A General Privacy Loss Aggregation Framework for Distributed Constraint Reasoning*

Jimmy H.M. Lee
Dept of Computer Science & Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
jlee@cse.cuhk.edu.hk

Terrence W.K. Mak
NICTA Victoria Lab &
University of Melbourne
VIC 3010, Australia
Terrence.Mak@nicta.com.au

Yuxiang Shi
Dept of Computer Science & Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
yshi@cse.cuhk.edu.hk

***Abstract*—Distributed constraint solving are useful in tackling constrained problems when agents are not allowed to share his/her private information to others and/or gathering all necessary information to solve the problem in a centralized manner is infeasible. With these two limitations, distributed algorithms solve the problem by coordinating agents to negotiate with each other. However, once information is exchanged during negotiation, the private information may be leaked from one agent to another. We propose and design a framework based on Valuation of Possible States (VPS) to evaluate how well a distributed algorithm preserves the totality of all private information on the entire system when solving distributed constraint optimization problems, by allowing the uses of different aggregators aggregating agents' individual privacy loss. Two classes of aggregators: idempotent aggregators and risk based aggregators are proposed. We further proposed generalized inference rules to infer privacy loss of individual agents. We implement our work on four distributed constraint solving algorithms: Synchronous Branch and Bound (SynchBB), Asynchronous Distributed Constraint Optimization (ADOPT), Branch and Bound ADOPT (BnB-ADOPT), and Distributed Pseudo-tree Optimization Procedure (DPOP). Preliminary experimental evaluations on two benchmarks, Distributed Multi-Event Scheduling Problem (DiMES) and Random Distributed COP, comparing the four algorithms are performed.**

***Keywords*-Distributed reasoning, privacy loss, aggregation axiom**

I. Introduction

The task at hand is that of a Distributed Multi-Event Scheduling Problem (DiMES) [10] where there are a set of events, a group of attendees, and a set of free timeslots for each of the attendees. Each attendee is required to attend a list of events, and there are preferences (modeled by penalty costs) for each of his/her free timeslots. The goal of the problem is to assign free timeslots to events for each of the attendees such that all attendees can attend the events he/she is required to attend and the total penalty cost is minimized. For each attendee, the information on which events he/she needs to participate, the available timeslots for him/her, and the preference for each timeslot should not be revealed or known by the others. For this type of problems, gathering all information and tackling the problem in a centralized manner is not an ideal way since the agent solving the problem can gather all the attendees' (private) information. For security-sensitive applications, there may also be geographical restrictions disallowing information to be dispersed to third parties. To overcome these limitations, we usually tackle the problem via message exchanges in a distributed manner but aim at minimising privacy loss.

Distributed Constraint Optimization Problem (DCOP) is a framework for modeling these problems, and various algorithms, e.g. Asynchronous Distributed Constraint Optimization (ADOPT) [11], Branch and Bound ADOPT (BnB-ADOPT) [16], and Distributed Pseudo-tree Optimization Procedure (DPOP) [13], have been proposed for solving DCOP. Many of these algorithms utilize message exchange methods (i.e. negotiations in DiMES) to communicate with other agents (i.e. attendees in DiMES). However, once an agent exchanges messages with other parties/agents, there are chances to leak information. In real life situations, some of these information may be confidential, and we hope to minimize leakage of such information as much as we can. How well a distributed algorithm keeps the privacy of agents is always one of the major concerns in distributed reasoning. In this paper, our primary focus is to study and design a framework to evaluate and compare distributed algorithms on how well they protect agents' private information.

Much work [4–6, 9, 14] has been proposed to compute and measure the privacy loss of individual agents. However, for comparing system privacy loss with different algorithms, many just utilize simple aggregation techniques (e.g. max and average) to aggregate the privacy loss of individual agents and use the aggregated results to indicate how well the algorithm performs. Note that different aggregation techniques may give different results. It is worthwhile to understand and further study what kinds of aggregators we should use in an application. Aggregation and fusion of information are basic concerns [2] for all kinds of knowledge based systems. In general, we can view aggregation as simultaneously utilizing different pieces of information in order to come up with a decision. Restricting our attention to only simple aggregators may not be adequate to compare the quality of algorithms. Sometimes, it may even be worthwhile to use more complex

*We are grateful to the anonymous referees for their constructive comments. The work was generously supported by grants CUHK413808, CUHK413710 and CUHK413713 from the Research Grants Council of Hong Kong SAR. The work of Mak was performed while he was at CUHK.

aggregators or even hybrid aggregators.

Our primary goal is to propose a principled system level aggregation framework allowing us to devise aggregators for combing individual agents' privacy loss, by classifying and characterizing the properties of different types of aggregators. We then give an outline on how to compute and derive the amount of privacy of an agent deduced by others via a set of *basic inference rules* in general DisCOPs. These rules are generalized from previous work on Distributed Multi-Event Scheduling. We then implement our framework and inference rules on four distributed constraint solving algorithms: Synchronous Branch and Bound (SynchBB), Asynchronous Distributed Constraint Optimization (ADOPT), Branch and Bound ADOPT (BnB-ADOPT), and Distributed Pseudo-tree Optimization Procedure (DPOP). Preliminary experimental evaluations on two benchmarks comparing the four algorithms are performed.

II. Background

In this section, we first give the definition of DCOP, followed by showing the description for the Distributed Multi-Event Scheduling Problem. We then describe the VPS framework by Maheswaran *et al*.

