Privacy Sensitive Construction of Junction Tree Agent Organization for Multiagent Graphical Models

Yang Xiang YXIANG@UOGUELPH.CA
Abdulrahman Alshememry AALSHEME@UOGUELPH.CA
University of Guelph, Canada

Abstract

Junction trees (JTs) are not only effective structures for single-agent probabilistic graphical models (PGMs), but also effective agent organizations in multiagent graphical models, such as multiply sectioned Bayesian networks. A natural decomposition of agent environment may not allow construction of a JT organization. Hence, re-decomposition of the environment is necessary. However, re-decomposition incurs loss of agent privacy that ultimately translates to loss of intellectual property of agent suppliers. We propose a novel algorithm DAER (Distributed Agent Environment Re-decomposition) that re-decomposes the environment to enable a JT organization and incurs significantly less privacy loss than existing JT organization construction methods.

Keywords: Multiply sectioned Bayesian networks; Multiagent systems; Multiagent PGMs; Junction tree agent organization; Privacy in PGMs.

1. Introduction

JTs are not only effective structures for single-agent PGMs, but also effective agent organizations (governing which pairs of agents communicate directly) in multiagent systems. Multiply sectioned Bayesian networks (MSBNs, Xiang (1996)) are earliest multiagent PGMs based on JT organizations, where each agent uses a JT as the runtime model, and all agents are also organized into a JT (referred to as *hypertree*). JT organizations are also applied to distributed constraint optimization (Vinyals et al. (2010); Brito and Meseguer (2010)) and decentralized decision making (Xiang and Hanshar (2015)). Some multiagent systems use pseudotrees (Hoang et al. (2016); Le et al. (2016)). It has been shown (Vinyals et al. (2010)) that JT organizations are superior to pseudotrees.

Privacy is an important issue in multiagent systems (Faltings et al. (2008); Yokoo et al. (2005); Maheswaran et al. (2006)). We identify two distinct types of privacy. In meeting scheduling by personal agents (Maheswaran et al. (2006)), information to be protected concerns private constraints and preferences of humans, referred to here as *user privacy*. In equipment monitoring, e.g., a fertilizer plant (Xiang (2008)), agents are built by independent developers who supply corresponding plant subsystems. What is to be protected is the intellectual property of each supplier in terms of domain knowledge built into the agent. This is referred to as *agent privacy*, which has rarely been emphasized in literature, and is the focus of this work.

Very few work has been done on protecting agent privacy during JT organization construction. Construction techniques in several frameworks that depend on JT organizations, e.g., Xiang (2002); Vinyals et al. (2010); Brito and Meseguer (2010), compromise agent privacy on private variables, shared variables, agent identities and adjacency relations, as shown in Xiang and Srinivasan (2016).

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., Yokoo et al. (2005). It requires multiple external servers that may not always

be available or justifiable for the benefit. The second approach minimizes the amount of private information transmitted, e.g., Maheswaran et al. (2006).

Adopting the second approach, HTBS (HyperTree construction based on Boundary Set) algorithm (Xiang and Srinivasan (2016)) determines whether a decomposition of agent environment admits a JT organization. If so, HTBS constructs one without privacy loss on private variables, shared variables, agent identities and adjacency relations. Hence, when environment admits a JT organization, HTBS is superior than alternative construction methods, such as those in Action-GDL (Vinyals et al. (2010)) and DCTE (Brito and Meseguer (2010)), which incur agent privacy loss.

A decomposition of agent environment may not admit a JT organization. In such cases, methods in Action-GDL and DCTE revise the decomposition to construct a JT organization, while incurring privacy loss. HTBS, on the other hand, terminates without constructing a JT organization.

The main contribution of this work is a novel algorithm DAER that builds on top of HTBS and enables JT organization construction when agent environment decomposition does not admit one. DAER does so by modifying the decomposition, as Action-GDL and DCTE do, but with less agent privacy loss. This advancement significantly improves agent privacy in multiagent PGMs, such as multiply sectioned Bayesian networks, as well as any multiagent systems that utilize JT agent organizations.

