Valuations of Possible States (VPS): A Quantitative Framework for Analysis of Privacy Loss Among Collaborative Personal Assistant Agents

Rajiv T. Maheswaran, Jonathan P. Pearce, Pradeep Varakantham,
Emma Bowring and Milind Tambe
University of Southern California
(maheswar,jppearce,varakant,bowring, tambe)@usc.edu

ABSTRACT

For agents deployed in real-world settings, such as businesses, universities and research laboratories, it is critical that agents protect their individual users' privacy when interacting with others entities. Indeed, privacy is recognized as a key motivating factor in design of several multiagent algorithms, such as distributed constraint optimization (DCOP) algorithms. Unfortunately, rigorous and general quantitative metrics for analysis and comparison of such multiagent algorithms with respect to privacy loss are lacking. This paper takes a key step towards developing a general quantitative model from which one can analyze and generate metrics of privacy loss by introducing the VPS (Valuations of Possible States) framework. VPS is shown to capture various existing measures of privacy created for specific domains of distributed constraint satisfactions problems (DCSPs). The utility of VPS is further illustrated via analysis of DCOP algorithms, when such algorithms are used by personal assistant agents to schedule meetings among users. In addition, VPS allows us to quantitatively evaluate the properties of several privacy metrics generated through qualitative notions. We obtain the unexpected result that decentralization does not automatically guarantee superior protection of privacy.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Security

Keywords

Privacy, Personal Assistant Agents, Collaborative Negotiation

1. INTRODUCTION

Personal assistant agents are an emerging application whose integration into office environments promises to enhance productivity by performing routine or mundane tasks and expediting coordinated activities [1]. 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 on their users' behalves, 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. 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. 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.

While research on security has provided methods to protect a collaborating set of agents from outside parties obtaining internal information, we focus on privacy which captures information loss inherent in the negotiation within the collaborating set. Earlier work focused on using cryptographic methods to provide privacy, however this required the existence of multiple external servers which is not always desirable or available [13]. While there has been additional recent work on addressing privacy in constraint satisfaction, and though these investigations have quantitative elements, they are not immediately portable to other models where one might want to optimize instead of satisfy [2, 12, 6]. What is lacking is a principled quantitative approach to deriving metrics for privacy for general domains. To address this need, we propose the Valuation of Possible States (VPS) framework to quantitatively evaluate privacy loss in multiagent settings. We apply these ideas in a distributed meeting scheduling domain modeled as a distributed constraint optimization problem (DCOP). We then develop techniques to analyze privacy loss when using the OptAPO [5] and SynchBB [3] algorithms to solve the DCOP.

A key implication of our experiments is that centralization can outperform some dececntralized approaches for constraint optimization with respect to privacy loss over many metrics. Another is that 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. Finally, we observe that the key to preserving privacy is to minimize the inferences that other agents can make about one's possible states.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

2. VALUATIONS OF POSSIBLE STATES FRAMEWORK

Given a setting where a group of N agents, indexed by the set $\mathcal{N} := \{1, \dots N\}$, 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. This private information can be modeled as a state among a set of possible states. In a meeting scheduling domain, agents might need to protect calendar information while in a task allocation problem, a personal assistant agent might want to protect information about its users' capabilities or priorities. Consider a scenario where personal assistant agents collaborate to order lunch [9]. Each user could prefer one of three restaurants. To observers, a person can exist in one of three possible states. Each observer also has an estimate of the likelihood that a person lies in each of these possible states. Privacy can then be interpreted as a valuation on the other agents' estimates about the possible states that one lives in. Simply put, privacy is our level of comfort with what others know about us. In this paper, we will often use the term *agent* to refer to an entity with private information though *person* or *user* can be equivalently substituted.

