : \tilde{s}^i = s_i} P^j(\tilde{s})$$

where $\tilde{s}^i$ refers to the state of the ith agent in the tuple $\tilde{s} \in \mathbb{S}_{-j}$. In Example 1, the probability that agent A thinks that agent B is in the state [0 1] is the sum of of the probabilities that it thinks agents B and C are in the states {[0 1 0 0], [0 1 0 1], [0 1 1 0], [0 1 1 1]}.

The knowledge that other $N-1$ (observer) agents have about the ith (observed) agent can then be expressed as follows:

$$\mathbb{P}_i(S_i) = [\mathbb{P}_i^1(S_i)\ \ \mathbb{P}_i^2(S_i)\ \ \cdots\ \ \mathbb{P}_i^{i-1}(S_i)\ \ \mathbb{P}_i^{i+1}(S_i)\ \ \cdots\ \ \mathbb{P}_i^{N-1}(S_i)\ \ \mathbb{P}_i^N(S_i)]$$

where $\mathbb{P}_i^j(S_i)$ is as defined in Eq. (1). In Example 1, the information other agents have about agent A is $\mathbb{P}_A(S_A) = [\mathbb{P}_A^B(S_A)\ \mathbb{P}_A^C(S_A)]$. The above model assumes that agents do not share there information or estimates about other agents, i.e. there is no collusion to gain information, and the beliefs are independent. If sharing does occur, then $\mathbb{P}_i^G(S_i)$ denotes G's belief about the ith agent, where $G \subset \mathcal{N}$ is a group of agents that share information to obtain a better estimate of the ith agent's state, where $i \notin G$. In this case $\mathbb{P}_i(S_i)$ would be composed of group estimates $\mathbb{P}_i^G(S_i)$, where G is an element of the power set of $\mathcal{N}$. These concepts can be extended to the case where the beliefs of observer agents or groups of observer agents are not independent.

The ith agent can then put a value for each distribution that the collection of other agents could hold, yielding a value function $\mathbb{V}_i(\mathbb{P}_i(S_i))$. A simple value function is the number of possible states that other agents have not eliminated (i.e. states with nonzero probability). If an observed agent's valuation of privacy treats each observer agent's beliefs independently, we can represent $\mathbb{V}_i(\mathbb{P}_i(S_i)) = \sum_{j \neq i} \mathbb{V}_i^j(\mathbb{P}_i^j(S_i))$ where $\mathbb{V}_i^j(\cdot)$ is the valuations of privacy with respect to the jth observing agent. Given these assumptions and incorporating the number of remaining possible states as a metric, we have

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) = \sum_{j \neq i} \sum_{s_i \in S_i} I_{\{\mathbb{P}_i^j(s_i) > 0\}} \tag{2}$$

where $I_{\{\cdot\}}$ is an indicator function. This is just one possible metric expressed in VPS, and additional metrics for meeting scheduling will be discussed in Section 4.

An agent's privacy loss during a negotiation is the difference in the valuations of the observer agents' beliefs before negotiations and their beliefs after the negotiation. If $\mathbb{P}_{i,0}(S_i)$ is the probability distribution that observer agents attribute to the ith agent before negotiation and $\mathbb{P}_{i,F}(S_i)$ is the probability distribution that observer agents attribute to the ith agent after negotiation, then the privacy loss for the ith agent is

$$\mathbb{V}_i(\mathbb{P}_{i,0}(S_i)) - \mathbb{V}_i(\mathbb{P}_{i,F}(S_i)).$$

The privacy of the system is indicated by some function $f(\mathbb{V}_1, \ldots, \mathbb{V}_N)$ which aggregates the individual valuations of possible states for the entire set of agents. A flow diagram for the VPS framework is displayed in Fig. 1.

Example 2 *Privacy loss in meeting scheduling.* Consider the scenario proposed in Example 1 where the valuation function for all agents are as described in Eq. (2) and the aggregation function is:

$$f(\mathbb{V}_A, \mathbb{V}_B, \mathbb{V}_C) = \mathbb{V}_A(\cdot) + \mathbb{V}_B(\cdot) + \mathbb{V}_C(\cdot).$$

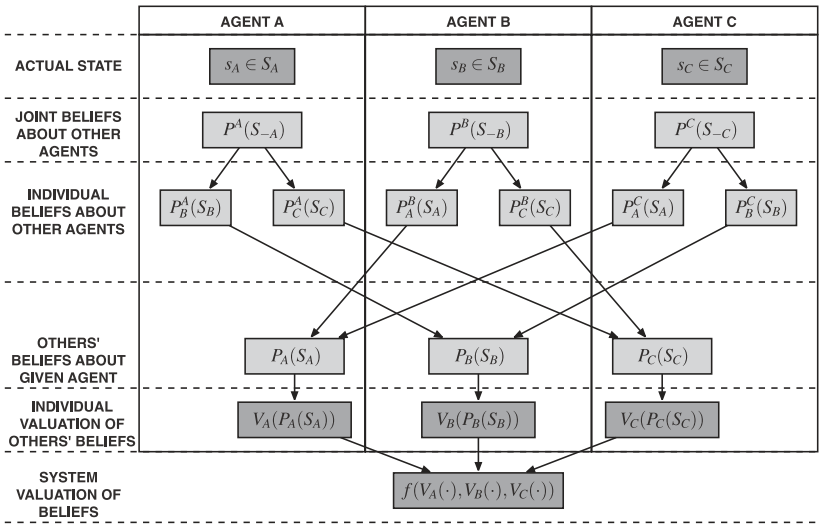

Fig. 1 VPS flow diagram

Let us say that the preferences of the agents are denoted by the states $s_A = s_B = [0\ 1]$ and $s_C = [0\ 0]$. Before a multiagent coordination protocol begins, the agents have no information about the other agents, i.e. each state is equally likely. Thus, agent A's model of agents B and C are captured by $\mathbb{P}^A(S_{-A})$ where $P^A(s_{-A}) = 1/16, \forall s_{-A} \in S_{-A}$, which implies that agent A's model of agent B states $P_B^A(s_B) = 1/4, \forall s_B \in S_B$. The same is true for agent A's model of agent C. With analogous belief structures for agent B and agent C, we can calculate the individual privacy level before negotiation for all agents as 8 (the sum of states with positive probabilites in the beliefs of the two observing agents), and the system privacy level as 24. Let us say that after the protocol has terminated, and inference has occurred by analyzing the exchanged messages, agent A's model of others has been reduced to $P^A([0\ 1\ 0\ 0]) = 1/2$ and $P^A([0\ 0\ 0\ 1]) = 1/2$; agent B's model of others is analogous to that of agent A; agent C has narrowed its belief to $P^C([0\ 1\ 0\ 1]) = 1$. This implies that $P_B^A([0\ 1]) = 1/2$ and $P_B^A([0\ 0]) = 1/2$, with the same distribution for agent A's model of agent C. Again, agent B's model of agent A and agent C are analogous to that of agent A's model of agent B and agent C. Agent C, however, has $P_A^C([0\ 1]) = P_B^C([0\ 1]) = 1$. After negotiation, the valuation of privacy for agents A and B ($\mathbb{V}_A(\mathbb{P}_A(S_A))$ and $\mathbb{V}_B(\mathbb{P}_B(S_B))$, respectively) has now been reduced to 3, while agent C's valuation of privacy is at 4; the system privacy is at 10. Since the individual privacy level varies from 2 to 8, agents A and B have lost $(8-3)/(8-2) = 5/6$ of their privacy, while agent C has lost $(8-4)/(8-2) = 2/3$ of its privacy. The system privacy level varies from 6 to 24, so during this collaboration, the system privacy loss was $(24-10)/(24-6) = 7/9$ of the maximum possible loss.

This example shows how normalization will be a key element of VPS when engaging in cross-metric comparison. In this paper, as in the example, in any fixed scenario, we assume that all agents employ the same valuation function (which may vary over scenarios). The valuation function is homogenous with respect to all observers and the aggregation function is simply the arithmetic average of the individual valuations. Thus, when we measure loss in system privacy, it is the average loss in privacy among all the agents in the system, i.e. the difference between the average privacy level before negotiation and the average privacy level after negotiation. We note that VPS is a theoretical framework for capturing privacy loss. In

practice, agents may not have to computational or storage ability to record and analyze their exchanges in this manner, especially in large-scale systems. However, our goal is to study the privacy loss inherent in the algorithms and thus, ignore complexity and bounded rationality issues that could mask their privacy loss.

2.2. Unification

One of the motivations for introducing VPS was to build a unifying framework for privacy. A successful model must then be able to capture existing notions of privacy. In this section, we show that VPS indeed passes this test by representing three metrics proposed by prominent researchers in the field within our framework. While some of the metrics were expressed quantitatively, by presenting them in VPS, we connect them to a common fundamental framework which facilitates cross-metric comparison.

– In [25], they consider DisCSPs where agents have a cost associated with the revelation of whether some tuple of values (such as a meeting location and time) is feasible (i.e. the agent's user is willing to have a meeting at that place and time). The agents begin exchanging messages and each agent pays a cost if the feasibility of some tuple is fully determined by other agents. This continues until a solution is reached or to the point where the cost of a tuple whose feasibility is about to be fully revealed is greater than the potential reward of the collaboration. If the latter occurs, the negotiation is terminated. Putting this in VPS form, we have S_i is the set of all vectors of length T_i whose components are either 0 or 1, where T_i is the cardinality of all tuples of the ith agent. The ith agent is then characterized by some element $s_i \in S_i$ where $s_i(t)$ denotes the feasibility of tuple t. This metric of privacy can expressed as:

$$\mathbb{V}_i(\mathbb{P}_i^G(S_i)) := \sum_{t=1}^{T_i} c_i(t)\left[I_{\{\mathbb{P}_i^G(S_i^t)=0\}} + I_{\{\mathbb{P}_i^G(S_i^t)=1\}}\right]$$

where $G = \mathcal{N} \setminus i$, $c_i(t)$ is the cost of revealing tuple t, $I_{\{\cdot\}}$ is an indicator function,

$$S_i^t := \{s_i \in S_i : s_i(t) = 0\}, \quad \text{and} \quad \mathbb{P}_i^G(S_i^t) = \sum_{s \in S_i^t} P_i^G(s).$$

Since revelation for the ith agent is considered with respect to information gathered by all other agents G, we consider the joint knowledge of all other agents, $\mathbb{P}_i^G$. The expression for the valuation captures that a cost $c_i(t)$ is paid whenever the feasibility of that tuple has been identified. The expressions inside the indicator functions capture whether a tuple has been identified by seeing if the probability on a tuple being identified as available is zero or one, i.e. anything else would indicate a distribution on more than one possibility.

– In [8], Franzin, Rossi, Freuder and Wallace consider a distributed meeting scheduling problem, where each agent assigns a preference from the discrete set $\{0.1, 0.2, \ldots, 1\}$ to each location/time-slot combination. The measure of privacy loss is entropy with respect to the size of the possible state space that can exist. Thus, in VPS, S_i is the set of all vectors of length $T_i L_i$ where T_i is the number of time slots and L_i is the number of locations, where each component of the vector can take one of 10 values. Privacy metric, which applies entropy to the uncertainty in valuation for each particular location/time-slot combination, can be expressed as:

$$\mathbb{V}_i(\mathbb{P}_i^G(S_i)) := \sum_{k=1}^{T_i L_i} \log_2 \left(\frac{\sum_{j=1}^{10} I_{\{\max_{s_i \in S_i : s_i(k)=j/10} \mathbb{P}_i^G(s_i(k)=j/10)>0\}}}{10} \right)$$

where $G = \mathcal{N} \backslash i$ is the set of all agents except the ith agent as information sharing is part of the assumption in privacy loss. The indicator function in the numerator is because the authors consider whether a particular valuation has been eliminated as viable for a time slot, hence the key difference is whether the probability is positive or zero. When assigning uniform probability over viable time slots, the probability multiplier before the log in the entropy function is eliminated ($\sum_{i=1}^{N}(1/N)\log(1/N) = \log(1/N)$). The 10 in the denominator indicates that all 10 preferences are possible at the beginning of negotiation.

