A Probabilistic Framework for Cooperative Multi-agent Distributed Interpretation and Optimization of Communication

Y. Xiang

Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
E-mail: yxiang@cs.uregina.ca

To appear in *Artificial Intelligence*, fall, 1996

Abbreviated title: Probabilistic Framework for Multi-agent Systems

Abstract

Multiply sectioned Bayesian networks for single-agent systems are extended into a framework for cooperative multi-agent distributed interpretation systems. Each agent is represented as a Bayesian subnet. We show that the semantics of the joint probability distribution of such a system is well defined under reasonable conditions.

Unlike in single-agent systems where evidence is entered one subnet at a time, multiple agents may acquire evidence asynchronously in parallel. New communication operations are thus proposed to maintain global consistency. It may not be practical to maintain such consistency constantly due to the inter-agent 'distance'. We show that, if the new operations are followed, between two successive communications, answers to queries from an agent are consistent with all local evidence, and are consistent with all global evidence gathered up to the last communication.

During a communication operation, each agent is not available to process evidence for a period of time (called *off-line time*). Two criteria for the minimization of the off-line time, which may commonly be used, are considered. We derive, under each criterion, the optimal schedules when the communication is initiated from an arbitrarily selected agent.

1 Introduction

Probabilistic reasoning in Bayesian networks (BNs), as commonly applied, assumes a single-agent paradigm. That is, a single processor accesses a single global network representation, updates the joint probability distribution over the network variables as evidence becomes available and answers queries. Concurrency, as applied to BNs, primarily aims at performance and decentralization of control [15, 9], but not at modeling inference among multiple agents with multiple perspectives. The resultant individual concurrent element is thus 'fine-grained', e.g., a node in a BN [15] or a clique in the junction tree representation of a BN [9].

The single-agent paradigm is inadequate when uncertain reasoning is performed by elements of a system between which there is some 'distance', which may be spatial, temporal or semantic (elements are specialized differently) [1]. Such systems pose special issues that need to be addressed. A multi-agent view is thus required where each agent is an autonomous intelligent subsystem. Each agent holds its own partial domain knowledge, accesses some external information source and consumes some computational resource. Each agent communicates with other agents to achieve the system's goal cooperatively.

Distributed artificial intelligence (DAI) addresses the problems of designing and analyzing such 'large-grained' coordinating multi-agent systems [2, 6]. Main stream approaches in DAI, e.g., blackboard systems [5], contract nets [3] and open systems [8] are essentially logic-based. To our best knowledge, little has been reported to explore probabilistic approach in DAI.

This paper reports our pilot study on applying the probabilistic approach to cooperative multi-agent reasoning. We address the problem of distributed interpretation, a subclass of problems in DAI. As defined originally by Lesser and Erman [12], an *interpretation* system accepts evidence from some environment and produces higher level descriptions of objects and events in the environment. A *distributed* interpretation system is needed when sensors for collecting evidence are distributed, and communication of all evidence to a centralized site is undesirable. Examples of such systems include sensor networks, medical diagnosis by multiple specialists, trouble-shotting of complex artifacts and distributed image interpretation. Multi-agent systems may consist of *cooperative* or *self-interested* agents. We consider only cooperative agents in this paper.

Our representation is based on multiply sectioned Bayesian networks (MSBNs)[19], which were developed for single-agent-oriented and modular knowledge representation, and more efficient inference[18]. We demonstrate that the modularity of MSBNs allows a natural extension into a multi-agent reasoning framework. In particular, we show that the semantics of the joint probability distribution of a cooperative multi-agent system is well defined under reasonable conditions. We propose new communication operations that are used to maintain inter-agent consistency. We derive communication schedules that optimize the time efficiency of the communication.

Section 2 briefly introduces BNs and single-agent MSBNs. Section 3 presents the semantic extension of single-agent MSBNs to multi-agent MSBNs. Each cooperative agent is represented as a Bayesian subnet that consumes its own computational resource, gathers its own evidence and can answer queries. When agents are cooperative, are conditionally independent given the intersections of their subdomains and have a common initial belief on their intersections, then it is shown that a joint probability distribution of the multi-agent system is uniquely defined and is consistent with the belief of every agent in the system.

Unlike single-agent systems where evidence is entered one subnet at a time, multiple agents may acquire evidence asynchronously in parallel. Section 4 discusses consistency-related issues that arise from the extension. Section 5 adds new belief propagation operations to the set of single-agent MSBN operations for inter-agent communication. We show that agents in the system will be globally consistent after the proposed operations are performed. Inter-agent 'distance' and the associated communication cost may prevent constant maintenance of inter-agent consistency. We prove that when the proposed operations are used, between two successive communications, the answers to queries from an agent are consistent with all local evidence gathered so far and are consistent with all global evidence gathered up to the last communication. Section 6 presents an experimental demonstration how a multi-agent MSBN may be used to perform a distributed interpretation task.

During a communication operation, each agent is not available to process new evidence for a period of time (called *off-line time*). Such non-availability imposes restriction on time-critical applications. Therefore, the length of the off-line time should be minimized.

Section 7 defines two criteria for the minimization of the off-line time which may commonly be used. One is based on the total length of the off-line time for the entire multi-agent system. The other is based on the average length of the off-line time across all agents in the system. To facilitate the study of the optimal communication schedules, we abstract the activities during the communication into a graphical model, and identify the factors that can be manipulated in optimizing these schedules.

Section 8 reduces the communication scheduling into two independent subproblems and establishes the duality of the two subproblems. This result allows the optimal communication schedules be derived by solving only one of the subproblms. Section 9 derives, for each minimization criterion, the communication schedules that yield the minimum off-line time when the communication is initiated from an arbitrarily selected agent.

Section 10 discusses some general issues related to this work. Our presentation assumes a general terminology of graph theory.

2 Multiply Sectioned Bayesian Networks

2.1 Bayesian networks

A BN [15, 13, 11, 9] is a triplet (N, E, P). N is a set of nodes. Each node is labeled with a variable associated with a space. We shall use 'node' and 'variable' interchangeably. Therefore, N represents a problem domain. E is a set of arcs such that $D = (N, E)$ is a directed acyclic graph (DAG). We refer to D as the *structure* of the BN. The arcs signify directed dependencies between the linked variables. For each node $A_i \in N$, the strengths of the dependencies from its parent nodes π_i are quantified by a conditional probability distribution $p(A_i|\pi_i)$ of A_i conditioned on the values of A_i's parents. For any three sets X, Y and Z of variables, X and Y are said to be *conditionally independent* given Z under probability distribution P if $P(X|YZ) = P(X|Z)$ whenever $P(YZ) > 0$. The basic dependency assumption embedded in BNs is that a variable is conditionally independent of its non-descendants given its parents. This assumption allows P, the joint probability distribution (jpd), to be specified by the product $P = \prod_i p(A_i|\pi_i)$.

2.2 Single agent oriented MSBNs

To make the paper self-contained, we briefly introduce the single-agent oriented MSBNs [19, 18].

A MSBN M consists of a set of interrelated Bayesian subnets. Each subnet represents dependencies of a subdomain in a large problem domain or total universe. Each subnet shares a non-empty set of variables with at least one other subnet. The intersection between each pair of subnets satisfies the *d-sepset* condition.

Definition 1 (d-sepset) *Let* $D^i = (N^i, E^i)$ $(i = 1, 2)$ *be two DAGs such that* $D = (N^1 \cup N^2, E^1 \cup E^2)$ *is a DAG. The intersection* $I = N^1 \cap N^2$ *is a* `d-sepset` *between* D^1 *and* D^2 *if, for every* $A_i \in I$ *with its parents* π_i *in* D*, either* $\pi_i \subseteq N^1$ *or* $\pi_i \subseteq N^2$*.*

It can be shown that, when a pair of subnets are isolated from M, their d-sepset renders them conditionally independent. Figure 1 (left) shows the structure of a MSBN for diagnosis of Median nerve lesion (Medn), Carpal tunnel syndrome (Cts) and Plexus upper trunk lesion (Pxut).[1] It consists of three subnets D^i $(i = 1, 2, 3)$ for clinical, electromyography and nerve conduction subdomains, respectively. The d-sepset between each pair of subnets is $\{Medn, Cts, Pxut\}$. In general, d-sepsets between different pairs of subnets of M may be different.

Subnets of M are organized into a *hypertree* structure. Each hypernode is a subnet of M. Each hyperlink is a d-sepset between a pair of subnets. A hypertree structured M ensures that each hyperlink render the two parts of M that it connects conditionally independent. The subnets in Figure 1 (left) can be organized into the hypertree in Figure 1 (middle). Figure 1 (right) depicts a general hypertree structured MSBN.

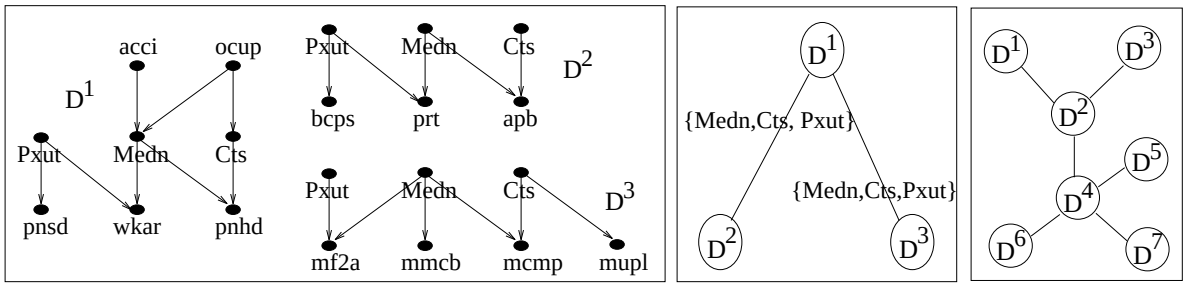

Figure 1: Left: An example MSBN for neural muscular diagnosis. Middle: The hypertree organization of the MSBN in the left. Right: A general hypertree structured MSBN.

Each subnet in M may be multiply connected (more than one path between a pair of nodes), e.g., D^1. In order to perform inference more efficiently in each subnet, the hypertree structured M is converted into a *linked junction forest* (LJF) F of the identical structure as its run time representation. Each hypernode in the hypertree is a *junction tree* (JT) (clique tree) converted from the corresponding subnet through moralization and triangulation [11, 9]. Each hyperlink in the hypertree is a set of *linkages* which covers the d-sepset between the two corresponding subnets.

The need for linkages can be understood as follows: When evidence is obtained in one subnet/JT, it can be propagated to an adjacent JT by passing the probability distribution over the d-sepset I. This may not be efficient if the cardinality of I is large. The efficiency can be improved by exploiting the conditional independence within I. Linkages form a

[1]The example is taken from a fraction of PAINULIM [18] with modification.

decomposition of I based on conditional independence. Once linkages are defined, the probability distribution over I can be passed by passing distributions over linkages, which is more efficient. We will illustrate this later in this subsection. Linkages are obtained as follows:

Definition 2 (linkage) *Let I be the d-sepset between JTs T^a and T^b in a LJF.*

First remove recursively every leaf clique C of T^a that satisfies one of the following conditions. (1) $C \cap I = \phi$. (2) $C \cap I$ is a subset of another clique. Denote the resultant graph by T'.

Then remove recursively either a node from a clique of T' or a clique from T' as follows. (a) If a node $x \notin I$ is contained in a single clique C, remove x from C. (b) If a clique C becomes a subset of an adjacent clique D after (a), union C into D.

The resultant is a `linkage tree` *$Y^{a \to b}$ of T^a relative to T^b. Each clique l of $Y^{a \to b}$ is a* `linkage` *from T^a to T^b. The clique of T^a that contains a linkage l is the* `linkage host` *of l.*

It can be shown that a linkage tree is a JT. It can also be shown that belief propagation between JTs through linkages can be performed correctly if and only if $Y^{a \to b}$ and $Y^{b \to a}$ are identical.

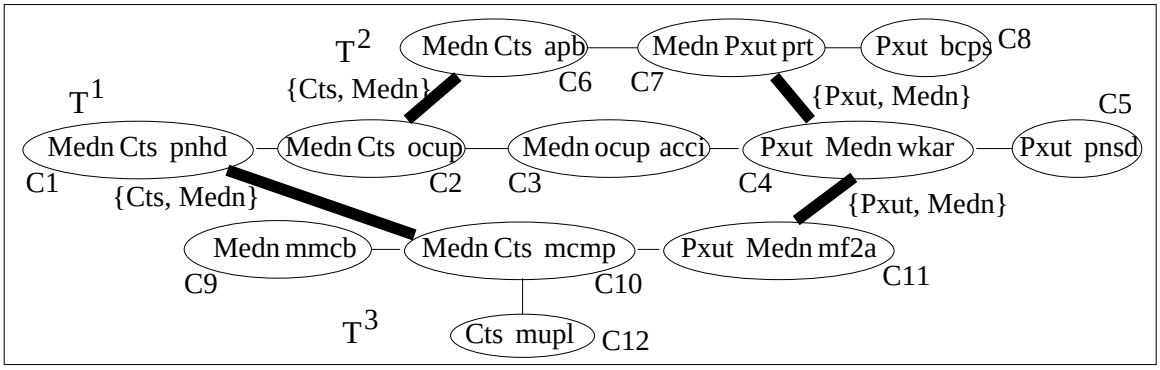

Figure 2: A linked junction forest of the MSBN in Figure 1.