The remainder is organized as follows. Section 2 reviews JT organization 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

Let $A = \{A_0, ..., A_{\eta-1}\}$ be a set of cooperative (and independent) agents, whose environment is described by a collection V of environment 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*.

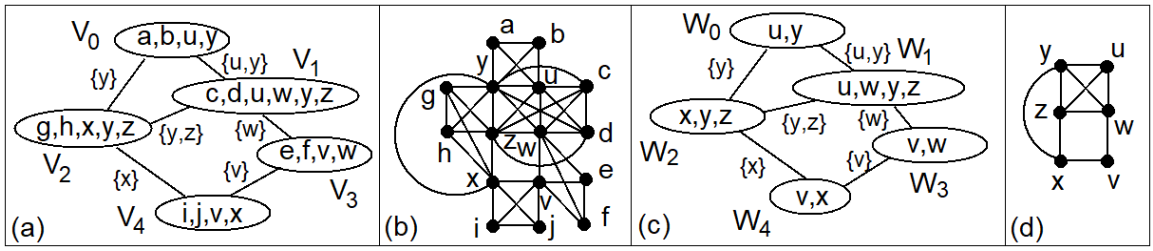

Figure 1: (a) Environment decomposition cluster graph. (b) Undirected environment decomposition graph. (c) Communication graph. (d) Boundary graph.

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 a border. It 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

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

If the system is an MSBN (Xiang (2002)), each V_i will be encoded as a Bayesian subnet. For distributed constraint optimization, each V_i will be encoded as a constraint subnet.

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 only shared variables.

A JT organization is a tree subgraph (with all clusters) of the environment decomposition cluster graph with running intersection. A decomposition may not admit a JT organization. Hence, construction of JT organization involves 3 related tasks: (a) Determine whether a JT organization exists. (b) If so, construct one. (c) If not, revise environment decomposition to construct one.

When agents are developed by independent suppliers, e.g., in an MSBN system for monitoring a fertilizer plant, a natural environment decomposition embeds 4 types of private information relative to each agent: (1) private variables, (2) variables shared with adjacent agents, (3) agent identity, and (4) bordering relations. These pieces of information may be leaked when conducting tasks (a), (b), and (c), which we refer to as *privacy loss*. A private variable may be leaked to other agents, a shared variable to non-sharing agents, or identity of an agent and its bordering relation to non-bordering agents. Although other private knowledges exist, they are not involved during JT construction and are immune to privacy loss during multiagent inference (Xiang and Srinivasan (2016)).

For instance, JT organization construction in MSBNs performs tasks (a) and (b). It leaks shared variables, agent identities and bordering relations to a coordinator agent (Xiang (2002)). JT constructions in Action-GDL and DCTE perform tasks (b) and (c). Action-GDL is based on pseudotree conversion and leaks information on private and shared variables (Vinyals et al. (2010)). DCTE (Brito and Meseguer (2010)) is based on variable propagation and incurs privacy loss on all 4 types.

HTBS is a recent algorithm for tasks (a) and (b) without privacy loss (Xiang and Srinivasan (2016)). It classifies boundary sets as follows. A (maximal) clique in a BG is *boundary contained* if it is a subset of a boundary. A boundary set has one of 3 types. It is type 1 if the boundary graph is chordal and its cliques are boundary contained. It is type 2 if the boundary graph is not chordal. It is type 3 if the boundary graph is chordal but some clique is not boundary contained. The boundary graph in Fig. 1 (d) is non-chordal. Hence, the above boundary set W is type 2. An environment decomposition admits a JT organization iff its boundary set is type 1 (Xiang and Srinivasan (2016)). Since W is type 2, no JT organization exists for the above decomposition Ω.

HTBS is based on recursive agent self-elimination. An agent A_i with boundary W_i can be *self-eliminated relative to* a bordering 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. In that case, a JT organization emerges from the trace saved during HTBS. Otherwise, no JT exists.