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. The user of each agent has a preference or availability denoted by 0 or 1 for each time. Thus, each agent (user) belongs in one of four states* ([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. At the beginning of the negotiation, all agents model the other agents with some probability distribution of over these states. A uniform distribution represents no* a priori *knowledge. After negotiation, agents will alter these probabilities due to inferences being made from the messages received. Given a metric for privacy, represented as a valuation function over a probability distribution, we can measure the loss of privacy by subtracting the value of the probability distribution at the end of negotiation from the value of the probability distribution at the beginning of negotiation.* □

To express these ideas more formally, let the private information of the i^{th} agent be modeled as a state $s_i \in S_i$, where S_i is a set of possible states that the i^{th} 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. Then, let

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

be the set of all possible states of all agents except the j^{th} agent. The j^{th} 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 j^{th} agent's knowledge as a probability distribution over the possible states of all other agents $\mathbb{S}_{-j}$, i.e. $\mathbb{P}^j(\mathbb{S}_{-j})$. 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 $K_1 = 16$.

Thus, an agent's knowledge of other agents is a joint probability mass function over the product set of possible states of all other agents. The j^{th} agent's knowledge of a particular agent, say the i^{th} agent, is then the marginal probability of this distribution with respect to i, as follows:

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

where $\tilde{s}^i$ refers to the state of the i^{th} 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 B-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 $N-1$ agents have about the i^{th} agent can then be expressed as follows:

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

where $\mathbb{P}_i^j(S_i)$ is as defined in (1). In Example 1, the information other agents have about agent A is $\mathbb{P}_A((S_A)^2) = [\mathbb{P}_A^B(S_A)\ \mathbb{P}_A^C(S_A)]$. The above model assumes that agents do not share there information or estimates about other agents. If sharing does occur, this can be captured by the $\mathbb{P}_i^G(S_i)$ where $G \subset \mathcal{N}$ is a group of agents that share information to obtain a better estimate of the i^{th} agent's state, where $i \notin G$. In this case $\mathbb{P}_i((S_i)^{N-1})$ would be composed of group estimates $\mathbb{P}_i^G(S_i)$ where G is an element of the power set of $\mathcal{N}$.

The i^{th} agent can then put a value for each distribution that the collection of other agents could hold, yielding a value function $\mathbb{V}_i(\mathbb{P}_i((S_i)^{N-1}))$. For example, in the lunch ordering scenario, one agent may value highly the ability to hide his restaurant preferences (it wants $\mathbb{P}_i^j(S_i)$ to be uniformly distributed over $S_i, \forall j \neq i$) while another may value highly that others know its preference (it wants $\mathbb{P}_i^j(S_i)$ to be an accurate delta function $\forall j \neq i$), e.g. if the agent (or more appropriately, its user) is a vegetarian. In Example 1, a simple value function is the number of possible states that other agents have not eliminated (i.e. states with nonzero probability). Additional metrics for meeting scheduling will be discussed later. This framework is essentially a utilitarian model of information states and thus captures the notion of privacy as a special case. Given this framework, a single agent can measure loss of privacy in a collaboration by measuring the changes in $\mathbb{V}_i$ and an organization can measure the loss of privacy from a process by calculating the changes in some function $f(\mathbb{V}_1, \cdots, \mathbb{V}_N)$ which aggregates the valuation of possible states for the entire set of agents.

3. UNIFICATION OF EXPRESSION WITH VALUATIONS OF POSSIBLE STATES FRAMEWORK

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 [10], Silaghi and Faltings consider Distributed Constraint Satisfaction Problems (DCSPs) 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 i^{th} agent. The i^{th} 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,

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

Since revelation for the i^{th} 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 [2], 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_iL_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:

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

where $G = \mathcal{N} \setminus i$ is the set of all agents except the i^{th} agent as information sharing is part of the assumption in privacy loss. The indicator function in the denominator 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 (hence, no probability multiplier before the log). The 10 in the numerator indicates that all 10 preferences are possible at the beginning of negotiation.

- In [11], 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 amongst 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 is modeled as an element s_i of the set S_i which is the set of all vectors of length $T_iL_iM_i$ 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 i^{th} 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 i^{th} agent must be a delta function (all probability on one state) and therefore, the i^{th} agent's state is known exactly by the group G.