– In Silaghi and Mitra present a privacy model for a setting where each agent has a cost for scheduling a particular meeting at a particular time and location. They propose a model where agents can share information among each other. The privacy metric is the size of the smallest coalition necessary to deduce a particular agent's costs exactly. In VPS, each agent's private information is modeled as an element s_i of the set S_i which is the set of all vectors of length $T_i L_i M_i$ where T_i is the number of time slots, L_i is the number of locations and M_i is the number of meetings. The components of the vector are some elements of a finite set of costs. Even this distinctive model can be captured in VPS as follows:

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) := \min_{G \in \mathcal{G}} |G| \quad \text{where}$$

$$\mathcal{G} := \left\{ G \subset \mathcal{N} : \sum_{s_i \in S_i} P_i^G(s_i) \log P_i^G(s_i) = 0 \right\}.$$

The set $\mathcal{G}$ is the set of all coalitions that have deduced the ith agent's costs exactly. Deducing the costs exactly is identical to saying that the entropy of the knowledge distribution is zero. If the entropy measure on $\mathbb{P}_i^G$ is zero, then the estimate of the group G about the ith agent must be a delta function (all probability on one state) and therefore, the ith agent's state is known exactly by the group G. Alternately, we could define

$$\mathcal{G} := \left\{ G \subset \mathcal{N} : \prod_{s_i \in S_i} (1 - P_i^G(s_i)) = 0 \right\}.$$

The fact that VPS can capture such a diverse set of metrics indicates not only its ability to unify expression of privacy but also that it mathematically represents the basic and intrinsic properties of privacy.

3. Distributed meeting scheduling model

To investigate VPS in a relevant privacy problem, we apply it to a personal assistant agent domain: distributed meeting scheduling. Here, we have an environment where private information must be exchanged to obtain a team-optimal solution, i.e. an optimal schedule. However, we also wish to minimize the privacy lost through inference from the messages sent in any multiagent negotiation/coordination protocol. We present here the distributed multi-event scheduling (DiMES) model presented in [13] that captures many fundamental characteristics of distributed scheduling in an optimization framework.[2] We then describe how we can map

[2] While we choose DiMES as it effectively captures our example domain of meeting scheduling, VPS and the analysis in the following sections are not DiMES-dependent and can be extended to other models [12,21].

the DiMES problem to a DCOP problem, which can be solved by agents on a structure that prevents a priori privacy loss.

3.1. DiMES

The original DiMES model mapped the scheduling of arbitrary resources. Here, DiMES is instantiated to address a meeting-scheduling problem. We begin with a set of people $\mathcal{R} := \{R_1, \ldots, R_N\}$ of cardinality N and an event set $\mathcal{E} := \{E^1, \ldots, E^K\}$ of cardinality K. Let us consider the minimal expression for the time interval $[T_{\text{earliest}}, T_{\text{latest}}]$ over which all events are to be scheduled. Let $T \in \mathbb{N}$ be a natural number and Δ be a length such that $T \cdot \Delta = T_{\text{latest}} - T_{\text{earliest}}$. We can then characterize the time domain by the set $\mathcal{T} := \{1, \ldots, T\}$ of cardinality T where the element $t \in \mathcal{T}$ refers to the time interval $[T_{\text{earliest}} + (t-1)\Delta, T_{\text{earliest}} + t\Delta]$. Thus, a business day from 8 AM–6 PM partitioned into half-hour time slots would be represented by $\mathcal{T} = \{1, \ldots, 20\}$, where time slot 8 is the interval [11:30 AM, 12:00 PM]. Here, we assume equal-length time slots, though this can easily be relaxed.

Let us characterize the kth event with the tuple $E^k := (A^k, L^k; V^k)$ where $A^k \subset \mathcal{R}$ is the subset of people that are required to attend. $L^k \in \mathcal{T}$, is the length of the event in contiguous time slots. In a meeting scheduling, an event is characterized by its attendees, the duration and the importance of the event to its attendees. The heterogeneous importance of an event to each attendee is described in a value vector V^k. If $R_n \in A^k$ (R_n is a required attendee of the kth event), then V_n^k will be an element of V^k which denotes the value per time slot to the nth person for scheduling event k. Let $V_n^0(t) : \mathcal{T} \to \mathbb{R}^+$ denote the nth person's valuation for keeping time slot t free (or committed to its current status). These valuations allow agents to compare the relative importance of multiple events to be scheduled, and also to compare the importance of an event to be scheduled to the current value of a particular time slot.

Given the above framework, we now present the scheduling problem. Let us define a schedule S as a mapping from the event set to the time domain where $S(E^k) \subset \mathcal{T}$ denotes the time slots committed for event k. All people in A^k must agree to assign the time slots $S(E^k)$ to event E^k in order for the event to be considered *scheduled*, thus allowing the people to obtain the utility for attending it. This assumption also could be relaxed in an extended framework.

Let us define a person's utility to be the difference between the sum of the values from scheduled events and the aggregated values of the time slots utilized for scheduled events if they were kept free. This measures the net gain between the opportunity benefit and opportunity cost of scheduling various events. The organization wants to maximize the sum of utilities of all its members as it represents the best use of all assets within the team. Thus, we define the fundamental problem in this general framework as:

$$\max_S \left\{ \sum_{k=1}^{K} \sum_{n \in A^k} \sum_{t \in S(E^k)} \left(V_n^k - V_n^0(t) \right) \right\}$$

such that $S(E^{k_1}) \cap S(E^{k_2}) = \emptyset, \quad \forall k_1, k_2 \in \{1, \ldots, K\}, \; k_1 \neq k_2, \; A^{k_1} \cap A^{k_2} \neq \emptyset$. Intuitively, we want to schedule the meetings that are most important using the least valuable time slots, while making sure that all attendees can attend without creating any conflicts.

3.2. PEAV-DCOP

Given a problem captured by the DiMES framework, we need an approach to obtain the optimal solution. As we are optimizing a global objective with local restrictions (eliminating conflicts in resource assignment), DCOP [17] presents itself as a useful and appropriate approach.

A DCOP consists of a variables set $X = \{x_1, \ldots, x_N\}$ distributed among agents where the variable x_i takes a value from the finite discrete domain D_i. The goal is to choose values for variables to optimize the aggregation of utility functions, each of which depend on the values of a particular subset of variables in X. If all the utility functions depend on exactly two variables, it can be modeled with a graph, where nodes represent variables and utility functions can be captured as edge weights. For each edge $(i, j) \in E$, (where E denotes a set of edges whose endpoints belong to a set isomorphic to X), we have a function $f_{ij}(x_i, x_j) : D_i \times D_j \to \mathbb{R}$. Our goal is to choose an assignment $a^* \in A := D_1 \times \cdots \times D_N$, such that $a^* = \arg\max_{a \in A} \sum_{(i,j) \in E} f_{ij}\left(x_i = a_i, x_j = a_j\right)$. Figure 2 shows a DCOP structure where each variable must choose from an identical two-value domain and the global objective function is captured through four edges with identical constraint utility functions.

Our challenge is to convert a given DiMES problem into a DCOP. We choose to convert to a DCOP with binary constraints as all prominent fully decentralized algorithms in the community depend on a binary graph. We may then apply any type of algorithm developed for DCOP to obtain a solution. In [13], three DCOP formulations for DiMES were proposed. The formulation chosen will affect privacy loss because each variable must have knowledge of the constraint utility function between it and all other connected variables. These constraint utility function may contain private information depending on the formulation.

Example 3 Consider the scenario with three agents ($\{A, B, C\}$) trying to schedule two meetings where the attendees for each meeting are $A^1 = \{A, B\}$ and $A^2 = \{B, C\}$. The EAV (Events As Variables, where there is one variable in the DCOP for each meeting) and PEAV (Private Events As Variables, where the DCOP has multiple variables for each meeting, one for each attendee) DCOP graphs for this problem are shown in Fig. 3. In the EAV formulation, we have a variable representing E^1, and one agent would have to reveal all its private information to another (say B to A). Similarly for E^2, B or C would have to obtain all private

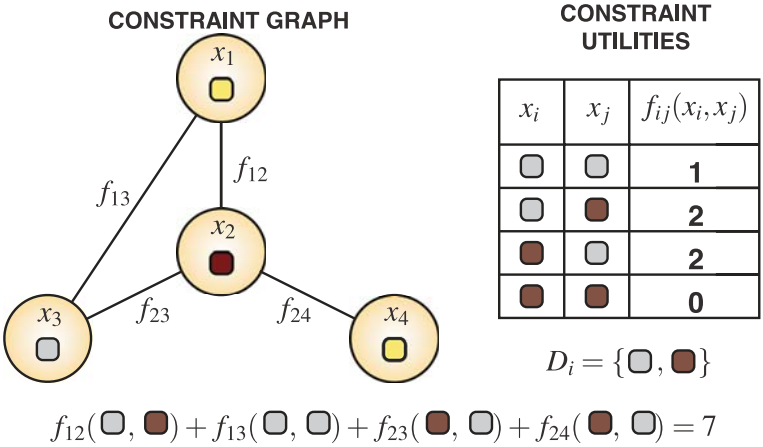

Fig. 2 DCOP structure

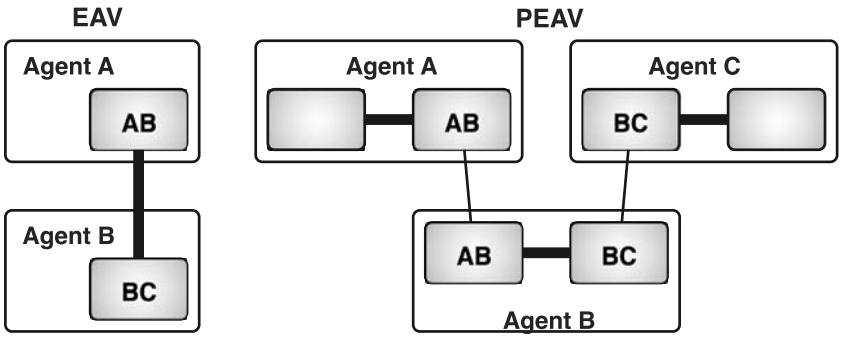

Fig. 3 EAV and PEAV DCOP graphs for AB and BC events

information about each other. Furthermore, since this information would have to lie on the constraint between the two variables, both agents controlling each variable would have full knowledge of all private information before any DCOP algorithm even began. If for E^1, agent A revealed its private information to agent B, and for E^2, agent C revealed its private information to agent B, then it would be identical to a centralized solution. However, in the PEAV formulation, because each agent creates its own variable for each event, private information can be stored on internal constraints while inter-agent constraints simply enact a large penalty if event times disagree and are zero otherwise. Formulations analogous to PEAV have been utilized by others for the sake of privacy when investigating meeting scheduling in DisCSP formulations [16,18].

To utilize PEAV, we occasionally have unary constraints for agents who are attendees for only one event. Thus, we create a "dummy" variable which takes on a single value, to ensure having the private information on an internal constraint. As we are investigating privacy, we choose the PEAV formulation, which was created such that there would be no loss of private information prior to negotiation. The details of constructing the PEAV constraints are discussed in [13]. Thus, given events and values, we are able to construct a graph and assign constraint link utilities from which a group of personal assistant agents can apply a DCOP algorithm and obtain an optimal solution to the DiMES problem.

4. Privacy

In this section, we address privacy loss when applying DCOP algorithms to scheduling problems in the DiMES model. We first generate several instantiations of valuations to quantitatively measure privacy levels and express them in the VPS framework. In addition, VPS helps identify dimensions by which we can map and compare the various metrics. We then discuss how privacy is lost due to the mechanics of the DCOP algorithms, including details about how inference is conducted on messages passed in a distributed protocol. We use algorithms from three areas: a centralized one, a partially centralized one, and one that attempts full distribution. We select OptAPO [15] from the partially centralized space because it is the primary and prominent member of that class. For the distributed case, several candidates are available. We focus on SynchBB [11] because (i) as a synchronous algorithm with fewer messages than its asynchronous counterparts, it provided an illustrative testbed for privacy loss analysis; (ii) its fewer messages were hypothesized to lead to lower privacy loss; (iii) it was simpler to express the impact of uncertainty and quantify inference.

4.1. Privacy metrics for PEAV-DCOPs for DiMES

The initial task is to identify the information that an agent should consider private, i.e. the data that identifies the state of its human user. In DiMES, it is clear that the valuation of time, $V_i^0(t)$, explicitly captures the preferences that will be used in the collaborative process. Users may wish to keep these preferences private as it may reveal whether there are important individual activities going on in particular time slots, or because it reveals their preferences for working early or working late. In addition, the rewards for attending various events $\{V_i^k : i \in A^k\}$ is another component that agents may wish to keep private. For the sake of simplicity, we will assume a setting where event rewards are public, though the analysis can be extended to capture situations where this information is private (if the event rewards are private, our analysis could assume that these rewards take values over some known set with some given a priori distribution). If $V_i^0(t) \in \mathcal{V}$ where $\mathcal{V}$ is a discrete set and there are T time slots in a schedule, the state s_i of the ith agent is an element of the set $S_i = \mathcal{V}^T$ and can be expressed as a vector of length T. This is because users have assigned a valuation from $\mathcal{V}$ to each of their T time slots based on their preferences.