A. Definition of DCOP

A distributed constraint optimization problem (DCOP) $\mathcal{P}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{A}, \phi)$ where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of *variables*, $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite *domains* for each of the variables in $\mathcal{X}$, C is a set of *constraints* (also called cost functions), and $\mathcal{A} = \{a_1, \ldots, a_k\}$ is a set of (distributed) *agents*. ϕ is a *mapping*: $\mathcal{X} \mapsto \mathcal{A}$ which maps each variable in $\mathcal{X}$ to an agent in $\mathcal{A}$. We denote $x_i = v_i$ an *assignment* assigning value $v_i \in D_i$ to variable x_i, and the set of assignments $l = \{x_1 = v_1, x_2 = v_2, \ldots, x_n = v_n\}$ is a *complete assignment* on variables in $\mathcal{X}$, where v_i is the value assigned to x_i. A *partial assignment* $l[S]$ is a projection of l onto variables in $S \subseteq \mathcal{X}$. Each constraint $C_S \in \mathcal{C}$ is a constraint over a set $S = \{x_{S_1}, x_{S_2}, \ldots, x_{S_k}\}$ of variables, defining real-valued costs for all possible joint assignments $\{x_{S_1} = v_{S_1}, x_{S_2} = v_{S_2}, \ldots, x_{S_k} = v_{S_k} | v_{S_1} \in D_{S_1}, v_{S_2} \in D_{S_2}, \ldots, v_{S_k} \in D_{S_k}\}$. The *cost* of a complete assignment l in $\mathcal{X}$ is defined as: $cost(l) = \sum_{C_S \in \mathcal{C}} C_S(l[S])$. The objective is to find a complete assignment with minimum costs.

For simplicity, we write C_i for the unary constraint on variable x_i and $C_i(v)$ for the cost of constraint C_i with the value assignment $\{x_i = v\}$. Note that in DCOP, variables and constraints are distributed among agents. An agent a_j holds a variable x_i iff $\phi(x_i) = a_j$, and a_j holds a constraint C_S iff $\exists x_i \in S s.t. \phi(x_i) = a_j$. If ϕ is surjective, then all agents must hold at least one variable. If ϕ is injective, then no two (or more) variables are being held by the same agent. For DCOPs, we follow previous work [4, 6, 9] by considering an agent's *private information* to be the unary costs of his/her variables. In other words, an agent does not want to disclose his/her variables' unary costs to other agents.

B. Distributed Multi-Event Scheduling

The *Distributed Multi-Event Scheduling* [10] framework (DiMES) includes a set of N *people* $\mathcal{R} = \{R_1, \ldots, R_N\}$, a set of K *events* (also called event set) $\mathcal{E} = \{E_1, \ldots, E_K\}$, and a set of T *timeslots* $\mathcal{T} = \{1, \ldots, T\}$ where $t \in \mathcal{T}$ refers to the t^{th} timeslot. We have a set $F = \{F_1, F_2, \ldots, F_N\}$ of cost functions. Cost functions F_i are known by R_i. For each $F_i \in F$, $F_i(t)$ gives the costs for person R_i keeping timeslot t free. Each event E_j is a tuple $(A_j, L_j; W_j)$, where $A_j \subseteq \mathcal{R}$ is the subset of people that are required to attend for event E_j, $L_j \in \mathcal{T}$ is the timeslot the event E_j will be going to be held. W_j is a set $\{W_j(R_1), W_j(R_2), \ldots, W_j(R_N)\}$, where $W_j(R_i)$ denotes the reward of person R_i for attending event E_j. Overall, the reward that a person R_i obtains from scheduling event E_j in timeslot t is $\Delta(R_i, E_j, t) = W_j(R_i) - F_i(t)$. Our goal is to find a set of timeslots $\{L_1, \ldots, L_K\}$ for the set of events $\mathcal{E} = \{E_1, \ldots, E_K\}$ s.t.

$$\sum_{E_j = (A_j, L_j; W_j) \in \mathcal{E}} [\sum_{R_i \in A_j} \Delta(R_i, E_j, L_j)] \text{ is maximized.}$$

To model DiMES as a DCOP, we use the model called *Private Events as Variables* (PEAV) [10]. Each person $R_i \in \mathcal{R}$ will be modeled as an agent $a_i \in \mathcal{A}$ in the DCOP. For each agent a_i (i.e. person R_i), he/she will own a set of variables $\{x_{i_j} | R_i \in A_j \wedge E_j = (A_j, L_j; W_j) \in \mathcal{E}\}$, where each x_{i_j} corresponds to the event E_j he/she needs to attend and $\phi(x_{i_j}) = a_i$. We set the domain of each variable x_{i_j} to be the set of timeslots $\mathcal{T} = \{1, \ldots, T\}$. We have (inter-agent) constraints between agents to restrict that different agents choose the same time for the same event, and also have (intra-agent) constraints to restrict no timeslot for an agent has been assigned to two events. These two types of constraints will return either a cost of 0 (i.e. satisfiable) or a cost of ∞ (i.e. unsatisfiable). Unary constraints C_{i_j} on variables x_{i_j} will be used to return costs for Δ, where $\forall t \in \mathcal{T}, C_{i_j}(t) = \Delta(R_i, E_j, t)$. Note that for DiMES, sometimes we can assume the reward $W_j(R_i)$ of a particular person R_i attending an event E_j is known by all agents [9]. In this case, we may further assume the private information of a person R_i is the costs $F_i(t)$ for R_i keeping a timeslot t free.

C. Valuation of Possible States

Valuation of Possible States (VPS) [9] is a privacy loss metric framework. The framework views an agent's privacy loss as the amount of privacy leaked by him/her to other agents. We call this type of privacy loss *individual privacy loss*. By combining and fusioning different agent's individual privacy loss, we obtain the privacy loss of an algorithm/system, namely called *system privacy loss*. VPS computes the privacy loss of an agent from the viewpoint of other agents. To calculate the privacy loss V_i of an agent a_i, the framework first calculates the amount of a_i's private information V_i^j revealed by others (i.e. all agents a_j except a_i himself). The framework allows us to use different metrics, e.g. entropy-based metrics [3, 4, 7], proportional metrics [9], and state-guessing metrics [5, 14] to measure and compute V_i^j. Then, the

framework computes V_i by $\sum_{a_j \in \mathcal{A}, j \neq i} V_i^j$. In other words, V_i is the privacy loss deduced by other agents. Before execution of a distributed algorithm, we assume V_i of each agent a_i is the amount of common knowledge, and V_i will only be increased when message exchanges occur (executed by the distributed algorithm). The privacy loss p_i of an agent a_i during the solving procedure will then be the difference on V_i before and after executing the procedure. Finally, VPS combines the privacy loss of all agents by taking *average* to represent the loss of an algorithm/system.

