Boundary Set Based Existence Recognition and Construction of Hypertree Agent Organization

Yang Xiang and Kamala Srinivasan

School of Computer Science, University of Guelph, Canada

Abstract. Some of the essential tasks of a multiagent system (MAS) include distributed probabilistic reasoning, constraint reasoning, and decision making. Junction tree (JT) based agent organizations have been adopted by some MAS frameworks for their advantages of efficient communication and sound inference. In addition, JT organizations have the potential capacity to support a high degree of agent privacy. This potential, however, has not been fully realized. We present two necessary and sufficient conditions on the existence of JT organization given a MAS. Following these conditions, we propose a new algorithm suite, based on elimination in the so called *boundary set* of a MAS, that recognizes JT organization existence and constructs one if exists, while guaranteeing agent privacy on private variables, shared variables and agent identities.

1 Introduction

Some of the essential tasks of a multiagent system (MAS) include distributed probabilistic reasoning, constraint reasoning, and decision making (decision theoretic). Existing frameworks include AEBN [8] and MSBN [10] for probabilistic reasoning, ABT [4], ADOPT [5], DPOP [7], Action-GDL [9], DCTE [1], and MSCN [12] for constraint reasoning, and RMM [2], MAID [3], and CDN [11] for decision making. Frameworks such as AEBN and MAID do not assume specific agent organization. ABT assumes a total order among agents. ADOPT and DPOPS use a pseudo-tree organization. MSBN, MSCN, CDN, Action-GDL, and DCTE all use a junction tree (JT) organization (known as *hypertree* in the first three frameworks), which is the focus of this work.

In JT-based frameworks, the application environment is represented by a set of variables, referred to as the *env*. The env is decomposed into a set of overlapping *subenvs*, each being a subset of env. The subenvs are one-to-one mapped to agents of the MAS. Subenvs (and hence agents) are organized into a JT. A JT is a tree where each node is associated with a set of variables called a *cluster*. The tree is structured such that, the intersection of any two clusters is contained in every cluster on the path between the two (*running intersection*). In a JT agent organization, each cluster corresponds to a subenv and its agent. The organization prescribes communication pathways: an agent can send a message to another, iff they are adjacent in the JT.

JT-based agent organizations support several desirable properties. First, they allow efficient communication. For agents to cooperate by utilizing information available locally at individual agents, it suffices to pass two messages along each link of the JT. Hence, the time complexity of communication is linear in the number of agents. Second,

O. Zaïane and S. Zilles (Eds.): Canadian AI 2013, LNAI 7884, pp. 187–198, 2013.

they support sound inference. Global consistency is guaranteed by local consistency, e.g., probabilistic reasoning in MSBN [10] is exact, constraint reasoning in MSCN [12] is complete, and decision making in CDN [11] is globally optimal.

Third, the above two properties are enabled while message contents involve only shared variables (those in the intersection of subenvs). Hence, JT organizations have the potential capacity to support a high degree of agent privacy. However, such potential has not been fully realized in existing JT-based MAS frameworks.

In Action-GDL [9], the JT is built through a centralized mapping operation from a pseudo-tree where each node is an env variable. The centralized mapping discloses identities of all variables to the mapping agent. The JT in DCTE [1] is constructed according to [6], where clusters (one per agent) of env variables are initially organized into a tree that is generally not a JT. Variable identities are passed along the tree links to transform the tree into a JT, and hence are disclosed beyond the agent initially associated with them.

In MSBN [10], MSCN [12], and CDN [11] frameworks, the JT organization is constructed by a coordinator agent with knowledge of variables shared by agents. Since coordinator knows nothing about private variables (those that are contained in a single subenv), their privacy is ensured. However, shared variables and agent identities are disclosed to the coordinator.