3. DAER for Environment Re-decomposition

For a multiagent PGM based on JT organization, if the environment decomposition does not admit a JT organization, it must be revised. Assuming that no variable can be removed, the only option is to share some variables beyond 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 bordering relations, as shown below. The challenge is how to minimize such loss. We present DAER, a novel algorithm suite, that re-decomposes the environment to construct a JT organization. It differs form methods in Action-GDL and DCTE as follows: (1) DAER operates on CGs or BGs, rather than on environment decomposition cluster graphs or their undirected equivalent. Hence, DAER is free from privacy loss on private variables. (2) DAER is based on agent self-elimination, which naturally extends HTBS. (3) 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.

3.1 Overview of DAER

Given an agent environment (A, Ω, W), DAER (Algo. 1) takes (A, W) as input. It runs in multiple rounds. Each round consists of an HTBS stage and an elimination-expansion (EE) stage. Note that, to fully utilize space, we include Algo. 2 here that is not referenced until Section 3.2. A pointer of Algo. 2 will be placed there, when it is first referenced, to mitigate the inconvenience to reader.

Algorithm 1 $DAER(A, W)$

1 *do*
2 *hasJT = HTBS(A, W);*
3 *if hasJT = true, halt;*
4 *each active agent plans a best neighbor for expansion;*
5 *select one active agent A_X with boundary X and its active neighbor A_Y with boundary Y;*
6 *A_X shares $X \setminus Y$ with A_Y, self-eliminates, and be inactive;*
7 *A_Y expands Y into $Y \cup X$;*

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 HTBS stage (lines 2 and 3) 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 organization, and one is constructed (Xiang and Srinivasan (2016)). Otherwise, the EE stage (lines 4 to 7) is run, in which only remaining active agents participate. Before presenting algorithmic details of the EE stage, we illustrate with the example in Fig. 1.

Suppose that HTBS starts by 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 announces non-existence of JT.

During the EE stage, active agents operate on a reduced CG, without W_0 and incident links (see Fig. 2 (a)). First, each agent plans a best neighbor for boundary expansion. 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.

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

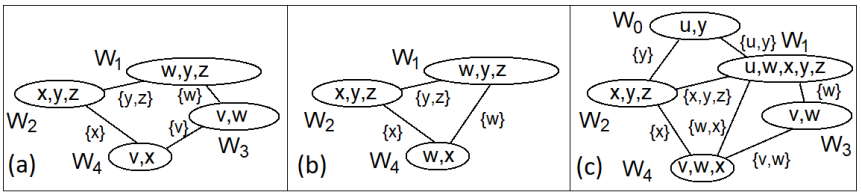

Figure 2: Reduced communication graphs after elimination of A_0 (a) and A_3 (b). (c) Re-decomposed communication graph.

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 through a distributed depth-first-search (DFS) over the CG in Fig. 2 (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). As agents report back the best loss found to their parents, 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 $\{v, w, x\}$, and allows A_3 to self-eliminate. Fig. 2 (b) shows the new CG. As the result of sharing w, a border is introduced between A_4 and A_1. HTBS is run once again led by A_4. Since no agent can self-eliminate, A_4 eventually announces non-existence of JT organization. During EE, suppose A_2 is selected by DFS. It shares x with A_1 and self-eliminates. After expansion, W_1 becomes $\{w, x\}$.

HTBS is then run the third time and concludes existence of JT organization. The re-decomposed CG is shown in Fig. 2 (c). The resultant boundary set is type 1. It can be seen by tracing HTBS on Fig. 2 (c). The boundaries can be self-eliminated in the order (W_0, W_2, W_3, W_1) with a single W_4 remaining. After presenting relevant algorithms, we establish soundness of DAER in general.

3.2 Leader Agents

Although Algo. 1 seems centrally controlled (to ease comprehension), the computation is a sequence of distributed HTBS and EE stages, each initiated by a leader agent. We describe how each leader agent is selected and its role. Since any agent may be the leader for a given stage, we present Algo. 2 (at start of Section 3.1) that every agent Ag executes in response to relevant messages or events.