The MSBN in Figure 1 (left and middle) can be converted into the LJF in Figure 2. The three subnets D^i $(i = 1, 2, 3)$ are converted into three JTs T^i $(i = 1, 2, 3)$. Then linkages (shown as heavy links) between pairs of JTs are defined. The linkage tree of T^2 relative to T^1 is obtained by first removing the clique $C8$, and then removing the variable *apb* from the clique $C6$ and removing *prt* from $C7$. We then obtained the linkage tree with two cliques $\{Cts, Medn\}$ and $\{Pxut, Medn\}$ each of which is a linkage between T^1 and T^2. Their linkage hosts in T^2 are $C6$ and $C7$, and their hosts in T^1 are $C2$ and $C4$.

Parallel to the structure conversion, the conditional probability tables stored at nodes of M are converted to *belief tables* (unnormalized probability distributions) of cliques in

JTs of F such that a *joint system belief* of F, assembled from the belief tables, is equivalent to the jpd of M. The belief table of a JT T is

$$B_T(N) = \frac{\prod_i B_{C_i}(C_i)}{\prod_j B_{S_j}(S_j)}, \tag{1}$$

where N is the set of domain variables of T, $B_{C_i}(C_i)$ is the belief table of clique C_i and $B_{S_j}(S_j)$ is the belief table of clique separator S_j. Subscripts are used to denote the object that a belief table is associated with. Let $B_I(I)$ be the belief table of a d-sepset I assembled from belief tables of linkages in the corresponding linkage tree in the similar fashion as Equation 1 (recall that a linkage tree is a JT). The joint system belief of F takes the form

$$B_F(U) = \frac{\prod_i B_{T^i}(N^i)}{\prod_j B_{I^j}(I^j)}, \tag{2}$$

where $U = \cup_i N^i$ is the total universe. Since belief tables are unnormalized probability distributions, $B_F(U)$ is proportional to the jpd of F

$$P_F(U) = \frac{\prod_i P_{T^i}(N^i)}{\prod_j P_{I^j}(I^j)} \tag{3}$$

where P denotes a probability distribution.

To answer queries by efficient local computation in F, it must be consistent. F is *locally consistent* if all JTs are internally consistent, i.e., when marginalized onto the same set of variables, different belief tables in a JT yield the identical marginal distribution. F is *boundary consistent* if each pair of adjacent JTs are consistent with respect to their d-sepset. F is *globally consistent* if it is both locally consistent and boundary consistent.

A set of operations are developed to achieve consistency during evidential reasoning: We assume that F is initially globally consistent. Details on initialization can be found in the above reference.

After evidence is entered into a JT, the JT is no longer internally consistent and F is no longer globally consistent. `UnifyBelief` brings a JT internally consistent. It is defined in terms of `DistributeEvidence` (an outward belief propagation within a JT) and `CollectEvidence` (an inward belief propagation) proposed by Jensen et al [9].

Operation 3 (UnifyBelief) *Let T be a JT in a LJF. When* `UnifyBelief` *is initiated in T, the following are performed: (1) A clique C is arbitrarily selected. (2)* `CollectEvidence` *is called in C. (3) When C has finished* `CollectEvidence`, `DistributeEvidence` *is called in C.*

For example, suppose `UnifyBelief` is performed in T^3 of Figure 2 and the clique $C9$ is selected. During `CollectEvidence`, first belief propagates from $C11$ to $C10$ and from $C12$ to $C10$, and then belief propagates from $C10$ to $C9$. During `DistributeEvidence`,

first belief propagates from $C9$ to $C10$, and then from $C10$ to $C11$ and $C12$. This brings T^3 internally consistent.

When evidence is available relative to variables in a JT, it is entered by `EnterEvidence`. `EnterEvidence` enters evidence by multiplying the belief tables of relevant cliques with the evidence function and then brings the JT internally consistent again by calling `UnifyBelief`.

For example, suppose Median motor conduction block ($mmcb = true$) is observed in the nerve conduction study of a patient. In Figure 2, the clique $C9$ contains the variable $mmcb$. During `EnterEvidence`, the belief table of $C9$ will be modified such that all configurations of $\{Medn, mmcb\}$ incompatible with the observation will be set to 0. Then `UnifyBelief` is called in $C9$.

Belief propagation between adjacent JTs in F are performed with `UpdateBelief`. It updates the belief of a JT T relative to an adjacent JT, and brings T internally consistent. It is defined in terms of a lower level operation `AbsorbThroughLinkage`. Given a linkage and its two hosts (one at each JT involved), `AbsorbThroughLinkage` updates the belief table of one host by the marginalization of the belief table of the other host over the linkage.

Operation 4 (UpdateBelief) *Let $L = \{L_1, \ldots, L_k\}$ be the set of linkages between JTs T^a and T^b. Let C_i^a and C_i^b be the linkage hosts of L_i in T^a and T^b, respectively. When* `UpdateBelief` *is called in T^a to update its belief relative to T^b, the following are performed:* `AbsorbThroughLinkage` *is called in each C_i^a to absorb from C_i^b through L_i. After each* `AbsorbThroughLinkage`*,* `DistributeEvidence` *is called in C_i^a.*

In Figure 2, suppose `UpdateBelief` is called in T^2 to update its belief relative to T^1. First, belief propagates from $C2$ to $C6$ through the linkage $\{Cts, Medn\}$ followed by `DistributeEvidence` in $C6$. Then belief propagates from $C4$ to $C7$ through the linkage $\{Pxut, Medn\}$ followed by `DistributeEvidence` in $C7$. Note that if all variables in the d-sepset $\{Medn, Cts, Pxut\}$ has three possible values, then the probability distribution over the d-sepset has 27-1=26 independent values. By exploring the conditional independence within the d-sepset, we only pass the distributions over the two linkages with 8+8=16 values.

`DistributeBelief` initiated at a JT T causes an outward belief propagation in F. If F was globally consistent before evidence is entered to T, then after `EnterEvidence` in T followed by `DistributeBelief` in T, F is again globally consistent.

Operation 5 (DistributeBelief) *Let T^i and T^j be two adjacent JTs in a LJF. When* `DistributeBelief` *is called in T^i by T^j, the following are performed: (1) T^i updates its belief relative to T^j by* `UpdateBelief`*. (2) T^i calls* `DistributeBelief` *in all adjacent JTs except T^j.*

In Figure 2, suppose T^1 has acquired new clinical evidence. When `DistributeBelief` is called in T^1 (the caller is the system and only the step (2) is performed), it calls T^2 and T^3 to `DistributeBelief`. The two JTs will then update their belief relative to T^1.

In single agent MSBNs, `DistributeBelief` is only needed for initialization. Global consistency during evidential reasoning is maintained by a more efficient operation `ShiftAttention`. After the user has entered multiple pieces of evidence into a JT, `ShiftAttention` allows the user to shift attention to another target JT. It maintains consistency along the hyperpath in the hypertree from the current JT to the target JT.

Operation 6 (ShiftAttention) *Let $T^0, T^1, \ldots, T^j$ be a subset of JTs in a LJF that form a simple path in the hypertree from T^0 to T^j. When* `ShiftAttention` *is called to shift attention from T^0 to T^j, for $i = 1$ to j,* `UpdateBelief` *is called in T^i to update its belief relative to T^{i-1}.*

In Figure 1 (right), suppose the user currently focuses his attention on D^1/T^1.[2] If he wants to shift attention to D^3, `ShiftAttention` will propagate belief from D^1 to D^2 and then to D^3. Note that subnets $D^4, \ldots, D^7$ are not computed during this `ShiftAttention`, hence the efficiency over `DistributeBelief`.

A user may start with a particular JT, enter some evidence, query the subnet, shift attention to another JT, and repeat these actions for a number of times. Note that, in such a single agent context, evidence is always entered into the current JT. It can be shown that, after `ShiftAttention`, the target JT is aways consistent at the *global level* in the sense that answers to queries provided by the JT is consistent with all the evidence entered so far in the entire LJF. We will come back to this point in Section 4.

3 Representing Multiple Agents in MSBNs

In this section, we extend the single agent MSBNs to *cooperative* and *homogeneous* multi-agent systems for distributed interpretation and consider the semantics of such a system.

3.1 The semantics of joint system belief

As described in Section 2, a MSBN represents a large problem domain by representing each subdomain with a subnet. From the viewpoint of reasoning agents, a MSBN represents the *coherent* multiple perspectives of a single agent. For example, PAINULIM [18] consists of three subnets which represents a neurologist's three different perspectives of the neuromuscular diagnostic domain: clinical, EMG and nerve conduction perspectives. The jpd of the MSBN represents the subjective belief of the single agent.

[2]The diagram in fact shows the MSBN M, not the LJF F. We abuse the illustration a bit since M and F share the same hypertree structure.

In a multi-agent system, each agent can be considered as holding its own perspective of the domain. This partial perspective may be over a specialty, over a period of time, or over a spatial area. The modular representation of MSBN allows a natural extension to multi-agent systems, with a modification of the semantics: Instead of representing *one* agent's *multiple* perspectives of a domain, a multi-agent MSBN represents *multiple* agents in a domain each of which holds *one* distinct perspective of the domain. Each subnet corresponds to one such perspective.

A natural question arises: What is the interpretation of the jpd of such a system? Whose belief does it represent? We will first discuss this issue intuitively and then justify our interpretation formally.

Consider a computer system. It processes information coherently as a whole, even though its components are commonly supplied by different vendors. This coherence is achieved since each vendor follows a set of protocols in designing the functional *interface* of a component. As long as the interface follows a common protocol, a vendor has the freedom to determine the internal structure of a component and the entire system will function as if it follows a single will. How much knowledge is necessary to the designer of the system? He only needs to know the functional interfaces of components and not their internal structures. In a sense, the system is built by a *group* of designers including all vendors who supply components as well as the system designer. Building complex systems in such a way has become a common practice. Procedural abstraction and data abstraction are commonly applied to develop complex software systems by team work [7]. Layered approach is commonly used in operating systems [17] and computer networks [16].

Next consider a human 'system' consisting of a patient and a family doctor. Suppose that the patient has *no* medical knowledge of his problem and he trusts the doctor's expertise *completely*. Suppose that the doctor is also giving the best diagnosis and treatment he can. When they meet, the patient will tell all that the doctor needs to know for diagnosis. After the doctor reaches a diagnosis, he will prescribe a therapy which the patient will follow. Even though the doctor does not experience the symptom himself and the patient does not understand how the diagnosis is reached, the system as a whole demonstrates a coherent belief on symptoms (the doctor uses to reach the diagnosis) and the diagnosis (the patient follows the therapy). Situations like this are not uncommon when a user is seeking advice from a specialist. Who is the designer of this system? It's the demand and supply (of medical expertise).

The above two scenarios illustrate that, under certain conditions, a system consisting of different agents may demonstrate a coherent joint belief or will consistent with that of each individual agent. Clearly one of the conditions is that agents are *cooperative*. An agent must trust the information supplied by others and must also supply others with what he really believes. This is possible if all agents in the system are working towards a common goal (vs self-interested).

Another condition is *conditional independence*. It is not necessary for each agent to

supply others with *all* that he believes. A component in a complex system only needs to pass to other components the information specified in the protocol, and it can and should hide other details regarding how the supplied information is obtained. In structured programming, a procedure header only specifies the input and output parameters. How the mapping from input to output is performed needs not be concerned by the caller of the procedure. A doctor only needs to inform the patient of the diagnosis and the therapy. He does not need to explain how the diagnosis is reached. In general, to a particular agent engaged in a particular task, there is usually a certain amount of information from other agents, once exchanged, that is sufficient to help the agent to perform its own task. Beyond that amount, the information about how other agents think is irrelevant, namely, the agent is conditionally independent of other agents conditioned on that certain amount of information.

Let us formalize the above idea by first introducing a third condition.

Definition 7 *Let $N = A \cup B$ be a problem domain such that $A \cap B \neq \phi$. Let $Q(A)$ and $R(B)$ be probability distributions over A and B. $Q(A)$ and $R(B)$ are said to be* `consistent` *if $\sum_{A \setminus B} Q(A) = \sum_{B \setminus A} R(B)$, where the summation represents marginalization.*

In other words, $Q(A)$ and $R(B)$ are consistent if they yield the same distribution when marginalized to $A \cap B$.

The following lemma is due to Dawid and Lauritzen. We reformulated in our notation.

Lemma 8 *[4] Let $N = A \cup B$ be a set of variables. Let $Q(A)$ and $R(B)$ be probability distributions over A and B and let them be consistent. Then there exists a* unique *probability distribution*

$$P(N) = Q(A)R(B|A \cap B) \text{ whenever } R(A \cap B) > 0$$

such that (1) $\sum_{N \setminus A} P(N) = Q(A)$, (2) $\sum_{N \setminus B} P(N) = R(B)$ and (3) A is conditionally independent of B given $A \cap B$ under P.