The contribution of this work is a new algorithm suite for JT organization based on the so-called *boundary-set* (see below) of the MAS. Given the boundary-set of a MAS, the algorithm first determines distributively whether a JT organization exists. If it does, then a JT will be constructed. The entire process preserves agent privacy on private variables, shared variables, and agent identities. To the best of our knowledge, no known JT-based MAS frameworks provide the same degree of agent privacy, except an alternative approach for distributed JT construction that we report in a related work [13].

The next section defines the problem that we tackle. The subsequent section presents a necessary and sufficient condition on the existence of JT organization. This insight allows a classification of MAS subenv decompositions relative to JT organization existence. The next section gives another necessary and sufficient condition on JT organization existence, which leads to an agent privacy preserving algorithm suite for JT organization existence recognition and construction, illustrated and then specified in the following two sections.

2 Problem Definition

Consider a MAS populated by a set $\mathscr{A} = \{A_0, ..., A_{\eta-1}\}$ of $\eta > 1$ agents. Let V be the set of env variables, and be decomposed into a collection of *subenvs*, $\boldsymbol{\Omega} = \{V_0, ..., V_{\eta-1}\}$, such that $\cup_{i=0}^{\eta-1} V_i = V$. Each V_i is associated with a unique agent A_i, and vice versa. We refer to the set of shared variables, $I_{ij} = V_i \cap V_j \neq \emptyset$, between A_i and A_j as their *border*. For each agent A_i, we denote the set $W_i = \cup_{j \neq i} I_{ij}$ as its *boundary*, and we refer to $W = \{W_0, ..., W_{\eta-1}\}$ as the *boundary set* of the MAS. Fig. 1 (a) shows a subenv decomposition of a trivial env, with agent boundaries shown in (b). A_i knows A_j and can communicate with A_j, iff they have a border. The message between them can only involve variables included in their border.

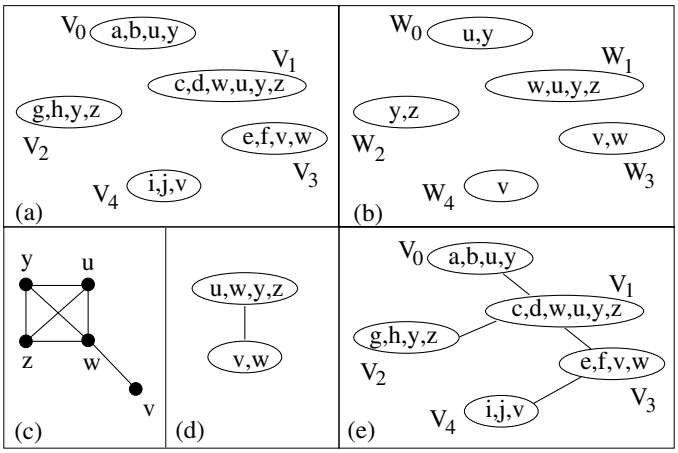

Fig. 1. (a) Subenv decomposition. (b) Agent boundaries. (c) Boundary graph. (d) JT from (c). (e) JT organization for (a).

The general task of JT agent organization is to construct a JT with subenvs as clusters such that running intersection holds.

A variable $x \in V$ is *private* if x is contained in a unique subenv V_i. Agent privacy on private variables is preserved, if no information on any private variable is disclosed to other agents, including its existence, identity, domain (of possible values), associated conditional or marginal probability distribution (in the case of probabilistic reasoning), or constraint (in the cases of constraint reasoning and decision making), and observed or assigned value.

A variable $x \in V$ is *shared* by A_i and A_j, if $x \in I_{ij}$. Agent privacy on shared variables is preserved, if no information on any shared variable is disclosed beyond agents who share it.

An agent is known to another agent, iff they share a border. Privacy on agent identities is preserved, if for every agent, its identity is not disclosed to any non-bordering agent.

The boundary set of a MAS can play an important role in privacy preserving construction of JT organizations, as established by a proposition from [13].

