Privacy Sensitive Environment Re-Decomposition for Junction Tree Agent Organization Construction

Yang Xiang YXIANG@UOGUELPH.CA

Abdulrahman Alshememry AALSHEME@UOGUELPH.CA

University of Guelph, Canada

Abstract

A number of frameworks for decentralized probabilistic reasoning, constraint reasoning, and decision theoretic reasoning assume a junction tree agent organization (JT-org). A natural decomposition of agent environment may not admit a JT-org. Hence, JT-org construction involves three related tasks: 1. Recognize whether a JT-org exists for a given environment decomposition. 2. When JT-orgs exist, construct one. 3. When no JT-org exists, revise the environment decomposition so that one exists and then construct it. Task 3 requires re-decomposition of the environment. However, re-decomposition incurs loss of JT-org linked privacy, including agent privacy, topology privacy, privacy on private variables, and privacy on shared variables. We propose a novel algorithm suite DAER (Distributed Agent Environment Re-decomposition) that accomplishes all three tasks distributively. For Tasks 1 and 2, DAER incurs no loss of JT-org linked privacy. For Task 3, it incurs significantly less privacy loss than existing JT-org construction methods. Performance of DAER is formally analyzed and empirically evaluated.

Keywords: Coordination and control models for multiagent systems, Organisations and institutions, Self-organization, Privacy in MAS

1. Introduction

Decentralized probabilistic reasoning, constraint reasoning, and decision theoretic reasoning are essential tasks of cooperative multiagent systems. Many frameworks exist to perform those tasks in multi-agent systems. Some do not assume specific agent organization, e.g., [6, 15]. Some assume a total order among agents, e.g., [9, 3]. Some are based on a lattice topology, e.g., [11]. Some use a pseudotree, e.g., [12, 13, 5, 7]. Multiply Sectioned Bayesian Networks (MSBNs) [17, 19] are the earliest multiagent systems based on JT organizations (JT-orgs). JT-orgs have now been applied to other frameworks, e.g., distributed constraint optimization [16, 2] and decentralized decision theoretic reasoning [20]. It has been shown [16] that JT-orgs are superior than pseudotrees. More generally, for a number of distinct information processing tasks, JT-orgs are shown to guarantee exact answers [1].

Privacy is an important issue in multiagent systems [8, 22, 10]. For instance, four types of privacy have been identified [8] in distributed constraint reasoning: agent privacy, topology privacy, constraint privacy, and decision privacy. Very few works exist on protecting privacy during JT-org construction. Construction techniques employed by several frameworks that depend on JT-orgs, e.g., [18, 16, 2], compromise privacy on private variables, shared variables, agent identities and adjacency relations, as shown in [21]. The above four types of privacy from distributed constraint reasoning do not adequately capture these privacy losses. First, JT-org construction does not involve constraint privacy and decision privacy. Second, although loss on agent identities correspond to agent privacy, and loss on agent adjacency relations correspond to topology privacy, privacy losses

on private and shared variables are not covered by any of the four types. In this paper, we refer to the overall privacy related to JT-org construction as *JT-org linked privacy*. It includes four specific types of privacy: We refer to privacy on agent identities and adjacency relations as *agent privacy* and *topology privacy*, respectively. We refer to the remaining two types of privacy as *privacy on private variables* and *privacy on shared variables*.

Two fundamentally different approaches on privacy exist. The first transmits private information into the public domain, but makes it unintelligible to unintended receivers by cryptographic techniques, e.g., [22]. Although it may require multiple external servers that are not always available or justifiable for the benefit, recent progresses, e.g., [8, 14], are able to protect privacy without using external servers. The second approach minimizes the amount of private information transmitted, e.g., [10]. It is the approach we take in this work.

A natural decomposition of agent environment may not admit a JT-org. Hence, JT-org construction involves three related tasks:

1. Recognize whether a JT-org exists for a given environment decomposition.
2. When JT-orgs exist, construct one.
3. If no JT-org exists, revise environment decomposition so that one exists and then construct it.

HTBS (HyperTree construction based on Boundary Set) is an algorithm that performs Tasks 1 and 2 [21]. It does so without JT-org linked privacy loss. Hence, when environment admits JT-orgs, HTBS is superior than alternative construction methods, such as those in Action-GDL [16] and DCTE [2], which incur JT-org linked privacy loss. Action-GDL and DCTE do not perform Task 1 explicitly.

When no JT-orgs exist for the given environment decomposition, methods in Action-GDL and DCTE perform Task 3, while incurring privacy loss. HTBS, on the other hand, terminates after Task 1, without constructing a JT-org.

The main contribution of this work is a novel algorithm DAER that integrates and significantly extends HTBS, and accomplishes all three tasks. When agent environment decomposition admits no JT-orgs, DAER performs Task 3 by modifying the decomposition, as Action-GDL and DCTE do, but with considerably less JT-org linked privacy loss (as demonstrated experimentally in Section 6). This advancement significantly improves JT-org linked privacy in multiagent systems that utilize JT agent organizations, such as MSBN [17], Action-GDL [16], and DCTE [2].

Table 1: Summary of distributed JT-org construction methods

Method	Tk 1	Priv Loss	Tk 2	Priv Loss	Tk 3	Priv Loss	Amt Loss
ActGDL	no	n/a	yes	priv, shar	yes	priv, shar	medium
DCTE	no	n/a	yes	priv, shar, id, adj	yes	priv, shar, id, adj	high
HTBS	yes	none	yes	none	no	n/a	n/a
DAER	yes	none	yes	none	yes	shar, id, adj	very low

Table 1 summarizes main properties of three distributed JT-org construction methods in comparison with those of DAER. The *Tk i* column specifies whether Task i is explicitly performed by each method. The *Priv Loss* column specifies whether the method incurs JT-org linked privacy loss, and if so the type of loss: *priv* for private variables, *shar* for shared variables, *id* for agent identity,

and *adj* for agent adjacency relation. The *Amt Loss* column records the relative amount of privacy loss on Task 3. The comparisons covered by the last 3 columns on Task 3 and the last row on DAER are detailed in this paper. Details on the remaining comparisons can be found in [21].

The remainder is organized as follows. Section 2 reviews JT-org of agents and HTBS. Sections 3 through 5 present DAER, its assessment on privacy loss, and analysis of its soundness. The empirical evaluation of DAER is reported in Section 6, followed by conclusion in Section 7.

2. Background

Consider a set $A = \{A_0, ..., A_{\eta-1}\}$ of cooperative and independent agents, whose environment is described by a collection V of variables. The *environment* V is decomposed into a set of overlapping *subenvironments* $\Omega = \{V_0, ..., V_{\eta-1}\}$, where $\cup_{i=0}^{\eta-1} V_i = V$, such that agent A_i controls V_i. A variable that appears in a unique subenvironment V_i is a *private* variable. Otherwise, it is a *shared* variable. If A_i and A_j ($j \neq i$) share variables, their *border* is the set of variables that they share, $I_{ij} = V_i \cap V_j \neq \emptyset$, and the two agents are *adjacent*.

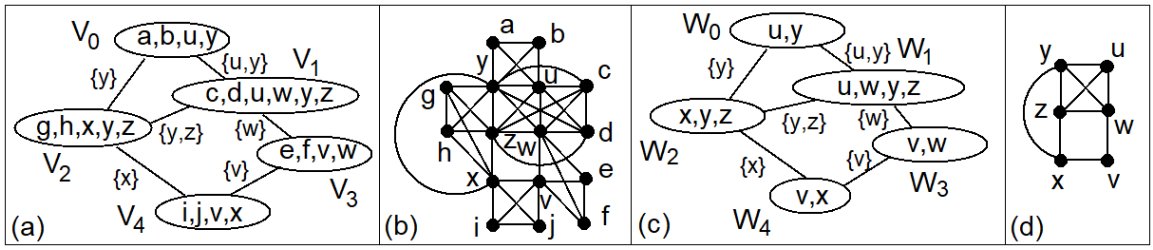

Figure 1: Environment decomposition cluster graph (a), undirected environment decomposition graph (b), communication graph (c), and boundary graph (d)

Environment decomposition Ω can be depicted by an *environment decomposition cluster graph* (Fig. 1 (a)), where each cluster is a subenvironment and each link between two clusters is labeled by their border. Ω can also be depicted by an *undirected environment decomposition graph* (Fig. 1 (b)), obtained from the above cluster graph by mapping each distinct cluster member (a variable) to a node and connecting members of each cluster pairwise. Fig. 1 (a) and (b) illustrate a (trivial) environment

$$\Omega = \{V_0 = \{a, b, u, y\}, V_1 = \{c, d, u, w, y, z\}, V_2 = \{g, h, x, y, z\},$$
$$V_3 = \{e, f, v, w\}, V_4 = \{i, j, v, x\}\}.$$

For distributed probabilistic reasoning, probabilistic knowledge over each V_i can be encoded as a Bayesian subnet, e.g., MSBN [19]. For decentralized constraint optimization, constraints over each V_i can be encoded as a constraint subnet, e.g., Action-GDL [16] and DCTE [2].

The *boundary* of an agent is the union of its borders. The boundary of A_1 in the above example is $W_1 = \{u, w, y, z\}$. The *boundary set* of a multiagent system is the collection of boundaries of its agents. The boundary set of the above example is

$$W = \{W_0 = \{u, y\}, W_1 = \{u, w, y, z\}, W_2 = \{x, y, z\}, W_3 = \{v, w\}, W_4 = \{v, x\}\}.$$

A boundary set can be depicted by a cluster graph, called *communication graph* (CG), where each cluster is a boundary and each link between two clusters is a border (see Fig. 1 (c)). It can also be

depicted by an undirected graph, called *boundary graph* (BG), obtained from CG by mapping each distinct cluster member to a node and connecting members of each cluster pairwise (see Fig. 1 (d)). CG and BG involve shared variables only.

A JT-org is a tree subgraph (including all clusters) of the environment decomposition cluster graph, such that intersection of any two subenvironments is contained in every subenvironment on the path between the two. Fig. 2 (a) shows an environment decomposition cluster graph and (b) is a corresponding JT-org. The intersection of V_0 and V_2 is $\{y\}$ and is contained in V_1. A JT-org can be equivalently expressed as a tree subgraph of the communication graph, as shown in (c). This alternative representation will be applied below as it is more concise (without private variables).