Now let α and β be two cooperative agents. Suppose α can only perceive the subdomain A and β can only perceive the subdomain B. Let the subjective belief of α be represented by $Q(A)$ and that of β be represented by $R(B)$. Suppose knowing the other agent's belief on the intersection $A \cap B$ is sufficient to coordinate the tasks of α and β. Suppose $Q(A)$ and $R(B)$ are consistent, i.e., the two agents share the same belief $Q(A \cap B) = R(A \cap B)$ on the intersection $A \cap B$. Then, according to Lemma 8, there exists a unique probability distribution $P(N)$ that is consistent with both $Q(A)$ and $R(B)$ and that it satisfies the conditional independence of A and B conditioned on $A \cap B$.

Coming back to our extension of MSBNs to multi-agent systems, suppose we form a MSBN M with the two agents α and β. Suppose M consists of two subnets S^α and S^β over the subdomains A and B such that their d-sepset is $A \cap B$. Now the distribution of S^α corresponds to α's belief and the distribution of S^β corresponds to β's belief. The

boundary consistency of M corresponds to the consistency of the two agents' belief. When M is globally consistent, the jpd defined by Equation 3 is identical to $P(N)$ in Lemma 8.

Dawid and Lauritzen have generalized Lemma 8 to the case of more than two distributions. We reformulated in our notation as follows:

Theorem 9 *[4] Let N be a set of variables. Let T be a junction tree and C_i be a clique of T such that $\cup_i C_i = N$. Let $Q_{C_i}(C_i)$ be the probability distribution over the clique C_i such that distributions for each pair of adjacent cliques in T are consistent. Let S_j be a clique separator in T and $Q_{S_j}(S_j)$ be the distribution over S_j computed from the distribution of any one of its adjacent cliques. Then there exists a* unique *probability distribution*

$$P_T(N) = \frac{\prod_i Q_{C_i}(C_i)}{\prod_j Q_{S_j}(S_j)}$$

such that (1) for each clique C_i, $\sum_{N \setminus C_i} P_T(N) = Q_{C_i}(C_i)$ and (2) for each pair of adjacent cliques C_i and C_j with their separator S_k, C_i is conditionally independent of C_j given S_k under P_T.

According to Theorem 9, if we organize a set of *cooperative* agents into a MSBN such that adjacent agents in the hypertree are *conditionally independent* and *consistent*, then the jpd of the MSBN defines a coherent joint belief among all agents. This implies that a multi-agent MSBN can be constructed by multiple developers. Each developer builds one computational agent based on its own expertise in a subdomain. The theory of MSBNs provides the guidance as how the agents should be connected, i.e., the intersection between subdomains should be conditionally independent, the interface of subnets should be a d-sepset, the overall organization should be a hypertree, etc., and how belief propagation should be performed.

3.2 MSBNs ensure disciplined communication

One might still wonder what difference a multi-agent MSBN makes compared to a same set of agents without organized into a MSBN. Can each agent cooperate with others by sending messages and treating messages received as evidence?

It should be clear that the belief propagation in MSBNs performed by `UpdateBelief` is in fact message passing. The messages are the probability distributions over linkages. However, message passing in a MSBN is *disciplined*.

First, the distribution on the entire d-sepset between a pair of subnets, in the form of belief tables over all linkages, must be passed each time. Passing less information is not allowed since it would not be sufficient to inform the other agent such that a coherent joint system belief can be warranted. Passing more information is also not allowed since it is useless.

More importantly, messages in a MSBN must flow along the hypertree in a regulated fashion as in `DistributeBelief` and `ShiftAttention`. Otherwise, the following sequence

of events is possible. Initially an agent α may send a message to an agent β based on a piece of evidence. After updating its belief, β may send a message to an agent γ. After updating its belief, γ may send a message to α. Not knowing the message from γ is based on the same evidence originated from α, α will update its belief and count the same evidence twice. Such circular evidence propagation causes *no* problem if all agents are *deterministic* or *logical*. However, it will create *false* belief with no evidential support if agents' knowledge is *uncertain* or *probabilistic*. The problem of circular evidence propagation in probabilistic reasoning is discussed in [15]. The hypertree structure of MSBNs and the way `DistributeBelief` and `ShiftAttention` operate ensures that no circular evidence propagation occur among agents.

3.3 Illustration of multi-agent MSBNs

We use two examples to illustrate the above general discussion.

First, let us consider monitoring or troubleshooting a complex artifact system as shown in Figure 3. The system is made of five subsystems $U0$ through $U4$. The external input variables of the system are a, b, c, h, m, r and w, and the external output variables are u, v, y and z, as shown in the figure. Suppose subsystems are manufactured by different vendors. Each vendor may build a computational agent that encodes the knowledge of the functional and the faulty behavior of parts and of the internal structure of the subsystem. Each agent is capable of monitoring or troubleshooting the corresponding subsystem. To monitor the entire artifact system, we can form a multi-agent system and let these agents cooperate.

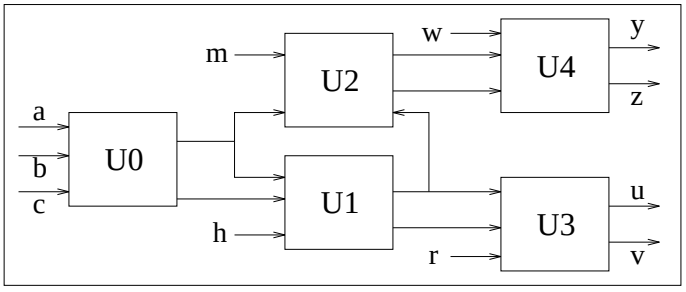

Figure 3: An artifact consisting of five subsystems.

Suppose external inputs are independent of each other. Then each computational agent is independent of other agents given the variables that connect the agent to others (not labeled in Figure 3). Finally, agents must agree on a prior distribution over their interfacing variables. We assume there is no feedback between subsystems as is the case in this example. Since $U1$ receives input from $U0$, the agent for $U1$ simply accepts the prior distribution of the input variables set by the agent for $U0$. Since $U2$ receives input

from both $U0$ and $U1$, the agent for $U2$ simply accepts the prior distribution of the input variables set by the other two agents. Now all the semantic conditions required (cooperative, conditional independent and consistent) are met and we can organize the agents into a MSBN.

The above illustration is independent of the particular application domain of an artifact system. To make the example more concrete, we fill each box of Figure 3 with a digital circuit as in Figure 4. Digital circuits are used simply because the reader's knowledge to understand the example can be safely assumed. Note that, in integrating the MSBN, only the knowledge of the interface of each circuit as shown in Figure 3 is needed and the knowledge of the internal structure of each circuit is not necessary. Furthermore, although the function of each gate is commonly defined, its faulty behavior may vary from vendor to vendor. For example, $U1$ and $U2$ may be supplied by different vendors. An AND gate in $U1$ may have the stuck-at-0 faulty behavior, but an AND gate in $U2$ may output correctly 40% of time when it is faulty. The vendor of each circuit, not the designer of the artifact system, is in the best position to encode such knowledge. If a circuit from a vendor is replaced by another with the same functional interface but from a different vendor, we simply replace the corresponding agent without disturbing the rest of the MSBN. The new MSBN will still perform inference coherently. Figure 5 (left) shows the five computational agents in the form of Bayesian subnets for the five circuits in Figure 4. Figure 5 (right) shows the hypertree organization of the MSBN. Who is the designer of the MSBN? There is a group of them including those who build agents and the one who integrates agents. Such a multi-agent system may be used by multiple users each interacting with one computational agent. The interaction consists of local evidence entering and queries.

Our second example extends that by Lauritzen and Spiegelhalter [11] for a single agent system:

Example 10 Dyspnoea (δ) may be due to tuberculosis (τ), lung cancer (ι) or bronchitis (β). A recent visit to Asia (ν) increases the chances of τ, while smoking (ζ) is a known risk factor for both τ and ι. After an initial diagnosis based on the above information, to further discriminate between τ and ι, a clinician may request lab tests from a radiology lab and a biology lab. Radiology lab has two relevant tests for τ and ι: X-ray (χ) and laminagraphy (α). Biology lab has two relevant tests as well: sputum test (ρ) and biopsy (o).

This fictitious example involves three medical subdomains: clinical, radiological and biological. In order to diagnose a patient with dyspnoea, expertise from all three subdomains may be needed (cooperative). A human decision maker often needs assistants to help gather relevant information from other decision makers and to help make decisions. Each medical practitioner in a subdomain (either a doctor or a radiologist or a biologist) may be assisted by a computational agent (a BN) specialized on the subdomain helping him or her diagnose and communicate with other practitioners during diagnosis. As we

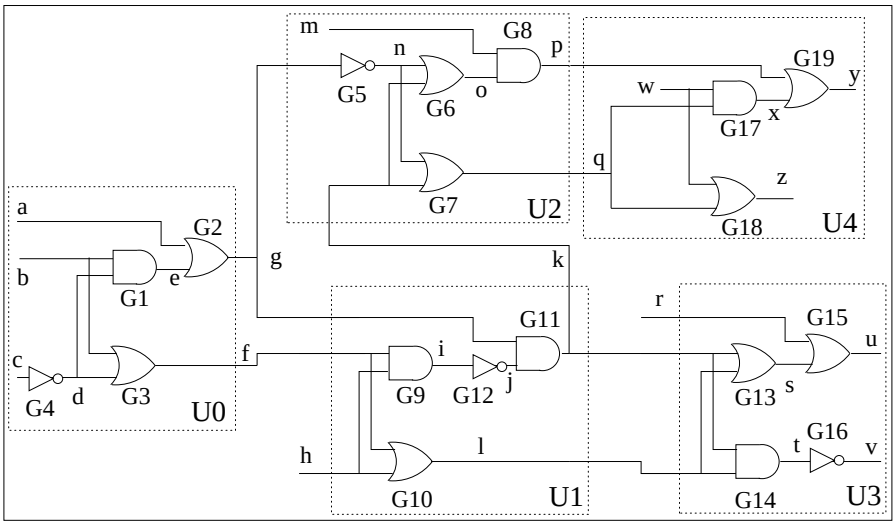

Figure 4: A digital system consisting of five circuits.

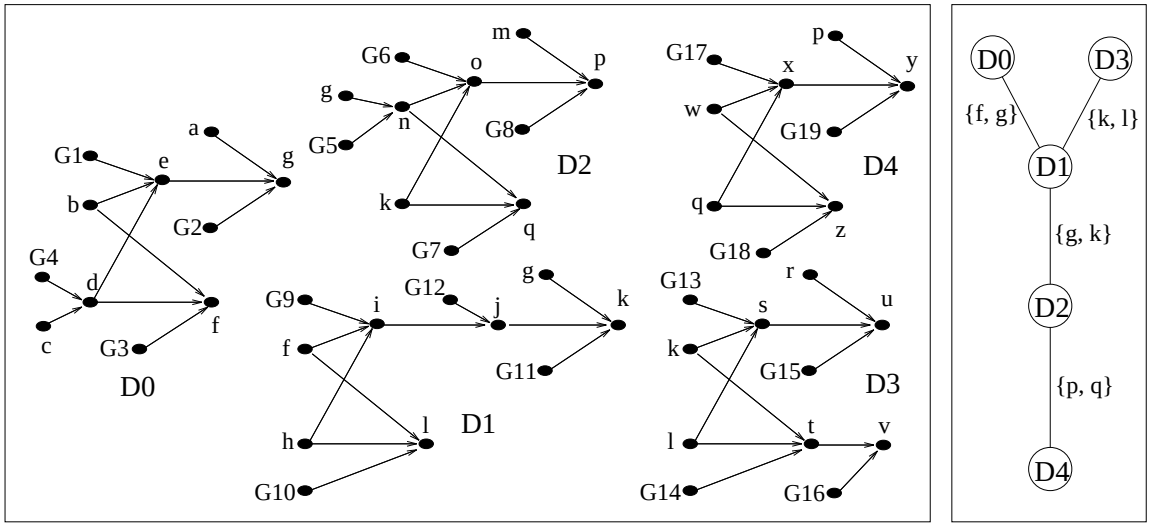

Figure 5: Left: The computational agents in the form of Bayesian subnets for the five circuits in Figure 4. Right: The hypertree organization of the five agents as a MSBN.

argued in Section 3.2, these agents should be organized as a MSBN to ensure the coherent communication. Each individual agent may be developed by an independent developer. The MSBN may be integrated before any diagnosis is to be performed. This is similar to the digital system example. It is also possible that an agent may demand assistance from a pool of potentially cooperative agents such that a MSBN is formed on the fly. This is the approach taken in the contract net [3]. However, as is proposed and commonly applied, the contract net does not have a mechanism to perform probable reasoning coherently. Multi-agent MSBNs will provide such a mechanism, although the dynamic formulation of a MSBN is beyond the scope of this paper.