III. Aggregation Framework for Privacy Loss

VPS is a coherent framework in measuring the privacy loss of individual agents during message exchanges. To indicate the privacy loss of an algorithm/system, they choose to take average on agents' individual privacy losses. In fact, comparing and checking which system/algorithm is better is a kind of decision making process. In many cases, the decision making process needs to consider multiple criteria or attributes [8]. In such decision making process, the decision making person/agent is faced with a set of decision alternatives and the ratings per alternative per criterion which then have to be fused together in order to produce an overall rating, and the fusion process is carried out by some form of aggregation operation [1]. In many cases, this operation is in some form of weighted arithmetic mean [1], and this works well in situations in which any differences are viewed as being in conflict because the operator reflects a form of compromise behaviour among the various criteria [8]. However, when these conditions do not hold, arithmetic mean is a poor choice [8]. Thus, in general, we should also provide aggregation operations/functions reflecting the user's individual choice of behaviour among these different criteria. For aggregating privacy losses, similarly, we observe there are many more ways to aggregate individual privacy loss other than taking average. We propose a principled framework to study and investigate the possibilities to incorporate other useful aggregators. Case studies on several popular aggregation functions will be investigated. We then propose to incorporate and utilize risk based aggregations [8] to aggregate privacy losses.

A. Fundamental Axioms

Without loss of generality, we normalize privacy losses p_i of an agent a_i to be real values from the domain of $[0, 1]$, where $p_i = 0$ means there are no privacy loss for the agent a_i and $p_i = 1$ means the agent a_i loses all privacy to all the others, i.e., all the other agents know all the privacy of agent a_i. We write $\mathbb{0}^N$ and $\mathbb{1}^N$ to be an ordered N-tuple containing all zeroes and an ordered N-tuple containing all ones respectively. We now define the aggregation function for aggregating individual privacy losses based on several fundamental axioms [1].

Definition 1. *Given a DCOP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{A}, \phi)$. An aggregation function* Agg *for aggregating $N \geq 1$ privacy losses $p_1, p_2, \ldots, p_N$ for agents $a_1, a_2, \ldots, a_N$ in $\mathcal{A}$ is a function: $[0, 1]^N \mapsto [0, 1]$. The input is a vector (or tuple) whose elements are the privacy loss of each agent and the output is the system privacy loss. We require the function to satisfy the following axioms:*

$$\text{If } N = 1: \ \forall p_1 \in [0, 1], \text{Agg}([p_1]) = p_1 \tag{1}$$

$$\forall N \geq 1: \ \text{Agg}(\mathbb{0}^N) = 0 \wedge \text{Agg}(\mathbb{1}^N) = 1 \tag{2}$$

$$\forall N \geq 1: \ p_1 \leq p'_1, p_2 \leq p'_2, \ldots, p_N \leq p'_N \implies \text{Agg}([p_1, p_2, \ldots, p_N]) \leq \text{Agg}([p'_1, p'_2, \ldots, p'_N]) \tag{3}$$

Axiom (1) requires any aggregation function aggregating the *privacy loss of a single agent* must not alter the privacy loss. Axiom (2) gives the *boundary conditions* of the aggregated privacy loss. If there are no privacy losses for all of the agents, the aggregated privacy loss must also be 0. Similarly, if all agents lose all privacy, then the aggregated privacy loss must be 1. Axiom (3) restricts all aggregation functions must be *monotonic*, i.e. a non-decreasing function. The aggregated privacy loss must increases if the list of individual privacy loss is increasing.

Finding arbitrary functions satisfying the definition of aggregation functions is not hard. However when analysing algorithms, we usually have goals, and these goals may vary for different users. It is useful if we can clearly identify which classes/types of aggregation functions allow the user to achieve his/her goal. In this paper, we identify and discuss two classes of aggregation functions: idempotent aggregation and risk-based aggregation.

B. Idempotent Aggregation

Many common used functions, e.g. minimization, maximization, and average, are idempotent aggregation functions. We first introduce this class of aggregation functions as many functions from this class is popular or common to users.

Definition 2. *Given a DCOP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{A}, \phi)$. An idempotent aggregation function* Agg *for aggregating $N \geq 1$ privacy losses $p_1, p_2, \ldots, p_N$ for agent $a_1, a_2, \ldots, a_N$ in $\mathcal{A}$ is an aggregation function with the idempotent condition:*

$$p_1 = p_2 = \ldots = p_N = c \implies \text{Agg}([p_1, p_2, \ldots, p_N]) = c,$$

where c is a constant in $[0, 1]$.

Idempotence means if all the agents' privacy loss is the same, the aggregated (system) privacy loss should be equal to the agents' privacy loss.

Lemma 1 (Upper and Lower Bound)**.** *Given an idempotent aggregation function* Agg *with $N \geq 1$ privacy losses $p_1, p_2, \ldots, p_N$ for agents $a_1, a_2, \ldots, a_N$ in a DCOP $\mathcal{P}$.*

$$\min_{i=1}^{N} p_i \leq \text{Agg}([p_1, p_2, \ldots, p_N]) \leq \max_{i=1}^{N} p_i$$

We skip the proof as it is easy to derive by using the idempotent condition and the monotonicity condition. Similar proof can be found and derived from the work of Yager *et al.* [15]. A useful corollary for this lemma is that given privacy losses of a set of agents, the aggregated privacy loss generated by an idempotent aggregation must be bounded by

the minimum and maximum losses. All functions from the class of idempotent aggregation functions must not return values beyond the range of the input values, i.e. return values larger (smaller) than the largest (smallest) input. We can also view this class of aggregation functions as a kind of compromised aggregation [8].

Lemma 2. *Given a DCOP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{A}, \phi)$ and $N \geq 1$ privacy losses where each p_i corresponds to an agent $a_i \in \mathcal{A}$. Function k^{th} max which chooses the k^{th} largest privacy loss from these agents' privacy loss and function weighted average $\sum_{i=1}^{n}(w_i \times p_i)$, where $\sum_{i=1}^{n} w_i = 1$ are both idempotent aggregation functions.*

We skip the proof as it can be easily constructed according to the basic definition of idempotence and the three axioms for aggregation functions. Note that maximization and minimization are special cases of k^{th} max and arithmetic mean is a special case of weighted average. An immediate corollary is that these functions are all idempotent aggregation functions.