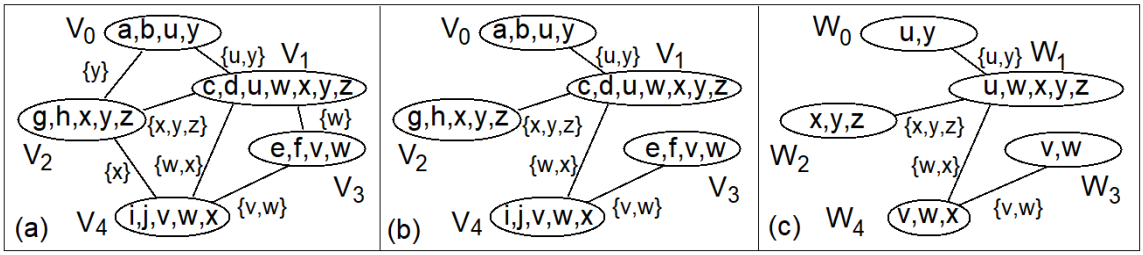

Figure 2: Environment decomposition cluster graph (a), JT-org (b), and JT-org over boundaries (c)

A decomposition may not admit a JT-org. The environment decomposition in Fig. 1 (a) is an example: Its subenvironments cannot be organized into a JT-org. Hence, construction of JT-org involves three related tasks: (1) Recognize whether a JT-org exists for a given environment decomposition. (2) When JT-orgs exist, construct one. (3) When no JT-org exists, revise environment decomposition so that one exists and then construct it.

When agents are developed by independent suppliers, e.g., in an MSBN system for monitoring a fertilizer plant, a natural environment decomposition embeds at least four types of private information relative to each agent:

1. private variables
2. shared variables each shared with a particular subset of adjacent agents
3. agent identity
4. agent adjacency relations

These types of information may be leaked when agents conduct Tasks 1, 2, and 3, which we refer to as *privacy loss*. A private variable may be leaked to other agents, e.g., b of A_0 in Fig. 1 (a) leaked to A_1. A shared variable may be leaked to non-adjacent agents, e.g., u shared by A_0 and A_1 in Fig. 1 (a) leaked to A_2. The identity of an agent and its adjacency relation may be leaked to non-adjacent agents. For instance, the identity of A_0 and the adjacency of A_0 and A_1 may be leaked to A_3. Additional private knowledges exist, such as the structure of Bayesian subnet or constraint subnet within each subenvironment. They are not involved during JT-org construction and are immune to privacy loss once the organization operates due to characteristics of associated inference algorithms (see [21]).

Privacy loss is common in existing multiagent frameworks that depend on JT-org. For instance, JT-org construction in MSBNs performs Tasks 1 and 2. It leaks shared variables, agent identities and adjacent relations to a coordinator agent [18]. JT-org constructions in Action-GDL and DCTE perform Tasks 2 and 3. Action-GDL is based on pseudotree conversion and leaks information on

private and shared variables [16]. DCTE [2] is based on variable propagation and incurs loss on all four types of JT-org linked privacy.

HTBS is a recent algorithm for Tasks 1 and 2 without privacy loss [21]. It is founded on a classification of boundary sets as follows: A *clique* in an undirected graph is a maximal set of nodes that are pairwise connected, e.g., $\{u, w, y, z\}$ in Fig. 1 (d). A clique in a BG is *boundary contained*, if it is a subset of a boundary. A graph is *chordal* if every cycle of length > 3 has a *chord*, which is a link outside the cycle but connecting 2 nodes of the cycle. For instance, $\langle u, w, z, y, u\rangle$ in Fig. 1 (d) is a cycle of length 4. The link $\langle u, z\rangle$ is a chord of the cycle, and so is $\langle y, w\rangle$. The cycle $\langle z, w, v, x, z\rangle$ also has length 4, but it does not have a chord. Due to existence of this chordless cycle, Fig. 1 (d) is not chordal. If we add a link $\langle x, w\rangle$ to Fig. 1 (d), the cycle will have a chord, and the graph will be chordal.

A boundary set W has three possible types, depending on the nature of its BG. W is Type 1, if the BG is chordal and its cliques are boundary contained. W is Type 2, if the BG is not chordal. W is Type 3, if the BG is chordal but some clique is not boundary contained. The BG in Fig. 1 (d) is non-chordal, as shown above. Hence, the above boundary set W is Type 2. An environment decomposition admits a JT-org iff its boundary set is Type 1 [21]. Since W is Type 2, no JT-org exists for the above decomposition Ω.

Soundness and completeness of boundary set typing rest on the relation between undirected graphs and JTs whose clusters are cliques of the graphs. It is well known that a JT exists whose clusters are cliques of an undirected graph, iff the graph is chordal. This relation directly leads to the conclusion drawn on Type 2. Since each cluster in a JT-org must also be a boundary, this leads to the conclusions drawn on Type 3 and Type 1. Formal analysis can be found in [21].

HTBS consists of recursive agent self-eliminations. An agent A_i with boundary W_i can be *self-eliminated relative to* an adjacent agent A_j with W_j, if W_i equals their border, $W_i = I_{ij}$. After A_i is eliminated, W_j is updated by removing any variable that A_j uniquely shared with A_i. In Fig. 1 (c), A_0 can be self-eliminated relative to A_1, and W_1 is reduced to $\{w, y, z\}$ (with u removed). A boundary set is Type 1 iff a single agent remains after all possible self-eliminations (Theorem 2 in [21]). In that case, a JT-org emerges from the trace saved during HTBS. Otherwise, no JT-org exists.

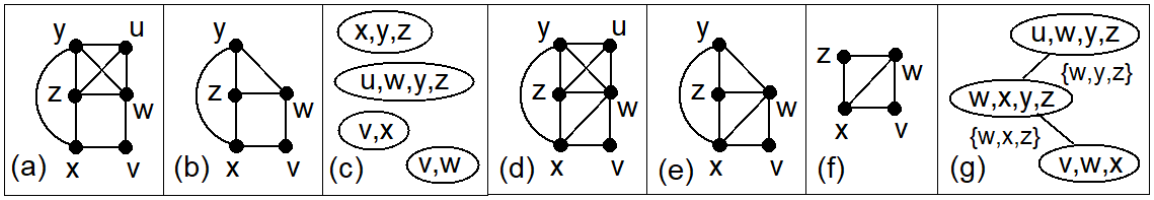

Figure 3: (a) Graph from Fig. 1 (d). (b) After node u is eliminated from (a). (c) Cliques of the graph in (a). (d) A graph from adding one link to (a). (e) After node u is eliminated from (d). (f) After node y is eliminated from (e). (g) The JT from cliques of (d).

Soundness and completeness of HTBS rest on the relation between graphical elimination and existence of JTs. A node x in an undirected graph is *eliminated* if nodes adjacent to x are pairwise connected (with links added if necessary), and then x is removed from the graph. Fig. 3 (b) shows the result of eliminating node u from the graph in (a).

A JT, whose clusters are cliques of an undirected graph, exists iff all nodes of the graph can be eliminated one by one, and no link is added in the process. After u is eliminated from the graph in Fig. 3 (a), no node can be eliminated from the graph in (b) without requiring addition of a link. For instance, to eliminate node v next, link $\langle w, x\rangle$ must be added. Therefore, no JT exists whose clusters are cliques of the graph in (a). These clusters are shown in Fig. 3 (c). No matter how they are connected, the result cannot be a JT. Next, consider the graph in (d), which differs from (a) by the extra link $\langle w, x\rangle$. After eliminating node u, the result is shown in (e). After eliminating y from (e), the result is shown in (f). The remaining nodes can be eliminated in the order (v, z, w, x), and no link is ever added in the entire process. Hence, a JT exists whose clusters are cliques of the graph in (d). The JT is shown in Fig. 3 (g).

More details on the relation between graphical elimination and existence of JTs can be found from [4, 18]. Self-elimination of agents in HTBS extends this relation by grouping the elimination operations within each agent, and catching the agent elimination trace. Note that self-elimination of agents does not enable existence of JT-org. It simply identifies JT-org existence.

3. DAER for Environment Re-decomposition

For a multiagent system that is based on JT-org, if the environment decomposition does not admit a JT-org, it must be revised. Assuming that no variable can be removed, the only option is to share some variables beyond their original scope. This results in privacy loss on these variables. If they are shared with non-adjacent agents, privacy loss occurs on agent identity, as well as on agent adjacency relations, as shown below. The challenge is to minimize such loss.

We present DAER, a novel algorithm suite, that integrates and significantly extends HTBS to accomplish all three tasks of JT-org construction distributively. DAER performs Tasks 1 and 2 without privacy loss. When agent environment decomposition admits no JT-org, DAER re-decomposes the environment and constructs a JT-org, with significantly lower privacy loss than Action-GDL and DCTE (shown in Section 6). It differs form methods in Action-GDL and DCTE as follows: (1) DAER operates on CGs or BGs, rather than environment decomposition cluster graphs or their undirected equivalent. Hence, DAER is free from privacy loss on private variables. (2) DAER is based on a numerical evaluation of privacy loss, so that agent developers can influence privacy loss and trade among different types of privacy loss.

We make the standard assumption that agent communication is reliable such that messages do not get lost, and are received in the order they were sent. We assume that agents are honest, but curious: They follow the intended algorithm/protocol, but are interested in learning private information of other agents from messages received.

3.1 Overview of DAER

Algo. 1 presents DAER at a high abstraction level. Given an agent environment (A, Ω, W), DAER takes (A, W) as input. It runs in multiple rounds. Each round consists of an HTBS stage and an elimination-expansion (EE) stage.

The HTBS stage (lines 2 to 4) runs HTBS, during which agents self-eliminate (becoming inactive), until no such elimination is possible. If a single active agent is left, then (A, Ω, W) admits a JT-org, and one emerges distributively [21], as indicated in line 3. Otherwise, (A, Ω, W) does not admit JT-org, as indicated by line 4. Hence, in the first round, DARE accomplishes Tasks 1 and 2 of JT-org construction through lines 2 to 4.

Algorithm 1 $DAER(A, W)$

1 do
2 hasJT = HTBS(A, W);
3 if hasJT = true, return JT-org;
4 declare "no JT-org";
5 each active agent plans a best neighbor for expansion;
6 select one active agent A_X with boundary X and its active neighbor A_Y with boundary Y;
7 A_X shares $X \setminus Y$ with A_Y, and A_Y expands Y into $Y \cup X$;
8 A_X self-eliminates relative to A_Y, and becomes inactive;

If the first round continues to line 4, then (A, Ω, W) does not admit JT-org, and the EE stage (lines 5 to 8) is run, in which only remaining active agents participate. The EE stage of the first round and the remaining rounds of DAER perform Task 3. Before presenting algorithmic details of the EE stage, we illustrate with the example in Fig. 1.