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.

4. DISTRIBUTED MEETING SCHEDULING MODEL

To investigate the usefulness of VPS as a generative tool, we applied it to a personal assistant domain: distributed meeting scheduling. We present here the distributed multi-event scheduling (DiMES) model presented in [4] that captures many fundamental characteristics of distributed scheduling in an optimization framework. We then describe how we can map the DiMES problem to a distributed constraint optimization problem (DCOP), which can be solved by agents on a structure that prevents *a priori* privacy loss.

4.1 DiMES

The original DiMES model mapped the scheduling of arbitrary resources. Here, we 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 $[T_{earliest}, T_{latest}]$ over which all events are to be scheduled. Let $T \in \mathbb{N}$ be a natural number and Δ be a length such that $T \cdot \Delta = T_{latest} - T_{earliest}$. 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 $[T_{earliest} + (t-1)\Delta, T_{earliest} + t\Delta]$. Thus, a business day from 8AM - 6PM 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 k^{th} 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. The heterogeneous importance of an event to each attendee is described in a value vector V^k. If $R_n \in A^k$, then V_n^k will be an element of V^k which denotes the value per time slot to the n^{th} person for scheduling event k. Let $V_n^0(t) : \mathcal{T} \to \mathbb{R}^+$ denote the n^{th} person's valuation for keeping time slot t free. These valuations allow agents to compare the relative importance of events and also to compare the importance of the event to the value of a person's time.

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.

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 $S(E^{k_1}) \cap S(E^{k_2}) = \emptyset \ \forall k_1, k_2 \in \{1, \ldots, K\},\ k_1 \neq k_2,\ A^{k_1} \cap A^{k_2} \neq \emptyset$.

4.2 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 [7] presents itself as a useful and appropriate approach.

Our challenge is to convert a given DiMES problem into a DCOP with binary constraints. We may then apply any algorithm developed for DCOP to obtain a solution. In [4], three DCOP formulations for DiMES were proposed. 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.

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.

5. PRIVACY IN PEAV-DCOPS FOR DIMES

In this section, we apply our VPS ideas to the DiMES model and generate several instantiations of valuations to quantitatively measure the privacy loss when agents apply known DCOP algorithms to a distributed meeting scheduling scenario.

5.1 VPS 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. 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 $V_i^0(t) \in \mathcal{V}$ where $\mathcal{V}$ is a discrete set and there are T time slots in a schedule, the state s_i of the i^{th} 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 V to each of their T time slots based on their preferences. Before negotiation, each agent knows only that 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 $\mathbb{P}_i((S_i)^{N-1})$. 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 [10] does not apply here because we are not in a satisfaction setting and the method in [11] is not viable because information sharing is not an appropriate assumption in this domain. We do consider the entropy-based metric introduced in [2] and captured in VPS in Section 3. We remove the factor L_i that captures location and adjust to account for privacy loss to other individual agents:

$$\mathbb{V}_i(\mathbb{P}_i(S_i)) := \sum_{j \neq n} \sum_{k=1}^{T} \log_2 \left(\frac{\sum_{m=1}^{|\mathcal{V}|} I_{\{\max_{s_i \in S_i : s_i(k)=m} \mathbb{P}_i^j(s_i(k)=m)>0\}}}{|\mathcal{V}|} \right) \quad (2)$$

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. In this case, we have

$$\mathbb{V}_i(\mathbb{P}_i^G(S_i)) := \log_2 \left(\frac{\sum_{j=1}^{|\mathcal{V}|^T} I_{\{\mathbb{P}_i^G(s_j)>0\}}}{|\mathcal{V}|^T} \right) \quad (3)$$

where $G = \mathcal{N} \setminus i$. 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.