Before negotiation, each agent knows only that each of the other agents exist in one of $|\mathcal{V}|^T$ possible states. After negotiation, each agent will be modeled by all other agents whose estimate of the observed agent is captured by $\mathbb{P}_i(S_i)$. The question now is how an agent should assign values to these estimates of possible states through which others see it. The method introduced in [25] does not apply here because we are not in a pure satisfaction setting and the method in [26] is not viable because information sharing is not an appropriate assumption in this domain.

4.1.1. Entropy-based metrics

We do consider the entropy-based metric introduced in [8] and captured in VPS in Section 2.2. We remove the factor L_i that captures location and adjust to account for privacy loss to lack of information sharing:

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) := \sum_{j \neq i} \sum_{k=1}^{T} \log_2 \left(\frac{\sum_{m=1}^{|\mathcal{V}|} I_{\{\max_{s_i \in S_i : s_i(k)=m} \mathbb{P}_i^j(s_i(k)=m)>0\}}}{|\mathcal{V}|} \right). \tag{3}$$

We extend this to the case where entropy is applied to the distribution over the entire schedule as opposed to time slot by time slot. The "entire schedule" case has a single joint distribution over each possible valuation assignment for all time slots, while the "time slot by time slot" case is an aggregation of distributions for each time slot, where each distribution has support over a single time slot. In this case, we have

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) := \sum_{j \neq i} \log_2 \left(\frac{\sum_{m=1}^{|\mathcal{V}|^T} I_{\{\mathbb{P}_i^j(s_m)>0\}}}{|\mathcal{V}|^T} \right). \tag{4}$$

Using entropy, it is possible for the privacy loss to get arbitrarily high as the number of initial states increases (due to T or $|\mathcal{V}|$). To facilitate cross-metric comparison, we shift and normalize each metric $\tilde{\mathbb{V}} = 1 + \alpha \mathbb{V}$, with an appropriate constant α so that the valuation for the worst-case privacy level, i.e. the case where the entire schedule is known, is zero and the ideal level is one.

4.1.2. Proportional metrics

Due to the nature of the messaging in DCOPs, the most typical form of information gathered is the elimination of a possible state. Thus, a straightforward choice for $\mathbb{V}_i$ would be

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) = \sum_{j \neq i} \sum_{s_i \in S_i} I_{\{\mathbb{P}_i^j(s_i) > 0\}} \tag{5}$$

which can be extended to a time-slot-by-time-slot version:

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) = \sum_{j \neq i} \sum_{k=1}^{T} \sum_{m=1}^{|\mathcal{V}|} I_{\{\max_{s_i \in S_i : s_i(k)=m} \mathbb{P}_i^j(s_i(k)=m) > 0\}} \tag{6}$$

where $I_{\{\cdot\}}$ is an indicator function. The first essentially aggregates the number of states that have not been eliminated by an observing agent in the system. The second aggregates the number of valuations (per time slot) that have not been eliminated. We can scale both functions with a transformation of the form $\tilde{\mathbb{V}} = \alpha(\mathbb{V} - \beta)$ with appropriate choices of α and β such that the valuations span [0 1] with zero being the worst level and one being the ideal level of privacy.

4.1.3. State-guessing metrics

We note that the metrics above are linear functions in possible states. Consider when one agent has been able to eliminate one possible state of another agent. The observed agent may not value that loss equally if the observer went from 1000 states to 999, as opposed going from 2 to 1. To address this idea, we introduce the following nonlinear metrics for privacy:

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) = \sum_{j \neq i} \left(1 - \frac{1}{\sum_{s \in S_i} I_{\{\mathbb{P}_i^j(s) > 0\}}} \right) \tag{7}$$

and its per-time-slot analogue:

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) = \sum_{j \neq i} \sum_{k=1}^{T} \left(1 - \frac{1}{\sum_{m=1}^{|\mathcal{V}|} I_{\{\max_{s_i \in S_i : s_i(k)=m} \mathbb{P}_i^j(s_i(k)=m) > 0\}}} \right). \tag{8}$$

These valuations model privacy as the probability that an observer agent will be unable to guess the observed agent's state accurately given that their guess is chosen uniformly over their set of possible states for the observed agent. For the first, the other agents are trying to guess the entire schedule accurately while in the second they are guessing time slot by time slot. Again, we can scale both functions with a transformation of the form $\tilde{\mathbb{V}} = \alpha(\mathbb{V} - \beta)$ with appropriate choices of α and β such that the valuations span [0 1] with zero being the worst level and one being the ideal level of privacy. We note that all the metrics here can be written as

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) = \sum_{j \neq i} \mathbb{V}_i^j(\mathbb{P}_i^j(S_i))$$

where $\mathbb{V}_i^j(\cdot)$ represents the ith agent's valuation of privacy loss to the jth agent. Because of the normalization of an agent's privacy level to fall within [0 1] and the fact that each observer agent contributes equally to privacy level, the privacy level calculated from any single observer agent falls within $[0 \ \frac{1}{(N-1)}]$. We refer to the metrics presented here as EntropyTS (3), EntropyS (4), ProportionalS (5), ProportionalTS (6), GuessS (7) and GuessTS (8) where the numbers in parentheses refer to the equations that characterize them.

4.1.4. Classification

The advantage of the VPS framework is that we can take these six metrics and map them into a common space which allows us to identify dimensions along which we can classify privacy metrics. This can be useful in discovering new directions for generating metrics and uncovering gaps in existing metrics. In our case, there are two distinct dimensions that appear: (i) state-space partitioning, and (ii) degree of nonlinearity. The first dimension is revealed in the bifurcation between judging privacy by schedule versus by time slot. In VPS, this amounts to deciding how to partition the state space into independent supports on which privacy loss will be evaluated. For privacy by schedule, we take the entire state space, while for privacy by time slot, we filter the state space to evaluate the privacy loss in time slot independently. Thus, for this and other domains, identifying the independence in state space is a dimension to classify or generate privacy metrics. The second dimension is the degree of nonlinearity of the valuation applied to the part of the state space being evaluated. In our case, we have three types of valuations: linear (ProportionalS, ProportionalTS), asymptotic (GuessS, GuessTS) and logarithmic (EntropyS, EntropyTS). A table showing a classification of the metrics generated here along the dimensions mentioned above are displayed in Fig. 4. Another key principle whose identification was facilitated by VPS was the importance of normalization in cross-metric comparison. Using our normalization to [0 1], we display the degree of nonlinearity of the various functional types in Fig. 5. We see that as the number of states increase, the effects of the relative nonlinearities increase as well.

Finally, we must determine which function to use for the system privacy level. For simplicity, in this paper, we choose the arithmetic mean of the individual privacy levels, i.e.:

$$f(\mathbb{V}_1, \ldots, \mathbb{V}_N) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{V}_i(\cdot).$$

This reflects the notion that the privacy of all agents are equally valued in the system. Heterogeneous importance of agents could be addressed easily with a weighted sum over individual valuations.

		PARTITION OF STATE SPACE	
		Entire State Space	Time Slot by Time Slot
DEGREE OF NONLINEARITY	Linear x	ProportionalS	ProportionalTS
	Asymptotic $1 - 1/x$	GuessS	GuessTS
	Logarithmic $\log(x)$	EntropyS	EntropyTS

Fig. 4 Classification by metrics by VPS dimensions

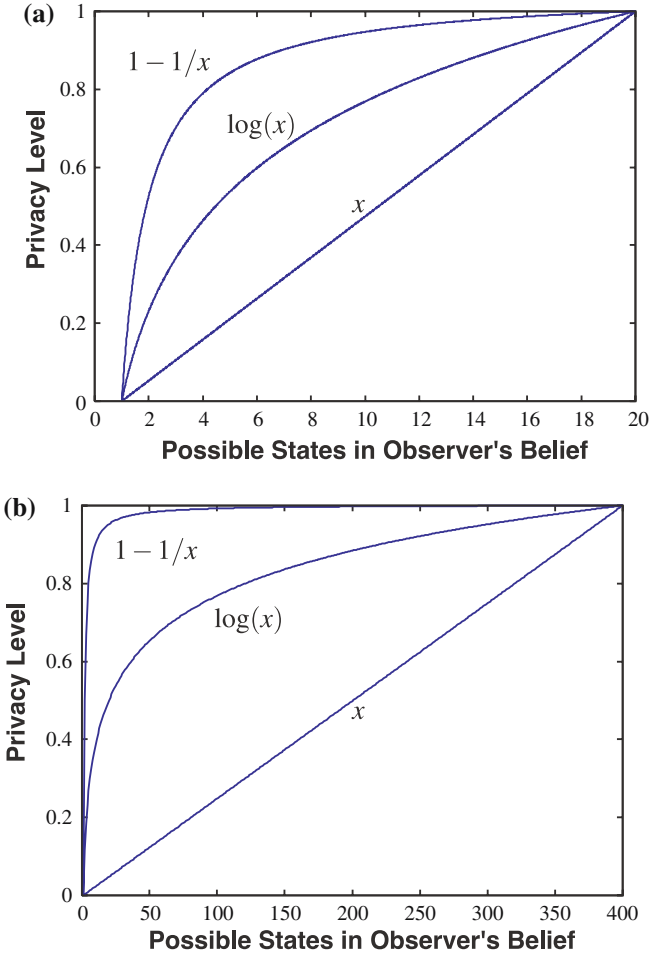

Fig. 5 Degree of nonlinearity of metrics for (**a**) 20 states and (**b**) 400 states

4.2. Privacy loss in DCOP algorithms

We now apply these metrics to DCOP algorithms. We consider the *partially centralized algorithm* OptAPO and the *distributed* algorithm SynchBB [11]. In addition, we address a baseline that is missing in much privacy analysis, which is the *centralized* solution. As detailed earlier, one of the main arguments for distributed solutions is the need to protect privacy. However, by ignoring centralization in analysis, the implicit assumption is that decentralization would automatically yield better privacy protection. Consequently, it is important to identify if and when this argument is justified. The metrics generated under the VPS framework give us an opportunity to compare various classes of protocols (centralized, partially centralized, decentralized) for a relevant problem (meeting scheduling) in a quantitative and rigorous manner.

4.2.1. Centralized

In an N-agent example, before any information is exchanged, all observers' models of an observed agent will not be able to eliminate any states, i.e., $\mathbb{V}_i^j(\cdot) = 1/(N-1)$ due to our normalization. Thus, the privacy level of all agents will be $\mathbb{V}_i(\cdot) = 1$. Then, the system level of privacy will also be $f(\cdot) = 1$. In a centralized solution, one agent will receive a set of utility change values $\{Delta_i^k(t)\}$, where i denotes the agent, k denotes the event and t denotes the time slot. Each utility change value, $\Delta_i^k(t) = V_i^k - V_i^0(t)$, represents the utility gain or decrease to the ith agent for scheduling the kth event at time t. Because the event rewards, $\{V_i^k\}$ are public, the centralized agent can calculate $\{V_i^0(t)\}$ for all other agents (if event rewards weren't public, we could apply some probabilistic analysis). After this solution is reached, the central agent's privacy level remains at $\mathbb{V}_{j^*}(\cdot) = 1$, as he has not revealed anything. For all other agents, they have revealed their state exactly to one agent while the remaining $N-2$ observer agents have exactly the same knowledge as they had before information was exchanged. Then, $\mathbb{V}_i^{j^*}(\cdot) = 0$ and $\mathbb{V}_i^j(\cdot) = 1/(N-1)$ for $j \neq j^*$. The privacy level of these agents after centralization is:

$$\mathbb{V}_i(\cdot) = \mathbb{V}_i^{j^*}(\cdot) + \sum_{j \neq i, j^*} \mathbb{V}_i^j(\cdot) = 0 + \frac{N-2}{N-1} = \frac{N-2}{N-1}.$$

The system level of privacy of all agents after centralization is:

$$f(\cdot) = \frac{1}{N}\left(\mathbb{V}_{j^*}(\cdot) + \sum_{i \neq j^*} \mathbb{V}_i(\cdot)\right) = \frac{1}{N}\left(1 + (N-1)\frac{N-2}{N-1}\right) = \frac{N-1}{N}$$

which implies the privacy loss due to centralization is $1/N$. This is the case for all the metrics we presented as they were all normalized to have the same ranges. Thus, $1/N$ is the baseline by which we must evaluate privacy loss in other algorithms.

Example 4 For the problem considered in Example 3 given our normalization, an observed agent has potentially one unit of privacy to lose in total, which translates to losing $1/(N-1) = 1/2$ to each observing agent, if they discover the exact state of the observed agent. If we use centralization, with agent B as the central agent, agent A will lose 1/2 to agent B, and 0 to agent C; agent C will lose 1/2 to agent B and 0 to agent A; agent B will not lose any privacy. The system privacy level after centralization is then $[V_A(\cdot) + V_B(\cdot) + V_C(\cdot)]/3 = [1/2 + 1 + 1/2]/3 = 2/3$ which is a privacy loss of $1/3 = 1/N$.