Figure 6 shows an example MSBN for the three medical subdomains mentioned above. In constructing the example, we have assumed that only other agents' belief on the two diseases matters to an agent in interpretating its own evidence (conditional independence). We also assume that, prior to observation of a patient, all three agents share the same belief on the likelihood of tuberculosis and lung cancer. That is, the three subnets have the same prior probability distribution over the two diseases (consistent). Therefore, the d-sepset between pairs of subnets is $\{\tau, \iota\}$. Each subnet encodes the knowledge how evidence in the corresponding subdomain should be used in diagnosing tuberculosis and lung cancer. Note that the evidence from different subdomains may be conflicting. For example, the result of X-ray may be positive, suggesting lung cancer to the radiologist, but the result of biopsy my be negative, suggesting the opposite to the biologist. Each subnet will interpret its own evidence according to its encoded knowledge. By communicating with other subnets using the belief propagation operations of MSBNs (some new operations are needed and will be introduced in Section 5), the three subnets will come to a coherent diagnosis.

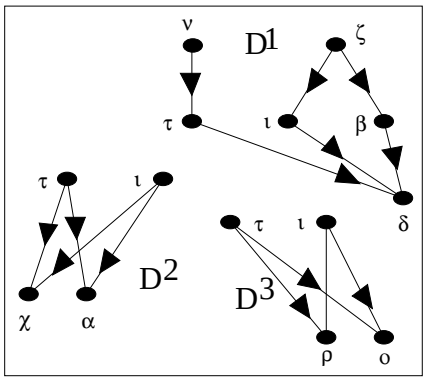

Figure 6: A multi-agent MSBN representing three medical specialists diagnosing a patient with dyspnoea. D^1: clinical subnet, D^2: radiological subnet, D^3: biological subnet.

The LJF for the MSBN is shown in Figure 7. Note that, for this example, only a single linkage is created between a pair of JTs, which is not the general case (compare with Figure 2). Note also that, the three agents are not only semantically different, but

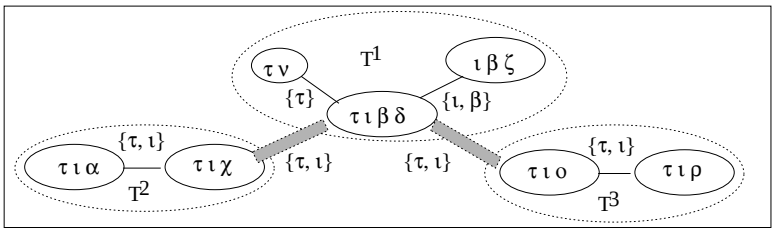

Figure 7: The LJF for the multi-agent MSBN in Figure 6. Separators between cliques are shown in solid lines. Linkages between JTs are shown in dotted bands.

may also be remotely located and must communicate through a computer network.

4 Consistency Issues in Multi-agent MSBNs

The semantic extension of MSBNs to multi-agent systems implies that all the technical constraints applicable to the construction of single-agent MSBNs must be followed in the construction of multi-agent MSBNs. In addition, evidential reasoning in multi-agent systems raises new issues regarding consistency, which must be addressed. To appreciate these issues, we use Figure 1 (right) to review how consistency is maintained in single-agent MSBNs.

For a single-agent MSBN, evidence always comes towards the subdomain that the single user is currently focusing on. The corresponding subdomain or subnet is called *active*. Suppose the user of the MSBN in Figure 1 (right) focuses attention on the subnet D^1, then D^3, and then D^5. The user enters some evidence into each of the subnets as it is active. Consistency of the MSBN can be maintained using either `DistributeBelief` or `ShiftAttention` (Section 2.2).

Suppose `DistributeBelief` (Operation 5) is used. As soon as the user has entered evidence to D^1 and wants to activate D^3, `DistributeBelief` can be called in D^1. It propagates new evidence to the entire MSBN. When D^3 is made active afterwards, it is up-to-date. Similarly, after the user has entered evidence to D^3 and before he activates D^5, `DistributeBelief` can be called in D^3. It brings all subnets up-to-date including D^5.

`DistributeBelief` always brings the entire MSBN globally consistent before a new subnet is made active, which may not be necessary. Alternatively, `ShiftAttention` only ensures the subnet that the user shifts attention to be up-to-date, since *no* evidence will be entered elsewhere and *no* query will be issued in any other subnet. This is achieved by propagating evidence only along the hyperpath in the hypertree MSBN from the currently active subnet to the next active subnet. For the above example, during the attention shift from D^1 to D^3, belief propagates from D^1 to D^2 and then to D^3. During the attention shift from D^3 to D^5, belief propagates from D^3 to D^2 to D^4 and then to D^5. It can be shown [19] that, as far as the target subnet (first D^3 and then D^5) is concerned, its belief

state after `ShiftAttention` is identical to that after `DistributeBelief`. The answers to queries at the target subnet is consistent with the evidence acquired in the entire MSBN, namely, the target subnet is consistent at *global level* even though the entire MSBN is *not* globally consistent. Since `ShiftAttention` only maintains *hyperpath consistency* and hence requires less computation than `DistributeBelief`, it is alway preferred in single-agent MSBNs.

4.1 How to regain global consistency?

The fact that `ShiftAttention` as well as `DistributeBelief` are sufficient to maintain consistency depends directly on the fact that evidence is always entered at the current subnet and nowhere else. This can be understood by noting that both operations propagate evidence from the current subnet (the source of new information) outward. In a multi-agent system, multiple agents may acquire evidence asynchronously in parallel. Since source of new information are now scattered throughout the hypertree, none of the two operations can be used to ensure global consistency any more. Belief propagation must follow a different process which we shall refer to as *communication*. We propose the corresponding operations and prove their properties in Section 5.

4.2 What is the consistency level between communications?

In a single-agent MSBN, after `ShiftAttention` the newly active subnet is consistent at a global level. If `EnterEvidence` (Section 2.2) is performed subsequently to enter evidence to the subnet and to bring it internally consistent, then the subnet is still consistent at a global level, i.e., answers to queries is consistent to evidence acquired in the entire MSBN.

In a multi-agent MSBN, the situation is quire different. After evidence is entered into two different subnets, neither has the knowledge of the evidence entered into the other subnet. Due to the inter-agent 'distance' and the associated communication cost, we may not be able to perform communication frequently enough to maintain global consistency constantly. A question that must be answered is, between two successive communications which regain global consistency, what is the consistency level of each agent after additional evidence is acquired? We prove a theorem to answer this in Section 5.

5 Maintaining Consistency through Communication

5.1 Added Operations for Regaining Global Consistency

As discussed in Section 4, parallel evidence acquisition at multiple agents renders invalid the single-agent MSBN operations for maintenance of global consistency. To regain consistency in a multi-agent MSBN, we extend the inward-outward belief propagation method in a single JT [9, 10] to the LJF of a MSBN. Jensen et al's method propagates belief

through a single information path (a unique path exists between any two cliques in a JT). Belief propagation in a LJF must be performed over multiple linkages. Fortunately, the latter problem has been solved in single-agent MSBNs with the operation `UpdateBelief` (Section 2).

Following this idea, we add two new operations `CollectNewBelief`[3] and `CommunicateBelief`[4]. `CollectNewBelief` causes an inward belief propagation in a LJF. `CommunicateBelief` calls `CollectNewBelief` and `DistributeBelief` to propagate evidence obtained from multiple agents (JTs) inward first and then outward to the entire LJF.

Operation 11 (CollectNewBelief) *Let T be a JT in a LJF. Let* `caller` *be either the LJF or an adjacent JT in the hypertree. When* `CollectNewBelief` *is called in T, the following are performed:*

1. *T calls* `CollectNewBelief` *in all adjacent JTs except* `caller` *if* `caller` *is a JT.*
2. *After each JT being called has finished* `CollectNewBelief`*, T updates its belief with respect to the JT by* `UpdateBelief`*.*

`CollectNewBelief` *is associated with JTs.*

Operation 12 (CommunicateBelief) *When* `CommunicateBelief` *is initiated at a LJF F, the following are performed:*

1. *A JT T in F is arbitrarily selected.*
2. `CollectNewBelief` *is called in T.*
3. *When T has finished* `CollectNewBelief`*,* `DistributeBelief` *is called in T.*

`CommunicateBelief` *is associated with the LJF.*

Example 13 Figure 8 shows how belief propagates through a LJF during `CommunicateBelief`. Each node corresponds to an agent in the system. The link between two adjacent agents corresponds to the set of linkages between them. Suppose the operation is initiated at an arbitrarily selected agent T^1. `CollectNewBelief` proceeds by first propagating control from T^1 towards terminal agents along solid arrows, and then propagating belief from terminal agents back to T^1 along dotted arrows. Then `DistributeBelief` proceeds by propagating belief from T^1 towards terminal agents along solid arrows. The time required for control propagation can usually be ignored compared with that for belief propagation.

[3]The operation `CollectBelief` [19] is similar in form to `CollectNewBelief`. But `CollectBelief` deals with a simpler consistency problem and can only be used for initialization. It is thus computationally less expensive than `CollectNewBelief`.

[4]The operation `BeliefInitialization` [19] is similar in form to `CommunicateBelief`. But the former deals with a simpler consistency problem (initialization). It is thus computationally less expensive than `CommunicateBelief`.

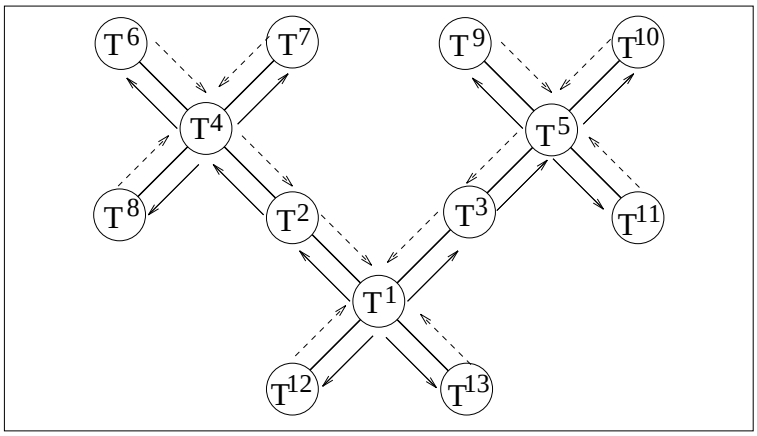

Figure 8: Belief propagation during `CommunicateBelief` in a LJF. Each node represents a JT. Each link between two nodes represents the set of linkages between the JTs. The operation is initiated from T^1. Dotted arrows illustrate belief propagation during `CollectNewBelief`, and solid arrows illustrate belief propagation during `DistributeBelief`.

Note that `CommunicateBelief` is operationally 'semi-parallel'. It is 'parallel' in that T^4 and T^5 may perform `UpdateBelief` relative to their terminal neighbors in parallel during `CollectNewBelief`. It is 'semi'-parallel in that T^4 must perform `UpdateBelief` relative to its terminal neighbors in sequence during `CollectNewBelief`. The same can be said on `DistributeBelief` as well.

5.2 Consistency After and Between Communications

We answer the two questions raised in Section 4. First, we show that after `CommunicateBelief` is performed, the multi-agent MSBN is globally consistent.

Theorem 14 (Multi-agent consistency) *Let F be a globally consistent LJF* [5] *converted from a MSBN of a hypertree structure. Let Z be a subset of JTs of F. After the following operations, F is globally consistent.*

1. *For each JT in Z, use* `EnterEvidence` *to enter finite pieces of evidence into the JT.*
2. *Use* `CommunicateBelief` *to communicate belief among JTs in F.*

[5] To be more precise, F should also be supportive and separable [19]. We will not go into that level of technical detail here.

Proof:

By assumption F is globally consistent before any `EnterEvidence`. We convert F conceptually into a JT Υ and prove the theorem using Υ. For each JT T^i of F with the domain N^i, union all its cliques into one huge clique denoted by W^i. Note that $W^i = N^i$. Let the nodes of Υ be those huge cliques. For each pair of adjacent JTs T^i and T^j of F in the hypertree, connect W^i and W^j in Υ and denote their separator by S^{ij}. Note that $S^{ij} = I^{ij}$ where I^{ij} is the intersection of T^i and T^j in F. The resultant graph Υ is a JT since F has a hypertree structure.

Assign to each clique W^i in Υ a belief table $B_{W^i}(N^i) = B_{T^i}(N^i)$. Assign to each separator S^{ij} a belief table $B_{S^{ij}}(I^{ij}) = B_{I^{ij}}(I^{ij})$. The resultant Υ is a consistent JT whose joint system belief is proportional to the joint system belief of F.

`EnterEvidence` in T^i of F corresponds to multiplying $B_{W^i}(N^i)$ in Υ by the evidence function. It updates the joint system belief and causes inconsistency. `CommunicateBelief` consists of the selection of a JT T^0 and a `CollectNewBelief` followed by a `DistributeBelief`. The `CollectNewBelief` in F corresponds to a `CollectEvidence` in Υ called in W^0. The `DistributeBelief` in F corresponds to a `DistributeEvidence` in Υ. The resultant Υ is again consistent.