Suppose that HTBS starts by a leader agent A_0. From Fig. 1 (c), since $W_0 = I_{01}$, A_0 self-eliminates relative to A_1, and A_1 leads subsequent computation. Elimination will be attempted in sequence by remaining agents, but none is able to. Eventually, A_1 declares non-existence of JT-org.

During the EE stage, active agents operate on a reduced CG, without W_0 and incident links (Fig. 4 (a)). First, each agent plans a best neighbor for boundary expansion (line 5). For instance, A_2 has two alternative options. It can share x with A_1, expanding W_1 into $\{w, x, y, z\}$, which allows A_2 to self-eliminate relative to A_1. Alternatively, A_2 can share y and z with A_4, expanding W_4 into $\{v, x, y, z\}$, which allows A_2 to self-eliminate relative to A_4.

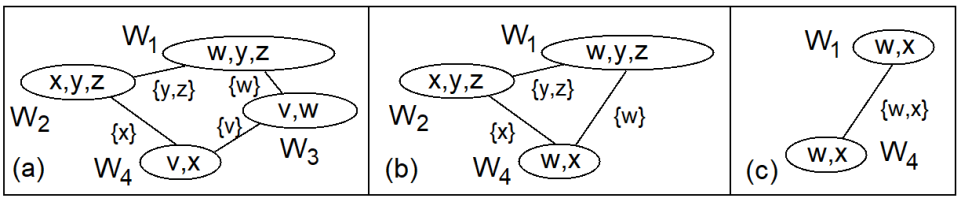

Figure 4: Reduced CGs after elimination of A_0 (a), A_3 (b), and A_2 (c)

The first option has a single newly shared variable, while the second option has two. If privacy loss is measured only by the number of newly shared variables, then the first option is better. We elaborate on measures of privacy loss and the local option evaluation in Section 4. For now, it suffices that each agent locally plans for a best neighbor to expand, assuming arbitrary tie-breaking.

Subsequently, active agents select a best expansion plan globally (line 6) through a distributed depth-first-search (DFS) over the CG in Fig. 4 (a). Since A_1 announces the outcome in the HTBS stage, it acts as the leader and root of DFS. Suppose that agents are activated during DFS in the order (A_1, A_2, A_4, A_3), where the expansion plan of A_3 has the minimum privacy loss L. The search organizes agents into a DFS tree, which is the chain (A_1, A_2, A_4, A_3) in this example. As agents report back to their parents the best loss found, eventually A_1 knows the best loss L. By propagating L down the DFS tree, A_3 knows that its expansion plan has been selected.

Suppose that the winning plan of A_3 is to share w with A_4. This expands W_4 into $\{v, w, x\}$ (line 7), and allows A_3 to self-eliminate (line 8). Fig. 4 (b) shows the new CG. As the result of sharing w, a border is introduced between A_4 and A_1. This terminates the first round of DAER.

In the second round of DAER, HTBS is run, led by A_4. Since no agent can self-eliminate, A_4 eventually declares non-existence of JT-org. During EE, suppose A_2 is selected by DFS. It shares x with A_1 and self-eliminates. After expansion, W_1 becomes $\{w, x\}$.

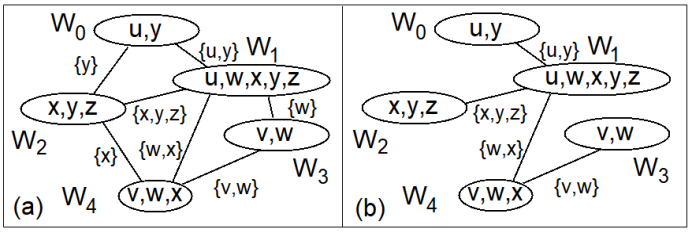

Figure 5: Re-decomposed CG (a) and the JT-org (b)

In the third round of DAER, HTBS is run in the CG of Fig. 4 (c), led by A_1. After A_1 self-eliminates, the single remaining agent A_4 declares "JT-org exists" and DAER terminates (line 3). The re-decomposed CG is shown in Fig. 5 (a), and the JT-org emerging from the trace saved during HTBS stages is in (b).

As illustrated above, the key operations that enable construction of the JT-org is the boundary expansions at selected agents. Their rationality can be intuitively understood as follows: Recall from the end of Section 2 on the relation between graphical elimination and existence of JTs: The elimination process plays the role of identification of existence of JT. If the undirected graph does not have a JT, then any elimination process will add links with certainty no matter in what order the nodes are eliminated. As a result, a JT exists whose clusters are cliques of the graph augmented with these links. General presentation on these graph-theoretic operations can be found in [4, 18]. After presenting relevant algorithms, we establish soundness of DAER formally.

3.2 Leader Agents

Although Algo. 1 appears centrally controlled (for easy comprehension), the computation consists of a sequence of distributed HTBS and EE stages, each initiated by a leader agent. Algo. 2 describes how a leader agent is selected and its role. Since any active agent may be the leader for a specific stage, Algo. 2 is executed by every agent Ag in response to relevant messages or events.

Algorithm 2 $Lead(m, e)$
Input: m is an incoming message; e is an event;

1 if e is external request to start HTBS,
2 initiate HTBS;
3 else if e is declaration of "no JT-org" by Ag,
4 initiate search for best expansion;
5 else if m requests boundary expansion,
6 expand boundary of Ag relative to sender of m;
7 initiate HTBS;
8 else if e is declaration of "JT-org exists" by Ag,
9 initiate halting of DAER;

The first round of DAER starts with an HTBS stage. The HTBS computation assumes an externally specified leader agent [21]. DAER assumes the same. It may be arbitrarily specified and the

choice does not affect soundness of DAER. This agent is the only leader agent externally specified for DAER. The only role of this agent is to lead the first HTBS stage in the first round of DAER, as lines 1 and 2. Leaders of all HTBS stages in subsequent rounds are internally determined during DAER, as shown below.

At end of an HTBS stage, an agent declares "no JT-org" or "JT-org exists". In the original HTBS [21], the agent then starts a halting process. DAER handles these events differently. When HTBS ends with "no JT-org", the announcing agent becomes leader of the next EE stage, as indicated in lines 3 and 4. When HTBS ends with "JT-org exists", the current, expanded boundary set is type 1 (established later). Each agent has a possibly expanded boundary. Using the trace saved, a JT-org is well defined and each agent knows its adjacent agents in the JT-org Lines 8 and 9 encapsulate the corresponding computation.

During EE stage, the best expansion among active agents is determined, and the winning agent A_X (line 6 of Algo. 1) is notified. A_X runs lines 7 and 8 of Algo. 1, and the expanding agent A_Y runs line 7, which ends the EE stage. The next HTBS stage is initialized by A_Y through lines 5 to 7 of Algo. 2. This is how the leader of each HTBS stage, except the first one, is selected.

3.3 Elimination-Expansion

Next, we present distributed algorithms for EE stage, which accomplish lines 6 to 8 of Algo. 1. We mention how computation in line 5 is integrated, but will elaborate later.

At the start of DAER, each agent knows adjacent agents in the current CG and the border with each, and hence its own boundary. For the decomposition in Fig. 1 (c), agent A_1 knows identities of A_0, A_2, and A_3, the borders $\{u, y\}$, $\{y, z\}$, and $\{w\}$, respectively, and its boundary $\{u, w, y, z\}$. During the first HTBS, A_0 is self-eliminated. Hence, at the start of the first EE stage, A_1 maintains *active* adjacent agents A_2 and A_3, and borders with them, as a partial view of the CG in Fig. 4 (a). Note that the active boundary of A_1 is reduced to $\{w, y, z\}$.

Each agent has a procedure $GetBestExpPlan$, which computes line 5 of Algo. 1 locally. We elaborate $GetBestExpPlan$ in Section 4. For now, it suffices to say that $GetBestExpPlan$ returns (among other things) the lowest (best) privacy loss score over its alternative expansion plans.

Below, we present a suite of algorithms for active agents to respond to messages from other active agents. They involve a distributed depth-first-search. Although depth-first-search is well known, we specify key steps unique to our current task. Since privacy loss is the main focus, the presentation makes it transparent what information is distributed between agents and what is not.

We refer to the agent executing the algorithm as A_i and the message sender as A_c. A_i has a flag $visited$ for itself and one for each active adjacent agent, that are initialized to false.

Algo. 3 DFS is activated either by the leader agent of EE stage (see lines 3 and 4 in Algo. 2), or by an active agent in response to a $DFS(inScore)$ message. In the first case, only lines 1 to 5 are run. In the second case, line 1 and lines 6 to 14 are run. Through DFS, A_i performs local bookkeeping during depth-first-search. Once $bestScore$ is initialized in line 1, A_i maintains and updates this value. During depth-first-search, agents are organized into a search tree, which is referred to as DFS-tree below.

Algorithm 3 DFS

1 run GetBestExpPlan to get bestScore;
2 if A_i is leader of the EE stage,
3 label self as root of DFS-tree; visited = true;
4 run ForwardDFS(bestScore);
5 return;
6 retrieve inScore from message;
7 label A_c as visited;
8 if A_i is visited,
9 label A_c as non-adjacent on DFS-tree;
10 send NonChildReport to A_c;
11 else
12 label A_c as parent on DFS-tree;
13 bestScore = min(bestScore, inScore);
14 run ForwardDFS(bestScore);

DFS calls Algo. 4 $ForwardDFS$ to extend depth-first-search to other agents.

Algorithm 4 $ForwardDFS(bestScore)$

1 select an unvisited active adjacent agent A_k;
2 if A_k exists,
3 send DFS(bestScore) message to A_k;
4 label A_k as visited;
5 else
6 if A_i is not the root of DFS-tree,
7 send ChildReport(bestScore) to A_c;
8 else run Notify(bestScore);

During depth-first-search, A_i runs Algo. 5 to respond to $NonChildReport$.

Algorithm 5 $RespondNonChildReport$

1 label A_c as non-adjacent on DFS-tree;
2 run ForwardDFS(bestScore);

During $ForwardDFS$, A_i runs Algo. 6 to respond to $ChildReport$.