In real-life applications, we may encounter complex/large problems involving different groups of agents, where each group may have different privacy requirements. For each of these groups, we may need to use different aggregation functions in measuring their privacy losses. To further obtain the system privacy loss, we need an aggregation function which aggregates the privacy losses of each group to indicate the loss for the whole problem/system. These combined aggregation functions are called *hybrid aggregations*.

Example 1. *Suppose we have a set of nine agents $\{a_1, \ldots, a_9\}$ and the privacy losses corresponding to these agents, $a_1, \ldots, a_9$, are represented as a vector $[p_1, \ldots, p_9]$. We want to create an aggregation function which first: 1) takes the maximum on the privacy losses of agent a_1 and a_2, 2) takes the median on the privacy losses of agent a_3, a_4, a_5, and a_6, and 3) takes the arithmetic mean on the privacy losses of agent a_7, a_8 and a_9, followed by: returning the maximum of the three aggregated answers as outputs. The hybrid function $\mathrm{Agg}(p_1, p_2, \ldots, p_9)$ will then be: $\max(\max\{p_1, p_2\}, \mathrm{median}\{p_3, p_4, p_5, p_6\}, [(p_7 + p_8 + p_9)/3])$.*

It is easy to observe the aggregation function above, Agg, is an idempotent aggregation. According to the basic definition of idempotence and the three axioms of privacy loss aggregator, we can easily obtain the following lemma.

Lemma 3. *A function combining idempotent aggregation functions using an idempotent aggregator is idempotent.*

Note that all these aggregation functions we mentioned above, in general, do not need any user guidance during computations and there are no required external user inputs other than privacy losses. These functions may not be an ideal choice when we need to consider human's behaviour.

C. Risk Based Aggregation

Evaluating and computing risks is an important factor to consider when people are making decisions. Privacy losses are also risks. The more privacy the agent loses, the more risky the agent is perceived to be. However for aggregating privacy losses, the choice of aggregators is likely to vary from user (algorithm designer) to user because their own view on how to do the aggregation can be different from each other. For example, when aggregating two privacy losses, some users might be optimistic and believe the result should not be higher than either of the two; some might be pessimistic and believe the result should not be lower than either of the two; and some might be more neutral and believe the result should be in-between [8]. For different situations or risk levels, the attitude of a user to consider the privacy loss can be also quite different. One classic approach is to set up a threshold so that privacy losses higher/lower than the threshold are regarded as being positive/negative. We then handle these two sets of privacy losses separately by using different types of aggregations. This makes the situation more complex. Whenever we want to compare algorithms, we should also consider the individual preferences and attitudes towards risk for each of the users [8]. Our goal is to introduce such notions to allow users devising aggregation function flexible enough to handle such criteria.

In this paper, we employ the standard assumption from the economic game theory community [8, 12] that users can exhibit three broad types of attitude toward risk: risk-seeking (optimistic), risk-averse (pessimistic) or risk neutral (neutral), and propose aggregation functions corresponding to each of these attitudes. When a user wants an aggregation to be risk-seeking (risk-averse resp.), the user is optimistic (pessimistic resp.), i.e. thinking the system is less (more resp.) privacy-sensitive. We also add the risk-neutral aggregation for users with neutral attitude, i.e. not risk-seeking and also not risk-averse. We define the privacy-loss threshold τ to be a real value $\in [0, 1]$. The threshold is used to represent the user's expected privacy loss of the system which varies from person to person. If a privacy loss p_i is larger (smaller resp.) than τ for an agent a_i, it means the loss is worse (better resp.) than the user's expectations. For such losses, we add an extra label $^+$ sign ($^-$ sign resp.) to p_i. Hence, a loss p_i can be either p_i^+ or p_i^-. By adding the two signs, we can then seperate all losses into two sets.

By following and extending the work of (Luo and Jennings, 2008), we now give risk-based aggregation functions for aggregating individual privacy losses of agents.

Definition 3 (Risk-based aggregation functions)**.** *Given a DCOP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{A}, \phi)$ and a user defined $\tau \in [0, 1]$. Suppose we have $N \geq 1$ privacy losses vector $\vec{G} = [p_1, p_2, \ldots, p_N]$ for agent $a_1, a_2, \ldots, a_N$ in $\mathcal{A}$. We partition $\vec{G}$ into two vectors: $\vec{G}^+$ and $\vec{G}^-$, where $\vec{G}^+$ ($\vec{G}^-$ resp.) contains all privacy losses $p_i \in \vec{G}$ s.t. $p_i > \tau$ ($p_i < \tau$ resp.).*

For all $p_1^+, \ldots, p_e^+ \in \vec{G}^+$, an aggregation function Agg *is:*

- *τ-high-pessimistic if* $\mathrm{Agg}(p_1^+, \ldots, p_e^+) \geq \max(p_1^+, \ldots, p_e^+)$
- *τ-high-optimistic if* $\tau \leq \mathrm{Agg}(p_1^+, \ldots, p_e^+) \leq \min(p_1^+, \ldots, p_e^+)$
- *τ-high-neutral if* $\min(p_1^+, \ldots, p_e^+) < \mathrm{Agg}(p_1^+, \ldots, p_e^+) < \max(p_1^+, \ldots, p_e^+)$

For all $p_1^-, \ldots, p_f^- \in \vec{G}^-$, an aggregation function Agg *is:*

- *τ-low-pessimistic if* $\tau \geq \text{Agg}(p_1^-, \ldots, p_f^-) \geq \max(p_1^-, \ldots, p_f^-)$
- *τ-low-optimistic if* $\text{Agg}(p_1^-, \ldots, p_f^-) \leq \min(p_1^-, \ldots, p_f^-)$
- *τ-low-neutral if* $\min(p_1^-, \ldots, p_f^-) < \text{Agg}(p_1^-, \ldots, p_f^-) < \max(p_1^-, \ldots, p_f^-)$