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 $\mathbb{V}_n$ would be

$$\mathbb{V}_i(\mathbb{P}_i((S_i)^{N-1})) = \sum_{j \neq i} \sum_{s_i \in S_i} I_{\{\mathbb{P}_i^j(s_i)>0\}} \quad (4)$$

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

$$\mathbb{V}_i(\mathbb{P}_i((S_i)^{N-1})) = \sum_{j \neq i} \sum_{k=1}^{T} \sum_{m=1}^{|\mathcal{V}|} I_{\{\max_{s_i \in S_i : s_i(k)=m} \mathbb{P}_i^j(s_i(k)=m)>0\}} \quad (5)$$

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.

We note that these 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:

$$\mathbb{V}_i(\mathbb{P}_i((S_i)^{N-1}) = \sum_{j \neq i} \left(1 - \frac{1}{\sum_{s \in S_i} I_{\{\mathbb{P}_i^j(s)>0\}}} \right) \quad (6)$$

and its per-time-slot analogue:

$$\mathbb{V}_i(\mathbb{P}_i((S_i)^{N-1}) = \sum_{j \neq i} \sum_{k=1}^{T} \left(1 - \frac{1}{\sum_{m=1}^{|\mathcal{V}|} I_{\{\max_{s_i \in S_i : s_i(k)=m} \mathbb{P}_i^j(s_i(k)=m)>0\}}} \right). \quad (7)$$

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 refer to these metrics as LogTS (2), LogS (3), Linear-S (4), LinearTS (5), GuessS (6) and GuessTS (7) where the numbers in parentheses refer to the equations that characterize them. We apply these metrics to various algorithms in an experimental domain discussed below.

5.2 Experimental Domain

We choose an instantiation of the DiMES problem where there are three personal assistant agents, each representing a single user, whose joint task is to schedule two meetings. One meeting involves all three agents and another involves only two. Each meeting lasts for one time slot. Formally, we have $\mathcal{R} = \{A, B, C\}, E^1 = [\{A, B, C\}, 1, V^1], E^2 = [\{A, B\}, 1, V^2]$. The private information accessible to the agents are the valuations of time $\{V_A^0, V_B^0, V_C^0\}$ which are vectors of length T, whose components can take values from the set $\mathcal{V} := \{1, \cdots, K\}$. In our experiments, we varied both K and T from the set $\{3, 4, 5, 6, 7\}$. For the privacy of the entire agent group, we choose to average their individual levels, i.e. $f(\mathbb{V}_A, \mathbb{V}_B, \mathbb{V}_C) = (\mathbb{V}_A + \mathbb{V}_B + \mathbb{V}_C)/3$. To solve this scheduling problem, a group of agents can employ a variety of techniques. We now analyze the privacy loss for several of these algorithms.

5.3 Analysis of Algorithms

As we have argued, privacy is a critical property in the realm of personal assistant agents. Thus, it is important to evaluate privacy loss when deciding which methods to embed in these agents for solving team problems. For meeting scheduling, we will apply our metrics on two DCOP algorithms: OptAPO [5] and SynchBB [3], in addition to the baseline centralized solution.

We first look at centralization as a solution because one main argument for decentralization is the need to protect privacy [4]. Consequently, it is important to identify if and when this argument is justified. A quantitative analysis tool gives us the opportunity to compare various algorithms in a rigorous manner. We calculate average privacy loss, where we aggregate the privacy loss by all the agents (which may be and often is asymmetric) and divide by the total possible privacy loss for all the agents. Each agent has the potential to use 2 units of privacy (one to each of the other two agents) and the system has 6 units to lose. In a centralized solution, all agents except one would give up their entire valuation vector to a single agent. The central agent will then be able to narrow all other agents' possible states to one, while non-central agents remain at their initial estimates. Because we have scaled our metrics to span [0 1], the two non-central agents go from a privacy level of two to a privacy level of one (due to the privacy loss to the central agent) and the central agent remains at two. Thus two agents lose one unit of privacy and and one agent loses no privacy, which leads to an average privacy loss of 2/6 = 1/3. For N agents, centralization would yield an average privacy loss of $1/N$ as we lose $N - 1$ units of privacy lost in the interaction and the total possible privacy loss for all N agents is $N(N - 1)$.