Algorithm 6 $RespondChildReport$

1 label A_c as a child on DFS-tree;
2 retrieve inScore from message and associate it with A_c;
3 bestScore = min(bestScore, inScore);
4 run ForwardDFS(bestScore);

Executions of Algos. 3 through 6 end with the root agent of DFS-tree knowing the best privacy loss score among all agents (line 3 of Algo. 6). It initiates a forward message propagation along the DFS tree to notify the winning agent (line 8 of Algo. 4). The root agent does so by running Algo. 7. Any other agent that executes Algo. 7 is activated by message from its DFS-tree parent (see line 5).

Algorithm 7 $Notify(winningScore)$

1 if A_i's bestScore = winningScore,
2 run ShareVariable;
3 else
4 select a DFS-tree child agent A_k such that score associated with A_k = winningScore;
5 message A_k to run $Notify(winningScore)$;

It may appear uncertain whether agent A_k in line 4 exists. That is guaranteed for the reason below: The $winningScore$ either originates from the root agent of the DFS-tree or a non-root agent. If it originates from the root, since $Notify(winningScore)$ is initiated by the root in line 8 of Algo. 4, the test in line 1 of Algo. 7 will succeed and line 4 will not be run. Therefore, Line 4 is only run when $winningScore$ originates from a non-root agent.

Since $winningScore$ originates from a non-root, it must be associated with a DFS-tree child of the root through $ChildReport$. That child agent is identified as A_k when the root runs Algo. 7 and executes line 4. Hence, success of finding A_k in line 4 is certain when the root runs line 4.

The same argument can be applied recursively to the sub-DFS-tree rooted at A_k. Therefore, success of finding A_k in line 4 is certain no matter A_i is a child of the root or is an agent further downstream on the DFS-tree. Hence, line 5 is well defined. If multiple expansion plans have the same $winningScore$, the ties are broken arbitrarily.

Propagation of $Notify$ operation eventually reaches the agent whose expansion plan generated $winningScore$. Note that throughout execution of Algos. 3 through 7, no expansion plan of any agent, including the expansion plan of the winning score, is included in any message among agents. Only scores of expansion plans are included messages. As will be elaborated in Section 4, the score of an expansion plan is a sum of multiple privacy weights, each of which is in $[0, 1]$. A given sum value can be made of a few weights close to 1, or many weights close to 0, giving no accurate indication of the number of weights (and the scale of the plan).

We now continue with realization of the winning plan. By line 2 of Algo. 7, the winning agent A_i runs Algo. 8. We refer to the agent to expand according to the winning plan as A_k.

Algorithm 8 $ShareVariable$

1 compute $Q = W_i \setminus I_{ik}$;
2 let S_Q be the set of agents sharing some variable in Q with A_i;
3 send message $Expand(Q, S_Q)$ to A_k;
4 A_i self-eliminates relative to A_k;

Algo. 8 is the primary source of privacy loss. The set Q of variables is shared with a new agent A_k. The identity of each agent in S_Q, if not already adjacent to A_k, is revealed to A_k (line 3). Consider the CG in Fig. 4 (a) reproduced in Fig. 6 (a). Suppose A_3 runs Algo. 8 with $A_k = A_4$.

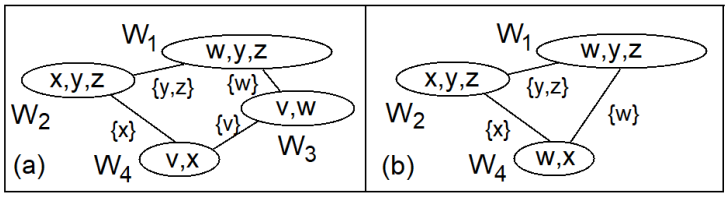

Figure 6: Illustration of Algo. 8 and Algo. 9, where Fig. 4 (a) and (b) are reproduced

Then $Q = \{v, w\} \setminus \{v\} = \{w\}$ and $S_Q = \{A_1\}$. When A_3 sends message $Expand(Q, S_Q)$ to A_4, variable w is disclosed to A_4, and so is the identity of A_1.

In response to the Expand message from Algo. 8, A_k (which we now refer to by A_i according to our convention) will perform Algo. 9. Lines 1 through 3 compute the set T of variables that A_i shared uniquely with A_c. Line 5 adds variables received from A_c to the boundary of A_i, and removes those in T since A_c is now self-eliminated (line 4 of Algo. 8).

Algorithm 9 $RespondExpand$

1 denote the set of adjacent agents of A_i other than A_c by S;
2 initialize a set T to be the border between A_i and A_c;
3 for each agent $A_j \in S$, $T = T \setminus I_{ij}$;
4 retrieve Q and S_Q from Expand message;
5 $W_i = (W_i \cup Q) \setminus T$;
6 become adjacent to each agent in S_Q if not already so;
7 $S = S \cup S_Q$;

Continue with the above example in Fig. 6 (a), where A_4 runs Algo. 9. We have $S = \{A_2\}$ (line 1) and T is initialized to $T = \{v\}$ (line 2). T remains the same after line 3. At line 5, $W_4 = (\{v, x\} \cup \{w\}) \setminus \{v\} = \{w, x\}$. After line 6, A_4 and A_1 become adjacent (Fig. 6 (b)). At line 7, $S = \{A_1, A_2\}$.

Note that through line 6, the identity of A_i is disclosed to agents in S_Q. In the example, the identity of A_4 is disclosed to A_1. We will elaborate on additional loss in Section 4. The execution of Algo. 9 ends an EE stage.

4. Evaluating Privacy Loss

We present how DAER agents evaluate privacy losses of their expansion plans. A new scheme of measure for privacy loss using privacy weights is developed in Section 4.1. How agents evaluate expansion plans using the measure is presented in Section 4.2. In Section 4.3, we present a protocol for regulating privacy weight assignment by agents, to prevent unfair game-theoretic behavior.

4.1 Measure of Privacy Loss

We have assumed that each agent can select an expansion plan by a method $GetBestExpPlan$, which depends on a measure of privacy loss for expansion plans. As stated in Section 2, JT-org construction may suffer from privacy loss on private and shared variables, on agent identities, and on agent adjacency relations. Few measures exist to evaluate such privacy loss. The measure in [10]

assumes that agent identities and their state spaces are publicly known, and hence is not applicable. Four types of privacy have been identified [8] in distributed constraint reasoning: agent privacy, topology privacy, constraint privacy, and decision privacy. They do not adequately capture JT-org linked privacy: JT-org construction does not involve constraint privacy and decision privacy, while privacy losses on private and shared variables during JT-org construction are not covered by the four types. The measure in [21] assumes that leak of each piece of private information contributes one unit to the total loss. The scheme does not admit that some information is more sensitive than others. It is suited for loss evaluation at the system level, but does not support evaluation at the agent level. We develop a new measure of privacy loss such that (1) it allows agents to trade off disclosure of private information of different sensitivity, and (2) privacy loss of each boundary expansion can be evaluated locally by the relevant agent.

In particular, we assume that each private variable x has a privacy weight $\omega_x \in [0, 1]$, assigned by agent in charge of x. Each shared variable x has a weight $\omega_x \in [0, 1]$, agreed by sharing agents. Each agent A has a weight $\omega_A \in [0, 1]$ for its identity. Value of ω_A is assigned by A and known to each adjacent agent. Each pair of adjacent agents A and B assign a weight $\omega_{AB} \in [0, 1]$ to their adjacency relation. A higher weight signifies that the information is more sensitive, and that the agent is less willing to disclose it. For instance, let weights of shared variables x and y of an agent be $\omega_x > \omega_y$. If one variable must be disclosed beyond original adjacent agents in order to construct JT-org, the agent rather discloses y than x.

There are multiple ways by which the above weights can be determined. Analysis of relative merits of such protocols is beyond the scope of this work, and is left for future research, except an example protocol is presented in Section 4.3. DAER assumes that these weights are part of the given environment decomposition, and DAER aims to produce a re-decomposition (if necessary) that minimizes privacy loss determined by these weights. Below, we consider how an agent evaluates expansion plans using the weights.

4.2 Evaluation of Privacy Loss

[**Variable based loss**] When an agent leaks a variable x to another agent, it incurs ω_x units of privacy loss. Let S be a set of variables, where each $x \in S$ is shared between agent A and some adjacent agent. Let B be an agent that does not share any variable in S with A. When A discloses S to B, the variable (var) based loss is $Loss_{var} = \sum_{x \in S} \omega_x$.

Note that $Loss_{var}$ applies to both private variables and shared variables. Since DAER work on CG, it only suffers loss on shared variables. In our experimental comparison (Section 6) with alternative JT-org construction methods, $Loss_{var}$ is calculated for both types of variables as appropriate.

[**Loss of agent privacy**] When a set S of variables is shared with a new agent, it also incurs loss of agent privacy. This was illustrated by the example of Fig. 6 (a), where agent A_3 is adjacent to A_1 and A_4. A_3 shares the variable w with A_1, but not A_4. When A_3 newly shares w with A_4, the identity of A_1 is disclosed to A_4 and vice versa.

In general, suppose agent A is adjacent to B and C. A shares variable x with B, but not C. When A newly shares x with C, potential identity loss arises. If B already shares variables with C, no identity loss occurs. Otherwise, identity of B is leaked to C, and identity of C is leaked to B, since B and C are now adjacent. No identity loss is incurred relative to agent A. Since the identity loss depends on whether B and C already share variables, we denote their relation by a binary function $ncv(B, C)$ (no common variable).

Definition 1 *If agents B and C have a common variable, then $ncv(B, C) = 0$. Otherwise, $ncv(B, C) = 1$.*

The following proposition characterizes the condition of identity loss.

Proposition 1 *Agent A is adjacent to B and C, and shares variable x with B, but not C. When A newly shares x with C, identity loss occurs between B and C, iff $ncv(B, C) = 1$.*

Proof: If $ncv(B, C) = 0$, by Def. 1, B and C have common variables and know the identity of each other before the sharing of x with C. Hence, no identify loss is incurred due to the sharing. If $ncv(B, C) = 1$, B and C are not adjacent before the sharing of x with C, but will be so after the sharing. Hence, identity loss occurs. □

Since agent A needs to evaluate the loss due to sharing x with C, we describe the knowledge state of A through an extended ncv function.

Definition 2 *If agents A, B, and C have a common variable, then $ncv(A, B, C) = 0$. Otherwise $ncv(A, B, C) = 1$.*

From Defs. 1 and 2, the lemma below follows:

Lemma 1 *For agents A, B, and C, $ncv(B, C) = 1$ implies $ncv(A, B, C) = 1$.*

Although the knowledge of A is $ncv(A, B, C)$, it sometimes allows A to evaluate identity loss with certainty.