For all $p_1^-, \ldots, p_f^- \in \vec{G}^-$ and for all $p_1^+, \ldots, p_e^+ \in \vec{G}^+$, an aggregation function Agg *is:*

- *τ-combined-pessimistic if* $\text{Agg}(p_1^+, \ldots, p_e^+, \tau) \leq \text{Agg}(p_1^-, \ldots, p_f^-, p_1^+, \ldots, p_e^+)$
- *τ-combined-optimistic if* $\text{Agg}(p_1^-, \ldots, p_f^-, \tau) \geq \text{Agg}(p_1^-, \ldots, p_f^-, p_1^+, \ldots, p_e^+)$
- *τ-combined-neutral if* $\text{Agg}(p_1^-, \ldots, p_f^-, \tau) < \text{Agg}(p_1^-, \ldots, p_f^-, p_1^+, \ldots, p_e^+) < \text{Agg}(p_1^+, \ldots, p_e^+, \tau)$

In the above definition, τ-high-pessimistic, τ-high-optimistic and τ-high-neutral means that the aggregation for the positive privacy losses is aggregated in a risk-adverse, risk-seeking, and risk-neutral way respectively. Similarly, we define τ-low-pessimistic, τ-low-optimistic, and τ-low-neutral to aggregate the negative privacy losses p_i^- in a risk-adverse, risk-seeking, and risk-neutral way. To combine both types of privacy losses (i.e. p_i^+ and p_i^-), functions τ-combined-pessimistic, τ-combined-optimistic, and τ-combined-neutral are defined to aggregate both types of privacy losses in a risk-adverse, risk-seeking, and risk-neutral way. Based on the above definitions, we can then construct appropriate risk-based aggregation functions for different users' behaviour towards risks, by: 1) setting τ to represent the user's expectation on the privacy loss of the system/algorithm, and 2) choosing the user's attitudes for τ-high, τ-low and τ-combined.

Example 2. *Suppose we have a set of agents $\{a_1, a_2, \ldots, a_5\}$ and the vector of privacy loss for each agent is $[p_1 = 0.2, p_2 = 0.4, p_3 = 0.6, p_4 = 0.5, p_5 = 0.7]$. Assume now agent a_1 expects the system privacy loss (τ) to be 0.5, and we have $\vec{G}^+ = [p_3 = 0.6, p_5 = 0.7]$ and $\vec{G}^- = [p_1 = 0.2, p_2 = 0.4]$. If we choose maximization function as our aggregation function, then according to the definition, it will be classified as τ-high-pessimistic, τ-low-pessimistic, and τ-combined-pessimistic. It is easy to observe choosing the maximum losses as a representative for the system privacy loss is a risk-adverse approach. Similarly, if we choose geometric mean as our aggregation function (*Agg*), we can observe the following.*

$$\begin{aligned}
\min(0.6, 0.7) = 0.6 &< \text{Agg}(0.6, 0.7) \approx 0.648 \\
&< \max(0.6, 0.7) = 0.7 \\
\min(0.2, 0.4) = 0.2 &< \text{Agg}(0.2, 0.4) \approx 0.283 \\
&< \max(0.2, 0.4) = 0.4 \\
\text{Agg}(0.2, 0.4, 0.5) \approx 0.342 &< \text{Agg}(0.2, 0.4, 0.6, 0.7) \approx 0.428 \\
&< \text{Agg}(0.5, 0.6, 0.7) \approx 0.594
\end{aligned}$$

Note that the above calculations for geometric mean are rounded to three significant figures. We can observe geometric mean is classified as τ-high-neutral, τ-low-neutral, and τ-combined-neutral. In fact, the three commonly found aggregators: maximization, minimization and average (arithmetic/geometric mean) are typical risk-adverse, risk-seeking, and risk-neutral aggregation functions. It is worth to note that one could ask for aggregation function which only considers the positive (or negative) losses. Suppose one wants to define a function Agg *which takes arithmetic mean only on the positive (or negative) losses,* $\text{Agg}([0.2, 0.4, 0.6, 0.5, 0.7]) = (0.6 + 0.7)/2 = 0.65$ *(or* $= (0.2 + 0.4)/2 = 0.3$*). This is particularly useful if one wants to create functions ignoring all losses below (or above) the threshold. Note that for the function to be an aggregation function, the function must not be undefined on the negative (or positive) losses. Assume the number of agents is greater than one. One possible amendment is to return constant value on cases where all of the privacy losses falls below (or above) the threshold.*

IV. Individual Privacy Losses

Before we use our introduced classes of aggregation functions to compare and analyse systems/algorithms, we have to compute the privacy losses for each of the individual agents. We follow the approach of VPS. Recall VPS [9] is a privacy loss metric framework which views an agent's privacy loss as the amount of privacy leaked by himself to others. To compute the loss of an agent, we alternatively compute the private information deduced by other agents. VPS has introduced methods called *inference rules* to deduce and compute an individual agent's private information (deduced by all the other agents). However, their work are focused on the DiMES problem and the inference rules are crafted for that specific problem on three algorithms: a) a centralized algorithm, b) a partial centralized algorithm OptAPO, and c) a distributed algorithm: SynchBB only.

Our work further extends VPS in the following ways. First, we further generalize their inference rules to compute individual privacy loss not only applicable to DiMES, but also on general/random DCOPs. Second, we extend and implement the rules on deducing and computing privacy losses for individual agents (for both DiMES and general/random DCOP) on three widely known distributed algorithms: ADOPT, BnB-ADOPT, and also DPOP. We also implement and cover our generalized rules on SynchBB. In addition, some of their methods have the assumption that the constraint graph of a DCOP is totally or partially known to all agents. This limit the applicability of their work as it breaks total topology privacy [3]. Our work do not require this assumption.

We will first introduce the principle of the inference rules in VPS, i.e. how an agent can infer information from the other agents. Then, we introduce how we generalize their rules for random DCOPs. Due to space limitations, we skip the detailed algorithms and implementation details on incorporating these (general) inference rules for the four algorithms.