After each `EnterEvidence`, the JT of F involved is internally consistent. Both `CollectNewBelief` and `DistributeBelief` call `UpdateBelief` which maintains the internal consistency of the JT involved. Therefore, after `CommunicateBelief`, F is locally consistent. The consistency of Υ implies boundary consistency of F. Hence F is globally consistent. □

`CommunicateBelief` involves both local computation at each agent and information exchange among multiple agents over 'distance'. Due to the cost involved `CommunicateBelief` may not be performed frequently. Therefore, each agent may acquire multiple pieces of evidence between two successive `CommunicateBelief` operations and may have to answer queries before the next `CommunicateBelief` can be performed. The second question we address is: what is the consistency level of these answers to queries. Theorem 15 shows that, between two successive communications, a JT is consistent with all local evidence acquired so far and is consistent with all global evidence acquired up to the last communication. This is the best that one can expect.

Theorem 15 (Semi-up-to-date) *Let F be a LJF converted from a MSBN of hypertree structure. Let Z be a subset of JTs of F.*

After a `CommunicateBelief` *in F followed by a finite number of* `EnterEvidence` *to each JT in Z, the marginal distributions obtained in a JT $T \in Z$ are identical as would be obtained if only the* `EnterEvidence` *operations in T were performed after the* `CommunicateBelief`.

Proof:

We shall refer to the `CommunicateBelief` mentioned in the theorem as the *first* `CommunicateBelief`. After the operation, F is globally consistent by Theorem 14. Among

all the `EnterEvidence` operations performed in Z, only those performed in T change the BTs in T, and none of the other operations has any effect on T. Suppose none of the `EnterEvidence` operations performed outside T was ever performed. We show that if a second `CommunicateBelief` is called in F with T as the initiating JT, the BTs of T will not change at all:

`CollectNewBelief` called in T does not change BTs in T since F is boundary-consistent after the first `CommunicateBelief`. That `DistributeBelief` does not change BTs in T is trivially true. □

6 An Experimental Demonstration

We demonstrate how communication works in a multi-agent distributed interpretation system with the digital system example in Figure 4. Recall that each agent (a subnet in Figure 5) may be built by a different vendor who encodes the knowledge about a subsystem (one circuit) into the agent. The designer of the MSBN only need the knowledge as depicted in Figure 3.

In simulating the cooperation in the five agent MSBN, we use the following arbitrarily chosen numeric parameters: For each external input (a, b, c, h, m, r, w), the prior probability is 0.5. For each gate, the prior probability that it is faulty is 0.01. For each AND gate, it produces the correct output 20% of time when it is faulty. For each OR gate and each NOT gate, the corresponding percentages are 70% and 50%, respectively. This uniform assignment of numeric parameters is not necessary but used simply for convenience.

For all simulations, we assume the following external input: $a = 0, b = 1, c = 0, h = 1, m = 1, r = 0$ and $w = 0$. We assume that the gates $G9$ in $U1$ and $G7$ in $U2$ are faulty and produce incorrect output, and every other gate is normal. This allows us to generate a complete description about the state of the system.

We assume that each agent only has a limited access to this true state of the world. First, each agent can only access the values (0 or 1) of input/output variables of gates within its subdomain with an exception that the value of variable k is *not* accessible to any agent. The values (*normal* or $faulty$) of gate variables are not accessible to any agent. For example, the agent $U2$ can only access the values of g, m, n, o, p and q but not values of $G5, G6, G7$ and $G8$. This assumption simulates the fact that an interpretation system cannot directly observe everything in the world but has to infer from the available evidence.

Second, the values of i/o variables are revealed to an agent sequentially in a random order. The agent has no control over this order. For instance, in one simulation, the agent $U2$ may receive the evidence in the order $o = 1, g = 1, p = 1, m = 1, \ldots$. In another simulation, it may receive the evidence in the order $p = 1, o = 1, q = 0, m = 1, \ldots$.

Three sets of simulations were run using the probabilistic reasoning environment WEBWEAVR-II, an extended version of WEBWEAVR [18]. The first set had no commu-

nication between agents. The second set had constant communication and the third set had infrequent communication between agents. We will compare the results from the first two sets to demonstrate the effect of cooperation among agents through the operations defined in Section 5. We will compare the results from the last two sets to demonstrate the effect of infrequent communication analyzed in Theorem 15.

6.1 No communication

In the first set of ten simulations, each agent 'observed' its subdomain and inferred the values of gates. No communication with other agents was performed. The goal of agents was to identify the faulty gates correctly as soon as possible. Since only $U1$ and $U2$ contained the faulty gates, we focused on these two agents.

For each agent, we first determined the belief state that it may reach when it had received all the information available from its subdomain. This state represents the best interpretation the agent can come up with given that it only works alone. When the complete set of observations was entered to the agent $U1$, it had the posterior probabilities $p_1(G9 = faulty|evi) = 1$, $p_1(G10 = normal|evi) = 0.993$, $p_1(G11 = normal|evi) = 0.99$ and $p_1(G12 = normal|evi) = 0.995$. Subscripts are used to identify the agent whose belief are expressed by the probability values. Call this belief state the *ideal* state of the agent. When the complete set of observations was entered to the agent $U2$, the ideal state consisted of $p_2(G7 = faulty|evi) = 0.6$, $p_2(G6 = faulty|evi) = 0.4$, $p_2(G8 = faulty|evi) = 0.002$ and $p_2(G5 = faulty|evi) = 0.005$. The agent detected that observations did not confirm to the normal function of the circuit, but it was uncertain as to whether $G6$ or $G7$ was faulty since a key variable k was not observable.

We then randomly generated ten sequences of observations for each of the two relevant agents and performed ten simulations. In each simulation, one sequence for an agent was used. For each sequence, we entered observations one by one using `EnterEvidence`. As soon as the agent reached the ideal belief state within a range of 10% of fluctuation, we stopped and recorded the number of observations that had been entered. The results are listed in columns 2 and 3 of Table 1. For the ten simulations, the average number of observations needed for the agent $U1$ to reach a belief state close to the ideal is 5.4 observations. The corresponding number for $U2$ is 4.0 observations. Since each observation is processed with some cost, i.e., time or other resources, the average number of observations gives an indication of the efficiency when the agent has to perform the interpretation task alone.

6.2 Constant communication

In the second set of ten simulations, we generated randomly ten sequences of observations for each of the five agents. In each simulation, one sequence for an agent was used. We entered observations from $U0$ to $U4$, one observation for each agent. Then

`CommunicateBelief` was performed. We then entered the next observation for each agent and repeated this process. An observation for a variable shared by two or more agents was entered only once. After each `CommunicateBelief`, we check the belief state of $U1$ and $U2$. We recorded the number of observations each agent had made when either $U1$ or $U2$ reached a belief state close to the ideal.

The ideal state for $U1$ was the same as before, but the ideal state for $U2$ was changed to $p_2(G7 = faulty|evi) = 0.999$ and $p_2(G6 = faulty|evi) = 0.007$. This much sharpened belief was obtained by agents' pooling together evidence about k, which was not possible when $U2$ was working alone.

The results of the simulations are listed in columns 4 and 5 of Table 1. For the ten simulations, the average number of observations per agent needed for $U1$ to reach a belief state close to the ideal is 4.26 observations. The corresponding number for $U2$ is 2.84 observations. The average number decreased by 21% and 29% for $U1$ and $U2$ due to cooperation.

6.3 Infrequent communication

In the last set of ten simulations, exactly the same observation sequences in the second set of simulations were used. The same experimental method was used except that `CommunicateBelief` was performed after each agent had made two observations instead of one.

Each agent still reached the ideal belief state after enough observations had been made. The results of the simulations are listed in columns 6 and 7 of Table 1. For the ten simulations, the average number of observations per agent needed for $U1$ to reach a belief state close to the ideal is 4.5 observations. The corresponding number for $U2$ is 3.2 observations. The average number decreased by 16% and 20% for $U1$ and $U2$ compared to the no cooperation case, which shows that infrequent communication is still better than no communication. More observations per agent were made compared to the case of constant communication, since evidence obtained locally was not exchanged in time.

It should be indicated that delays of information exchange due to infrequent communication may cause agents to believe quite differently. For example, suppose the following observations are entered into the agents:

$$\begin{array}{llllllll} U0: & d=1 & c=0 & U2: & q=0 & o=1 & U4: & w=0 \quad x=0 \\ U1: & h=1 & g=1 & U3: & l=1 & u=1 & & \end{array}$$

After `CommunicateBelief`, $U1$ and $U2$ have $p_1(k = 0|evi) = p_2(k = 0|evi) = 0.976$. Given that $g = 1$, $U2$ reasons from both input of the OR gate $G6$ being 0 and concludes that $G6$ is faulty ($p_2(G6 = faulty|evi) = 0.97$) producing $o = 1$ and that $G7$ is normal ($p_2(G7 = faulty|evi) = 0.035$). Suppose $U1$ subsequently observes $i = 0$, which makes $U1$ change its belief on k to $p_1(k = 0|evi) = 0.013$. At this moment, $U1$ and $U2$ believe totally opposite things regarding the value of k. If communication is performed at this

	Simulation Set 1		*Simulation Set 2*		*Simulation Set 3*	
	Obs (G9)	Obs (G7)	Obs (G9)	Obs (G7)	Obs (G9)	Obs (G7)
Simu0	5	6	4	2	4	2
Simu1	6	3	4.8	2	5	2
Simu2	6	6	5	3	5	4
Simu3	6	6	5	4.8	5	5
Simu4	6	3	5	1	5	2
Simu5	6	6	4	4.6	4	5
Simu6	4	3	3	4	4	4
Simu7	4	2	4	2	4	2
Simu8	6	3	4.8	2	5	2
Simu9	5	2	3	3	4	4
Aver	5.4	4	4.26	2.84	4.5	3.2

Table 1: Summary of the simulation results

moment, $U2$ will change its belief sharply to $p_2(G6 = faulty|evi) = 0.02$ and $p_2(G7 = faulty|evi) = 0.987$. However, without communication, $U2$ will continue its old belief which is out-of-date relative to the knowledge of the entire system.

Such situation is very natural in probabilistic (vs logic) reasoning since as long as an agent's belief on a proposition x is not certain, a new observation may change its belief on x to either extreme (0 or 1).

In addition to perform `CommunicateBelief` as frequently as possible, the above situation may be avoided by letting each agent perform a sensitivity analysis locally. For the above example, a sensitivity analysis at $U2$ will reveal that $U2$'s belief on the value of $G6$ and $G7$ is very sensitive to the value of k. Therefore, if $U2$ is also responsible to take some action, e.g., to replace a gate that is believed faulty, it may defer the action and wait for the next communication. Alternatively, $U2$ may perform a belief exchange with $U1$, which will bring $U2$ up-to-date and avoid the premature replacement of $G6$. Note that such exchange does not bring either $U1$ or $U2$ globally consistent. Neither has the knowledge of what is happening in other agents of the system. However, it may be sufficient for $U2$ to interpret its subdomain correctly. In general, a given agent's belief on its subdomain is more sensitive to the probabilistic information contained in agents close to it in the hypertree than that contained in agents far away from it. Therefore, it makes sense for an agent to exchange its belief with adjacent agents prior to a critical and sensitive decision in order to benefit from the knowledge of others. Such exchange does not require a full scale of `CommunicateBelief` and thus is less expensive. We leave the exploration of this opportunity to future work.

7 Off-line Time During Communication

7.1 Off-line time of each agent

During `CommunicateBelief`, the BT of a JT T with domain N may be changed through `CollectNewBelief` and `DistributeBelief` in the following ways:

1. (Through `CollectNewBelief`) When each adjacent JT of T (except the caller of `CollectNewBelief`) has completed its `CollectNewBelief`, the operation `UpdateBelief` that T performs relative to the JT may change $B_T(N)$.

2. (Through `DistributeBelief`) When `DistributeBelief` is called in T, the operation `UpdateBelief` that T performs relative to the caller may change $B_T(N)$.

It follows from the proof of Theorem 14 that $B_T(N)$ should not be modified by new external evidence between the first `UpdateBelief` (during `CollectNewBelief`) and the last `UpdateBelief` (during `DistributeBelief`) in the process of `CommunicateBelief`. That is, `EnterEvidence` should not be performed in T between the two `UpdateBelief` operations. Otherwise, `CommunicateBelief` will not regain the global consistency.

Since not being able to process evidence within a time interval imposes restriction on time-critical applications, the length of the interval should be minimized. We define such interval of time as follows:

Definition 16 (Junction Tree Off-line Time) *Let T be a JT in a LJF F. During a* `CommunicateBelief` *operation in F, let t be the instant of time when the first* `UpdateBelief` *involved by T is started. Let τ be the instant of time when the last* `UpdateBelief` *involved by T is completed. The* `off-line time` *of T during communication is $\Delta(T) = \tau - t$.*

7.2 Factors affecting off-line time

Different JTs in a LJF may have different off-line time during communication depending on several factors:

A `CommunicateBelief` operation is started in a LJF at an arbitrarily selected JT, which we refer to as the communication *root*. The choice of root is one factor that affects the off-line time of each agent in the system.

Example 17 We consider the effect of the root on T^4's off-line time in Figure 8. Since T^1 is selected as the root as shown in the Figure, during `CollectNewBelief`, T^4 must perform `UpdateBelief` relative to T^6, T^7 and T^8 (sequentially) first, and then allow T^2 to perform `UpdateBelief` relative to T^4. Afterwards, T^4 must wait for the completion of `CollectNewBelief` in the rest of the system and wait for its turn in `DistributeBelief`.