To apply decentralized techniques, we map the experiment to a DCOP graph using the PEAV structure as shown in Figure 1. In OptAPO, agents exchange their constraints with all their neighbors in the initial phase. In our example, all agents will be able to identify the exact states of other agents with the exception that agents B and C will not know anything about each other immediately after the initial phase. The dynamics of OptAPO after the initial phase are not deterministically predictable. It is possible that by the end,

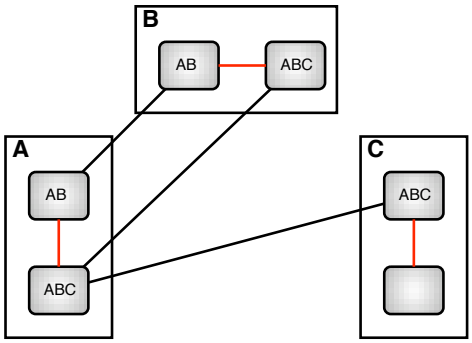

Figure 1: DCOP Graph of Experimental Domain in PEAV Structure

agents B and C 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. Here, agent A will go from 2 to 0 (as it reveals its preferences to agents B and C) while agents B and C go from 2 to 1 (as they reveal only to agent A). Thus, the average loss is 4/6=2/3. We see that OptAPO yields worse privacy loss than centralization.

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. 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.

We now consider SynchBB, another algorithm used for solving constraint satisfaction and optimization problems in a distributed setting. 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. Thus, the structure of the chain is a key factor in determining privacy loss. For our meeting scheduling example, we consider the chain structures displayed in Figure 2. 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. To illustrate how possible states can be eliminated in SynchBB, we outline the inferences that one can make from messages received in Chain 1.

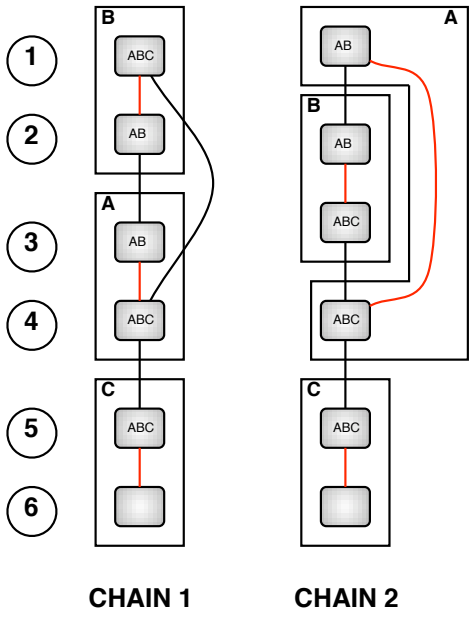

Figure 2: SynchBB Chain Structures

An upward message contains a number r, which is equal to the best currently known total reward for the chain. For PEAV, the total reward for the chain is equal to the sum of differences between the valuation of a scheduled meeting and the valuation of the time slot it occupies for every scheduled meeting for every person. We henceforth use the term *delta* to denote the change in utility (meeting valuation - time slot valuation) for a single agent when a single meeting is assigned a time slot (if the meeting is chosen to be unscheduled, the delta will be zero). If a node P receives an upward message from its child such that r has changed since the last message it received from its child, then P knows that the current context has allowed its child to find a better solution. Node P knows that $r = d + u$, where d = sum of deltas downstream from P and u = sum of deltas upstream from P. Because it has received it in a downstream message from its parent and P knows the value of u, P knows d exactly. If r has not changed, then P knows $r \leq d + u$, which gives P a lower bound on d. Since A knows when meeting ABC is scheduled, a message from C to A (m_{CA}) allows A to know $V_C(t_{ABC})$ (the valuation vector component of C at the time at which meeting ABC is scheduled). If m_{CA} contains a new r, then this value is known exactly, otherwise a lower bound is obtained. Since B knows when both meetings are scheduled, a message from A to B allows B to know $v_A(t_{AB}) + v_A(t_{ABC}) + v_C(t_{ABC})$. If m_{AB} contains a new r, then this value is known, otherwise a lower bound is obtained.