A. Inference Rules Principle

Distributed algorithms usually solve DCOPs via sending messages, and privacy may be leaked with message passing. We follow previous work by assuming the private information of an agent is the unary costs of its variables. The inference rules from Maheswaran *et al.* have three main characteristics.

(a) The inference rules *only* collect information from messages during message exchanges. In particular, many of these rules are interested on costs information only.
(b) The inference rules have the knowledge on how each agent computes costs.
(c) The inference rules can take account on the properties of a specific problem.

Point (a) is to limit the agent's ability to collect information from the others, by restricting them to collect information during message exchanges only. Messages with cost information is common in DCOP algorithms, e.g. *cost* messages in ADOPT/BnB-ADOPT, *down* and *up* messages in SynchBB and *UTIL* messages in DPOP. These kinds of messages usually carry information related to the costs generated from a particular groups of agents, e.g. a group of agents in a (pseudo-)subtree in ADOPT/BnB-ADOPT. By reverse engineering these costs, an agent can infer the range of unary costs of the variables. The more messages an agent collects, the more precise the range of costs the agent can infer. In other words, the more private information is being lost. However, for an agent to deduce costs information from messages he/she received, it is essential for him/her to know how others generate such cost messages. Point (b) is the required assumption, and its validity depends on the DCOP algorithm we use. In most DCOP algorithms, however, the reasoning logic is the same for every agents and all of the agents should know the algorithm they are going to execute beforehand. It is reasonable for us to assume all agents should know the exact procedure and behaviour of other agents and the assumption is automatically satisfied. Point (c) is mainly use to allow the rules becoming more accurate, by further utilizing the properties of a specific problem. For example in DiMES, all agents attending the same meeting must meet at the same time. The variable used to model that particular event across different agents must choose the same timeslot. By using such property, we can further devise rules specific to the problem to allow agents making better guesses on other's unary costs. Example 3 shows how to obtain privacy loss after an agent receives new cost messages.

Example 3. *Given a DiMES with three agents a_1, a_2 and a_3 having the same set of free timeslots $\mathcal{T} = \{t_1, t_2, t_3\}$. We assume the set of possible unary costs for these three agents for each timeslot is $\{1, 2, 3, 4\}$. Initially, a_1 only knows that a_2 and a_3 can choose one of the four costs in $\{1, 2, 3, 4\}$ for all of the timeslots, and he/she believes there are equal chances for the four costs. In the viewpoint of agent a_2, the probability for a_1 to guess correctly his/her costs is $1/4$. We denote the valuation, e.g. entropy, of agent a_1 on a_2 before applying the solving algorithm to be $V^b_{2,1}$. Note that we obtain the valuation from common knowledge only. Suppose agent a_1 and a_2 are required to attend an event E_w. In ADOPT, when a_1 receives a COST message with a value 2 from a_2 (assume no conflict penalty occurs), he/she can reveal that the total cost for the subtree rooted at a_2 is 2. Note that a_1 and a_2 both have a common event E_w, and hence, the timeslot t chosen by a_1 for E_w must also be the same timeslot t chosen by a_2 for E_w. Agent a_1 can then infer that the unary costs of a_2 for timeslot t is less than or equal to 2, i.e. $C_2(t) \in \{1, 2\}$. In the viewpoint of agent a_1, the probability to guess correctly $C_2(t)$ is $1/2$, which raises from $1/4$ to $1/2$. The reason behind is that agent a_1 reveals a tighter range of possible unary costs for agent a_2. During the solving procedure, agent a_1 receives more messages from a_2 and a_3, and the chances for a_1 to guess other agents' unary costs correctly is higher. If all the messages being received are stored, we can easily calculate the valuation of agent a_1 on a_2, denoted by $V^a_{2,1}$, according to the simple inference rules described above after the solving procedure. The privacy loss of a_2 revealed by a_1 will then be the difference between $V^a_{2,1}$ and $V^b_{2,1}$. The total privacy loss of agent a_i will then be the summation of all the privacy loss of a_i revealed by all the other agents.*

Readers may ask what happens if agent a_1 in Example 3 do not have the prior knowledge that a_2 can choose costs from $\{1, 2, 3, 4\}$. A naive way to handle is to set the initial cost range to infinity. However, this prior knowledge relates to the initial beliefs (in game theory) of agent a_1 on the other two agents, and usually in real-life, an agent can predict a reasonable cost range.

B. Generalized Inference Rules

The DiMES problem (with the PEAV representation) is a specific problem. In this problem, agents attending the same event will choose the same timeslot and these agents will have more information to infer a tighter cost range of the chosen timeslot for the other agents. In general, when solving DCOPs, an agent will not be able to determine which value the other chooses if an algorithm does not leak the information. Therefore, even though an agent receives a message containing cost information, he/she cannot infer which timeslot the cost information is referring to. As a result, it seems we are more secure as agents will not be able to get the mapping between value assignments and cost messages in a precise manner.

However, we observe that there are still risks for neighbouring agents to gain costs information from the messages. Suppose an agent a_i receives messages from a_j, a_i can make a weaker inference: $\exists v_j \in D_j$ s.t. $C_j(v_j) \in S$, where S is the newer cost range. In other words, an agent can still realize that there exists a value v_j in which the cost information is referring to. We demonstrate how we utilize such information to compute privacy losses in Example 4.