4.2.2. Partially centralized: OptAPO

As in the centralized case, due to our normalization the system level of privacy before the protocol has started is $f(\cdot) = 1$. In Optimal Asynchronous Partial Overlay (OptAPO) [15], there is an initial phase where agents exchange their constraints with all their neighbors. In our case, these constraints contain utility change information ($\{\Delta_i^k(t)\}$ described earlier) from which the private valuations of time can be deduced given public knowledge of event rewards ($\Delta_i^k(t) = V_i^k - V_i^0(t)$). The dynamics of OptAPO after the initial phase are not deterministically predictable. It is possible that by the end, all agents will be able to learn each others' preferences. However, it is also possible that the privacy loss may remain at the same level as after the initial phase. For purposes of analysis, we will assign to OptAPO the privacy loss after the initial phase, which is a lower bound on actual privacy loss.

Let L_i denote the set of agents who have links with ith agent in the DCOP graph, i.e., there exists a constraint between some variable for the ith agent and some variable for the jth agent for all $j \in L_i$. After the initial phase of OptAPO, we have $\mathbb{V}_i^j(\cdot) = 0$ for $j \in L_i$ and $\mathbb{V}_i^j(\cdot) = 1/(N-1)$ for $j \in \{L_i^C \setminus i\}$, where L_i^C is the set complement of L_i (we remove the element i because an agent does not measure privacy loss with respect to itself). This yields the privacy level of the ith agent after the initial phase of OptAPO as:

$$\mathbb{V}_i(\cdot) = \sum_{j \in L_i} \mathbb{V}_i^j(\cdot) + \sum_{j \in L_i^C, j \neq i} \mathbb{V}_i^j(\cdot) = \sum_{j \in \{L_i^C \setminus i\}} \frac{1}{N-1} = \frac{|\{L_i^C \setminus i\}|}{N-1},$$

which states the privacy level is the percentage of observer agents who are not neighbors of the ith agent. Then the system privacy level after the initial phase of OptAPO is

$$f(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{V}_i(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \frac{|\{L_i^C \setminus i\}|}{N-1} \leq \frac{1}{N} \sum_{i=1}^{N} \frac{N-2}{N-1} = \frac{N-2}{N-1}$$

where the inequality is because at best each agent will lose all information to only one neighbor. Because $(N-2)/(N-1) < (N-1)/N$ for $N > 1$, we have that OptAPO will always have worse privacy loss than centralization. Due to the normalization, this will be true for all the metrics we presented. Thus, if privacy protection is the main concern for a group of agents, it would be better for them to use a centralized solution rather than use OptAPO.

Example 5 Let us consider the problem in Example 3 when using OptAPO under the PEAV formulation shown in Fig. 3. As in the centralized case, given our normalization, an observed agent has potentially one unit of privacy to lose in total, which translates to losing $1/(N-1) = 1/2$ to each observing agent, if they discover the exact state of the observed agent. In the initial phase of OptAPO, agents will share their internal constraints with their neighbors. Thus, agent A will lose 1/2 to agent B, and 0 to agent C, and agent C will lose 1/2 to agent B and 0 to agent A, which is the same as centralization. However, in OptAPO, agent B will lose 1/2 to agent A and 1/2 to agent B. The system privacy level after OptAPO will then be at most $[V_A(\cdot) + V_B(\cdot) + V_C(\cdot)]/3 = [1/2 + 0 + 1/2]/3 = 1/3$, (assuming no more information is gained after the initial phase), which is a privacy loss of (at least) 2/3.

One reason for this phenomenon is that OptAPO was designed with speed of convergence as opposed to privacy in mind. However, it is important to note that the (partial) decentralization did not by itself protect privacy. We note that for more complex problems where there are multiple intra-agent constraints, it may be possible to prevent full privacy loss in the initial phase of OptAPO. Also, we note that our metric weights privacy loss equally with regard to the agent to whom privacy was lost. In some situations, where the weights are heterogeneous (an agent would prefer to tell certain agents about their preferences over other agents) and the central agent is chosen poorly, OptAPO may yield lower privacy loss than a centralized solution.

4.2.3. Decentralized: SynchBB

An algorithm used for solving constraint satisfaction and optimization problems in a distributed setting is Synchronous Branch and Bound (SynchBB) [11]. This approach can be characterized as simulating a centralized search in a distributed environment by imposing synchronous, sequential search among the agents. First, the constraint structure of the problem is converted into a chain. Synchronous execution starts with the variable at the root

selecting a value and sending it to the variable next in the ordering. The second variable then sends the value selected and the cost for that choice to its child. This process is repeated down to the leaf node. The leaf node, after calculating the cost for its selection, would send back the cost of the complete solution to its parent, which in turns uses the cost to limit the choice of values in its domain. After finding the best possible cost with its choices, each variable communicates with its parent and the process continues until all the choices are exhausted. As can be seen from above, branch and bound comes into effect when the cost of the best complete solution obtained during execution can be used as a bound to prune out the partial solutions at each node.

The loss of privacy in using SynchBB occurs by the inferences that variables in the chain can make about other variables in the chain through the cost messages that are passed. Determining these inferences and the consequent elimination of possible states is more complex in tree-like algorithms such as SynchBB, due to the partial and aggregated nature of information. In the following subsections, we discuss these inference processes in the cases where the agents are aware of the structure of the chain and also when the agents are not aware of the chain structure. While this inference is specific to the PEAV representation, it is illustrative of the types of inferences that may be feasible in SynchBB or graph-based algorithms in general. It is important to know that we cannot ensure that we are making all the possible inferences, as one could use domain information or more detailed exploitation of the algorithm to eliminate possible states. Thus, the privacy loss due to the inferences presented here for SynchBB represents a lower bound on actual privacy loss (as was the case for OptAPO).

4.2.4. Inference rules for SynchBB with graph knowledge

SynchBB requires that the DCOP graph be converted into a chain. If the process of conversion allows the agents to know the structure of the entire chain, they can employ this information when making inferences about other agents. The agents upstream know the agents involved in the DCOP downstream (and vice versa) and their variables in the chain, but as discussed earlier, agents do not know the valuations on time slots, $\{V_i^0(t)\}$, of other agents. In a SynchBB chain, a downstream message (from a parent to a child) denoted m_i^d where i is the agent receiving the message (in this case, the child), reports the context (instantiated values of variables) above the child and the associated partial utility of the solution.[3] An upstream message (from child to parent) reports only the best attainable utility for the entire chain given over all the contexts sent down by the parent. Thus, if a child sends up a utility value that is identical to a previous message, then the best attainable utility for the current context is less than or equal to the best attainable utility previously reported. Let us denote the associated partial utility reported in a downstream message as m_i^d where i is the agent receiving the message, i.e. the child. Let us denote the associated partial utility reported in an upstream message (which can be calculated by subtracting the partial utility from the downstream message from the total utility reported in the upstream message) as m_i^u where i again is the agent receiving the message, i.e. the parent. These messages are aggregations of utilities on constraints in the chain.

In a PEAV formulation, inter-agent links contribute either zero or a large negative utility to the aggregation. If the latter occurs, it will obfuscate any private information as an agent receiving a message with a large negative utility will not know how much to offset the value. Thus, any message that has a large negative value contains only the information

[3] Our discussion and analysis of SynchBB will use utility maximization principles even though SynchBB is implemented as a cost minimizing algorithm, as we can map one formulation to the other.

that a conflict has occurred. Because it is difficult to make inference when the penalties for conflicts can be arbitrarily picked (as long as they are sufficiently high), we will focus on the inference that can be done with messages that do not have conflict penalties included in the aggregates. A solution where no new meeting are scheduled has a utility of zero (as it does not change the previous value of time). Thus, there will be no conflicts in the final solution. If such a conflict occurs during negotiation, SynchBB will continue with more sets of values without conflicts. Thus, it is appropriate to just focus on inference on these messages that do not include conflict penalties. As mentioned earlier, this will give us a lower bound on the privacy loss in SynchBB.

The utility on the intra-agent links of the ith agent is the sum of the differences between the value gained for scheduling an event and the value of the time where it was scheduled, i.e.

$$\delta_i = \sum_{k \in \{E^k : i \in A^k\}} \Delta_i^k(t_k)$$

where δ_i is the change in utility due to the given schedule (captured by $\{t_k\}$), A^k are the attendees for the kth event E^k, and $\Delta_i^k(t_k) = V_i^k - V_i^0(t_k)$ is the utility change associated with scheduling E^k at time t_k.[4] It is these changes in utilities that are aggregated to form the upstream and downstream messages, which can use for inference with the following relationships:

$$m_i^d = \sum_{j \in \mathcal{A}(i)} \delta_j = \sum_{j \in \mathcal{A}(i)} \sum_{k \in \{E^k : j \in A^k\}} \Delta_j^k(t_k)$$

$$m_i^u = \sum_{j \in \mathcal{D}(i)} \delta_j = \sum_{j \in \mathcal{D}(i)} \sum_{k \in \{E^k : j \in A^k\}} \Delta_j^k(t_k),$$

where $\mathcal{A}(i)$ and $\mathcal{D}(i)$ denote the ancestor and descendent agents of the ith agent in the chain, respectively. We note that if the m_i^u is calculated from taking a value of the complete chain utility and subtracting m_i^d. If m_i^u is strictly greater than previously reported then the relationship above holds, however if there is no improvement, then we have

$$m_i^u \geq \sum_{i \in \mathcal{D}(i)} \sum_{k \in \{E^k : i \in A^k\}} \Delta_i^k(t_k).$$

Thus, every upstream message contains information of the form

$$\sum_{j \in \mathcal{J}} \Delta_{R_j}^{M_j}(t_{M_j}) = m_i^u \quad \text{or} \quad \sum_{j \in \mathcal{J}} \Delta_{R_j}^{M_j}(t_{M_j}) \leq m_i^u$$

where $\mathcal{J}$ is an index set, R_j is an attendee and M_j is an event. Every downstream message contains information of the form of the upstream equation with strict equality. By making the substitution

$$\Delta_{R_j}^{M_j}(t_{M_j}) = V_{R_j}^{M_j} - V_{R_j}^0(t_{M_j}),$$

we can transform the upstream message with inequality information to

[4] We assume that $\Delta_i^k(0) = 0$, i.e. choosing not to schedule a meeting ($t_k = 0$) does not change utility.

$$\sum_{j\in\mathcal{J}} V^0_{R_j}(t_{M_j}) \geq \sum_{j\in\mathcal{J}} V^{M_j}_{R_j} - m^u_i =: c^u_i$$

where c^u_i is defined as the right-hand side of the inequality above which is a constant given the message and our knowledge of $\{V^{M_j}_{R_j}\}$. Adding the complexity that t_{M_j} may not be known, we have

$$\sum_{j\in\mathcal{J}_1} V^0_{R_j}(t_{M_j}) + \sum_{j\in\mathcal{J}_2} V^0_{R_j}(\tilde{t}_{M_j}) \geq c^u_i \tag{9}$$

where $\mathcal{J}_1$ is an index set for known event-agent pairs, $\mathcal{J}_2$ is an index set for unknown event-agent pairs and $\tilde{t}_{M_j}$ indicates an unknown time. Similar transformations can be made for other messages. Thus, an agent can take every message it receives, turn it into an equation of the form in Eq. (9), and use the set of these equations to prune out possible states.

Example 6 In Fig. 6, we show the messages and the resulting inference equations for the problem in Example 3, where the chain is formed with agents in the order $A - B - C$. The downstream messages yield equality inferences and the upstream messages yield both inequality inferences (as shown) and equality inferences, depending on the value of the message with respect to previous messages.

We see through the inference relationships in this example how privacy protection in SynchBB can be enhanced. By passing down the entire context as opposed to the relevant context, agent C is aware of t_{AB} even though it plays no role in the calculation of δ_C. If agents (or variables) passed down contexts (i.e. meeting times) only to those agents who need to know it, we would have $t_{AB} \rightarrow \tilde{t}_{AB}$, meaning that agent C would not be aware of the time that agents A and B are considering for their meeting, which would result in the weaker inference:

$$m^d_C \;\Rightarrow\; V^0_A(\tilde{t}_{AB}) + V^0_B(\tilde{t}_{AB}) + V^0_B(t_{BC}) = c^d_C := V^{AB}_A + V^{AB}_B + V^{BC}_B - m^d_C.$$

The effects of this improvement were investigated and the results are shown and discussed in Section 5.