Proposition 2 *Agent A is adjacent to B and C, shares x with B, but not C, and $ncv(A, B, C) = 0$. If A newly shares x with C, it incurs no identity loss between B and C.*

Proof: From the contrapositive form of Lemma 1, $ncv(A, B, C) = 0$ implies $ncv(B, C) = 0$. By Prop. 1, the conclusion follows. □

The situation in Prop. 2 occurs when A shares a variable $y \neq x$ with both B and C. By Prop. 2, A knows that sharing x with C will not cause identity loss.

Unfortunately, the knowledge of $ncv(A, B, C)$ does not always allow agent A to determine identity loss due to sharing x with C. When A has no common variable with B and C, i.e., $ncv(A, B, C) = 1$, B and C may or may not share variables. That is, from $ncv(A, B, C) = 1$, A cannot infer $ncv(B, C)$, nor whether sharing x with C causes identity loss.

The above analysis leads to the following general evaluation of identity loss when A newly shares a set S of variables with an adjacent agent C.

Theorem 1 *Let A and C be adjacent agents, and S be a set of variables contained in A but not in C. Let A_S be any agent that contains some variable in S (hence is adjacent to A) and satisfies $ncv(A, A_S, C) = 1$. If A newly shares S with C, the agent privacy loss incurred is*

$$Loss_{id} = \sum_{A_S} ncv(A_S, C)\ (\omega_{A_S} + \omega_C).$$

Proof: Since the loss is additive over each A_S, it suffices to consider one A_S. If $ncv(A, A_S, C) = 0$, by Prop. 2, there is no identity loss between A_S and C.

When $ncv(A, A_S, C) = 1$, the loss depends on $ncv(A_S, C)$. By Prop. 1, if $ncv(A_S, C) = 0$, there is no loss. If $ncv(A_S, C) = 1$, the loss is $\omega_{A_S} + \omega_C$. $\square$

Since the value of $ncv(A_S, C)$ is unknown to A, so is that of $Loss_{id}$. Therefore, we let each agent evaluate identity based loss according to the upper bound of $Loss_{id}$

$$MaxLoss_{id} = \sum_{A_S}(\omega_{A_S} + \omega_C) \geq Loss_{id},$$

and minimize the upper bound.

[**Loss of topology privacy**] As analyzed above, explicit disclosure of private information over shared variables and agent identities occurs with Algos. 8 and 9. Implicit disclosure of agent adjacency relations occurs as well. In the example of Fig. 6 (a), agent A_3 is adjacent to A_1 and A_4. A_3 shares variable w with A_1, but not A_4. A_3, A_1, and A_4 do not share common variables. Hence, adjacency A_3-A_1 is unknown to A_4, and adjacency A_3-A_4 is unknown to A_1. When A_3 newly shares w with A_4, A_4 can infer A_3-A_1 adjacency, and A_1 can infer A_3-A_4 adjacency.

In general, suppose agent A is adjacent to B and C. A shares variable x with B, but not C. Furthermore, A, B, and C have no common variables, i.e., $ncv(A, B, C) = 1$. Hence, C does not know that A and B are adjacent, and B does not know that A and C are adjacent either. When A shares x with C, A informs C to become adjacent with B (Algo. 9). This allows C to infer the A-B adjacency relation (privacy loss). Furthermore, by being adjacent with C due to sharing x and observing A being self-eliminated, B can infer A-C adjacency. The loss does not depend on whether B and C already share variables. That is, the value of $ncv(B, C)$ is irrelevant.

Theorem 2 *Let A and C be adjacent agents, and S be a set of variables contained in A but not in C. Let A_S be any agent that contains some variable in S (hence is adjacent to A) and satisfies $ncv(A, A_S, C) = 1$. If A newly shares S with C, the loss of topology privacy incurred is*

$$Loss_{br} = \sum_{A_S}(\omega_{A,A_S} + \omega_{A,C}),$$

where br refers to border.

Proof: The loss is additive over each A_S. Hence, it suffices to consider one A_S. We analyze privacy loss by value of $ncv(A, A_S, C)$.

If $ncv(A, A_S, C) = 0$, adjacency between any pair among A, A_S and C is a prior knowledge of the third. Hence, the sharing of S does not incur loss of topology privacy.

If $ncv(A, A_S, C) = 1$, adjacency A-A_S was unknown to C but is known to C due to the sharing. The privacy loss incurred is ω_{A,A_S}. Similarly, the privacy loss incurred by disclosing adjacency A-C to A_S is $\omega_{A,C}$. The result of $Loss_{br}$ follows. $\square$

In summary, when agent A proposes to newly share a set S of variables with an adjacent agent C, the expansion plan is evaluated in procedure $GetBestExpPlan$. The evaluation computes the following upper bound of the privacy loss

$$MaxLoss = Loss_{var} + MaxLoss_{id} + Loss_{br}.$$

For example, suppose that A_2 in Fig. 6 (a) evaluates the expansion plan of sharing $S = \{y, z\}$ with A_4. Let the relevant privacy weights be the following:

$$\omega_x = 0.806, \omega_y = 0.622, \omega_z = 0.875, \omega_{A_0} = 0.723, \omega_{A_1} = 0.237, \omega_{A_2} = 0.991, \omega_{A_4} = 0.253,$$

$$\omega_{A_0,A_2} = 0.716, \omega_{A_1,A_2} = 0.162, \omega_{A_2,A_4} = 0.682.$$

We have included weights relevant to A_0 since it is adjacent to A_2 in the CG in Fig. 1 (c), even though A_0 has been eliminated in an earlier HTBS stage. The privacy loss is evaluated as follows:

$$Loss_{var} = 0.622 + 0.875, \quad MaxLoss_{id} = (0.723 + 0.253) + (0.237 + 0.253),$$

$$Loss_{br} = (0.716 + 0.682) + (0.162 + 0.682), \quad MaxLoss = 5.207.$$

A DAER execution consists of $K \geq 1$ rounds. If the environment decomposition Ω has a JT-org, then $K = 1$ and the first round consists of the HTBS stage only. The total privacy loss is 0. If Ω does not admit a JT-org, the first HTBS stage will conclude accordingly, an EE stage will follow, and $K > 1$. After $K - 1$ rounds with alternating HTBS and EE stages, the Kth round consists of the HTBS stage only, that has no privacy loss. The HTBS stages do not have privacy loss. Hence, the total privacy loss is accumulated from the first $K - 1$ EE stages. Since loss of each expansion is independent of other expansions, the total loss is the sum of losses from the $K - 1$ EE stages.

4.3 Possible Protocol for Assigning Privacy Weights

We have measured each piece of private information with a weight in $[0, 1]$. This allows an agent to trade disclosure of information of different sensitivities. We illustrate this function in Section 6.1. On the other hand, since DAER greedily minimizes the privacy loss of the agent society, individual agents can display game-theoretic behavior in assigning privacy weights: By assigning the largest possible weights for pieces of its own privacy information, an agent can avoid disclosing them during JT-org construction, at the price of increased disclosure of private information by other agents. Therefore, a protocol that regulates privacy weight assignment is desirable. Thorough investigation of viable protocols and analysis of their properties are beyond the scope of this work, and are an important direction of future work. As DAER functions with any privacy weights, unless reasonable weight assignment protocols are impossible, this issue does not undermine the soundness and generality of DAER. Below, we suggest a simple protocol of the kind, and show that meaningful weight assignment protocols do exist.

We believe that a reasonable privacy weight assignment protocol should have the following (minimum) properties: (1) It should prevent an agent from assigning maximum weights arbitrarily for each piece of private information. (2) It should not cause disclosure of private information. (3) Whether an agent followed the protocol is auditable in principle.

We consider a protocol which stipulates a constant $\mu \in (0, 1)$, e.g., $\mu = 0.5$, and requires each agent to assign weights with their average being equal to μ. Denote the number of privacy weights that an agent assigns by M. For instance, there is one weight for its agent identity, one weight for each agent adjacency, and so on. The sum of all weights assigned by an agent must be $\mu\, M$.

During DAER execution, a pair of agents can become adjacent and a new weight is needed. The protocol requires the new weight to be assigned a value μ. This ensures that the new weight maintains the $\mu\,(M + 1)$ average. The value stipulates a moderate sensitivity: It is preferable not to be disclosed, but can be if necessary.

Next, we consider how well the protocol addresses the 3 properties: Since the maximum sum of M weights is M, while the agent can only assign weights that sum to $\mu\ M < M$, the property (1) is satisfied. No additional information is disclosed by each agent, and hence the property (2) is satisfied. The weights are propagated in aggregated form during DAER execution. Hence, whether an agent assigned its privacy weight according to the protocol is auditable in principle: the property (3) is satisfied.

We view any weight assignment protocol as an associated component of the computational framework in which DAER functions, and not a component of DAER. Indeed, even if there is no regulation on the game-theoretic behavior such that agents assign weights close to 1 arbitrarily, DAER will still be able to minimize the overall privacy loss as measured by such weights.

The above protocol serves as a proof of existence of meaningful weight assignment protocols. It may need to be refined, and other alternatives are possible. We leave thorough investigation and analysis of alternative protocols to future work.

5. Soundness and Complexity

Assuming that the initial environment decomposition Ω does not admit a JT-org, Theorem 3 below shows that upon termination of DAER, the final expanded environment decomposition Ω' has a JT-org, while the privacy loss is greedily minimized.

Theorem 3 *Let Ω be an environment decomposition that does not admit a JT-org and DAER be run on Ω to completion. Let Ω' be the environment decomposition obtained by enlarging those boundaries of Ω as expanded by DAER. Then*

1. *no privacy loss is incurred at the HTBS stage of each round,*
2. *privacy loss at the EE stage of each round is upper bounded by that of the winning plan,*
3. *the upper bound is optimal for the round, and*
4. *Ω' has a JT-org.*

Proof: Condition 1 follows from the privacy loss analysis of HTBS in [21] (Section 8.2).

Condition 2 follows from Theorem 1, Theorem 2, the $MaxLoss$ used by $GetBestExpPlan$, and the depth-first-search employed in the EE stage. Condition 3 is also derived from the depth-first-search.

It may be argued that satisfaction of Condition 4 alone is insignificant. For instance, sharing all variables among all agents will do so trivially. The significance of Condition 4 here is its coexistence with Conditions 1 through 3. We prove Condition 4 below in such a context.