During `DistributeBelief`, T^4 must perform `UpdateBelief` relative to T^2 first and then allow T^6, T^7 and T^8 to perform `UpdateBelief` relative to T^4 (sequentially).

The above order of operations dictates that T^4 becomes off-line as soon as belief propagation from T^6, T^7 and T^8 to T^4 starts. T^4 remains off-line when belief further propagates to T^1 through T^2 and then back to T^4 from T^1. T^4 becomes available after belief propagates back to T^6, T^7, and T^8 through T^4. The total process involves 13 sequential `UpdateBelief` operations in the best case, where T^1 performs `DistributeBelief` relative to T^2 first among its four neighbors (16 in the worst case).

If T^6 instead T^1 is selected as the root, then T^4 will be off-line for only a period of time needed to perform 8 sequential `UpdateBelief`s.

Another factor is the order in which `CollectNewBelief` is performed by each agent relative to its neighbors.

Example 18 Consider T^7 in Figure 8 where the root is T^1. During `CollectNewBelief`, T^4 must perform `UpdateBelief` relative to T^6, T^7 and T^8 sequentially. If T^7 is first selected, T^7 must become off-line before T^6 and T^8, and its off-line time will be prolonged accordingly.

We refer to the order in which multiple neighbors are selected by an agent to perform `UpdateBelief` against, during `CollectNewBelief`, as the *collection order* of the agent.

Similarly, we refer to the order in which multiple neighbors are selected by an agent to perform `UpdateBelief`, during `DistributeBelief`, as the *distribution order* of the agent, which is a third factor affecting each agent's off-line time.

Example 19 Consider T^7 in Figure 8 where the root is T^1. During `DistributeBelief`, T^6, T^7 and T^8 must perform `UpdateBelief` relative to T^4 sequentially. If T^7 is first selected, T^7 can become available before T^6 and T^8, and its off-line time will be shortened accordingly.

The last factor is the time complexity of `UpdateBelief` by a JT T^i relative to a neighbor JT T^k. This time complexity is fixed once the LJF is constructed. Note that the time complexity of `UpdateBelief` by T^i relative to T^k may not be the same as the time complexity of `UpdateBelief` by T^k relative to T^i. This is because `UpdateBelief` performed by T^i relative to T^k involves multiple (the number of linkages between T^i and T^k) performance of `UnifyBelief` in T^i[19]. The time required by `UnifyBelief` in T^i is generally different than that in T^k.

7.3 Off-line time of a multi-agent system

The difference of off-line time across agents calls for a measurement of off-line time of the entire system. We consider two alternatives which may commonly be used:

Definition 20 (Absolute Off-line Time) *Let F be a LJF. Let t be the instant of time when the first JT in F becomes off-line during a* `CommunicateBelief` *operation. Let τ be the instant of time when the last JT becomes available again. The* `absolute off-line time` *of F is $\Delta_{abs} = \tau - t$.*

The absolute off-line time indicates the non-availability of the system as a whole even though some agents are available earlier than others.

Definition 21 (Average Off-line Time) *Let F be a LJF. Let $\Delta(T^i)$ be the off-line time of a JT T^i $(i = 1, \ldots, n)$ in F during a* `CommunicateBelief` *operation. The* `average off-line time` *of F is $\Delta_{ave} = (\sum_{i=1}^{n} \Delta(T^i))/n$.*

The average off-line time indicates the average non-availability of multiple agents.

Both definitions can be modified over a subset of JTs if availability is concerned with only the subset.

7.4 A graphical model for off-line time study

Based on the above analysis, given a LJF and a chosen off-line time measurement, we may manipulate the communication root, collection order and distribution order such that the off-line time is minimized. To avoid distraction by unnecessary details of communication, e.g., the number of linkages and the numerical belief computation, so that we can concentrate on the four factors that determine the off-line time, we abstract communication in a LJF into the following graphical model.

Model 22 (Graphical communication model)

Given an undirected and weighted tree, and an arbitrary node A as the root, the tree is converted to a rooted tree R.

For each node X of R, if $X \neq A$, place an *in-agent* at X. For each node Y of R, if Y has k children, place k *out-agents* at Y.

The agents traverse R according to the following rules:

1. To start with, each parent node Y with leaf children selects one child X, according to some order $O_{in}(Y)$. Once selected, X sends its in-agent to move from X to Y, which takes time $w_{in}(X)$ that is the weight associated with the link (X, Y) in the inward direction (from leaf to root).

   After one child's in-agent arrives at Y, the next child, selected according to $O_{in}(Y)$, sends its in-agent to Y.

   After a parent Y has received all the in-agents from its children, Y is ready for selection by its own parent Z. Once selected by Z, Y sends its in-agent to Z. The inward movement of in-agents continues in this fashion.

2. After the root A receives all in-agents from its children, the inward movement is completed and an outward movement starts.

   A selects one child X, according to some order $O_{out}(A)$. A then sends one out-agent to move from A to X, which takes time $w_{out}(X)$ that is the weight associated with the link (A, X) in the outward direction.

   After an out-agent of A reaches the destination, A selects another child according to $O_{out}(A)$ and sends another out-agent to the child. The process continues until all out-agents of A reach their destination.

   After an out-agent from A reaches a child X, X selects its own children, according to $O_{out}(X)$, and sends its out-agents to child nodes in sequence.

   The process continues in this fashion until the last out-agent in R reaches its leaf destination, and the outward movement halts.

The above model characterizes the `CommunicateBelief` operation correctly as far as the off-line time is concerned:

1. The original undirected tree corresponds to the LJF. Each node corresponds to a JT of the LJF.

2. The root A corresponds to the communication root.

3. The inward movement of in-agents corresponds to the belief propagation during `CollectNewBelief`, and the outward movement of out-agents corresponds to the belief propagation during `DistributeBelief`.

4. Given a parent node Y and a child node X, $w_{in}(X)$ corresponds to the time required for Y to perform `UpdateBelief` relative to X, and $w_{out}(X)$ corresponds to the time required for X to perform `UpdateBelief` relative to Y.

5. $O_{in}(X)$ corresponds to the collection order (Section 7.2) of X, and $O_{out}(Y)$ corresponds to the distribution order of Y.

6. The time instant, when the first in-agent from a child of a node X moves towards X, corresponds to the time instant when the corresponding JT becomes off-line. The time instant when the last out-agent from X arrives at a child of X corresponds to the time instant when the corresponding JT becomes available for entering evidence. The interval between the two instants thus corresponds to the off-line time of the JT represented by X.

Note that the graphical model reflects the semi-parallel pattern of communication that is discussed in Example 13. It will be seen that the key aspect of minimization of off-line time is to increase the degree of parallelism by manipulating the relevant factors.

We will use the following notations: We shall say that a non-leaf node X is *off* at the time instant when the in-agent from the first child selected by X starts moving to X. We use $t_{off}(X)$ to denote the instant. If X is a leaf node in the rooted tree, then X is off as soon as its in-agent leaves X.

We shall say that a non-leaf node X is *on* at the time instant when its last out-agent arrives at a child of X. We use $t_{on}(X)$ to denote the instant. If X is a leaf, then X is on when it receives the out-agent from its parent. We shall say that the off-line time of the node X is $\Delta(X) = t_{on}(X) - t_{off}(X)$.

We shall call the inward movement of in-agents *collection* and the outward movement of out-agents *distribution*. For collection, we use $t_{rdy}(Y)$ to denote the time instant when a non-leaf node Y receives the last in-agent from its children and is *ready* for its parent to select. For a leaf node Y, we assign $t_{rdy}(Y)$ to be the instant when collection starts. We use $t_{wat}(Y)$ to denote the time instant when the in-agent of Y arrives at its parent and Y starts to *wait* for an out-agent to come from its parent. For the root A, we assign $t_{wat}(A) = t_{rdy}(A)$.

For distribution, we use $t_{sel}(X)$ to denote the time instant when a node X is *selected* by its parent Y such that Y is about to send an out-agent to X. For the root A, we assign $t_{sel}(A)$ to be the instant when distribution starts. We use $t_{cpt}(X)$ to denote the time instant when X receives the out-agent from Y (distribution relative to Y is *completed*). For the root A, we assign $t_{cpt}(A) = t_{sel}(A)$.

Example 23 Figure 9 illustrates the graphical communication model for a seven-agent system. The figure in the left shows the rooted tree R with the root A, and leaves D, E, F and G. It also shows the in-agent and out-agents of each node, and the in-weight and the out-weight of each link.

Figure 9 (middle) shows collection in the rooted tree R. The collection order used is from left to right for each parent node. At time instant $t = 0$, E is off. Its in-agent arrives at B at $t = 5$. Thus E starts to wait for distribution at $t = 5$, and B is ready for selection by A at the same point in time.

Parallel to the above activity, at $t = 0$, F (the leftmost child) is selected by C to send its in-agent which arrives at C at $t = 1$. Then C selects the next child G, and the in-agent of G arrives at C at $t = 4$. Thus, F is waiting at $t = 1$, G is waiting at $t = 4$, and C is ready at $t = 4$.

According to the left-to-right order, A selects B at $t = 5$. The in-agent of B arrives at A at $t = 9$. Then C is selected, whose in-agent arrives at A at $t = 11$. Then D is selected, whose in-agent arrives at $t = 19$, and collection is completed.

Figure 9 (right) shows distribution which follows collection immediately. The distribution order used is right-to-left. At $t = 19$, A sends an out-agent to D, which arrives at $t = 26$. D is on at this moment since it has no child.

C is then selected and another out-agent of A arrives at C at $t = 29$.

The last out-agent of A arrives at B at $t = 35$, and A is on at this moment. Parallel to the movement of the last out-agent of A, at $t = 29$, C sends its first out-agent to G

which arrives at $t = 31$, and then G is on. C then sends another out-agent to F which arrives at $t = 35$. At this moment, both C and F are on.

At $t = 35$, B sends its out-agent to E, which arrives at $t = 40$. At this moment, both B and E are on. Distribution, as well as the entire communication, is then completed.

All activities during the communication are fully specified by the tuples used to label nodes of the two figures. The off-line time of each node and the entire system can thus be calculated. For instance, $\Delta(C) = 35 - 0 = 35$ and $\Delta(D) = 26 - 11 = 15$. The absolute off-line time of the system is $\Delta_{abs} = 40 - 0 = 40$. The average off-line time is $\Delta_{ave} = (30 + 40 + 35 + 15 + 40 + 35 + 30)/7 = 32.1$.

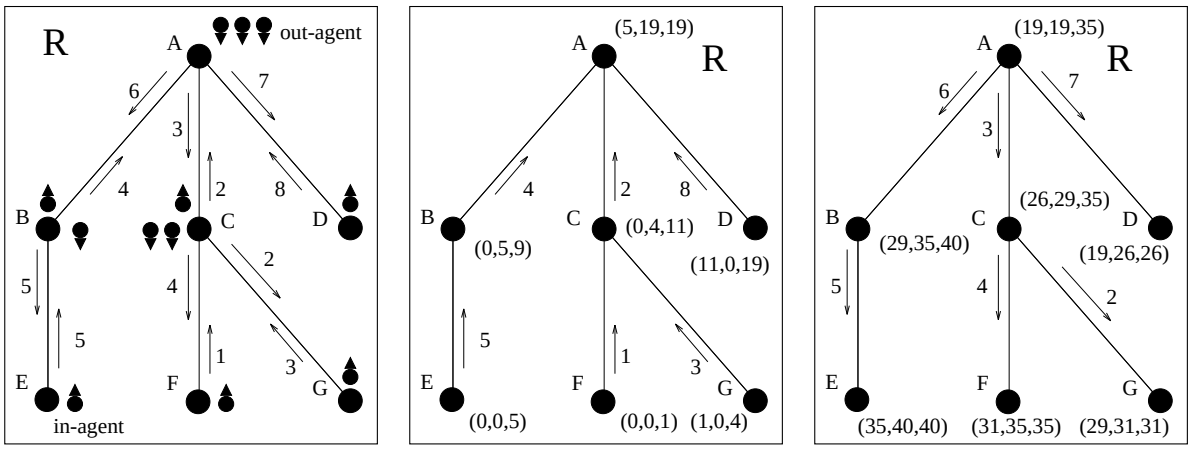

Figure 9: A graphical model for communication in a seven-agent system. The in-weight of a link is indicated by an upward arrow and the associated label, and the out-weight of a link is indicated by a downward arrow and the associated label. Left: The weighted tree R rooted at A. The in-agent and out-agents of each node, as well as the in-weight and the out-weight of each link are shown. Middle: Collection in R with only the in-weight of each link shown. Each node X is labeled will a triple $(t_{off}(X), t_{rdy}(X), t_{wat}(X))$. Right: Distribution in R with only the out-weight of each link shown. Each node X is labeled will a triple $(t_{sel}(X), t_{cpt}(X), t_{on}(X))$.