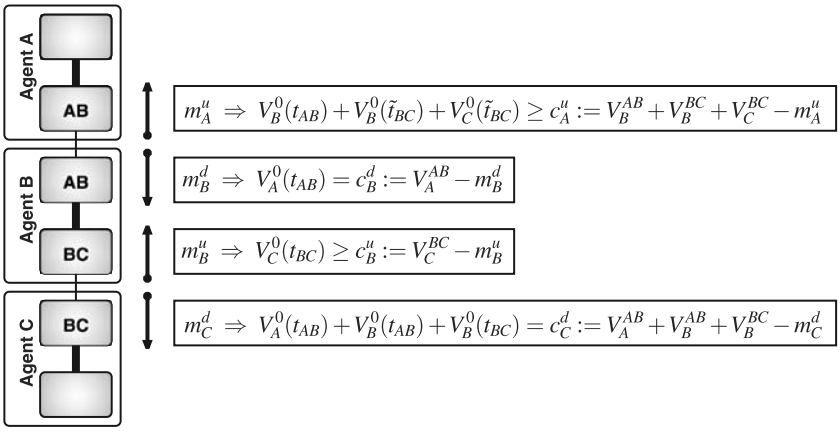

Fig. 6 Example of inference in SynchBB with graph information

4.2.5. Inference rules for SynchBB without graph knowledge

Let us now consider the case where, when converting the DCOP graph into a chain, the structure of the graph is not revealed to the agents. While the structure is not fully known, agents have a bound K on the number of terms $\{\Delta_{R_j}^{M_j}(t_{M_j})\}$ that could exist in the chain because of domain knowledge about limits on numbers and types of meetings that could be scheduled simultaneously. While agents do not know all of the Δ-terms that exist in the chain, they are aware of some of them because of the meetings for which they are attendees. Agents know that other attendees must have Δ-terms for these meetings and agents are also aware of whether these terms are above them or below them in the chain for communication purposes.

As opposed to the case where the entire graph structure is known, we cannot account for all the Δ's in the upstream and downstream messages by by adding all the Δ's for the upstream and downstream agents. Let $\mathcal{K}_i^u$ denote the set of Δ's known to be part of upstream messages (i.e. from variables below the ith agent) due to common meetings, $\mathcal{K}_i^d$ denote the set of Δ's known to be part of downstream messages (i.e. from variables above the ith agent) due to common meetings and $\mathcal{K}_i$ denote the set of Δ's within the ith agent.[5] Then $\widehat{K}_i := K - |\mathcal{K}_i^u| - |\mathcal{K}_i^d| - |\mathcal{K}_i|$ is the number of potential Δ's that may exist outside the scope of the ith agent's knowledge. Thus, given that the bound on the total number of Δ's in the entire chain is K, then the number of potential Δ's that are unknown to the ith agent is $\widehat{K}_i$.

To prevent making inaccurate inference, the ith agent must assume that messages imply relationships of the form:

$$m_i^u = \sum_{j \in \mathcal{J}_i^u} \Delta_{R_j}^{M_j}(t_{M_j}) + \sum_{j=1}^{\widehat{K}_i} \Delta_j \tag{10}$$

and

$$m_i^d = \sum_{j \in \mathcal{J}_i^d} \Delta_{R_j}^{M_j}(t_{M_j}) + \sum_{j=1}^{\widehat{K}_i} \Delta_j \tag{11}$$

where the utility changes take values $\Delta_j \in \{V_{\min}^k - V_{\max}^0, \ldots, V_{\max}^k - V_{\min}^0\}$ if the meeting valuations take values $V_i^k \in \{V_{\min}^k, \ldots, V_{\max}^k\}$ and the private valuations of time take values $V_i^0(t) \in \{V_{\min}^0, \ldots, V_{\max}^0\}$. Again, we must consider the possibility that the upstream relationship in Eq. (10) is an inequality. Let us consider first the case, where Eq. (10) is an equality due to the report of a strictly greater chain utility. By making the appropriate substitutions and using the limits of Δ_j, we have

$$\sum_{j \in \mathcal{J}_i^u} V_{R_j}^0(t_{M_j}) \geq \widehat{K}_i(V_{\min}^k - V_{\max}^0) + \sum_{j \in \mathcal{J}_i^u} V_{R_j}^{M_j} - m_i^u \tag{12}$$

and

$$\sum_{j \in \mathcal{J}_i^u} V_{R_j}^0(t_{M_j}) \leq \widehat{K}_i(V_{\max}^k - V_{\min}^0) + \sum_{j \in \mathcal{J}_i^u} V_{R_j}^{M_j} - m_i^u. \tag{13}$$

Thus, an equality relationship from m_i^u due to an improvement from variables below the ith agent yields two inference equations: a lower bound and an upper bound. If it was the

[5] We assume implicitly that all variables within a single agent will be connected sequentially. While we can modify our results for chains where this is not the case, for privacy protection, it is prudent to minimize messaging. Sequential ordering of agent variables ensures no more than two outgoing messages per agent.

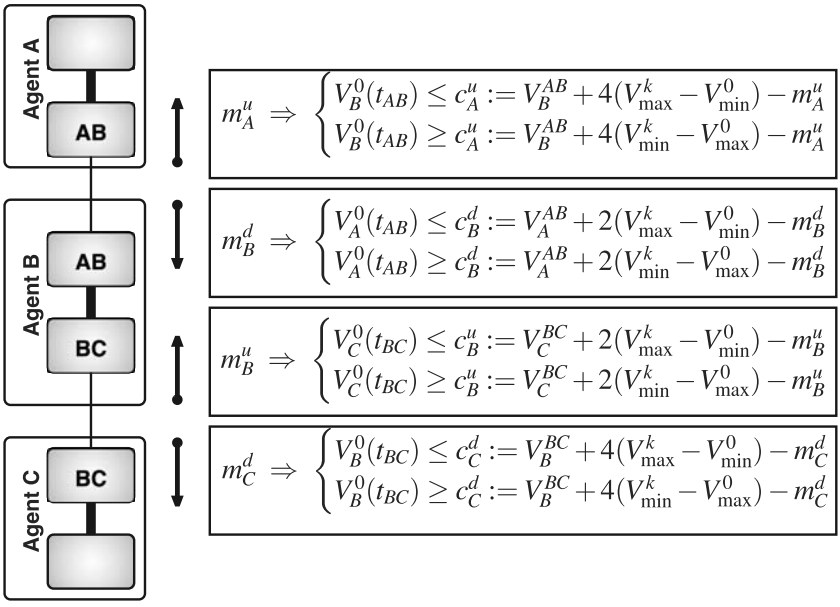

Fig. 7 Example of inference in SynchBB without graph information

case that the upstream message implied an inequality in Eq. (10), then we would only have Eq. (12), the lower bound, as the sole resulting inference equation.

Example 7 In Fig. 7, we show the messages and inferences for the chain in Example 6. Both the lower bound and potential upper bound that could result are displayed for upstream messages. We note that $\widehat{K}_A = 4$, $\widehat{K}_B = 2$ and $\widehat{K}_C = 4$.

5. Experiments

In this section, we describe the experimental domain detailing various scenarios which are modeled as PEAV-DCOPs for DiMES problems. We then solve these problems utilizing various DCOP algorithms for which we present and analyze privacy loss with respect to the metrics generated in Section 4. Implications of the choice of algorithm and metric along with phenomena such as uncertainty and collusion are discussed. This paper provides the most thorough empirical investigation of privacy loss in DCOP, with a total of 39,000 measurements over 6500 separate simulations taken according to six privacy metrics over seven meeting scenarios, using various combinations of environmental parameters.

5.1. Experimental domains

The majority of scheduling instances in a functional personal assistant agent system will consist of a small number of meetings that need to be negotiated simultaneously. This notion of a small number of meetings is also shared in the work motivated by [7,18], and as members of a research organization, this is the situation that the authors commonly observe. While larger-scale problems may present themselves, if privacy is a critical factor, the coordination

protocols must be effective for these small-scale instances. The instantiations of the DiMES problems that we investigated are described below:

- *Agents*: We consider scenarios where there are either three ($\mathcal{R} = \{A, B, C\}$) or four ($\mathcal{R} = \{A, B, C, D\}$) personal assistant agents, each representing a single user, whose joint task is to schedule a set of events (meetings).
- *Events*: We consider seven scenarios. For simplicity, all events last for one time slot. The attendee sets for the meetings in each scenario are as follows:
  - Scenario 1: $\{AB, BC\}$
  - Scenario 2: $\{AB, BC, AC\}$
  - Scenario 3: $\{ABC, BC\}$ with chain order $A - B - C$
  - Scenario 4: $\{ABC, BC\}$ with chain order $C - B - A$
  - Scenario 5: $\{AB, BC, CD\}$
  - Scenario 6: $\{ABCD, BC\}$
  - Scenario 7: $\{ABCD, BD, AD\}$

  The PEAV formulations of each scenario, displayed as chains, are shown in Fig. 8. This chain structure is relevant for the analysis of privacy loss for SynchBB.
- *Valuations and Timeslots*: For each experiment for a given scenario, we chose the number of time slots, denoted by T and a value for the number of possible valuations for a single time slot, denoted by $|\mathcal{V}|$. The valuations of time slots, $\{V_i^0(t)\}$, were chosen uniformly from the set $\mathcal{V} = \{1, \ldots, |\mathcal{V}|\}$ and the valuations of meetings, $\{V_i^k\}$, were chosen uniformly from the set $\{2, \ldots, |\mathcal{V}|\}$. For an event to be scheduled at time t, we required that $\Delta_i^k(t) = V_i^k - V_i^0(t) > 0$. Thus, we use $V_{\max}^0 = |\mathcal{V}|$, $V_{\max}^k = |\mathcal{V}|$, $V_{\min}^0 = 1$, and $V_{\min}^k = 2$ in our inference equations. For the three-agent scenarios, we varied T from $\{3, 4, 5, 6, 7\}$ while holding $|\mathcal{V}| = 3$. Then, we varied $|\mathcal{V}|$ from $\{3, 4, 5, 6, 7\}$, while holding $T = 3$. For reasons of computational complexity, we chose not to vary T and $V_{\max}^0$ for the four-agent scenarios (scenarios 5, 6 and 7) to the same degree as for the three-agent scenarios. For example, using $T = 7$ in a four-agent scenario (with $|\mathcal{V}| = 3$) would require an agent to consider $3^{(3 \cdot 7)}$ possible states, i.e. over a billion states. To keep the possible state space under 10^7, the cases we varied $|\mathcal{V}|$ from $\{3, 4, 5\}$ while holding $T = 3$, and varied $|\mathcal{V}|$ from $\{3, 4\}$ while holding $|\mathcal{V}| = 3$.
- *Privacy*: The system privacy level, as mentioned earlier, is the arithmetic mean of the individual privacy levels, i.e. $f(\mathbb{V}_A, \mathbb{V}_B, \mathbb{V}_C) = (\mathbb{V}_A + \mathbb{V}_B + \mathbb{V}_C)/3$ for three-agent scenarios and $f(\mathbb{V}_A, \mathbb{V}_B, \mathbb{V}_C, \mathbb{V}_D) = (\mathbb{V}_A + \mathbb{V}_B + \mathbb{V}_C + \mathbb{V}_D)/4$ for four-agent scenarios, where the individual privacy levels are obtained from the metrics described in Section 4.

We now discuss the results of solving multiple experiments for each scenario, under various DCOP algorithm and the analysis of the privacy loss with respect to the metrics and procedures discussed in Section 4.