Proposition 1. *Let V be env of a MAS, Ω be its subenv decomposition, W be the boundary set, and T be a JT with boundaries in W as clusters. Let T' be a cluster tree with subenvs of Ω as clusters, such that it is isomorphic to T with each subenv cluster mapped to the corresponding boundary cluster in T. Then T' is a JT.*

Based on Prop. 1, we take the approach to construct a JT organization from the boundary set W, rather than from the subenv decomposition Ω. This approach guarantees agent privacy on private variables, because these variables are excluded from input of the task.

Given a boundary set, a JT made of its boundary clusters may or may not exist. Hence, the problem we tackle in this work is stated as follows. Given the boundary set of a MAS, determine whether a JT agent organization exists and, if so, construct a JT,

such that agent privacy on private variables, shared variables, and agent identities are preserved in the process.

3 Boundary Graph Based Condition on Hypertree Existence

Before the existence of JT organization can be determined algorithmically, we analyze conditions of its existence, through an alternative representation of the boundary set. Let W be the boundary set of a MAS. An undirected graph BG is the *boundary graph* of the MAS, if its set of nodes is $N = \cup_{i=0}^{\eta-1} W_i$, and its links are connected so that each W_i is complete (elements are pairwise connected).

Prop. 2 identifies a condition under which a JT can be constructed from a boundary graph such that each JT cluster is a boundary. A set of nodes in a graph is a *clique*, if they are maximally pairwise connected. Two clusters are *comparable*, if one is a subset of the other.

Proposition 2. *Let W be the boundary set of a MAS, and BG be its boundary graph, such that*

1. *BG is chordal, and*
2. *for each clique C of BG, there exists $W_i \in W$ with $C \subseteq W_i$.*

Let T be a JT whose clusters are cliques of BG, and no two clusters in T are comparable. Then for every cluster C of T, there exists a boundary $W_i = C$.

Proof: From subcondition 1, the JT T exists. We prove by contradiction. Suppose there exists a cluster C in T such that $C \neq W_i$ for every $W_i \in W$.

From subcondition 2, there exists W_i such that $C \subseteq W_i$. Since $C \neq W_i$, it follows that $C \subset W_i$. Because BG is a boundary graph, W_i is complete in BG. Therefore, there exists a cluster C_i in T such that $W_i \subseteq C_i$. From $C \subset W_i$ and $W_i \subseteq C_i$, we have $C \subset C_i$. That is, T contains two comparable clusters: a contradiction. Hence, every cluster in T is a boundary. □

Example 1. *Fig. 1 (a) shows an env decomposition. The set of boundaries is shown in (b). The BG is shown in (c), and it satisfies the two conditions. The JT from the BG is shown in (d), where the two clusters are boundaries W_1 and W_3.*

Utilizing Prop. 2, Theorem 1 establishes a necessary and sufficient condition for the existence of JT organization.

Theorem 1. *Let W be the boundary set of a MAS and BG be its boundary graph. A JT agent organization exists, iff the following hold.*

1. *BG is chordal, and*
2. *for each clique C of BG, there exists $W_i \in W$ such that $C \subseteq W_i$.*

Proof: [Necessity] Suppose a JT H exists, whose clusters are subenvs. For each cluster in H, remove its private variables. The resultant cluster tree T is still a JT, and its corresponding undirected graph is BG. From T being a JT, it follows that BG is chordal. Hence, subcondition 1 holds. The clusters of T are one-to-one mapped to boundaries of agents, from which subcondition 2 follows.

[Sufficiency] Suppose both subconditions hold. We prove by construction.

Since BG is chordal, a JT T exists whose clusters are cliques of BG. Without losing generality, assume that clusters of T are not comparable. By Prop. 2, every cluster in T is a boundary. Hence, for every cluster C such that $C = W_i$ for some i, we can associate C with an agent A_i.

If not every agent is associated with a cluster yet, consider a remaining agent A_i without being associated with any cluster of T yet. Since W_i is complete in BG, there exists a cluster C in T such that $W_i \subseteq C$. Add to T a new cluster W_i, make it adjacent to cluster C, and associate the new cluster with A_i. Repeat this for each remaining agent, until each agent is associated with a cluster in T.