We will refer to a specification of the timing of every node's activity during collection, such as the labeling of Figure 9 (middle), as a collection *schedule*. A distribution schedule is similarly defined. Our goal is to find schedules with the minimum off-line time. In both schedules of Example 23, we assumed that a node engages in its activity as soon as the activity is possible without any delay. Since unnecessary idling can not contribute positively to our goal, we will exclude from our consideration those schedules in which some nodes delay their activities unnecessarily. On the other hand, if there is any practical

reason to delay the belief propagation, e.g., the computer network delay, we assume that the delay has been modeled in the link weights.

The schedule in Example 23 is not optimal. We illustrate this using the absolute off-line time: During collection, A follows the left-to-right order, and thus waits until $t = 5$ when B is ready. However, another child D of A is ready at $t = 0$, but is selected only at $t = 11$. If D is selected first, both A and D can be engaged in their activity earlier, and A will complete collection and start distribution earlier. The consequence is a shorter absolute off-line time. The difference made by switching the first child to D is that the resultant schedule has a higher degree of parallelism: The parallel activities E to B, and F, G to C in the previous schedule are now also paralleled by the activity D to A.

In the remaining part of this paper, we use the graphical model to study the minimization of the off-line time with an arbitrarily given communication root. Limited by the space, the optimization of the root is beyond the scope of this paper.

8 Reduction of Communication Scheduling

In this section, we reduce the communication scheduling problem into two subproblems: the collection scheduling and the distribution scheduling. We then show that the two are dual problems in the sense that the solution to one of them can be extended directly into the solution of the other.

8.1 Problem reduction

Let us first consider the minimization of the absolute off-line time. Let collection start at the time instant $t = t_0$ and terminate at $t = t_1$. Let distribution start at the time instant $t = t_1$ and terminate at $t = t_2$. Denote the time interval between t_0 and t_1 by Δ_{0-1}, and denote the time interval between t_1 and t_2 by Δ_{1-2}. We have $\Delta_{abs} = \Delta_{0-1} + \Delta_{1-2}$. Since Δ_{0-1} is independent of Δ_{1-2},

$$min(\Delta_{abs}) = min(\Delta_{0-1}) + min(\Delta_{1-2}),$$

where the minimization in the left-hand side of the equation is over all collection and distribution schedules, the first minimization in the right-hand side is over all collection schedules, and the second in the right-hand side is over all possible distribution schedules. We can thus study the optimal communication schedules by studying the optimal collection schedules and the optimal distribution schedules independently.

Next, let us consider the minimization of the average off-line time. For each node X, the off-line time is $\Delta(X) = t_{on}(X) - t_{off}(X) = (t_{on}(X) - t_1) + (t_1 - t_{off}(X))$. Note that $t_1 - t_{off}(X)$ is the off-line time of X during collection, and $t_{on}(X) - t_1$ is the off-line time

of X during distribution. We therefore obtain

$$\begin{aligned}\min_{S_c,S_d}(\Delta_{ave}) &= \min_{S_c,S_d}(\frac{1}{N}\sum_{i=1}^{N}\Delta(X_i)) = \frac{1}{N}\min_{S_c,S_d}(\sum_{i=1}^{N}(t_1 - t_{off}(X_i)) + \sum_{i=1}^{N}(t_{on}(X_i) - t_1))) \\ &= \frac{1}{N}(\min_{S_c,S_d}(\sum_{i=1}^{N}(t_1 - t_{off}(X_i))) + \min_{S_c,S_d}(\sum_{i=1}^{N}(t_{on}(X_i) - t_1))),\end{aligned}$$

where the minimization is over all possible collection schedules S_c and distribution schedules S_d. Since the minimization over S_d has no effect on the first summation, we have

$$\min_{S_c,S_d}(\Delta_{ave}) = \frac{1}{N}(\min_{S_c}(\sum_{i=1}^{N}(t_1 - t_{off}(X_i))) + \min_{S_c,S_d}(\sum_{i=1}^{N}(t_{on}(X_i) - t_1))).$$

Although the minimization over S_c affects the value of t_1, it has no effect on the length of interval $t_{on}(X_i) - t_1$. Thus the second summation is only affected by the minimization over S_d. We therefore obtain

$$\min_{S_c,S_d}(\Delta_{ave}) = \frac{1}{N}(\min_{S_c}(\sum_{i=1}^{N}(t_1 - t_{off}(X_i))) + \min_{S_d}(\sum_{i=1}^{N}(t_{on}(X_i) - t_1))).$$

This again implies that the optimal communication schedules can be studies by studying the optimal collection schedules and the optimal distribution schedules independently.

8.2 Duality of Collection and Distribution

A comparison of collection and distribution shows the great similarity between the two activities. Both proceed through the tree structure in a semi-parallel pattern. Inward movement of in-agents from the children of a node must proceed sequentially. So must outward movement of out-agents of a node to its children. Two nodes of a common ancestor may be engaged in collection relative to their children in parallel. So may they be engaged in distribution relative to their children in parallel.

We establish the duality of the two activities with respect to the off-line time in Theorem 24. The proof is in Appendix.

Theorem 24 (Duality)

Let R be a weighted tree rooted at A with the in-weight and out-weight of each link identical.

Let S_d be a distribution schedule which starts from A at t_d and terminates at τ_d. Let the distribution order of a parent node Y be denoted as $O_d(Y)$ in this schedule.

A schedule S_c for collection, that starts at t_c and terminates at A at τ_c, can be obtained by requiring each parent node Y to follow a collection order that is opposite to $O_d(Y)$ such that the followings hold:

1. $\tau_c - t_c = \tau_d - t_d$, *and*

2. *for every node* X, $\tau_c - t_{off}(X)$ *in* S_c *is identical to* $t_{on}(X) - t_d$ *in* S_d.

The converse (obtaining S_d *from* S_c*) is also true.*

Given a rooted tree R, Theorem 24 implies that, to find the optimal collection schedule in R, we can treat w_{in} for each link as w_{out} for the link, and find the optimal distribution schedule in the modified tree. Once the optimal distribution schedule is found, we can reverse the distribution order and the resultant is the optimal collection schedule. The converse is also correct. We therefore only need to find the optimal schedule for one of the two activities for any given off-line time criterion.

Example 25 Figure 10 (left) is a rooted tree R for distribution with a distribution schedule specified. The distribution order for each parent is right-to-left.

Figure 10 (right) shows a rooted tree R' identical to R except the w_{out} of each link in R becomes the w_{in} of the same link. A collection schedule for R' is specified in the figure that satisfies the two conditions of Theorem 24. The collection order for each parent is left-to-right.

Note that we have $\tau_d - t_d = 14 - 0$ in R, and $\tau_c - t_c = 14 - 0$ in R'. Comparing the four-tuples that label corresponding nodes in R and R', we see that the last attributes ($t_{on}(X) - t_d$ in R and $\tau_c - t_{off}(X)$ in R') have identical values.

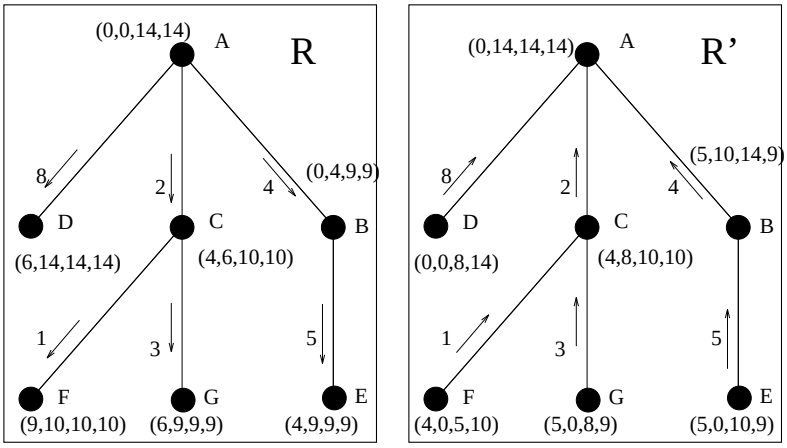

Figure 10: Left: A rooted tree R for distribution. The distribution schedule is shown by labeling each node X with $(t_{sel}(X), t_{cpt}(X), t_{on}(X), t_{on}(X) - t_d)$ where $t_d = 0$. Right: R' is obtained from R by treating w_{in} as w_{out}. A collection schedule with $t_c = 0$ is obtained from the schedule in R as described by Theorem 24. The schedule is shown by labeling each node X with $(t_{off}(X), t_{rdy}(X), t_{wat}(X), \tau_c - t_{off}(X))$.

9 Optimal Communication Schedules

In this section, we derive distribution schedules with the minimum absolute off-line time and distribution schedules with the minimum average off-line time, given a rooted tree. These results can then be used to obtain the optimal collection schedules through duality.

9.1 Distribution Schedules with Minimum Absolute Off-line Time

Distribution starts from the root. Consider first a rooted tree with depth 2 and with branching factor 2 as shown in Figure 11. Distribution in the root A can be performed with two possible orders: $O^{(1)}_{out}(A) = (B, C)$ and $O^{(2)}_{out}(A) = (C, B)$.

Using $O^{(1)}_{out}(A)$, we obtain

$$\Delta^{(1)}_{1-2} = max(w_{out}(B) + w_{out}(C) + w_{out}(F) + w_{out}(G), w_{out}(B) + w_{out}(D) + w_{out}(E)).$$

Since distribution from a parent of leaves to the leaf children is sequential, all leaf children of the same parent can be treated equivalently as a single leaf with its w_{out} being the sum of w_{out}s of the original leaves involved. We can therefore write

$$\Delta^{(1)}_{1-2} = max(w_{out}(B) + w_{out}(C) + w_{out}(FG), w_{out}(B) + w_{out}(DE)),$$

where $w_{out}(FG) = w_{out}(F) + w_{out}(G)$ and FG denotes the equivalent single leaf.

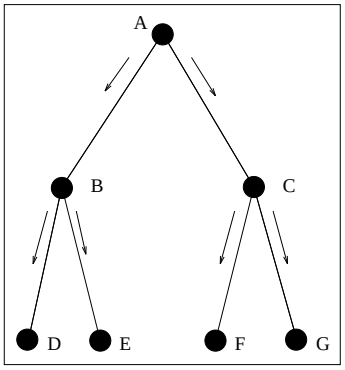

Figure 11: A rooted tree of depth 2 for distribution.

Similarly, using $O^{(2)}_{out}(A)$, we obtain

$$\Delta^{(2)}_{1-2} = max(w_{out}(C) + w_{out}(B) + w_{out}(DE), w_{out}(C) + w_{out}(FG)).$$

The optimal order is the one with a smaller Δ_{1-2}.

We claim that if $w_{out}(DE) \leq w_{out}(FG)$, then $O^{(2)}_{out}(A)$ is the optimal order: Under the condition, $\Delta^{(1)}_{1-2}$ can be simplified into $\Delta^{(1)}_{1-2} = w_{out}(B) + w_{out}(C) + w_{out}(FG)$ which is larger than or equal to both entries of $\Delta^{(2)}_{1-2}$, namely, $w_{out}(C) + w_{out}(B) + w_{out}(DE)$ and $w_{out}(C) + w_{out}(FG)$.

The above result suggests that the out-weights at the top level (i.e., $w_{out}(B)$ and $w_{out}(C)$) are not critical, but the sums of out-weights at the bottom level (i.e., $w_{out}(DE)$ and $w_{out}(FG)$) are. This result in fact is general as is shown in Proposition 26. The proof is in Appendix.

Proposition 26 (Optimal distribution schedule in trees of depth 2)

Let R be a tree rooted at A for distribution. Let the depth of R be 2. Let the children of root be $X_1, \ldots, X_n$. Let the sum of out-weights of children of X_i be v_i such that $v_1 \leq v_2 \leq \ldots \leq v_n$.

The absolute distribution off-line time Δ_{1-2} in R is minimized if the distribution order of A is $O_{out}(A) = (X_n, \ldots, X_1)$.

Example 27 According to Proposition 26, Δ_{1-2} for R in Figure 9 will be minimized if $O_{out}(A) = (C, B, D)$. The minimum Δ_{1-2} is $\Delta_{1-2} = max(3+(4+2), 3+6+5, 3+6+7+0) = 16$ which is a 24% improvement over $\Delta_{1-2} = 21$ in Example 23. A corresponding optimal schedule is shown in Figure 12. The distribution order used is left-to-right.

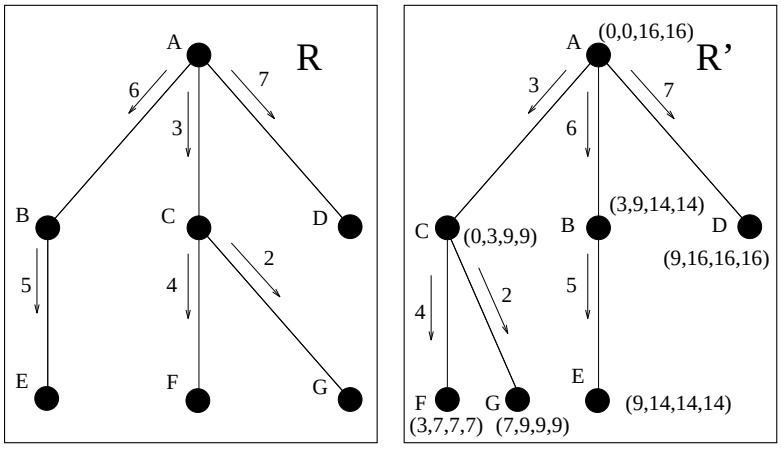

Figure 12: R: A rooted tree for distribution. R': R with the left-right order of nodes B, C, D rearranged according to $O_{out}(A) = (C, B, D)$ determined by Proposition 26. The distribution schedule is shown by the label $(t_{sel}(X), t_{cpt}(X), t_{on}(X), t_{on}(X) - t_d)$ at each node X. The distribution starts at $t_d = 0$.