Example 4. *Given a general DCOP with three agents: a_1, a_2 and a_3, each of these agents a_i holds a variable x_i with a domain of $\{a, b, c, d\}$, and there are only 4 possible choices of unary costs: 1, 2, 3, and 4 for all of the unary constraints. We use ADOPT to solve the problem and use entropy to measure the deduced agent privacy of an agent. Suppose agent a_2 receives three cost messages from a_3 (during the execution of an algorithm) with a cost of 2, 1, and 3 respectively for each of the messages, and he/she deduces the following inference inequalities: $\exists v_i \in D_3$ s.t. $C_3(v_i) \leq 2; \exists v_j \in D_3$ s.t. $C_3(v_j) \leq 1; \exists v_k \in D_3$ s.t. $C_3(v_k) \leq 3$. Since each of the value: v_i, v_j and v_k can either be a, b,*

c, or d. There are 4^3 possible combinations of choices for v_i, v_j and v_k. In Figure 1, we divide these combinations into 5 situations. We denote P_i to be the probability and p_3^i to be the privacy loss for each combination of situation i. Situation 1 shows the case when all of the three values are not equal to each other. This gives a total of $\binom{4}{1} \times \binom{3}{1} \times \binom{2}{1}$ possible combinations. We assume the agents have no extra information other than the three inequalities and the different combinations of choices are independent to each other. The best choice an agent can play is to guess in a uniform random manner with probability of $1/4^3$ for each combination. This gives the probability P_1 of the first situation to be $\binom{4}{1} \times \binom{3}{1} \times \binom{2}{1}/(4^3)$. To calculate privacy loss p_3^1, the first step is to build the mapping between the inequalities and values in D_3. Then, we can calculate the privacy loss by using similar method in Example 3. After obtaining the privacy loss for each of the $\binom{4}{1} \times \binom{3}{1} \times \binom{2}{1}$ combinations, we then sum up all these losses and store the average result over all the combinations. Note that it is possible to reduce redundant computations by exploiting the fact that individual deduced agent privacy for different possible combinations in one situation are the same according to entropy metric. The computation of privacy losses for the other situations is similar. The only difference is that we can further obtain tighter inferencing by using the fact that some of the values are equal. For example in situation 2, we can combine the first two inference inequalities and obtain the inference: $\exists v_j \in D_3, C_3(v_j) \leq 1$. The final privacy loss of a_3 to a_2 according to these three messages is: $\sum_{i=1}^{5}(P_i \times p_3^i)$.

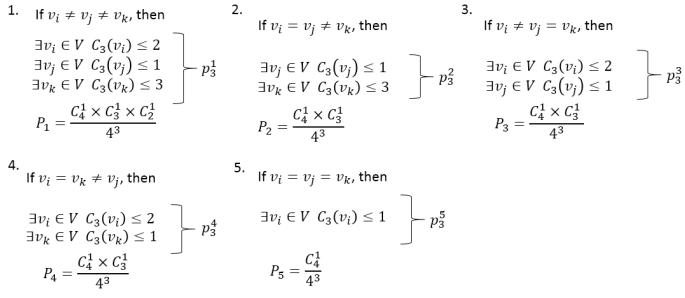

Fig. 1. Generalized Inference Rules for Example 4

By utilizing the computations in Example 4, we can then infer privacy loss for general DCOPs.

V. Experimental Evaluations

In this section, we show some preliminary experimental results. We compare the system privacy loss of SynchBB, ADOPT, BnB-ADOPT, and DPOP on two benchmarks: randomly generated DCOPs and DiMES using three different aggregations: arithmetic mean (Average), maximization (Max), and a risk-based aggregation function (Risk) where it is: a) τ-high-pessimistic when all the privacy loss of agents is higher than τ, b) τ-low-optimistic for all the agents' privacy loss is lower than τ, and c) τ-combined-pessimistic for all the other cases. We use maximization and minimization for modeling τ-high-pessimistic and τ-low-optimistic. For τ-combined-pessimistic, we use average for only the positive privacy losses. We follow the idea of VPS and use the entropy-based metric in measuring the private information leaked from one agent to another. Note that privacy loss equals to 0 means no privacy loss. If the privacy loss is 1, it means the agent losses all its private information to all the other agents. We generate 10 instances for each parameter setting on the two benchmarks and run each instance 5 times. Results are taken average. The main focus of this paper is to compare and study the system privacy loss on different algorithms. At current stage, we assume agents have sufficient computation power to compute these losses and we have not incorporate timing requirements in our study.

Randomly generated DCOPs We randomly generate DCOPs with parameter (b,u,d,p,s), where b gives the set $\{1, \ldots, b\}$ of binary costs, u gives the set $\{1, \ldots, u\}$ of unary costs, d is the maximum domain size, p is the constraint density, and s indicates the shape of the constraint graphs: balance binary tree (T), clique (Q), star (S), and chain (C). All instances in this benchmark have seven agents and each agent owns a variable (i.e. a total of 7 variables). Unary costs and binary costs are generated in uniform random according to the specified range. We set $\tau = 0.01$ for the risk-based aggregation function. Table I shows the result.

When applying the average aggregation function, we observe the privacy loss for the four algorithms decreases when the binary cost range increases. A possible reason is that when there are large (random) binary costs, the costs inferred by the inference rules is mixed with binary costs apart from unary costs. This could prevent the inference rules from inferring the unary cost precisely. Similar trend can also be observed on the maximization aggregation function. Comparing DPOP to the other three algorithms, we observe DPOP has far less privacy loss than the others in all three types of aggregation functions. Note that the risk-based aggregation function will return 0 if no privacy loss is greater than τ.

Distributed Multi-Event Scheduling (DiMES) We randomly generate DiMES with parameter (u, d, s), where u gives the set $\{1, \ldots, u\}$ of unary costs (i.e. range for the timeslot preferences of each attendee), d gives the number of timeslots for each attendee, s is the shape of the constraint graphs: balance binary tree (T), clique (Q), star (S), and chain (C). All instances in this benchmark have seven agents aiming to schedule an event. We set $\tau = 0.083$ for the risk-based aggregation function. Table II shows the result.

We observe the privacy loss of ADOPT, BnB-ADOPT, and DPOP are the same on the star graph across the three aggregations. One possible explanation is that for the pseudo-tree of the star graph, the nodes are all leaves (except the root node) and these three algorithms have similar routines for computing costs being sent from leaves to root. It is worth noting DPOP on clique graphs leaks the least amount of privacy among the four algorithms, and in general, the amount of privacy loss for DPOP is better than the others. One possible reason is that the *UTIL* phase of DPOP works in a synchronous manner on different sub-trees of the pseudo-tree. This fuses the unary costs of different values together, and inferring a precise cost range for individual value is hard.