Next, for each agent, add its private variables to its associated cluster in T. The resultant T is a JT agent organization with each cluster being a subenv. □

Theorem 1 provides the following insight. As far as the existence of JT organization is concerned, MAS subenv decompositions can be classified into three types.

Type 1 Boundary graphs are chordal, and their cliques are boundary contained.
Type 2 Boundary graphs are not chordal.
Type 3 Boundary graphs are chordal, but their cliques are not boundary contained.

Example 2. *The boundary graph for the subenv decomposition in Fig. 1 (a) is shown in (c). The subenv decomposition is type 1. The JT of the boundary graph is show in (d). The two clusters are associated with A_1 and A_3. For each of the three remaining agents, a cluster can be added to the JT. The JT organization is shown in (e).*

Example 3. *Fig. 2 (a) shows another subenv decomposition, with agent boundaries in (b) and BG in (c). Since BG is not chordal, the subenv decomposition is type 2. By Theorem 1, it has no JT organization.*

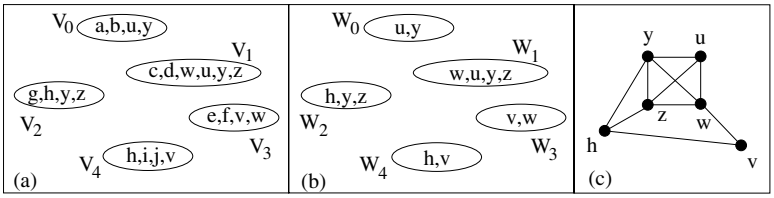

Fig. 2. (a) Subenv decomposition. (b) Agent boundaries. (c) Boundary graph.

Example 4. *For agent boundaries in Fig. 3 (a), the BG is shown in (b). It has two cliques. Since one of them, $\{h,v,w\}$, is not contained in any boundary, the subenv decomposition is type 3. By Theorem 1, it has no JT organization.*

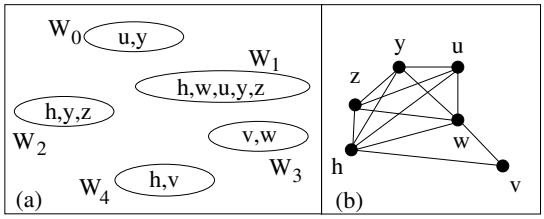

Fig. 3. (a) Agent boundaries. (b) Boundary graph.

4 Boundary Set Based Condition of Hypertree Existence

First, we define an operation to eliminate a boundary from a boundary set W. When a boundary $W_i \in W$ is *eliminated* from W *relative to* boundary $W_j \in W$, where $W_i \cap W_j \neq \emptyset$, it yields a *reduced boundary set* $W' = (W \setminus \{W_i, W_j\}) \cup \{W'_j\}$, where

$$W'_j = \bigcup_{W_k \in W, k \neq i, k \neq j} (W_j \cap W_k).$$

That is, the set W' resultant from eliminating W_i relative to W_j is obtained by deleting W_i and W_j from W, and replacing with W'_j. The W'_j is obtained by the union of borders of A_j, except the border with A_i. In other words, W'_j is the boundary W_j without variables that A_j uniquely shares with A_i. Consider the boundary set for Fig. 1 (b), $W = \{W_0, ..., W_4\}$. After W_0 is eliminated relative to W_1, the *reduced boundary set* is $W' = \{W'_1, W_2, W_3, W_4\}$, where $W'_1 = \{w, y, z\}$.

Without confusion, we refer to each element of W' as a *boundary*, whether or not it is identical to an element of the original boundary set. The elimination operation is well defined on the reduced boundary set, and hence can be performed iteratively.