Downward messages contain a number u, which is exactly equal to the current sum of deltas that exists upstream from the sender. They also contain the current values of all variables upstream from the receiver of the message. Since the values of all meetings are known, this means that a message from B to A (m_{BA}) allows A to know the exact value of $v_B(t_{AB}) + v_B(t_{ABC})$. A message from A to C (m_{AC}) allows C to know the exact value of $v_A(t_{ABC}) + v_A(t_{AB}) + v_B(t_{ABC}) + v_B(t_{AB})$. By collecting these relationships and making appropriate reductions in possible states ,we can obtain the privacy loss for SynchBB for the PEAV-DCOP privacy metrics introduced earlier.

We ran 10 cases for each chain and each $(K, T) \in \{3, 4, 5, 6, 7\}^2$,

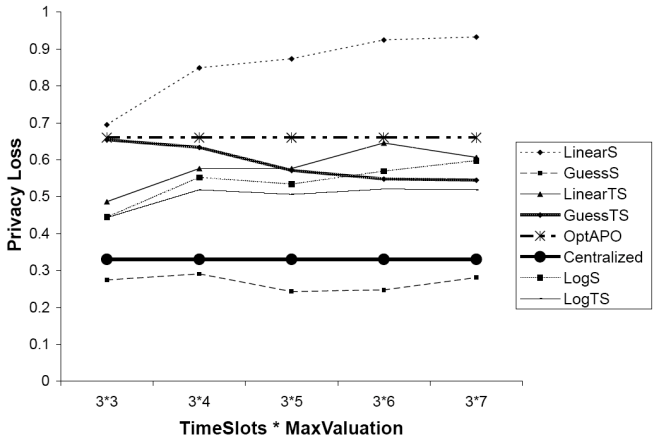

Figure 3: Privacy loss as K increases (chain 1)

where each case has valuations chosen uniformly from $\mathcal{V}$ for each time slot for all agents. Figures 3, 4, 5, and 6 provide privacy loss results for increasing K and T, for two different chains and the six metrics along with OptAPO and centralized search. In all the graphs, the y-axis denotes the average privacy loss (averaged over 10 runs) , while the x-axis denotes the problem type, $K * T$. Figure 3 presents results of privacy loss as K is increased from $3 - 7$ for the six metrics presented in Section 5.1 along with OptAPO and Centralized. Each line indicates the privacy loss associated with a metric, for instance the LinearS metric gives that the privacy loss varies from $0.7 - 0.9$ as K increases. Note that OptAPO and Centralized are fixed at 2/3 and 1/3, regardless of metric. Here are the key observations that can be made from Figure 3:

1. Different metrics give distinct values of privacy loss. For instance, LogS provides privacy loss in the range 0.45-0.6, while LinearS is in the range (0.7 - 0.9).

2. The phenomena (the relative privacy loss) observed across different metrics is not identical. For instance, the relative privacy loss (as K increases) associated with LinearS is "0.2" (0.9 at 3*7 - 0.7 at 3*3), whereas for GuessS it is "-0.1" (0.55 at 3*7 - 0.65 at 3*3).

3. All the metrics (except GuessS) indicate that the privacy lost using a centralized approach is far less than that lost by distributed approaches, SBB and OptAPO. For instance, all metrics except GuessS are in the range 0.45-0.9, while centralized has a privacy loss of 0.33.