TABLE I
RANDOMLY GENERATED DCOPS

	Average				Max				Risk			
(b, u, d, p, s)	ADOPT	SynchBB	DPOP	BnB-ADOPT	ADOPT	SynchBB	DPOP	BnB-ADOPT	ADOPT	SynchBB	DPOP	BnB-ADOPT
(2, 6, 3, 0.5, C)	0.047889	0.004768	0.000809	0.030785	0.084570	0.024859	0.006319	0.094631	0.059380	0.016874	0.0	0.053167
(3, 6, 3, 0.5, C)	0.038064	0.003776	0.000219	0.024525	0.072015	0.022540	0.001751	0.066727	0.055382	0.015918	0.0	0.042376
(4, 6, 3, 0.5, C)	0.036724	0.002149	0.000130	0.014416	0.080133	0.012034	0.001042	0.042132	0.059428	0.008903	0.0	0.030026
(5, 6, 3, 0.5, C)	0.031234	0.002074	0.000113	0.013703	0.064639	0.011125	0.000903	0.054023	0.050493	0.007281	0.0	0.038958

TABLE II
DISTRIBUTED MULTI-EVENT SCHEDULING (DIMES)

	Average				Max				Risk			
(u, d, s)	ADOPT	SynchBB	DPOP	BnB-ADOPT	ADOPT	SynchBB	DPOP	BnB-ADOPT	ADOPT	SynchBB	DPOP	BnB-ADOPT
(2, 4, S)	0.070833	0.064286	0.070833	0.070833	0.129167	0.175000	0.129167	0.129167	0.101736	0.126667	0.101736	0.101736
(2, 4, T)	0.047619	0.062500	0.046428	0.069643	0.125	0.191667	0.125	0.133333	0.103125	0.137500	0.103819	0.103542
(2, 4, C)	0.052381	0.093452	0.026786	0.073214	0.137500	0.195833	0.120833	0.141667	0.110902	0.162778	0.110416	0.110000
(2, 4, Q)	0.065476	0.044643	0.002976	0.065476	0.125	0.125	0.020834	0.129167	0.099792	0.082361	0.0	0.101805

On the other hand, ADOPT/BnB-ADOPT are asynchronous. Agent do not necessarily need to wait for (cost) messages from children before sending cost messages. This give chances for us to deduce a tighter cost range. Comparing SynchBB and DPOP which are both synchronous, we observe DPOP performs better than SynchBB in general (except in DiMES on star graph). In SynchBB, cost messages can be sent from higher agents to lower agents (down messages) and also from lower agents to higher agents (up messages). However, DPOP only sends cost messages from lower agents to higher agents. This gives more chances to higher agents in preserving privacy.

VI. CONCLUSION

In this paper, we propose a framework based on VPS to evaluate how well a distributed algorithm preserves the totality of agents' private information by allowing the uses of different aggregators aggregating agents' individual privacy loss. We propose two classes of aggregators: idempotent aggregators and risk based aggregators. Risk based aggregators consider user's attitude towards the privacy loss and allow us to define a wider class of aggregators other than classical approach. We outline the concept of inference rules from VPS which focus on deducing the amount of privacy loss by an agent in DiMES. We further generalize and extend these rules so that they are not only applicable to DiMES, but also capable of deducing privacy loss in general DCOPs. We implement our work on SynchBB, ADOPT, BnB-ADOPT, and DPOP to demonstrate the feasibility of our approach. Preliminary experiments on two benchmarks comparing the four algorithms are performed. In depth investigations on more types of benchmarks and algorithms will be left as future works. Other future directions includes incorporating complex aggregators for real life applications, improving existing algorithms to minimize privacy loss, and giving theoretical comparison between algorithms.

REFERENCES

[1] Tomasa Calvo, Anna Kolesárová, Magda Komorníková, and Radko Mesiar, *Aggregation operators: properties, classes and construction methods*, Aggregation operators, Physica-Verlag GmbH, 2002, pp. 3–104.

[2] M. Detyniecki, *Fundamentals on aggregation operators*, AGOP'01, 2001.

[3] B. Faltings, T. Léauté, and A. Petcu, *Privacy guarantees through distributed constraint satisfaction*, WI-IAT'08, 2008, pp. 350–358.

[4] M.S. Franzin, E.C. Freuder, F. Rossi, and R. Wallace, *Multi-agent constraint systems with preferences: Efficiency, solution quality, and privacy loss*, Computational intelligence **20** (2004), no. 2, 264–286.

[5] E.C. Freuder, M. Minca, and R.J. Wallace, *Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents*, IJCAI DCR Workshop, 2001, pp. 63–72.

[6] R. Greenstadt, J.P. Pearce, and M. Tambe, *Analysis of privacy loss in distributed constraint optimization*, AAAI'06, 2006, pp. 647–653.

[7] T. Léauté and B. Faltings, *Privacy-preserving multi-agent constraint satisfaction*, CSE, vol. 3, 2009, pp. 17–25.

[8] X. Luo and N.R. Jennings, *A spectrum of compromise aggregation operators for multi-attribute decision making*, AI **171** (2007), no. 2, 161–184.

[9] R.T. Maheswaran, J.P. Pearce, E. Bowring, P. Varakantham, and M. Tambe, *Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications*, AAMAS **13** (2006), no. 1, 27–60.

[10] 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*, AAMAS'04, 2004, pp. 310–317.

[11] P.J. Modi, W.M. Shen, M. Tambe, and M. Yokoo, *Adopt: Asynchronous distributed constraint optimization with quality guarantees*, AI **161** (2005), no. 1-2, 149–180.

[12] John Von Neumann and Oskar Morgenstern, *Theory of games and economic behavior*, Princeton University Press, 1944.

[13] A. Petcu and B. Faltings, *A scalable method for multi-agent constraint optimization*, IJCAI'05, 2005, pp. 266–271.

[14] M.C. Silaghi and B. Faltings, *A comparison of distributed constraint satisfaction techniques with respect to privacy*, AAMAS DCR Workshop, 2002.

[15] R.R. Yager, *On prioritized multiple-criteria aggregation*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **42** (2012), no. 5, 1297–1305.

[16] W. Yeoh, A. Felner, and S. Koenig, *BnB-ADOPT: An asynchronous branch-and-bound DCOP alorithm*, JAIR **38** (2010), 85–133.