The first round of DAER starts with an HTBS stage. The HTBS computation assumes an externally specified leader agent (Xiang and Srinivasan (2016)). 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 2 and 3. 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" or "JT exists". In the original HTBS (Xiang and Srinivasan (2016)), the agent then starts a halting process. DAER handles these events differently. When HTBS ends with "no JT", the announcing agent becomes leader of the next EE stage, as indicated in lines 3 and 4. When HTBS ends with "JT exists", the current, expanded boundary set is type 1 (established later). Each agent has a possibly expanded boundary. Using the trace saved, a JT is well defined and each agent knows its adjacent agents in the organization. Lines 8 and 9 encapsulate the corresponding computation.

During EE stage, the best expansion among active agents will be determined, and the winning agent A_X (line 5 of Algo. 1) will be notified. A_X runs line 6 of Algo. 1, and the expanding agent A_Y runs line 7, which ends the EE stage. The next HTBS stage is initialized by lines 5 to 7 of Algo. 2. Note that 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 5 to 7 of Algo. 1. We mention how computation in line 4 is integrated, but will elaborate later.

At 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 $\{u, w, y, z\}$. Since A_0 is self-eliminated during the first HTBS, at 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. 2 (a). Note that the active boundary of A_1 is reduced to $\{w, y, z\}$.

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

Below, we present a suite of algorithms executed by active agents in response 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 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. Once $bestScore$ is initialized in line 1, A_i maintains and updates this value.

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);
```

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);
```

Algorithm 5 $RespondNonChildReport$

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

DFS calls Algo. 4 ForwardDFS. During DFS, to respond to $NonChildReport$, A_i runs Algo. 5. During $ForwardDFS$, to respond to $ChildReport$, A_i runs Algo. 6.

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);

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 $Notify(winningScore)$;

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

It may appear uncertain whether agent A_k in line 4 exists. It is guaranteed for the reason below. Line 4 is only run if A_i is not the root. Hence, $winningScore$ originates from a non-root agent on the DFS tree. It must be associated with a DFS tree child of the root, and that agent is A_k when the root executes 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, and 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 Algo. 3 through 7, no agent knows the expansion plan of any other agent. This condition holds since the expansion plan is not included in any message among agents. As a consequence, no agent knows the expansion plan of the winning score, except the agent that generated 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;

Algorithm 9 $RespondExpand$

1 denote the set of adjacent agents of A_i by S;
2 initialize a set T to be the border between A_i and A_c;
3 for each agent $A_j \in S$ such that $A_j \neq A_c$, $T = T \setminus I_{ij}$;
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 $S = S \cup S_Q$;

Algo. 8 is a 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. In Algo. 9, the identity of A_k is disclosed to them as well. We will elaborate on additional loss in Section 4.

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. The execution of Algo. 9 ends an EE stage.

4. Evaluating 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 construction may suffer from agent privacy loss on private and shared variables, on agent identities, and on agent bordering relations. Few measures exist to evaluate such privacy loss. The measure in Maheswaran et al. (2006) assumes that agent identities and their state spaces are publicly known, and is not applicable. The measure in Xiang and Srinivasan (2016) 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, 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 bordering relation.

[**Variable based loss**] Next, we consider how an agent evaluates expansion plans by the measure. When an agent leaks a private 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 shared variable (sv) based loss is $Loss_{sv} = \sum_{x \in S} \omega_x$.

[**Agent identity based loss**] When a set S of variables is shared with a new agent, it also incurs identity based loss, mentioned after Algo. 8. Suppose variable x is shared by agents A and B, but not by 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). If B and C have a common variable, then $ncv(B, C) = 0$, and otherwise $ncv(B, C) = 1$. Similarly, we define a function $ncv(A, B, C)$. If A, B and C have a common variable, then $ncv(A, B, C) = 0$, and otherwise $ncv(A, B, C) = 1$. Note that if $ncv(B, C) = 1$, then $ncv(A, B, C) = 1$.