5.4 Implications

This first key result of our experiments is the variance in levels of privacy loss for the same communication message exchange across various metrics. Most current research focuses on cross-algorithm performance for a fixed metric. VPS and the idea behind unifying the metrics into a common framework and critically within a common span for best and worst case privacy levels (by choosing the appropriate α and β parameters above) lead to the first cross-metric comparison in our community. As we can see, for a fixed algorithm, the measure of its performance with respect to privacy can be selected arbitrarily by the capriciousness in selecting a metric. It is not only the absolute measure that varies, but also the phenomena as we vary the number of valuations and number of timeslots. We see that for some metrics, the privacy loss increases and for other metrics the privacy loss decreases as the possible world

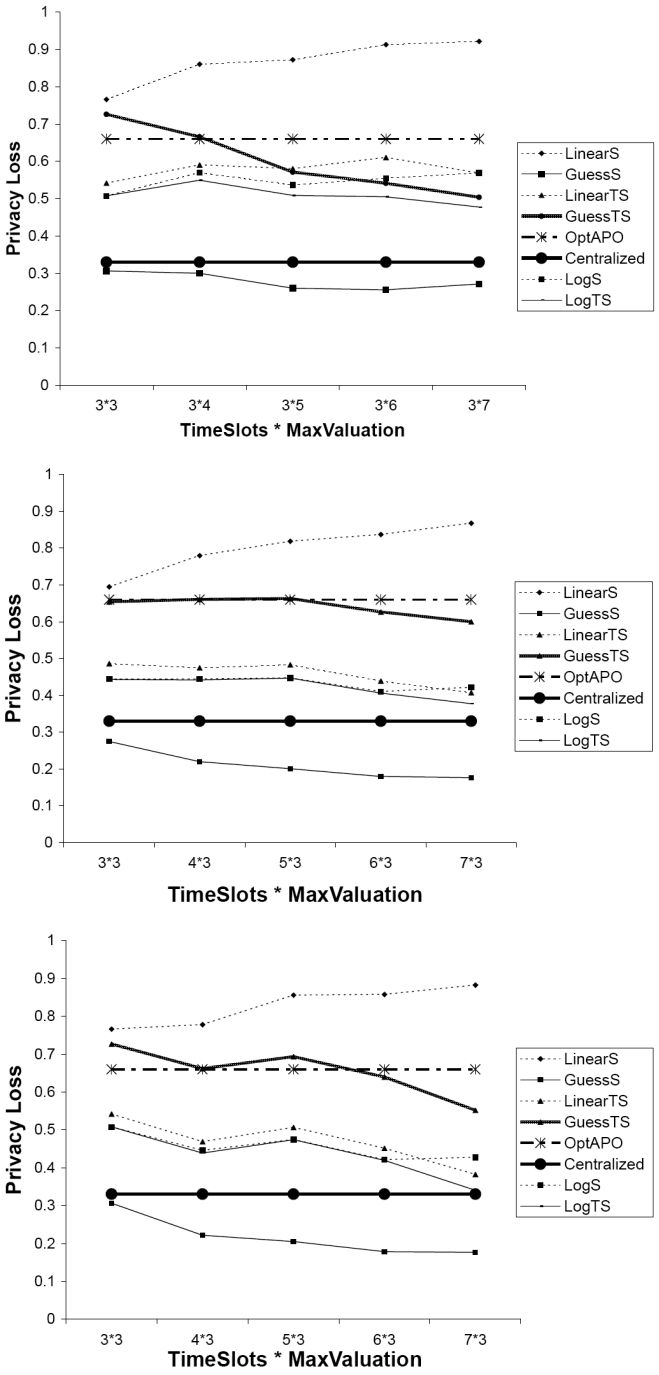

Figure 6: Privacy loss as T increases (chain 2)

space increases. These results imply that while investigating privacy, one must spend considerable effort justifying the appropriateness of a particular measure of privacy for a particular domain. All the metrics suggested here could be considered "reasonable" as they were generated to meet some justifiable qualitative property, yet they have vastly different implications on the conclusions about privacy loss in collaboration. Developing methodologies to evaluate privacy metrics will be a key challenge and VPS provides a quantitative framework from which to pursue this investigation.