Example 5. *For the boundary set of Fig. 1, elimination can be performed iteratively as follows.*

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

$$Eliminate\ \{u, y\}\ wrt\ \{w, u, y, z\} : W' = \{\{w, y, z\}, \{y, z\}, \{v, w\}, \{v\}\};$$

$$Eliminate\ \{v\}\ wrt\ \{v, w\} : W' = \{\{w, y, z\}, \{y, z\}, \{w\}\};$$

$$Eliminate\ \{w\}\ wrt\ \{w, y, z\} : W' = \{\{y, z\}, \{y, z\}\};$$

$$Eliminate\ \{y, z\}\ wrt\ \{y, z\} : W' = \{\{y, z\}\}.$$

Note that, each W_i eliminated relative to a W_j has been so chosen to satisfy $W_i \subseteq W_j$. The significance of such a choice will be seen below.

Note also that each reduced boundary set W' (except the final singleton) is a well-defined boundary set, in the sense that each variable is shared by at least two boundaries in W'. Take $W' = \{\{w, y, z\}, \{y, z\}, \{w\}\}$ for example, each of w, y, and z is shared by two boundaries.

Next, we establish another necessary and sufficient condition on hypertree existence, based on boundary elimination.

Theorem 2. *A MAS with the boundary set W has a JT agent organization, iff W can be eliminated iteratively into a singleton, such that each W_i eliminated relative to a W_j satisfies $W_i \subseteq W_j$.*

Sketch of proof: For necessity, suppose a JT H exists. Remove private variables in each cluster. The resultant cluster tree T is a JT, whose set of clusters is W. A leaf cluster satisfying the condition can be found in T, and eliminated iteratively.

For sufficiency, suppose W can be eliminated into a singleton. Denote the sequence of reduced boundary sets as $W^\eta, W^{\eta-1}, ..., W^2, W^1$, where $W^\eta = W$, W^1 is the final singleton, and the superscript indicates the number of boundaries in the set. Boundaries in each W^x, for $x = 2, ..., \eta$, can be organized into a JT. □

5 Distributed Recognition of Hypertree Existence

The condition $W_i \subseteq W_j$ in Theorem 2 is equivalent to $W_i = I_{ij}$. Hence, Theorem 2 suggests a privacy preserving, distributed computation to identify JT organization existence. Agents are self-eliminated one by one as long as possible. An agent A_i can be eliminated if its boundary is equal to the border with another remaining agent A_j. After A_i is eliminated relative to A_j, A_j removes from its boundary the variables that it shares uniquely with A_i. If all agents are eliminated except one, then a JT organization exists for the MAS. Otherwise, the JT does not exist.

We assume that a token is passed between bordering agents, according to depth-first-traversal. The first round of traversal starts at an arbitrary agent, who possesses the token tok^1. If an agent A_i who holds the token has its boundary equal to the border with another agent A_j, then A_i signifies to each bordering agent that it is eliminated, and passes a new token tok^2 to A_j.

A_j then starts the second round of traversal among remaining agents, using tok^2. If an agent starts a new round of traversal, and finds that it has no uneliminated bordering agent, then it announces existence of a JT organization.

On the other hand, suppose an agent A_j starts a new round of traversal, with at least another uneliminated bordering agent. After the token has traversed every uneliminated agent, and comes back to A_j, if A_j still has uneliminated bordering agents, then A_j announces non-existence of JT organization.

Example 6. *Consider agents in Fig. 4 (a) with their boundaries shown in ovals, where each link shows a bordering relation. Note that the subenv decomposition is type 1.*

Suppose A_0 starts first round with tok^1. It announces its elimination and passes the token to A_1. In response, A_1 reduces its boundary, as in (b), and starts second round with tok^2. It passes tok^2 to A_2. A_2 announces its elimination and passes tok^3 to A_1. In response, A_1 reduces its boundary again, as in (c), and starts third round. It announces its elimination and passes tok^4 to A_3. In response, A_3 reduces its boundary, as in (d), and starts fourth round. It announces its elimination and passes tok^5 to A_4. Finally, A_4 announces existence of a JT organization.