5.2. Results and analysis

For each $(T, |\mathcal{V}|)$ pair and a given scenario, we ran 50 experiments where time slot and meeting valuations were chosen randomly as discussed earlier. For each experiment, we ran SynchBB and each agent generated a set of inference equations or inequalities about the possible states of other agents from all the upward and downward messages it received. The agent used the simplest of these equations (those with only one $V_i^0(t)$ term, of the form $V_i^0(t) = c$ or $V_i^0(t) \leq c$) to quickly determine the maximum and minimum possible values for each of the time slot valuations of the other agents. Then, the agent considered every possible state

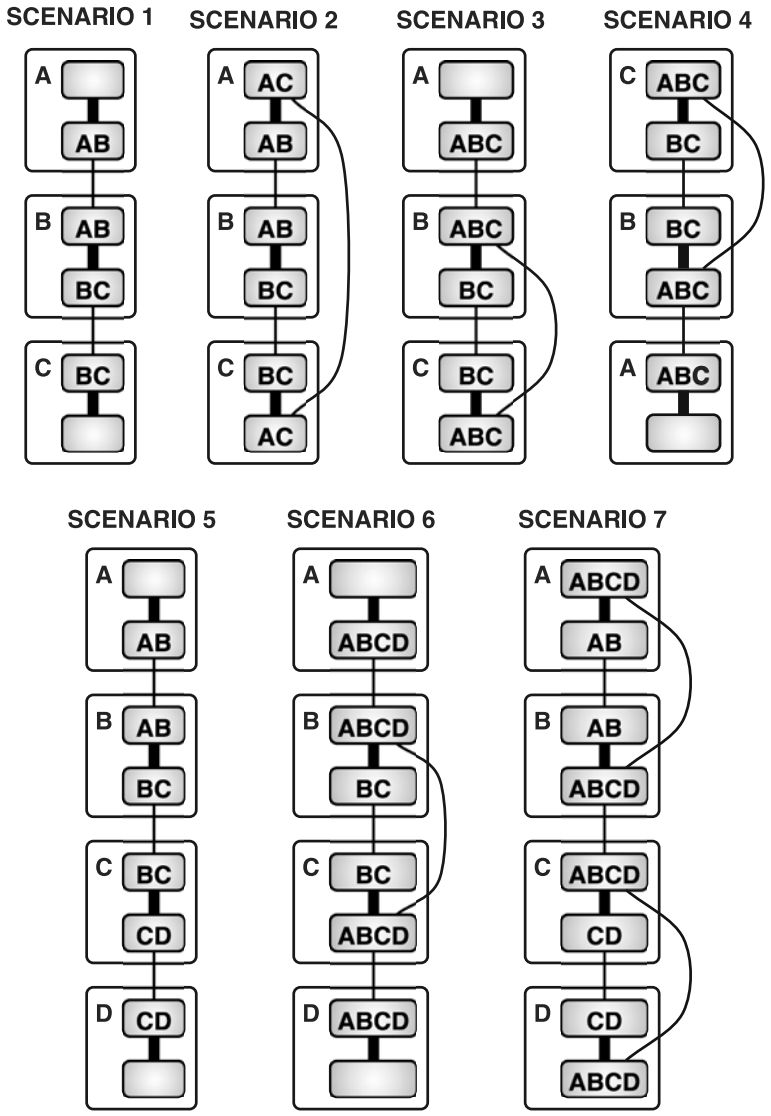

Fig. 8 PEAV formulations of scenarios as chains

in which the valuations fall within these boundaries, and checked this state against its set of remaining equations. If the world state satisfied all the equations, the agent considered it to be an active state (i.e. a state not eliminated) in its terminal belief. To compare the privacy loss in SynchBB [11] against other DCOP algorithms, we used the six metrics discussed in Section 4 metrics for PEAV-DCOPs for DiMES which were functions of agents' terminal beliefs.

For each metric, we took the mean of the privacy loss calculated from the 50 runs of each scenario. We compared SynchBB against the partially centralized DCOP algorithm, OptAPO [15], as well as a completely centralized method, in which all agents send all information to a single agent that computes the solution. As mentioned in Section 4.2.2 rather than measuring

privacy loss in OptAPO exactly, we obtain a lower bound by noting that, in the initial phase of the algorithm, all variables send complete information about their constraints to all their neighbors. In both centralization and OptAPO, agent-to-agent privacy loss is "all or nothing," (because messages share all internal constraints) and we have normalized the privacy losses across all metrics (such that "all" is a privacy loss of 1, and "nothing" is a privacy loss of 0), the system-wide privacy loss is identical for all experimental runs, regardless of which of the six metrics is used. The "all or nothing" properties of centralization and OptAPO, can be seen in Examples 4 and 5 in Sections 4.2.1 and 4.2.2 respectively. Below, we present two sets of results. Section 5 presents results comparing the original SynchBB with full chain knowledge against OptAPO and centralization for all scenarios and all metrics. Section 5.4 presents results for all scenarios comparing original SynchBB with full chain knowledge, improved SynchBB with full chain knowledge, improved SynchBB with uncertain chain knowledge, OptAPO and centralization.

5.3. Comparison of existing algorithms

Figure 9(a) presents the privacy loss in the original SynchBB algorithm for Scenario 1 as measured by the six different metrics and compares it to the privacy loss of OptAPO and a centralized algorithm (for which all metrics give the same result). Average system-wide privacy loss is represented on the y-axis and can vary from 0 to 1 where 0 means that no agents can make any inference about any other agent's time valuations and 1 means that all agents know all of each other's valuations. The x-axis shows the number of time slots in each agent's schedule. Each data point represents an average over 50 runs of experiments run with $|\mathcal{V}| = 3$. We can see that, regardless of the metric chosen to measure the loss of privacy in SynchBB, it is greater than the privacy loss in a centralized method, which in this scenario is 1/3. We can also see that OptAPO (with privacy loss of 2/3) loses more privacy than the centralized method, as shown in Section 4.2.2. Figure 9(b) presents the same privacy loss measures for Scenario 1 but the number of time slots, T, is held fixed at 3 and now the number of possible valuations, $|\mathcal{V}|$, is varied. Once again the average system-wide privacy loss is represented on the y-axis. On the x-axis we measure the number of possible valuations agents can have for each of their time slots.

The key observation from Fig. 9(a) and 9(b) is that the centralized method preserves greater average privacy than OptAPO and SynchBB, regardless of the chosen metric. In other words, simply distributing computation as done in SynchBB is inadequate by itself to preserve greater privacy compared to the centralized method. Figure 9(c)–(h) and 10(a)–(f) present the same graphs for each of the other 6 scenarios. Other than the GuessS metric for Scenario 4, the superiority of centralization as a privacy-preserving choice persists across all scenarios and metrics.

We note that these values for privacy loss are lower bounds. Indeed, whenever analyzing privacy loss, one can only discuss lower bounds as it is difficult to prove the nonexistence of additional inference rules. However, our experiments were conducted to investigate the assumption of the field that distribution alone was sufficient to provide greater privacy than centralization, and that assumption is clearly not supported. In fact, there is little room for privacy loss under centralization to increase as the central agent extracts the maximum possible information and the remaining agents have only the final meeting times to use for potential inference. Even if additional inference for noncentral agents in a centralized system was possible, it seem unlikely that this inference could close the gap in bounds of privacy loss with SynchBB. Furthermore, even if the gap was closed and the privacy loss between centralized and decentralized were identical, it is problematic to use privacy loss as a justification for

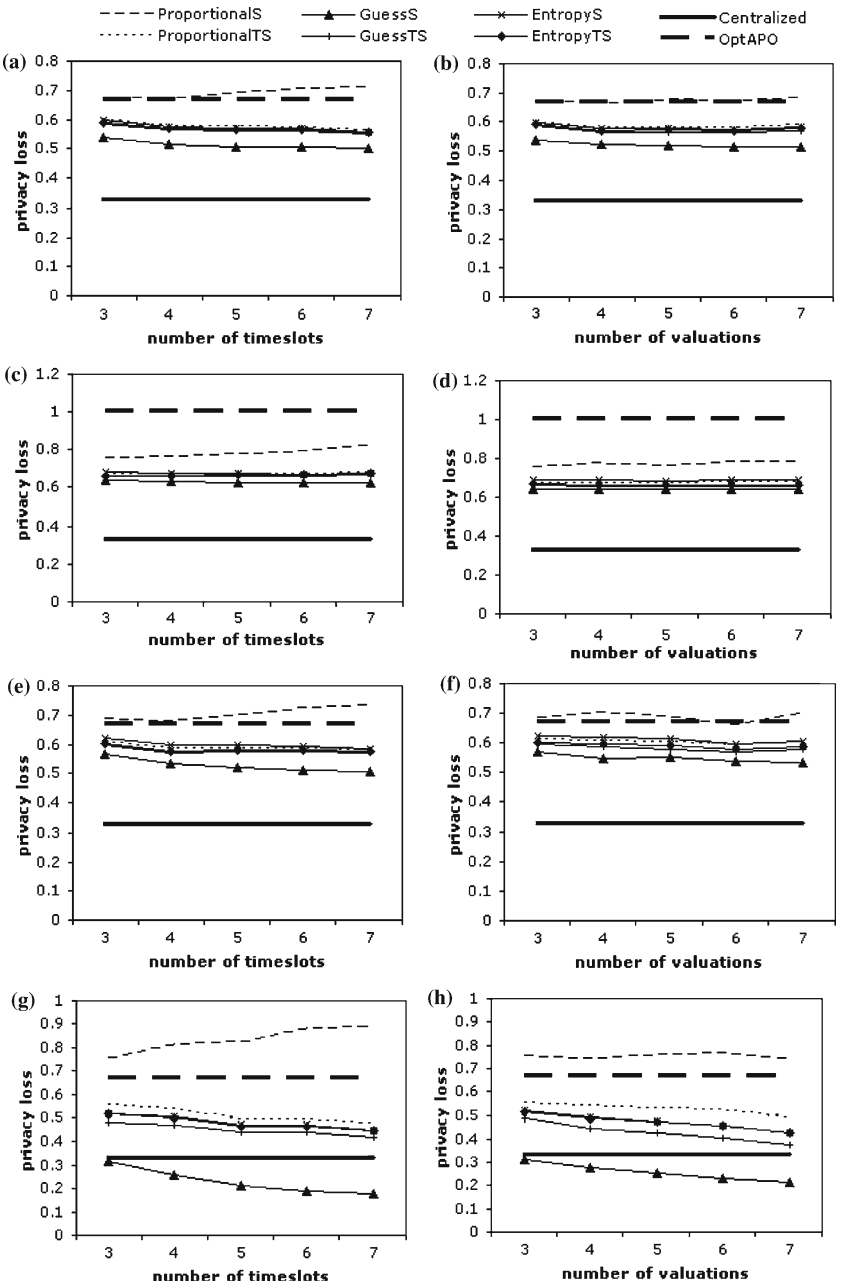

Fig. 9 Privacy loss versus number of time slots (*left column*) and privacy loss versus number of valuations (*right column*) for the three-agent scenarios (1–4)

decentralization which has, in general, a higher implementation cost. These results do not preclude the possibility that decentralization might be better for protection of privacy in other domains or under different decentralized schemes. We do note that centralization has its own

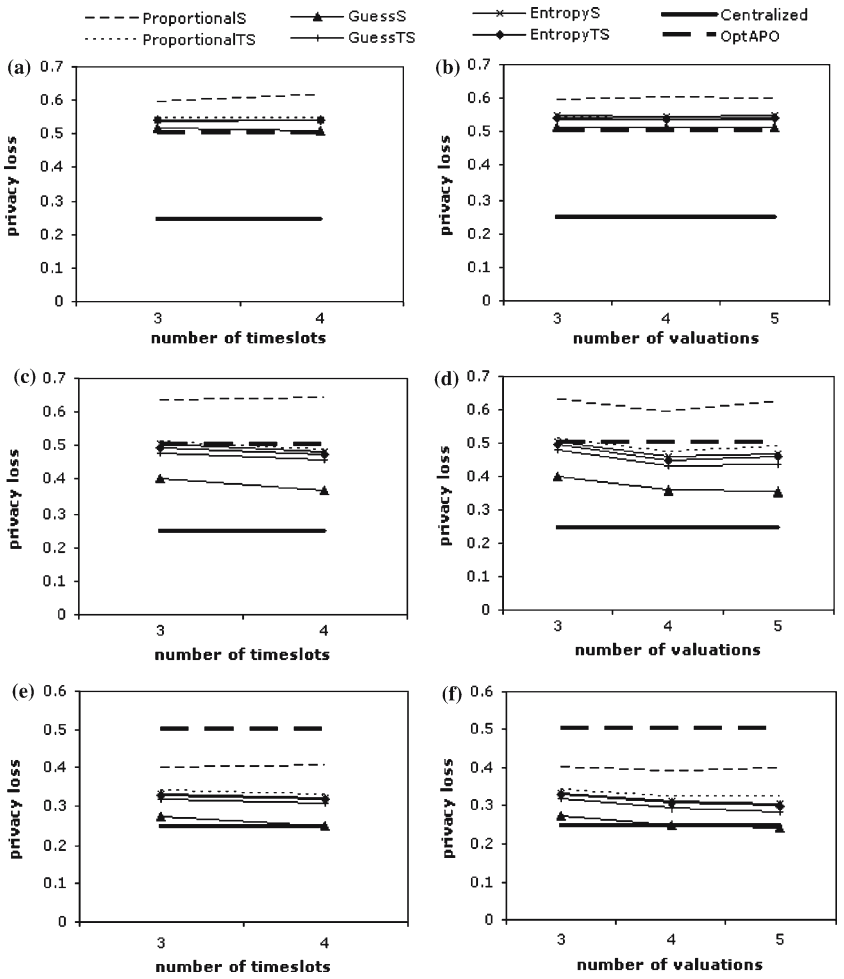

Fig. 10 Privacy loss versus number of time slots (*left column*) and privacy loss versus number of valuations (*right column*) for the four-agent scenarios (5–7)

drawbacks such as lack of robustness, in the case of failures or delays in the central agent's computation.