Since agent A needs to evaluate the loss due to sharing x with C, we describe knowledge state of A through the ncv function. If A shares a variable $y \neq x$ with both B and C, then A knows that sharing x with C will not cause identity loss. This condition is described by $ncv(A, B, C) = 0$, and identity loss relative to B and C is ruled out. If A has no common variable with B and C, i.e., $ncv(A, B, C) = 1$, then B and C may or may not share variables. Here, $ncv(A, B, C)$ is known to A, but $ncv(B, C)$ is not. As the value of $ncv(B, C)$ is unknown to A, it does not know whether sharing x with C causes identity loss. Therefore, when A newly shares a set S of variables with C, the agent identity based loss is $Loss_{id} = \sum_{A_S} ncv(A_S, C)\ (\omega_{A_S} + \omega_C)$, where A_S is an agent that contains some variable in S (hence adjacent to A) and $ncv(A, A_S, C) = 1$. Since the value of $ncv(A_S, C)$ is unknown to A, so is that of $Loss_{id}$. We let each agent evaluate identity based loss according to $MaxLoss_{id} = \sum_{A_S} (\omega_{A_S} + \omega_C)$, to minimize the maximum loss.

[**Border relation based loss**] Algos. 8 and 9 explicitly disclose private information over shared variables and agent identities. Agent bordering relations are implicitly disclosed as well. Suppose a

variable x is shared by agents A and B, but not by 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. When A shares x with C, A informs C to become adjacent with B. This allows C to infer the A-B bordering relation (a privacy loss). This loss does not depend on whether B and C already share variables. That is, the value of $ncv(B, C)$ is irrelevant.

In the above case, B did not know that A and C are adjacent either. By being adjacent with C due to sharing x and observing A becoming self-eliminated, B can also infer the A-C bordering relation. On the other hand, if $ncv(A, B, C) = 0$, adjacency between any pair among A, B and C is a prior knowledge of the third. Hence, the sharing does not incur border relation based loss. Therefore, when A newly shares a set S of variables with C, the border relation (br) based loss is $Loss_{br} = \sum_{A_S}(\omega_{A,A_S} + \omega_{A,C})$, where A_S is an agent with some variable in S and $ncv(A, A_S, C) = 1$.

In summary, when agent A proposes to newly share a set S of variables with an adjacent agent C, the expansion plan is evaluated by procedure $GetBestExpPlan$ according to

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

Consider an example where A_2 in Fig. 2 (a) evaluates the expansion plan of sharing y and 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 the first HTBS stage. The privacy loss is evaluated as follows:

$$Loss_{sv} = 0.622 + 0.875, \quad MaxLoss_{id} = (0.723 + 0.253) + (0.237 + 0.253),$$
$$Loss_{br} = (0.716 + 0.682) + (0.162 + 0.682), \quad MaxLoss = 5.207.$$

A DAER execution consists of $K \geq 1$ rounds. If environment decomposition Ω has a JT, 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 organization, 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.

5. Soundness and Complexity

Assuming that the initial environment decomposition Ω does not admit a JT organization, Theorem 1 below shows that upon termination of DAER, the final expanded environment decomposition Ω' has a JT organization, while the privacy loss is greedily minimized. Due to space limit, we omit proof.

Theorem 1 *Let Ω be an environment decomposition that does not admit a JT organization 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 organization.*

By Theorem 1, DAER re-decomposes the environment to enable construction of a JT organization. A JT organization can be specified distributively from the trace saved during HTBS (Xiang and Srinivasan (2016)), which can be directly extended to DAER. We omit the details due to space.