A second key result is the superiority of centralization in protecting privacy. Most research ignores centralization in cross-algorithm performance analysis as it is assumed that distribution is a superior course of action. We see here that it is not the case. In fact, the only case where distribution comes close to matching centralization is when the metric easiest to dismiss, GuessS. The GuessS metric captures another agent's ability to guess the observed agent's schedule accurately. Thus, if one began with 10,000 possible states, it would have to be whittled down to 20 before a 5% loss in privacy was observed. The GuessS metric is the one that depicts the least privacy loss from the experiments. According to all the other metrics, centralization significantly outperformed SynchBB and OptAPO. This implies that in building any distributed communication mechanism for privacy, its performance must always be compared with centralization as a baseline to justify distribution as a path to privacy preservation. It also implies that we must look more closely at the algorithms that we as a community develop, and investigate message protocols more deeply to ensure that privacy is protected.

From our experiments, we have extracted the notion that the key to managing privacy loss is minimizing the inferences that other agents are able to make about one's possible states from the messages transmitted. We chose to investigate SynchBB over Adopt [7] as it uses synchronization to greatly reduce the number of messages sent. We note that SynchBB sends costs in two directions while Adopt only sends costs in one. It would be interesting to see if this unidirectional feature can counter the vastly greater number of messages generated in Adopt. Another possible key for the source of our results is the size of the scheduling example. As justified in [8], meeting scheduling is an *incremental scheduling* problem where events to be scheduled arrive gradually. Thus, being able to schedule a small number of meetings that appear dynamically is a critical problem and thus, this example represents a significant canonical problem in the domain. To assure diversity, we varied the number of time slots and the range of valuations. To generate different inference rules, we considered both a clustered and an interspersed chain. In all scenarios, our two key findings, the variance in metrics and the stunning performance of centralization, were virtually identical.

6. SUMMARY

Three key contributions of this paper are (1) a general framework, Valuations of Possible Worlds (VPS) to quantitatively evaluate privacy loss in multiagent settings, (2) the unification of existing notions of privacy into VPS, and (3), the analysis of DCOP algorithms and privacy metrics using VPS. This analysis leads to the implications that (1) the conclusions that can be drawn about privacy loss for any algorithm can vary widely depending on the metric used, and (2) centralization should not be ignored when comparing algorithms with regard to privacy. This paper should serve as a call to arms for the community to improve privacy protection algorithms and further research on privacy.

Acknowledgments

This material is based upon work supported by DARPA, through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

7. REFERENCES

[1] *CALO: Cognitive Agent that Learns and Organizes*, 2003. http://www.ai.sri.com/project/CALO, http://calo.sri.com.

[2] M. S. Franzin, F. Rossi, E. C. Freuder, and R. Wallace. Multi-agent constraint systems with preferences: Efficiency, solution quality and privacy loss. *Computational Intelligence*, 20(2):264–286, 2004.

[3] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In G. Smolka, editor, *Principles and Practice of Constraint Programming*, pages 222–236, 1997.

[4] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: efficient complete solutions for distributed multi-event scheduling. In *AAMAS 2004*, New York, NY, July 2004.

[5] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 438–445, 2004.

[6] A. Meisels and O. Lavee. Using additional information in discsps search. In *Distributed Constraint Reasoning*, Toronto, CA, 2004.

[7] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems*, Sydney, Australia 2003.

[8] P. J. Modi and M. Veloso. Multiagent meeting scheduling with rescheduling. In *DCR*, 2004.

[9] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real-world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

[10] M. C. Silaghi and B. Faltings. A comparison of distributed constraint satisfaction approaches with respect to privacy. In *DCR*, 2002.

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

[12] M. C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with private constraints. In *Proceedings of Autonomous Agents*, pages 177–78, 2000.

[13] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *Principles and Practice of Constraint Programming - CP 2002, LNCS 2470*, pages 387–401, Berlin, 2002. Springer.