These experiments illustrate that one must carefully examine and justify any metric chosen to measure privacy loss. For instance, Fig. 9(g) shows how the choice of metric can lead to vastly different implications with respect to privacy loss. When comparing privacy loss under the GuessS and ProportionalS metrics, GuessS indicates that privacy loss decreases as the number of time slots increases and that SynchBB is better than centralization, while ProportionalS would indicate the opposite where privacy loss increases as the number of time slots increases and SynchBB is worse than centralization. We note that GuessS always gives the lowest level of privacy loss while ProportionalS always gives the highest level of privacy loss for SynchBB. Also, we note that ProportionalS and ProportionalTS give different qualitative properties, where ProportionalTS indicates that privacy loss decreases and the number of time slots increase. Similarly in Fig. 10(f), GuessS would indicate that SynchBB

matched the privacy loss in centralization, but ProportionalS would indicate a bigger loss for SynchBB. Thus, a careful choice of metrics is essential to avoiding misguided conclusions about SynchBB.

5.4. Impact of communication and uncertainty

Because SynchBB was not designed with privacy explicitly in mind, it can be easily modified to preserve more privacy. In SynchBB, each variable receives from its parent the values of all variables above it in the chain, and passes this information, along with its own value, to its child in the chain. As a result, many agents receive extra information about other agents with which they do not share constraints. This information can be used to make additional inferences. To avoid privacy loss to this extraneous communication, a variable can instead pass down only its own value (not those of its ancestors). However, this value needs to be passed to all the variable's descendants (not only its child), in order for them to have a sufficient context to choose their own values. We modified SynchBB to behave in this way. For example, in Scenario 2, with the original SynchBB, agent B would receive A's value for meeting AC, even though agent B does not need this information to make its decisions. With modified SynchBB, agent B would not receive this information, because A would pass it directly to C (along the AC–AC link) rather than relaying it through agent B.

To explore the effect of agents' knowledge of the constraint graph on the system-wide privacy loss, we introduced a degree of uncertainty that agents may have about the graph. We assumed that agents know an upper bound K on the sum of all attendees over all meetings, as given in Section 4.2.5, and then investigated how the tightness of this bound affected privacy loss.

Graphs comparing SynchBB with full graph knowledge, modified SynchBB with full graph knowledge, and modified SynchBB with uncertainty levels $K = \{K^*, K^* + 1\}$ (where K^* is the actual number of Δ's in the chain), are shown in Figs. 11 and 12. Here, "Uncertainty +0" refers to $K = K^*$, and "Uncertainty +1" refers to $K = K^* + 1$. We note that even though $K = K^*$ (the agent is aware of the number of Δ's), there is uncertainty as to how these are distributed in the chain. The graph presents the privacy loss of the different algorithms for each scenario as measured by the EntropyTS metric. Average privacy loss in the system is plotted on the y-axis. In the left column, the x-axis shows the number of time slots in each agent's schedule with the valuations chosen from a set of size $|\mathcal{V}| = 3$. The graphs in the right column have a fixed number of time slots, $T = 3$, but vary the number of possible valuations on the x-axis. Each data point represents an average over 50 runs. Once again the baseline performance provided by centralized is shown with the solid bold line. We can see that in some cases (Fig. 11(b) and Fig. 12(a), (b), (e) and (f)), the modified SynchBB algorithm preserves more privacy than a centralized algorithm. However, in all other cases, even the augmented SynchBB loses more privacy than the centralized algorithm.

Interestingly, even having no uncertainty about the number of Δ's in the chain is insufficient to guarantee greater privacy protection than centralization. Notably, in Fig. 11(e)–(h), the centralized algorithm still maintains more privacy than the modified SynchBB algorithm with uncertainty. However, when the graph uncertainty increases to "+1", the SynchBB algorithm preserves more privacy than centralization. In fact, privacy loss is virtually eliminated.

These experiments illustrate that while distribution by itself is inadequate to match the privacy loss in the centralized case, uncertainty of graph knowledge can begin to provide privacy in DCOP algorithms. Thus, if privacy preservation is crucial, agents must communicate

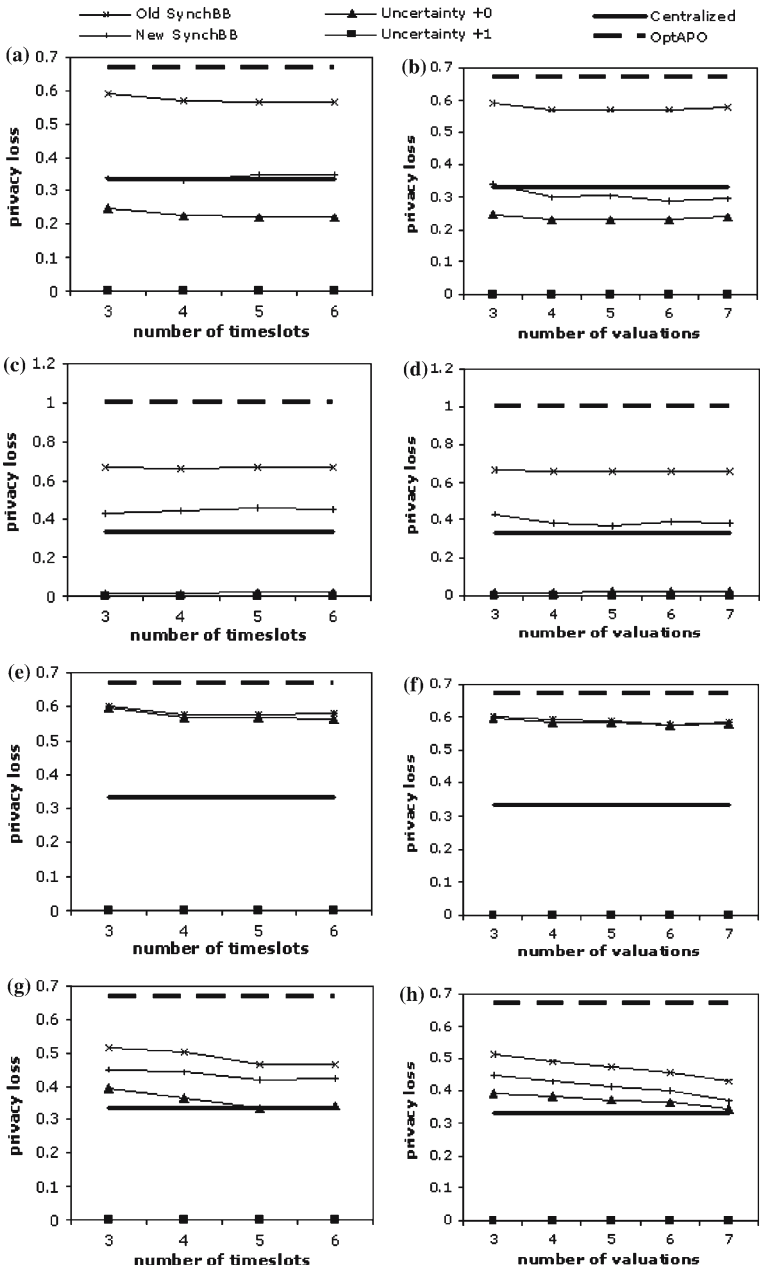

Fig. 11 Privacy loss versus number of time slots (*left column*) and privacy loss versus number of valuations (*right column*) for the three-agent scenarios (1–4)

only with the relevant agents in the DCOP and their choices should not be forwarded to agents uninvolved with local constraints. Indeed, even this step might be insufficient and knowledge of the graph structure itself may need to be hidden from others.

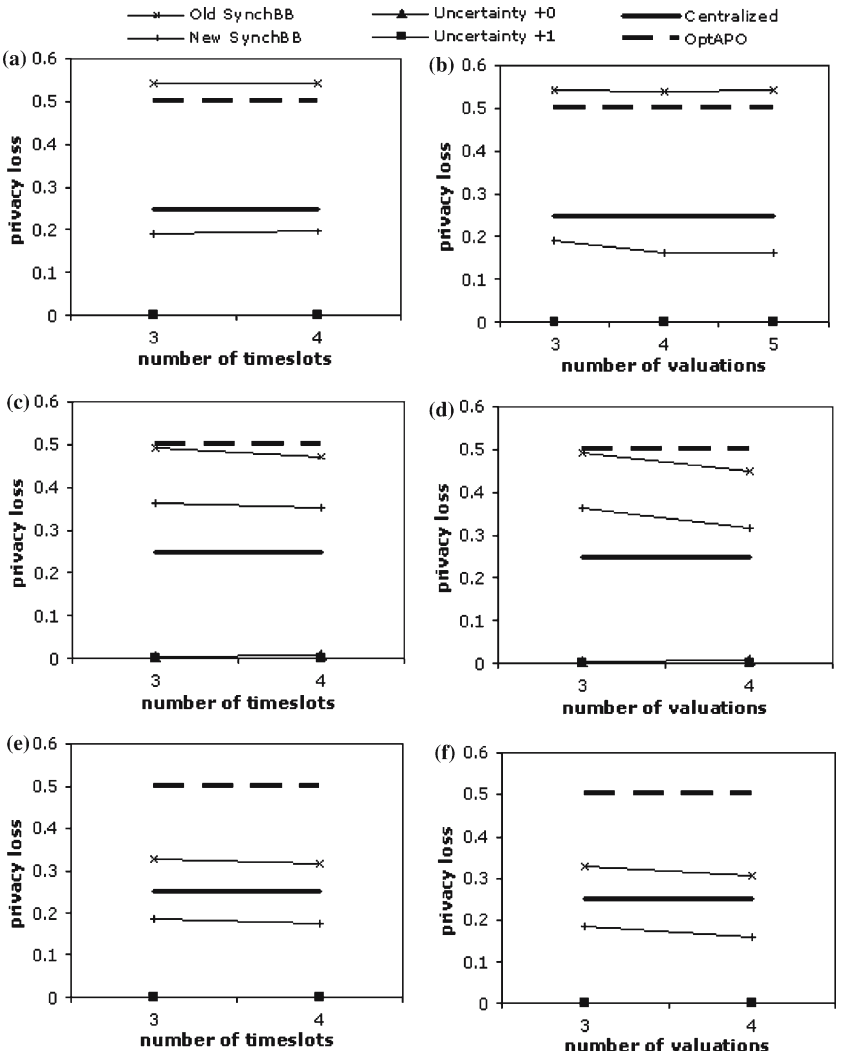

Fig. 12 Privacy loss versus number of time slots (*left column*) and privacy loss versus number of valuations (*right column*) for the four-agent scenarios (5–7)

6. Related work

This paper significantly extends our previous conference paper [14]. In particular, this paper provides significant new additional experiments and analysis, detailed and formal description of inference rules when detecting privacy loss, a significantly enhanced treatment of the formal VPS framework for privacy and the following detailed discussion of related work.

Given the focus of this paper, we start out with a discussion of privacy work in the context of DisCSP and DCOP, and then extend to other related work on privacy, particularly in agents and multiagent systems. As mentioned earlier, privacy is a major motivation for research on DisCSPs and DCOPs. Given the explosion of interest in applying DisCSPs and DCOPs in software personal assistants based on this motivation of privacy [2,13,18,26,31],

rigorous investigations of privacy loss are important. While the majority of research in this arena has focused on developing efficient distributed algorithms, which is also a crucial need, there has been some early work on privacy which we have discussed throughout this paper [8,25,26]. Indeed, this early work established the importance of a more rigorous understanding of privacy within DisCSP/DCOP and our VPS framework builds on this early work. The second Section illustrates how key metrics of privacy introduced in this earlier work can be captured within the VPS framework for comparison among metrics. We presented experimental results based on this idea and discussed key implications of these results for further research on privacy in DisCSP/DCOP. In addition, a key feature of VPS is that it steps beyond DisCSPs, which was the focus of this earlier work, into analysis of privacy loss in DCOP algorithms.

As discussed earlier, Yokoo et al. discuss a secure DisCSP algorithm [31]. The authors note that while privacy is a major motivation for DisCSP techniques, this issue is not rigorously dealt within existing algorithms; indeed, there is leakage of private information in search. They then introduce information security techniques to ensure that privacy is maintained during search. These techniques rely on public key encryption and require the introduction of multiple intermediate servers; thus, this paper provides the first instance of combining DisCSP and information security for the sake of privacy. The goal of this work is to ensure that within DisCSPs, each agent only knows the value assignment of its own variables and cannot obtain any additional information on the value assignment of variables that belong to other agents.