Since Ω does not admit a JT-org, DAER runs to completion in $K > 1$ rounds. In the first HTBS stage, zero or more agents are self-eliminated, and more than one agent remain at the end of the HTBS stage. By Algo. 1, the round 1 includes an EE stage.

During the EE stage, one agent A_X with boundary X is selected to expand the boundary Y of an adjacent agent A_Y. As the result, A_X is self-eliminated. Counting from both stages, a total of one or more agents (boundaries) are eliminated in round 1.

After round 1, boundaries of all agents remain the same, except that boundary Y is expanded into $Y \cup X$. Denote the updated environment decomposition by Ω_2^+. On the other hand, round 2

of DAER runs effectively on an environment decomposition which we denote by Ω_2^-. Ω_2^- does not include agents in Ω that were eliminated during round 1. The active boundary for A_Y is $Y \cup X$ without variables that were uniquely shared with A_X. Hence, Ω_2^+ and Ω_2^- characterize the initial context of round 2.

Using the same notation, the initial context of round 1 is characterized by Ω_1^+ and Ω_1^-, where $\Omega_1^+ = \Omega_1^- = \Omega$.

Each subsequent round $i \geq 2$ starts with Ω_i^+ and Ω_i^-. If Ω_i^- does not admit a JT-org, the HTBS stage in round i is followed by the EE stage, one or more agents are eliminated in round i, and DAER continues into round $i+1$. Otherwise, the HTBS stage in round i terminates DAER with the resultant environment decomposition $\Omega' = \Omega_i^+$.

Since Ω consists of a finite number (n) of boundaries, and at least one boundary is eliminated in each round, DAER terminates in $K < n$ rounds. In round K, the HTBS stage ends with a single agent left uneliminated.

Let $\rho = (A'_1, A'_2, ..., A'_{n-1})$ be the order in which the $n-1$ agents are eliminated during DAER. The order starts with agents eliminated during HTBS stage in round 1, followed by the agent eliminated during EE stage in round 1, followed by agents eliminated during HTBS stage in round 2, and so on. We show that starting with $\Omega' = \Omega_K^+$, n-1 agents can be self-eliminated in the order ρ, without any boundary expansion in between.

Suppose $A'_1, ..., A'_i$ were eliminated during the HTBS stage in round 1 of DAER. Since their boundaries are never modified, they remain the same in Ω'. For each boundary in $\Omega = \Omega_1^-$, its counter part in Ω' may have been expanded but has never been reduced. Hence, each of $A'_1, A'_2, ..., A'_i$ can be self-eliminated relative to the same corresponding agent as during round 1 of DAER.

The next agent in the order is A'_{i+1}, and it was eliminated during the EE stage of round 1. Suppose A'_{i+1} was eliminated relative to A_k. Since the boundary of A_k in Ω' has been expanded during round 1 into a superset of the boundary of A'_{i+1}, A'_{i+1} can still be eliminated relative to A_k.

Since each boundary ceases to change after its elimination during DAER and any boundary can only expand but never shrink, each remaining agent in ρ can be self-eliminated from Ω'. After their elimination, a single active agent is left, and hence Ω' is type 1. By Theorem 2 (briefly introduced at the end of Section 2) of [21], it follows that Ω' has a JT-org. $\square$

By Theorem 3, DAER re-decomposes the environment to enable construction of a JT-org. A JT-org can be specified distributively from the trace saved during HTBS ([21]), which can be directly extended to DAER. This is illustrated in Fig. 5 (b).

Theorem 4 establishes that DAER accomplishes all three tasks of JT-org construction., where each resultant condition corresponds to one such task.

Theorem 4 *Let Ω be an environment decomposition and DAER be run on Ω to completion. Then the following holds:*

1. *No matter whether Ω admits a JT-org, DAER will recognize so explicitly.*
2. *If Ω admits a JT-org, DAER will construct one without privacy loss.*
3. *Otherwise, DAER will construct one while greedily minimizing the privacy loss.*

Proof: Assume that DAER is run on Ω to completion. By the analysis immediately following Algo. 1, the conditions 1 and 2 hold. The condition 3 follows from Theorem 3. $\square$

Let η be the initial number of boundaries and e be the initial number of borders (we omit newly created borders during DAER as they are not substantial). Each round consists of an HTBS stage and an EE stage, each of which involves a depth-first-search over the currently active CG. In each depth-first-search, $O(e)$ messages are passed. Each round eliminates at least one boundary and a single boundary is active at the end of DAER. Hence, DAER concludes in $O(\eta)$ rounds. Its overall time complexity is $O(e\,\eta)$.

6. Experimental Evaluation

We report empirical evidence on the effectiveness of DAER. First, we provide a case study to demonstrate how the measure of privacy loss employed by DAER enables tradeoff of disclosure of private information of different sensitivity. It also connects components of DAER presented separately earlier into a coherent whole. Secondly, we compare performance of DAER with alternative distributed JT-org construction methods on their relative privacy loss.

6.1 Trading Privacy Disclosure

Suppose the CG of environment decomposition is that of Fig. 1 (c), and the privacy weights are assigned according to the protocol in Section 4.3 with the value $\mu = 0.5$. Since DAER is free from privacy loss on private variables, the protocol requires weights of the remaining three types of JT-org linked privacy to sum to μM. Privacy weights for agent identity and adjacency are shown in Table 2, and privacy weights on variable identity and domain for shared variables are shown in Table 3.

Table 2: Privacy weights for agent identity and adjacency (underlined are created during DAER)

Agt i	ω_{A_i}	Agt j	A_1	A_2	A_3	A_4
A_0	0.50	ω_{A_i,A_j}	0.34	0.62		
A_1	0.56			0.36	0.56	
A_2	0.01				<u>0.50</u>	0.33
A_3	0.99					0.94
A_4	0.41					

In our experimental implementation, we have differentiated privacy on a variable into that about its identity and that about its domain (see Table 3). This differentiation allows accommodation of JT-org construction methods, e.g., DCTE, where disclosure of identity for a variable does not necessitates disclosure of its domain (see Section 9.2 in [21]).

Table 3: Privacy weights on variable identity and domain for shared variables

Variable	u	v	w	x	y	z
Identity	0.32	0.18	0.16	0.78	0.59	0.83
Domain	0.99	0.19	0.48	0.67	0.14	0.67
Weight	1.31	0.37	0.64	1.45	0.73	1.50

For each agent with M weights, their sum is $\mu\ M$, which regulates the game-theoretic behavior of the agent. For instance, weights of A_0 sum to

$$\omega_{A_0} + \omega_{A_0,A_1} + \omega_{A_0,A_2} + \omega_u^{Id} + \omega_y^{Id} + \omega_u^{Dom} + \omega_y^{Dom} = 3.5 = 7\mu.$$

In round 1 of DAER, A_0 is self-eliminated relative to A_1 during HTBS stage, and the reduced CG is that of Fig. 4 (a). In the EE stage, active agents evaluate expansion plans locally. A_1 has two expansion plans. The first expands A_2 with $\{w\}$, which incurs privacy loss

$$\omega_w + \omega_{A_2} + \omega_{A_3} + \omega_{A_1,A_2} + \omega_{A_1,A_3} = 2.56.$$

The second plan expands A_3 with $\{y, z\}$, which incurs loss

$$\omega_y + \omega_z + \omega_{A_0} + \omega_{A_3} + \omega_{A_1,A_0} + \omega_{A_1,A_3} + \omega_{A_2} + \omega_{A_3} + \omega_{A_1,A_2} + \omega_{A_1,A_3} = 6.54.$$

Note that although A_0 has been eliminated, since A_0 contains y, when y is newly shared with A_3, the privacy loss relative to A_0 must be counted. In comparison, the first plan is best with privacy loss 2.56.

A_2 has two expansion plans: expanding A_1 with $\{x\}$ yielding loss 3.11, and expanding A_4 with $\{y, z\}$ yielding loss 5.75. The first plan is best with privacy loss 3.11. A_3 has two expansion plans: expanding A_1 with $\{v\}$ yielding loss 2.84, and expanding A_4 with $\{w\}$ yielding loss 3.11. The first plan is best with privacy loss 2.84. Finally, A_4 has two expansion plans: expanding A_2 with $\{v\}$ yielding loss 2.64, and expanding A_3 with $\{x\}$ yielding loss 3.72. The first plan is best with privacy loss 2.64.

Subsequently, through the distributed depth-first-search, the globally optimal expansion plan is selected as the first plan of A_1: expanding A_2 with $\{w\}$. It leads to the elimination of A_1, which reduces the CG to that of Fig. 7 (a). Since A_2 and A_3 become adjacent, a new weight for their adjacency is created (see Table 2).

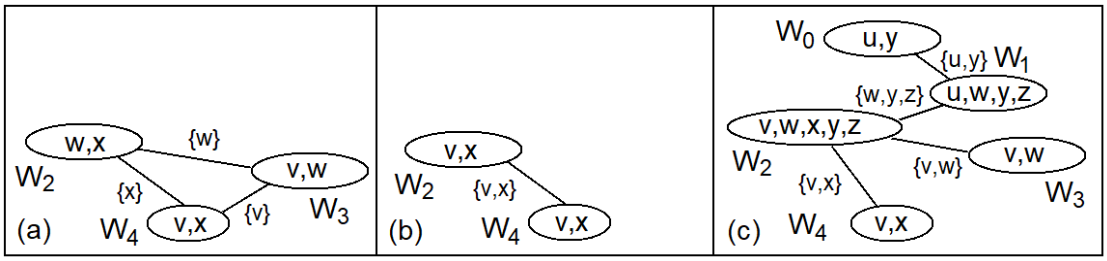

Figure 7: Reduced CG after round 1 (a), reduced CG after round 2 (b), and resultant JT-org (c)

The round 2 of DAER follows. No agent can be self-eliminated during the HTBS stage. In the EE stage, active agents evaluate expansion plans locally. A_2 has two plans, and the best plan is expanding A_3 with $\{x\}$ yielding loss

$$\omega_x + \omega_{A_3} + \omega_{A_4} + \omega_{A_2,A_3} + \omega_{A_2,A_4} = 3.68.$$

Note that since A_3 and A_4 are adjacent, the privacy loss measured by ω_{A_3} and ω_{A_4} does not really occur. However, the adjacency is unknown to A_2 and hence the upper found of 3.68. A_3 has two

plans, and the best plan is expanding A_2 with $\{v\}$ yielding loss 2.23. A_4 has two plans, and the best plan is expanding A_2 with $\{v\}$ yielding loss 2.64.