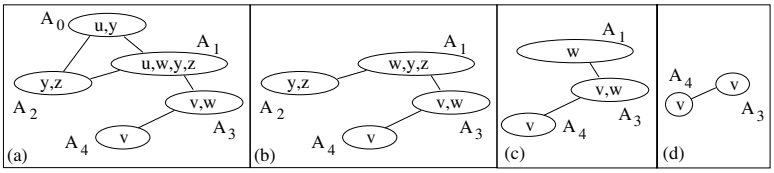

Fig. 4. Distributed recognition of JT organization with type 1 subenv decomposition

Example 7. *Consider agents and their boundaries in Fig. 5 (a). Note that the subenv decomposition is type 2.*

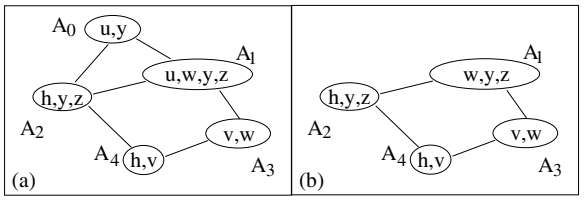

Fig. 5. Recognition of non-existence of JT organization with type 2 subenv decomposition

Suppose A_0 starts first round with tok^1, announces its elimination, and passes tok^2 to A_1. A_1 reduces its boundary, as in (b), starts second round, and passes tok^2 to A_2. A_2 passes tok^2 to A_4, who in turn passes tok^2 to A_3. A_3 has no unvisited agent to pass the token to, and returns tok^2 to A_4. A_4 returns tok^2 to A_2, who in turn returns to A_1. Finally, A_1 announces non-existence of JT organization.

Example 8. *Consider agents and their boundaries in Fig. 6 (a). Note that the subenv decomposition is type 3.*

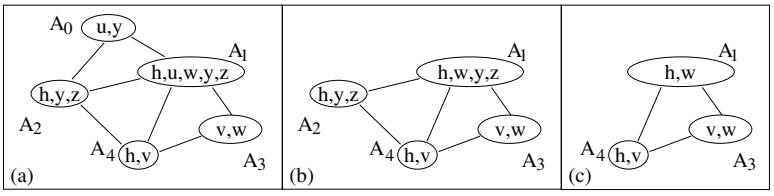

Fig. 6. Recognition of non-existence of JT organization with type 3 subenv decomposition

Suppose A_0 starts first round with tok^1, and passes tok^2 to A_1. A_1 reduces its boundary, as in (b), starts second round, and passes tok^2 to A_2. A_2 announces its elimination and passes tok^3 to A_1. A_1 further reduces its boundary, as in (c), starts third round,

and passes tok^3 to A_3. A_3 passes tok^3 to A_4. Eventually, tok^3 is returned to A_1, who announces non-existence of JT organization.

Note that during the traversal, although the *active boundary* for a remaining agent may be reduced, the border between any pair of agents never changes.

6 Algorithm for Hypertree Existence Recognition and Construction

Next, we specify a distributed algorithm suite, that agents execute to implement the computation described intuitively in the previous section. Each agent's activities are driven by responding to the following messages.

- A $StartNewDFT(tok)$ request that calls the receiver to start a new round of depth-first-traversal with the given token tok;
- An $Eliminated$ notification sent by an agent who has been self-eliminated;
- A $DFT(tok)$ request that calls the receiver to perform depth-first-traversal with the given token tok;
- A *Report* message sent by an agent who has been called to perform $DFT(tok)$, signifying either the called agent has been visited in the current round, or it has completed DFT and now backtracks to the caller.

We refer to the receiving agent of a message by A_i, who will act in response, and refer to the message sender by A_c. Every agent performs *Init* to initialize local data. Flag $state \in \{IN, OUT\}$ indicates whether A_i has been eliminated. Flag $nbsta(A_k) \in \{IN, OUT\}$ indicates the same for a bordering agent. Variable *curtok* keeps a token value after it has visited A_i, and $visited(A_k)$ indicates whether the token has visited the bordering agent. Y_i maintains the active boundary of A_i.