The goals of our work differ significantly from this earlier work that relies on information security and encryption techniques. We are focused on DCOP settings where agents cannot utilize such intermediate servers for reasons such as cost or availability. Indeed, in some business or office environments, users may be interested in some level of privacy, but may be willing to sacrifice some privacy to save costs. Indeed, if significant privacy could be obtained in DCOP without such information security techniques (e.g. due to uncertainty about the distributed constraint graph as mentioned earlier), the cost of such techniques may not be justifiable. Within this context of the absence of such information security techniques, we introduce a rigorous framework to unify expression of metrics measuring privacy loss, and illustrate concrete applications of this framework in comparing different metrics and algorithms in different contexts.

Even outside the context of DisCSP and DCOP, other research on distributed multiagent meeting scheduling has been motivated by notions of privacy [6,9,10,23]. Furthermore, this research has explored the tradeoffs in privacy and efficiency [6,9,10,23]. However, what has been missing so far in this research is a formal unifying framework to express privacy loss, and perform cross-metric comparisons. VPS has begun to close this gap. Note that VPS itself is not specific to DisCSPs and DCOPs and can be applied in these other systems.

Going beyond privacy in DCOPs and meeting scheduling, acting optimally while maintaining or hiding private information has emerged as an important topic of research in many multiagent systems. Indeed, the increased interest in personal software assistants and other software agents [1,13,22] to automate routine tasks in offices, in auctions and e-commerce, at home or all spheres of daily activity has led to increased concern about privacy. While such software agents need to use private user information to conduct business on behalf of users, this wealth of private information in possession of software agents is a great area of concern for users. This has led to many novel research thrusts aimed at protecting privacy, including use of cryptographic techniques, secure distributed computation (secure multiparty function evaluation) and randomization [3,4,19,28]. For instance, the randomization approach may be used when an agent's actions can be observed but must be kept private [20,24,28]. In this

approach, by choosing actions in a randomized fashion (in particular, relying on action strategies or policies that have high entropy), agents are able to provide minimal information to an adversary about their preferences, while attempting to reach some of their key objectives.

These different research thrusts are focused on developing novel techniques for protecting privacy, which complements the research presented in this article. Indeed, this other research emphasizes the increasing importance of defining a common framework for privacy metrics, although it does not define such a framework; we move towards this goal via our VPS framework. Finally, while our emphasis has been on understanding privacy loss within collaborative DisCSP/DCOP algorithms, it points the way to improving privacy preservation in such algorithms; techniques mentioned above may provide some insights into building such algorithms, although techniques such as randomization may not be directly applicable in a collaborative setting. Indeed, our investigation of the impact of uncertainty on privacy loss is a step in the direction of understanding principles that reduce privacy loss in DCOP algorithms.

7. Summary

In many emerging applications, particularly that of software personal assistant agents, protecting user privacy is a critical requirement. DCOP/DisCSP is an important approach to multiagent systems that promises to enable agents' distributed negotiation and conflict resolution while maintaining user's privacy; thus, several software personal assistant applications are being built around DCOP/DisCSP algorithms [1,2,10,13,18,26]. Unfortunately, a general quantitative framework to compare existing metrics for privacy loss, and identify dimensions along which to construct/classify new metrics is currently lacking. Indeed, privacy loss analysis has in general focused on DisCSPs rather than DCOPs, and within this arena, quantitative cross-metric comparisons of privacy loss due to different algorithms are currently missing.

This paper presents three key contributions to address these shortcomings. First, the paper presents the VPS framework, a general quantitative model from which one can analyze and generate metrics of privacy loss. VPS is shown to capture various existing measures of privacy created for specific domains of DisCSPs. The utility of VPS is further illustrated via analysis of privacy loss in DCOP algorithms, when such algorithms are used by personal assistant agents to schedule meetings among users. Second, the article presented key inference rules that may be used in analysis of privacy loss in DCOP algorithms, under different assumptions about agent knowledge. We provided such rules in the context of the fully centralized algorithm, an algorithm that is partially centralized (OptAPO) and finally an algorithm that attempts full distribution (SynchBB). These rules are illustrative of the inferences that may be feasible in general for other DCOP algorithms. Third, the article presented detailed experiments based on its VPS-driven analysis, leading to the following key results: (i) decentralization by itself does not provide superior protection of privacy in DisCSP/DCOP algorithms when compared with centralization; instead, privacy protection requires the additional presence of uncertainty in agents' knowledge of the constraint graph, (ii) one needs to carefully examine the metrics chosen to measure privacy loss; the qualitative properties of privacy loss and hence the conclusions that can be drawn about an algorithm can vary widely based on the metric chosen.

In terms of future work, several major issues suggest themselves immediately. First, researchers continue to investigate algorithms or preprocessing strategies that improve DCOP solution efficiency. If privacy is a major motivation for DCOP, then it is crucial to understand if these improvements ultimately cause a further erosion of privacy. Thus, researchers should

focus on privacy-preserving efficiency improvements at least if the DCOP algorithms are to be applied in domains such as software personal assistant agents, where preservation of privacy is crucial. Second, our current investigation weighed all privacy loss equally. However, it might be the case that privacy loss to some individuals is weighed less than to others. For instance, in inter-organizational negotiations, privacy loss within an organization might not be weighed as heavily as outside the organization. Understanding the impact of such weighted privacy loss is also a key issue for future work. Third, we have assumed a model where there is no information leakage, i.e. agents do not communicate inferred information to others in the system. While theoretically, information leakage can range from zero to complete (in potentially multi-dimensional ways, i.e. varying amounts to different agents), we chose zero as we believe it most closely approximates distributed meeting scheduling and similar domains. However, the assumptions about information leakage can alter the effectiveness of various algorithms in terms of privacy loss (e.g. in the extreme case, centralization would lead to total privacy loss under a complete information leakage assumption). In the general case, where information loss is not total, it is not obvious how algorithms will perform with respect to privacy loss, and thus, is a fertile area for research. This paper hopes to serve as a call to arms for the community to improve privacy protection algorithms and further research on privacy.

Acknowledgements This material is based upon work supported by DARPA, through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. We thank Pragnesh Jay Modi for providing us with an implementation of SynchBB for analysis. We also thank Rachel Greenstadt for her valuable comments on an earlier draft of this article.

References

1. Berry, P. M., Gervasio, M., Uribe, T. E., Myers, K., & Nitz, K. (2005). A personalized calendar assistant. In *AAAI spring symposium on persistent assistants: Living and working with AI.*
2. Bowring, E., Tambe, M., & Yokoo, M. (2005). Optimize my schedule but keep it flexible: Distributed multi-criteria coordination for personal assistants. In *AAAI spring symposium on persistent assistants: Living and working with AI*.
3. Brandt, F. (2001). Cryptographic protocols for secure second-price Auctions. In: *Cooperative information agents V, lecture notes in artificial intelligence (LNAI)*. (Vol. 2182) (pp. 154–165).
4. Brandt, F. (2003). Fully private auctions in a constant number of rounds. In *Proceedings of the 7th annual conference on financial cryptography (FC)*, (pp. 223–238).
5. Chalupsky, H., Gil, Y., Knoblock, C., Lerman, K., Oh, J., Pynadath, D., Russ, T., & Tambe, M. (2001). Electric elves: Applying agent technology to support human organizations. In *International conference on innovative applications of artificial intelligence*, (pp. 51–58).
6. Ephrati, E., Zlotkin, G., & Rosenschein, J. S. (1994). A non–manipulable meeting scheduling system. In *Proceedings of the 13th international workshop on distributed artificial intelligence*. Seattle, WA.
7. Franzin, M. S., Freuder, E. C., Rossi, F., & Wallace, R. (2002). Multi-agent meeting scheduling with preferences: Efficiency, privacy loss, and solution quality. In *Proceedings of the AAAI workshop on preference in AI and CP*. Edmonton, Canada.
8. Franzin, M. S., Freuder, E. C., Rossi, F., & Wallace, R. (2004). Multi-agent constraint systems with preferences: Efficiency solution quality and privacy loss. *Computational intelligence*, *20*(2), 264–286.
9. Garrido, L., & Sycara, K. (1996). Multi-agent meeting scheduling: Preliminary results. In *Proceedings of the 1996 international conference on multi-agent systems (ICMAS '96)*, (pp. 95–102).
10. Hassine, A. B., Défago, X., & Ho, T. (2004). Agent-based approach to dynamic meeting scheduling problems. In *Proceedings of the third international joint conference on autonomous agents and multi agent systems (AAMAS 2004)*, (pp. 1132–1139) New York, NY.
11. Hirayama, K., & Yokoo, M. (1997) Distributed partial constraint satisfaction problem. In G. Smolka (ed.), *Principles and practice of constraint programming*, (pp. 222–236).
12. Liu, J., & Sycara, K. P. (1996) Multiagent coordination in tightly coupled task scheduling. In *Proceedings of the second international conference on multiagent systems*, (pp. 181–187).

13. Maheswaran, R. T., Bowring, E., Pearce, J. P., Varakantham, P., & Tambe, M. (2004). Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proceedings of the third international joint conference on autonomous agents and multi agent systems (AAMAS 2004)*, (pp. 310–317) New York.
14. Maheswaran, R. T., Pearce, J. P., Varakantham, P., Bowring, E., & Tambe, M. (2005). Valuations of possible states (VPS): A unifying quantitative framework for analysis of privacy loss in collaboration. In *Proceedings of the fourth international joint conference on autonomous agents and multi agent systems (AAMAS 2005)*, (pp. 1030–1037) Utrecht, The Netherlands.
15. Mailler, R., & Lesser, V. (2004) Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, (pp. 438–445) New York, NY.
16. Meisels, A., & Lavee, O. (2004). Using additional information in DisCSPs search. In *Proceedings of the 5th workshop on distributed constraints reasoning (DCR-04)*. Toronto, CA.
17. Modi, P. J., Shen, W., Tambe, M., & Yokoo, M. (2003). An asynchronous complete method for distributed constraint optimization. In *Proceedings of the second international conference on autonomous agents and multi-agent systems*, (pp. 161–168).
18. Modi, J. P., & Veloso, M. (2005). Bumping strategies for the multiagent agreement problem. In *Proceedings of the fourth international joint conference on autonomous agents and multi agent systems (AAMAS 2005)*, (pp. 390–396). Utrecht, The Netherlands.
19. Naor, M., Pinkas, B., & Sumner, R. (1999). Privacy preserving auctions and mechanism design. In *Proceedings of the first ACM conference on electronic commerce*, (pp. 129–139).
20. Paruchuri, P., Tambe, M., Dini, D., Kraus, S., & Ordonez, F. (2005). Safety in multiagent systems via policy randomization. In *AAMAS workshop on safety and security in multiagent systems*.
21. Sadeh, N., & Fox, M. S. (1996). Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence*, *86*, 1–41.
22. Scerri, P., Pynadath, D., & Tambe, M. (2002). Towards adjustable autonomy for the real-world. *Journal of Artificial Intelligence Research*, *17*, 171–228.
23. Sen, S. (1997). Developing an automated distributed meeting scheduler. *IEEE Expert: Intelligent Systems and Their Applications*, *12*(4), 41–45.
24. Silaghi, M. (2004). Meeting scheduling guaranteeing n/2-privacy and resistant to statistical analysis (applicable to any DisCSP). In *3rd IC on web intelligence*, (pp. 711–715).
25. Silaghi, M. C., & Faltings, B. (2002). A Comparison of distributed constraint satisfaction approaches with respect to privacy. In *Proceedings of the 3rd workshop on distributed constraints reasoning (DCR-02)*. Bologna, Italy.
26. Silaghi, M. C., & Mitra, D. (2004). Distributed constraint satisfaction and optimization with privacy enforcement. In *Proceedings of the 2004 IEEE/WIC/ACM international conference on intelligent agent technology (IAT 2004)*, (pp. 531–535) Beijing, China.
27. Silaghi, M. C., Sam-Haroud, D., & Faltings, B. (2001). ABT with asynchronous reordering. In *Second Asia-Pacific conf. on intelligent agent technology*, (pp. 54–63) Maebashi, Japan.
28. van Otterloo, S. (2005). The value of privacy: optimal strategies for privacy minded agents. In *Proceedings of the fourth international joint conference on autonomous agents and multi agent systems (AAMAS 2005)*, (pp. 1015–1022) Utrecht, The Netherlands.
29. Yokoo, M., Durfee, E. H., Ishida, T., & Kuwabara, K. (1998). The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *10*(5), 673–685.
30. Yokoo, M., & Hirayama, K. (1996). Distributed breakout algorithm for solving distributed constraint satisfaction and optimization problems. In *Proceedings of the second international conference on multiagent systems*, (pp. 401–406) Kyoto, Japan.
31. Yokoo, M., Suzuki, K., & Hirayama, K. (2002). Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *Proceedings of the 8th international conference on principles and practice of constraint programming (CP 2002, LNCS 2470)*, (pp. 387–401) Ithaca, NY.