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 DFS over the currently active CG). In each DFS, $O(e)$ 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 evaluated DAER with 14 batches (Table 1) of randomly generated agent environments that do not admit a JT organization. Batches 1 to 7 have smaller boundaries, and batches 8 to 14 have larger ones. They are indexed in Table 1 by column B. Each batch contains 20 environments, whose numbers of agents (# Agents) and maximum boarder sizes (MxBdr) are shown in Table 1. The 20 environments are divided in 2 groups of 10 each. Each subenvironment in Group 1 has as many private variables as shared variables. Each in Group 2 has twice as many private variables. The maximum sizes of subenvironments in each group are indicated in columns MxSe1 and MxSe2. For each environment, privacy weights for variables, agent identities, and border relations are simulated. The total number of environments is 280, with the numbers of agents per environment between 4 and 306, the sizes of subenvironments between 6 to 54, and the border sizes between 2 and 15.

Table 1: Summary of experimental environments

B	# Agents	MxSe1	MxSe2	MxBdr	B	# Agents	MxSe1	MxSe2	MxBdr
1	$8 \sim 13$	$6 \sim 8$	$9 \sim 12$	$2 \sim 3$	8	$4 \sim 7$	$12 \sim 16$	$18 \sim 24$	$5 \sim 6$
2	$30 \sim 37$	$8 \sim 10$	$12 \sim 15$	$3 \sim 4$	9	$18 \sim 26$	$18 \sim 22$	$27 \sim 33$	$7 \sim 9$
3	$63 \sim 74$	$10 \sim 14$	$15 \sim 21$	$3 \sim 5$	10	$35 \sim 45$	$22 \sim 26$	$33 \sim 39$	$9 \sim 12$
4	$96 \sim 123$	$10 \sim 12$	$15 \sim 18$	$4 \sim 5$	11	$56 \sim 74$	$26 \sim 28$	$39 \sim 42$	$11 \sim 12$
5	$149 \sim 173$	$12 \sim 14$	$18 \sim 21$	$4 \sim 5$	12	$92 \sim 110$	$28 \sim 32$	$42 \sim 48$	$11 \sim 13$
6	$203 \sim 233$	$12 \sim 14$	$18 \sim 21$	$5 \sim 5$	13	$125 \sim 146$	$30 \sim 32$	$45 \sim 48$	$12 \sim 15$
7	$282 \sim 306$	$14 \sim 14$	$21 \sim 21$	$5 \sim 6$	14	$165 \sim 183$	$32 \sim 36$	$48 \sim 54$	$13 \sim 15$

For comparison, each environment is run by DAER, ActionGDL, and DCTE, until a JT organization is constructed (a total of 840 runs). 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.

Figs. 3 and 4 show privacy losses (in log10) by each algorithm, where the x-axis is indexed by the corresponding 70 environments. DAER has significantly less privacy loss than ActionGDL and DCTE. As subenvironments becomes larger (Fig. 4), privacy losses by DAER are further reduced by about one order of magnitude. It reveals that DAER can better utilize increased expansion options. Privacy losses by DCTE is almost doubled (1.8 times higher) moving from Group 1 to 2, as numbers of private variables are doubled. On the other hand, privacy losses by DAER are unchanged, as it does not suffer from loss by private variables.

Fig. 5 shows runtimes (msec in log10) by each algorithm over batches 8 to 14. ActionGDL is the fastest, while DAER and DCTE have comparable runtimes. Runtimes over batches 1 to 7 are similar and are omitted due to space limit.