Procedure 1 (Init)
1 $state = IN$; $parent = null$;
2 initialize current token to $curtok = null$;
3 set active boundary $Y_i = W_i$;
4 for each bordering agent A_k,
5 $nbsta(A_k) = IN$;
6 $visited(A_k) = false$;

At the start of each round of traversal, a remaining agent will be be messaged to *StartNewDFT*. In the first round, an arbitrary *leader* agent messages itself. Agent A_i being messaged does the following.

Procedure 2 (StartNewDFT(tok))
1 if A_c is another agent,
2 $nbsta(A_c) = OUT$;
3 if there exists no A_j with $nbsta(A_j) = IN$,
4 announce "hypertree exists";

```
5          return;
6       Y_i = ∅;
7       for each bordering A_k where nbsta(A_k) = IN,
8          Y_i = Y_i ∪ I_ik;
9   curtok = tok; parent = null;
10  run DFT; // A_i has IN bordering agents
```

When DFT below is run from $StartNewDFT(tok)$, A_i has $parent = null$, and have at least one remaining bordering agent A_j. If A_i can be eliminated relative to A_j, it will message A_j to $StartNewDFT$. Otherwise, it will send message $DFT(tok)$ to A_j.

When DFT is run from $DFT(tok)$ (see below), A_i has *parent* pointing to A_c, and may have no unvisited, remaining bordering agent other than A_c. If A_i cannot be eliminated relative to A_c, it must $Report$ to A_c.

Procedure 3 (DFT)

```
1   if there exists A_j with nbsta(A_j) = IN and Y_i = I_ij, // self-eliminate
2       state = OUT;
3       for each A_k ≠ A_j where nbsta(A_k) = IN, send Eliminated to A_k;
4       send StartNewDFT(curtok + 1) to A_j;
5   else // no IN agent satisfies Y_i = I_ij
6       for each A_k ≠ parent where nbsta(A_k) = IN, set visited(A_k) = false;
7       if there exists A_k ≠ parent where nbsta(A_k) = IN and visited(A_k) = false,
8          send A_k message DFT(curtok);
9       else send Report to parent;
```

When a remaining agent A_i receives from A_c the $Eliminated$ message, it responds by setting its $nbsta(A_c) = OUT$. When a remaining agent A_i receives from A_c the $DFT(tok)$ message, it performs the following.

Procedure 4 (DFT(tok))

```
1   if curtok = tok, // visited
2       send Report to A_c;
3   else // A_i has not seen tok before and has IN bordering agent other than A_c
4       curtok = tok;
5       parent = A_c; visited(A_c) = true;
6       run DFT;
```

After A_i has messaged A_j with $DFT(tok)$, it may receive a $Report$ from A_j. In response, A_i performs the following.

Procedure 5 (Respond to Report)

```
1   visited(A_j) = true;
2   if there exists A_k ≠ parent such that nbsta(A_k) = IN and visited(A_k) = false,
3       select A_k to send message DFT(curtok) to it;
4   else // no unvisited bordering agent
5       if parent = null, announce "no hypertree exists"; // DFT starter
6       else send Report to parent;
```

The algorithm suite, we refer to as HTBS, terminates when a remaining agent announces "hypertree exists" or otherwise. Its soundness and completeness is established below, which follows from Theorem 2.

Corollary 1. *A MAS with the boundary set W has a JT agent organization, iff HTBS terminates with announcement "hypertree exists". Otherwise, HTBS terminates with announcement "no hypertree exists".*

An important product of HTBS is the JT organization emerging upon positive announcement. For every agent A_i self-eliminated relative to A_j, A_i is the sender of $StartNewDFT$ message and A_j is the receiver. This relation implies that they are adjacent in the JT organization. Readers are encouraged to verify this by comparing Example 6 with Fig. 1 (e). This result is summarized below, whose proof is omitted due to space limitation.