The distributed depth-first-search selects the plan by A_3 as globally optimal. It leads to the elimination of A_3, which reduces the CG to that of Fig. 7 (b).

In round 3 of DAER, the HTBS stage eliminates one of the remaining agents, and terminates DAER with the JT-org in Fig. 7 (c), where A_2 has been expanded with $\{v, w\}$. Comparing Fig. 1 (c) and Fig. 7 (c), we see that under the given privacy weight assignment, w in A_1 and A_3 is newly shared with A_2, and v in A_3 and A_4 is newly shared with A_2.

Suppose v is regarded highly sensitive by A_3 and A_4, and they prefer to avoid disclosing v if possible. Hence, they assign the following weights differently from Tables 2 and 3:

$$\omega_{A_3} = 0.82, \;\; \omega_{A_4} = 0.24, \;\; \omega_v^{Id} = 0.68, \;\; \omega_v^{Dom} = 0.79.$$

Weights for agent identities are reduced from those in Table 2, weights for variable v are increased from those in Table 3, and the μM average is maintained. That is, the agents must be willing to sacrifice the privacy of some other information, in order to ensure preservation of privacy for variable v. We show the execution of DAER below:

In round 1 of DAER, A_0 is self-eliminated as above. In the EE stage, although each agent has the same optional expansion plans, their privacy losses are now different. The second plan of A_3, expanding A_4 with $\{w\}$, now has loss 2.01 and becomes the best. Subsequent depth-first-search selects it as the global optimal, which leads to the CG in Fig. 8 (a).

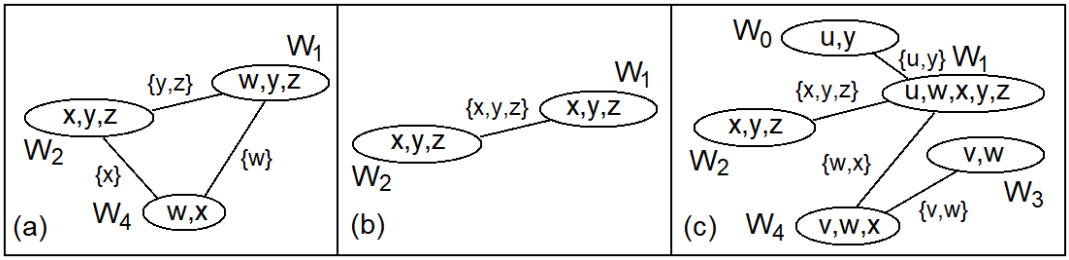

Figure 8: Reduced CG after round 1 (a), reduced CG after round 2 (b), and resultant JT-org (c)

In the EE stage of round 2, the best plan of A_4 is expanding A_1 with $\{x\}$ yielding loss 2.85. The depth-first-search selects it as globally optimal. It leads to the elimination of A_4, which reduces the CG to that of Fig. 8 (b).

In round 3 of DAER, the HTBS stage eliminates one of the remaining agents, and terminates DAER with the JT-org in Fig. 8 (c). Comparing Fig. 1 (c) and Fig. 8 (c), we see that w in A_1 and A_3 is newly shared with A_4, x in A_2 and A_4 is newly shared with A_1, and v is not shared beyond the original agents.

Table 4: Comparison of variable disclosure between two DAER executions

Execution	A_1	A_2	A_3	A_4
First	w		v, w	v
Second	w	x	w	x

In the previous execution of DAER, v in A_3 and A_4 is newly shared. In the alternative execution, this is avoided and replaced by newly sharing x in A_2 and A_4 (Table 4). From the perspective of A_4, disclosure of v is traded with disclosure of x, as well as its identity to A_1.

In general, by assigning privacy weights differently, an agent can trade disclosure of information with different sensitivities. The protocol exemplified ensures that they cannot reduce the amount of private information disclosed at the price of increased amount of disclosure by other agents. On the other hand, DAER will greedily minimize the overall privacy loss of the multiagent system no matter how privacy weights are assigned. The simple protocol for privacy weight assignment presented above only serves to show that such protocols exist. We leave the in-depth study of privacy weight assignment protocols to future research.

6.2 Evaluating Privacy Loss and Efficiency

To evaluate effectiveness of DAER in preserving JT-org linked privacy and its efficiency, we empirically compare DAER with alternative distributed JT-org construction methods in ActionGDL and DCTE. The experiment is conducted using simulated agent environments that do not admit a JT-org. We identify three independent parameters of environments that critically influence the performance of alternative methods. They are the number of agents (N), the maximum boundary size (Y), and the ratio between numbers of private and shared variables per subenvironment (R). Other relevant parameters exist, such as the maximum subenvironment size ($SubenvSz$) and the maximum border size ($BdrSz$). $SubenvSz$ is excluded as it is dependent on Y and R, and must increase as Y and R in order to be valid. $BdrSz$ is also dependent on Y, and must increase as Y. We therefore simulated agent environments according to N, Y, and R.

To evaluate how performance of JT-org construction methods depends on the environment parameters, we simulated three sets of environments. Environments in each set are identical on values of two parameters, with the third parameter varying its value. In particular, each set is divided into 5 groups differing by the value of the third parameter. Each group consists of 50 distinct environments of identical parameter values. The experimental environments are summarized in Table 5.

Table 5: Summary of experimental environments

Env Set	N	Y	R	# Env Groups	# Envs
1	10,50,100,150,200	20	1	5	250
2	50	10,15,20,25,30	1	5	250
3	50	15	1,2,3,4,5	5	250

Environments in the 1st set differ by the number of agents/subenvironments. For each environment, $Y = 20$ and $R = 1$. Hence, the maximum subenvironment size is 40 from $Y * (1 + R)$. The number of agents varies across groups from 10 per environment to 200. Each environment in the $N = 200$ group has about 2400 variables. The set consists of 5 groups with 50 environments per group, and has a total of 250 distinct environments.

Environments in the 2nd set differ by the maximum subenvironment size. Each environment has 50 subenvironments, with the maximum subenvironment size varies across groups from 20 to 60. Half of the variables of each agent are private.

Environments in the 3rd set differ by the ratio of the number of private variables over that of shared variables. Each environment has 50 subenvironments. The number of private variables per subenvironment varies across groups from the same number of shared variables to being 5 times as many. Hence, their maximum subenvironment size varies across groups from 30 to 90 variables.

For each environment, privacy weights for variables, agent identities, and border relations are simulated following the protocol in Section 4.3. The protocol requires each agent to assign all four types of privacy weights, with their average being $\mu = 0.5$.

Each environment is run by DAER, ActionGDL, and DCTE, until a JT-org is constructed. DCTE leaks identities of some variables, but not their domains. It can leak both identity and domain for other variables. For fair comparison, 2 weights are associated with each variable, one for leak of its identity and another for leak of its domain. The total number of distinct environments is 750, resulting in a total of 2250 runs.

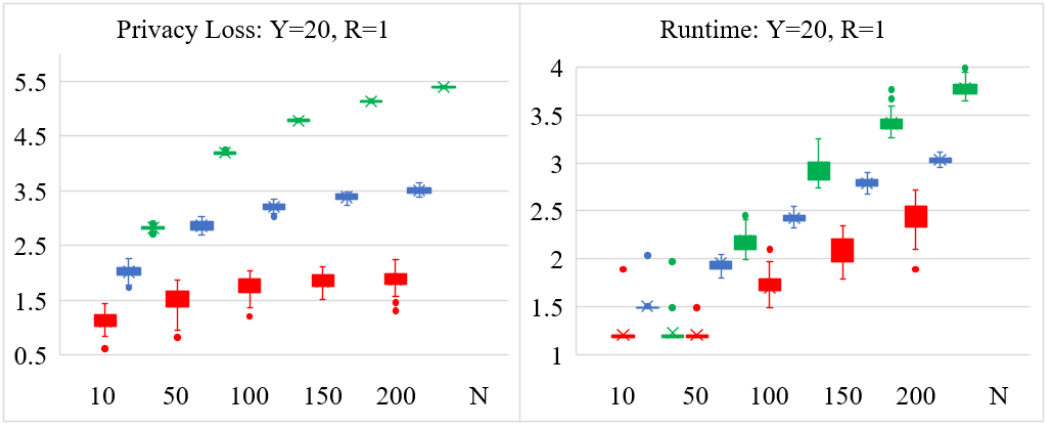

Figure 9: Privacy losses (left) and runtimes (right) for the 1st set of environments.

Fig. 9 (left) shows privacy losses (in log10) by each method for the 1st set of 250 environments. Each boxplot shows the results over 50 environments in the same group by a particular method. All boxplots by the same method have the same color. The 1st three boxplots in the left correspond to the environment group with $N = 10$, processed by DAER (red), ActionGDL (blue), and DCTE (green), respectively.

As the number of agents N increases from 10 to 200, privacy loss by each method increases. The average privacy losses by DAER range from 13.7 ($N = 10$) to 80.3 ($N = 200$). Those by ActionGDL range from 106.0 to 3210.1. Those by DCTE range from 660.6 to 240332.4. At $N = 200$, the average privacy loss by ActionGDL is 40 times higher than DAER, and that by DCTE is 2994 times higher.

Fig. 9 (right) shows runtimes (msec in log10) by each method. As N increases, runtime by each method increases. The average runtimes by DAER range from 16.8 msec to 288.4 msec. Those by ActionGDL range from 32.7 msec to 1076.5 msec. Those by DCTE range from 18.4 msec to 6073.3 msec. At $N = 200$, the average runtime by ActionGDL is 3 times higher than DAER, and that by DCTE is 21 times higher. Hence, DAER performed significantly better on both privacy loss and runtime, while the number of agents scales up.