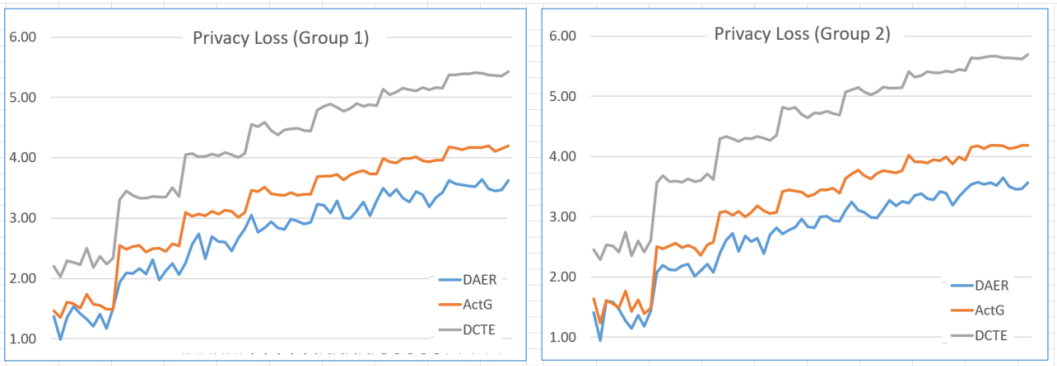

Figure 3: Privacy losses for batches 1 to 7 (smaller subenvironments)

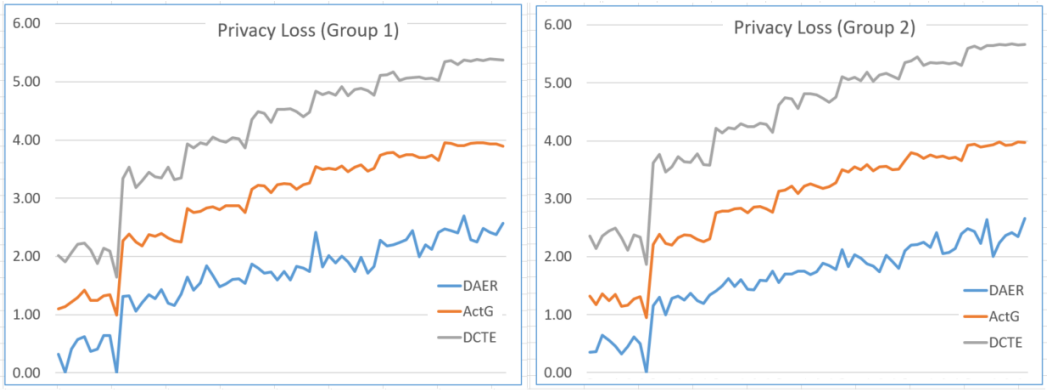

Figure 4: Privacy losses for batches 8 to 14 (larger subenvironments)

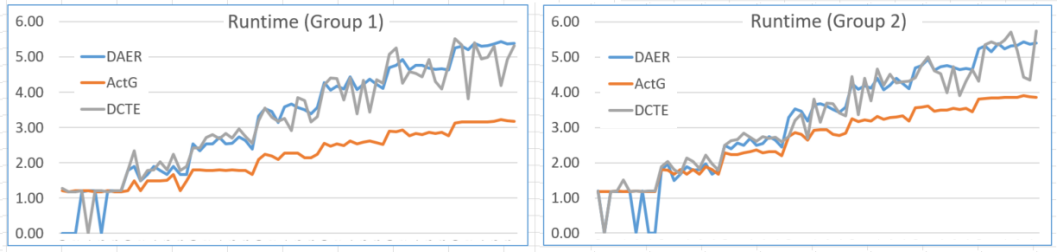

Figure 5: Runtimes for batches 8 to 14

7. Conclusion

The main contribution of this work is the DAER algorithm suite: a novel solution to re-decompose agent environment when it does not admit a JT organization. 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. DAER minimizes the total privacy loss by taking a greedy approach in selecting each boundary expansion among alternatives. 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 3 orders of magnitude relative to DCTE). It is categorically superior to 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, it takes longer runtime than ActionGDL, but is comparable with DCTE.

Acknowledgement

Financial support from NSERC Discovery Grant to first author is acknowledged.

References

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

B. Faltings, T. Leaute, and A. Petcu. Privacy guarantees through distributed constraint satisfaction. In *Proc. IEEE/WIC/ACM Intelligent Agent Technology*, pages 350–358, 2008.

K. 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.

T. Le, F. Fioretto, W. Yeoh, T. 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.

R. Maheswaran, J. 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.

M. Vinyals, J. 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.

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

Y. Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, Cambridge, UK, 2002.

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

Y. Xiang and F. 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.

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

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