Theorem 3. *If a MAS with the boundary set W has a JT agent organization, then agent adjacency in the JT is defined by $StartNewDFT$ sender-receiver relation during HTBS.*

Let e be the number of pairs of bordering agents. In each round of HTBS, at most $O(e)$ messages are passed. HTBS halts in at most $O(\eta)$ rounds. Hence, its time complexity is $O(e\ \eta)$.

Since HTBS is based on boundary set, agent privacy on private variables is guaranteed. Since HTBS messages contains no information on shared variables, agent privacy on shared variables is guaranteed. Since HTBS messages are passed between bordering agents only, and the message argument is a token only, privacy on agent identity is guaranteed.

7 Conclusion

The contributions of this research include the following. We proved two necessary and sufficient conditions for the existence of a JT organization given a MAS subenv decomposition. One of them provides insight and classification of subenv decompositions, and the other suggests a distributed reorganization of JT organization existence. Based on the second condition, we have presented an algorithm suite that recognize JT organization existence and construct it if exists. The algorithm guarantees agent privacy on private variables, shared variables, as well as agent identity. To the best of our knowledge, no existing JT-based MAS frameworks provide the same degree of agent privacy, except a related work based on distributed maximum spanning tree construction which we report in [13].

Our algorithm identifies correctly when no JT organization exists for a given subenv decomposition. Further research is needed for distributed revision of the subenv decomposition under the condition where no JT exists, while preserving agent privacy as much as possible.

Acknowledgement. We thank anonymous reviewers for their helpful comments. Financial support through Discovery Grant from NSERC, Canada is acknowledged.

References

1. Brito, I., Meseguer, P.: Cluster tree elimination for distributed constraint optimization with quality guarantees. Fundamenta Informaticae 102(3-4), 263–286 (2010)
2. Gmytrasiewicz, P., Durfee, E.: Rational communication in multi-agent environments. Auto. Agents and Multi-Agent Systems 4(3), 233–272 (2001)
3. Koller, D., Milch, B.: Multi-agent influence diagrams for representing and solving games. In: Proc. 17th Inter. Joint Conf. on Artificial Intelligence, pp. 1027–1034 (2001)
4. Maestre, A., Bessiere, C.: Improving asynchronous backtracking for dealing with complex local problems. In: Proc. 16th European Conf. on Artificial Intelligence, pp. 206–210 (2004)
5. Modi, P., Shen, W., Tambe, M., Yokoo, M.: Adopt: asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligences 161(1-2), 149–180 (2005)
6. Paskin, M., Guestrin, C., McFadden, J.: A robust architecture for distributed inference in sensor networks. In: Proc. Information Processing in Sensor Networks, pp. 55–62 (2005)
7. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: Proc. 19th Inter. Joint Conf. on Artificial Intelligence, pp. 266–271 (2005)
8. Valtorta, M., Kim, Y., Vomlel, J.: Soft evidential update for probabilistic multiagent systems. Int. J. Approximate Reasoning 29(1), 71–106 (2002)
9. Vinyals, M., Rodriguez-Aguilar, J., Cerquides, J.: 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)
10. Xiang, Y.: Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach. Cambridge University Press, Cambridge (2002)
11. Xiang, Y., Hanshar, F.: Multiagent expedition with graphical models. Inter. J. Uncertainty, Fuzziness and Knowledge-Based Systems 19(6), 939–976 (2011)
12. Xiang, Y., Mohamed, Y., Zhang, W.: Distributed constraint satisfaction with multiply sectioned constraint networks. accepted to appear in International J. Information and Decision Sciences (2013)
13. Xiang, Y., Srinivasan, K.: Construction of privacy preserving hypertree agent organization as distributed maximum spanning tree. In: ZaÏane, O., Zilles, S. (eds.) AI 2013. LNCS, vol. 7884, Springer, Heidelberg (2013)