Fig. 10 (left) shows privacy losses for the 2nd set of 250 environments, where the maximum boundary size Y increases from 10 to 30. As the boundary size increases, privacy loss by each method increases as well. The average privacy losses by DAER range from 24.4 ($Y = 10$) to 42.1 ($Y = 30$). Those by ActionGDL range from 277.2 to 1081.8. Those by DCTE range from 6807.2 to 21887.3. At $Y = 30$, the average privacy loss by ActionGDL is 25 times higher than DAER, and

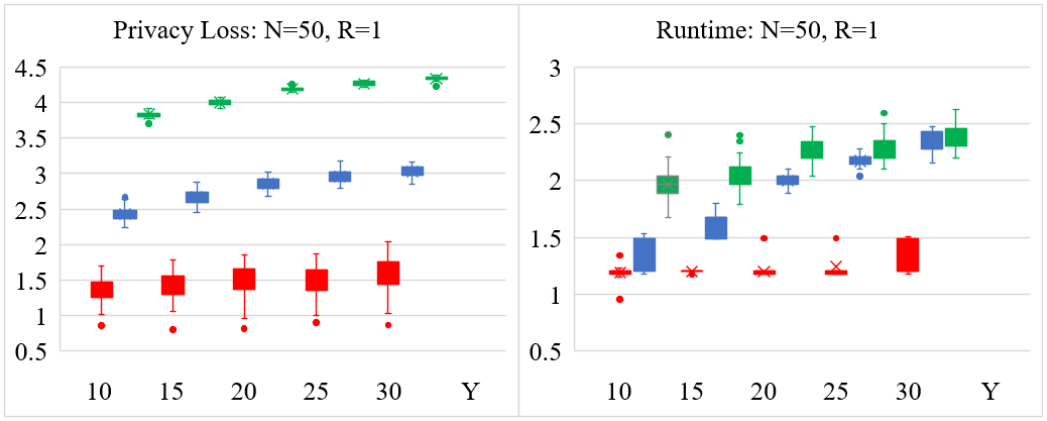

Figure 10: Privacy losses (left) and runtimes (right) for the 2nd set of environments.

that by DCTE is 520 times higher. Hence, DAER suffers significantly less privacy loss across the range of boundary sizes.

Fig. 10 (right) shows runtimes by each method. As Y increases, runtime by each method increases. The average runtimes by DAER range from 15.6 msec to 22.0 msec. Those by ActionGDL range from 25.5 msec to 230.5 msec. Those by DCTE range from 97.1 msec to 254.1 msec. At $Y = 30$, the average runtime by ActionGDL is 10 times higher than DAER, and that by DCTE is 11 times higher. Hence, runtime by DAER is significantly less than alternative methods, and more so when boundaries become larger.

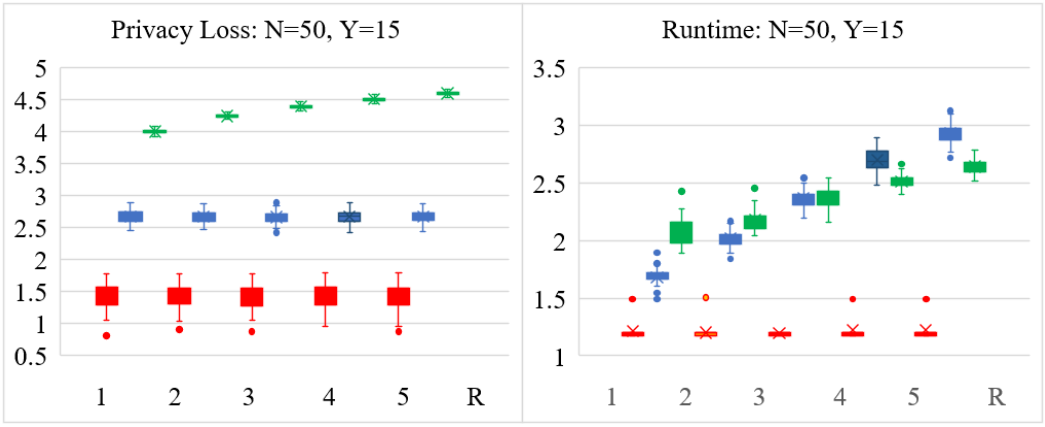

Figure 11: Privacy losses (left) and runtimes (right) for the 3rd set of environments.

Fig. 11 (left) shows privacy losses for the 3rd set of 250 environments, where the ratio R increases from 1 to 5. DAER is immune to privacy loss by private variables, and the average privacy loss remains about 29, unaffected by increase in R. Although ActionGDL is not immune to privacy loss by private variables (see analysis in [21]), the result shows that its loss remains about 470 (16 times higher than that of DAER), also unaffected by increase in R. On the other hand, the average privacy losses by DCTE increased from 10026.6 to 39434.5 (1359 times higher than that of DAER).

Fig. 11 (right) shows runtimes by each method. As R increases, runtime by DAER remains at about 17 msec. Those by ActionGDL range from 48.2 msec to 863.0 msec. Those by DCTE range from 127.4 msec to 440.1 msec. At $R = 5$, the average runtime by ActionGDL is 50 times higher than DAER, and that by DCTE is 25 times higher. Overall, privacy loss and runtime by DAER are both the lowest, and are unaffected by increases in private variables. The level of privacy loss by ActionGDL is in the middle of the three methods, is unaffected by increases in private variables, but its runtime grows faster than DCTE. DCTE has the highest privacy loss, the loss has the most significant growth with R, but its runtime growth is less significant than ActionGDL.

In summary, DAER has significantly less privacy loss and runtime as the number of agents, sizes of subenvironments, and the ratio of private variables scale up, when compared with alternative methods.

7. Conclusion

The main contribution of this work is the DAER algorithm suite. DAER is the first known algorithm that performs all three tasks of JT-org construction. When the environment decomposition admits a JT-org, DAER constructs one without JT-org linked privacy loss. Coupled with suitable protocols to regulate agent privacy weight assignment, DAER allows each agent to quantify sensitivity of different pieces of private information, so that less sensitive information is disclosed when it is necessary to disclose some. When the environment decomposition does not admit a JT-org, DAER re-decomposes the environment to construct a JT-org, while greedily minimizing the privacy loss. Although DAER does not guarantee the minimal total privacy loss, our experiments show that DAER dominates the alternative methods with significantly lower privacy loss (up to 4 orders of magnitude relative to DCTE). It is categorically superior over alternative methods by being immune to privacy loss over private variables.

DAER is efficient with linear time on the number of agents and the number of adjacent agent pairs. Experimentally, its runtime is significantly less than alternative methods across a range of values for the number of agents, the size of subenvironments, and the ratio of private variables.

DAER is general relative to privacy weight assignment, in the sense that no matter how agents quantify the sensitivity of their privacy, DAER always greedily minimizes the privacy loss over the agent society. Although game-theoretic behavior of agents is possible in their privacy weight assignment, protocols that regulate such behavior and prevent agents from doing so exist. We presented one such protocol to demonstrate their existence, and leave in-depth study of such protocols to future research.

Acknowledgement

Financial supports from the NSERC Discovery Grant to the first author, and the Scholarship from the Saudi Arabian Cultural Bureau to the second author are acknowledged. We thank anonymous reviewers for their time, effort, and constructive criticism.

References

[1] S.M. Aji and R.J. McEliece. The generalized distributive law. *IEEE Trans. Information Theory*, 46(2):325–343, 2000.

[2] I. Brito and P. Meseguer. Cluster tree elimination for distributed constraint optimization with quality guarantees. *Fundamenta Informaticae*, 102(3-4):263–286, 2010.

[3] Y. Gao, F. Toni, H. Wang, and F. Xu. Argumentation-based multi-agent decision making with privacy preserved. In J. Thangarajah, K. Tuyls, C. Jonker, and S. Marsella, editors, *Proc. 15th Inter. Conf. on Autonomous Agents and Multiagent Systems*, pages 1153–1161, 2016.

[4] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[5] K.D. Hoang, F. Fioretto, P. Hou, M. Yokoo, W. Yeoh, and R. Zivan. Proactive dynamic distributed constraint optimization. In J. Thangarajah, K. Tuyls, C. Jonker, and S. Marsella, editors, *Proc. 15th Inter. Conf. on Autonomous Agents and Multiagent Systems*, pages 597–605, 2016.

[6] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *Proc. 17th Inter. Joint Conf. on Artificial Intelligence*, pages 1027–1034, 2001.

[7] T. Le, F. Fioretto, W. Yeoh, T.C. Son, and E. Pontelli. Er-dcops: A framework for distributed constraint optimization with uncertainty in constraint utilities. In J. Thangarajah, K. Tuyls, C. Jonker, and S. Marsella, editors, *Proc. 15th Inter. Conf. on Autonomous Agents and Multiagent Systems*, pages 606–614, 2016.

[8] T. Leaute and B. Faltings. Protecting privacy through fistributed vomputation in multi-agent decision making. *J. Artificial Intelligence Research*, 47:649–695, 2013.

[9] A. Maestre and C. Bessiere. Improving asynchronous backtracking for dealing with complex local problems. In *Proc. 16th European Conf. on Artificial Intelligence*, pages 206–210, 2004.

[10] 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. *J. Autonomous Agents and Multi-Agent Systems*, 13(1):27–60, 2006.

[11] S. Mahmoud, S. Miles, and M. Luck. Cooperation emergence under resource-constrained peer punishment. In J. Thangarajah, K. Tuyls, C. Jonker, and S. Marsella, editors, *Proc. 15th Inter. Conf. on Autonomous Agents and Multiagent Systems*, pages 900–908, 2016.

[12] P.J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligences*, 161(1-2):149–180, 2005.

[13] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proc. 19th Inter. Joint Conf. on Artificial Intelligence*, pages 266–271, 2005.

[14] T. Tassa, T. Grinshpoun, and R. Zivan. Privacy preserving implementation of the Max-Sum algorithm and its variants. *J. Artificial Intelligence Research*, 59:311–349, 2017.

[15] M. Valtorta, Y.G. Kim, and J. Vomlel. Soft evidential update for probabilistic multiagent systems. *Int. J. Approximate Reasoning*, 29(1):71–106, 2002.

[16] M. Vinyals, J.A. Rodriguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *J. Autonomous Agents and Multi-Agent Systems*, 22(3):439–464, 2010.

[17] Yang Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *Artificial Intelligence*, 87(1-2):295–342, 1996.

[18] Yang Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, Cambridge, UK, 2002.

[19] Yang Xiang. Building intelligent sensor networks with multiagent graphical models. In N. Ichalkaranje G.P. Wren and L.C. Jain, editors, *Intelligent Decision Making: An AI-Based Approach*, pages 289–320. Springer-Verlag, 2008.

[20] Yang Xiang and Frank Hanshar. Multiagent decision making in collaborative decision networks by utility cluster based partial evaluation. *Inter. J. Uncertainty, Fuzziness and Knowledge-Based Systems*, 23(2):149–191, 2015.

[21] Yang Xiang and Kamala Srinivasan. Privacy preserving existence recognition and construction of hypertree agent organization. *J. Autonomous Agents and Multi-Agent Systems*, 30(2):220–258, 2016.

[22] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: reaching agreement without revealing private information. *Artificial Intelligence*, (161):229–246, 2005.