Algorithm 28 (Order arrangement for distribution)

Input: *A rooted tree of depth M with the out-weight of each link defined.*
Output: *The rooted tree with the left-right order of children of each parent re-arranged.*
begin

D := M-1
for each leaf Z of depth D, do
$v(Z) := 0$
for each node Y of depth D with child nodes $X_1, \ldots, X_n$, do
$v(Y) := \sum_{i=1}^{n} w_{out}(X_i)$
D := D-1
while $D \geq 0$, do
for each node Y of depth D with n child nodes, do
arrange children of Y and index them from left to right as $X_1, \ldots, X_n$ such that $v(X_1) \leq \ldots \leq v(X_n)$
$v(Y) := max(e_n, \ldots, e_1)$ *where* $e_i = (\sum_{k=i}^{n} w_{out}(X_k)) + v(X_i)$
for each leaf Z of depth D, do
$v(Z) := 0$
D := D-1

end

Algorithm 28 and Theorem 30 generalize Proposition 26 to an arbitrary rooted tree for distribution. Algorithm 28 rearranges the left-right order of children for each node such that the optimal distribution order becomes topologically explicit. Theorem 30 establishes the optimal schedule.

Example 29 Figure 13 illustrates Algorithm 28. After the first two *for* loops, we have $v(E) = 8$, $v(F) = 14$, $v(G) = 0$, $v(H) = 15$, $v(I) = 0$, and $v(J) = 18$. After the first pass of the *while* loop, children of B are arranged from left to right in the order G, E, F based on their v values. Children of D are arranged from left to right in the order I, H, J. The v values for nodes B, C, D are assigned as $v(B) = 19$, $v(C) = 0$, $v(D) = 27$.

After the second pass of the *while* loop, children of root A are arranged from left to right in the order C, B, D, and $v(A)$ is assigned the value $v(A) = 36$.

Theorem 30 (Optimal distribution schedule)

Let R be a rooted tree for distribution with root A. The absolute off-line time Δ_{1-2} is minimized if R is arranged according to Algorithm 28 and the distribution order for each node is right-to-left.

The minimum Δ_{1-2} is given by $v(A)$ as computed by Algorithm 28.

Proof:

We prove by induction. Let the depth of R be M. The statement is true if $M = 2$ according to Proposition 26.

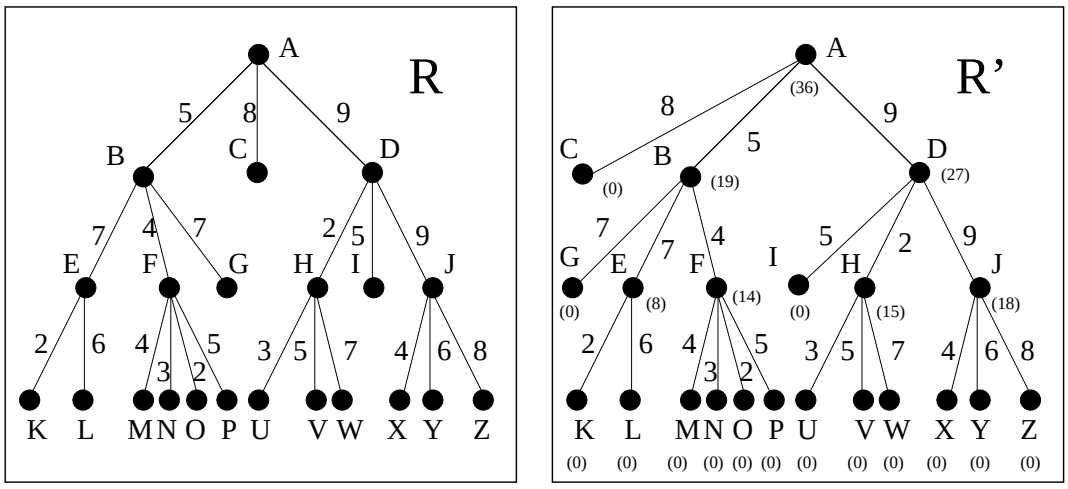

Figure 13: R: A rooted tree for distribution operation. R': R after being processed according to Algorithm 28. Each node X is labeled with $(v(X))$ as computed by Algorithm 28.

Assume that the statement is true for $M = K$ ($K \geq 2$). Consider $M = K+1$. Assume that R is arranged according to Algorithm 28, and the root A of R has n ($n \geq 1$) children $Y_1, \ldots, Y_n$. Each child is the root of a subtree.

By the inductive assumption, the distribution time for the subtree rooted at Y_i ($i \in \{1, \ldots, n\}$) is minimized if the right-to-left order is followed. The minimum time is $v(Y_i)$.

For $i = 1, \ldots, n$, replace the subtree rooted at Y_i by a single link (Y_i, X_i) with $w_{out}(X_i) = v(Y_i)$. The resultant is a tree of depth 2 rooted at A that satisfies the condition of Proposition 26. Therefore distribution using the right-to-left order at A will optimize the distribution time. Since distribution time for each subtree rooted at Y_i is minimized by induction assumption, and the distribution at root A is also optimized, Δ_{1-2} is minimal for $M = k + 1$ and the minimum value is given as $v(A)$. $\square$

Example 31 The absolute distribution off-line time for R in Figure 13 is minimized by arranging R as R' and following the right-to-left distribution order. The minimum Δ_{1-2} is $v(A) = 36$.

9.2 Distribution Schedules with Minimum Average Off-line Time

Let R be a tree rooted at A for distribution (Figure 14). Let the distribution operation start at the time instant t_1. We denote $\sum(t_{on}(Y) - t_1)$ by $\sigma_R(A)$ where the summation is over every node Y of R rooted at A. The optimal distribution schedule is one that minimizes $\sigma_R(A)$.

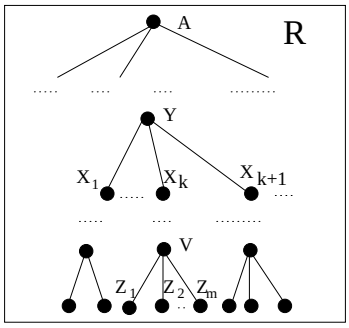

Figure 14: A general rooted tree for distribution

We first consider the contribution to $\sigma_R(A)$ made by the nodes in a subtree rooted at a parent node, say, V of m leaf children (see Figure 14). We have

$$\sigma_R(V) = (t_{on}(V) - t_1) + \sum_{k=1}^{m} (t_{on}(Z_k) - t_1).$$

If we substitute $t_{on}(V) = t_{cpt}(V) + \sum_{k=1}^{m} w_{out}(Z_k)$, we can rewrite

$$\sigma_R(V) = \sum_{k=1}^{m} w_{out}(Z_k) + (m+1)(t_{cpt}(V) - t_1) + \sum_{k=1}^{m} (t_{on}(Z_k) - t_{cpt}(V)) \qquad (4)$$

The first entry (summation) is independent of the distribution order. The second entry (product) is independent of the distribution order of V. It is dependent of the distribution orders of nodes outside the subtree rooted at V, and is dependent of the number $m+1$ of nodes in the subtree rooted at V. Only the last entry $\sum_{k=1}^{m}(t_{on}(Z_k) - t_{cpt}(V))$, which we denote by $\sigma(V)$, is dependent of the distribution order of V. Note that we have used $\sigma(V)$ instead of $\sigma_R(V)$ to denote the sum, to signify the fact that its minimization can be studied by isolating the subtree rooted at V from the rest of R.

Next, we consider the contribution to $\sigma_R(A)$ made by the nodes in a subtree rooted at an arbitrary non-leaf node Y. (Figure 14). We obtain

$$\begin{aligned} \sigma_R(Y) &= (t_{on}(Y) - t_1) + \sum_k \sigma_R(X_k) \\ &= \sum_k w_{out}(X_k) + (t_{cpt}(Y) - t_1) + \sum_k \sigma_R(X_k). \end{aligned}$$

Denote the number of descendants of X_k by n_k, we obtain

$$\begin{aligned} \sigma_R(X_k) &= \sigma(X_k) + (n_k+1)(t_{cpt}(X_k) - t_1) \\ &= \sigma(X_k) + (n_k+1)(t_{cpt}(X_k) - t_{cpt}(Y)) + (n_k+1)(t_{cpt}(Y) - t_1). \end{aligned}$$

If we substitute the above expression into $\sigma_R(Y)$ and denote the number of descendants of Y by η, we obtain

$$\begin{aligned}\sigma_R(Y) &= \sum_k w_{out}(X_k) + (\eta+1)(t_{cpt}(Y) - t_1) \\ &+ \sum_k \sigma(X_k) + \sum_k (n_k+1)(t_{cpt}(X_k) - t_{cpt}(Y)). \end{aligned} \quad (5)$$

The first entry (summation) is independent of the distribution order. The second entry (product) is dependent of the distribution orders of nodes outside the subtree rooted at Y, and is dependent of the number $\eta+1$ of nodes in the subtree rooted at Y. The third entry (summation) is dependent of the distribution orders of nodes in the subtree rooted at each child of Y. Only the last entry (summation) is dependent of the distribution order of Y.

Note that equation 4 is a special case of equation 5 if we let $Y = V$, $X_k = Z_k$, $n_k = 0$, $\sigma(X_k) = 0$ and $t_{cpt}(X_k) = t_{on}(Z_k)$.

Another special case of equation 5 is $\sigma_R(A)$. If we let $Y = A$, $t_{cpt}(Y) = t_1$, we obtain

$$\sigma_R(A) = \sum_k w_{out}(X_k) + \sum_k \sigma(X_k) + \sum_k (n_k+1)(t_{cpt}(X_k) - t_1). \quad (6)$$

The above analysis implies that, in order to find the optimal distribution schedule for R, we need only to search *locally*, i.e., to find the optimal distribution order of each node Y, taking into account the number of descendants of Y. The following Lemma prepares for Theorem 33. Its proof is in Appendix.

Lemma 32 *Let Y be a non-leaf node in a rooted tree. Let the child nodes of Y be $X_1, \ldots, X_m$ $(m > 0)$ where X_k has n_k descendants.*

The summation $\psi = \sum_{k=1}^{m}(n_k+1)(t_{cpt}(X_k) - t_{cpt}(Y))$ is minimized if

$$w_{out}(X_1)/(n_1+1) \le w_{out}(X_2)/(n_2+1) \le \ldots \le w_{out}(X_m)/(n_m+1)$$

and the distribution order of Y is $O_{out}(Y) = (X_1, \ldots, X_m)$.

Theorem 33 (Optimal distribution schedule)

Let R be a weighted tree rooted at A. For each non-leaf node Y of R, let the $m > 0$ child nodes of Y be indexed from left to right as $X_1, \ldots, X_m$, and let the number of descendants of X_i be n_i such that

$$w_{out}(X_1)/(n_1+1) \le w_{out}(X_2)/(n_2+1) \le \ldots \le w_{out}(X_m)/(n_m+1).$$

The average distribution off-line time $\sigma_R(A)$ is minimized if the distribution order of Y is left-to-right for every Y.

Proof:

First, consider the case where the depth of R is $D = 1$. Let the child nodes of root A be $X_1, X_2, \ldots$.

According to Equation 6, when $D = 1$, we obtain

$$\sigma_R(A) = \sum_k w_{out}(X_k) + \sum_k (t_{cpt}(X_k) - t_1).$$

To minimize $\sigma_R(A)$, we only need to minimize the second sum. According to Lemma 32, $\psi = \sum_k (t_{cpt}(Y_k) - t_1)$ can be minimized if the left-to-right order is followed. The statement is thus true when $D = 1$.

Assume that the statement is true when the depth of R is $D = d \geq 1$. That is, $\sigma_R(A)$ is minimized when the left-right order of child nodes is arranged as specified and the left-to-right distribution order is followed.

We consider

$$\sigma_R(A) = \sum_k w_{out}(X_k) + \sum_k \sigma(X_k) + \sum_k (n_k + 1)(t_{cpt}(X_k) - t_1)$$

when $D = d + 1$. Based on the analysis made with Equation 5, minimization of each $\sigma(X_k)$ and minimization of $\psi = \sum_k (n_k + 1)(t_{cpt}(X_k) - t_1)$ are independent, and can thus be performed separately. According to Lemma 32, ψ is minimized if the left-to-right order is followed. Each $\sigma(X_k)$ can be minimized by following the left-to-right order according to the inductive assumption. The statement is proven. □

Example 34 Figure 15 shows an optimal distribution schedule obtained according to Theorem 33. We can compare this schedule with the distribution schedule in Example 27. There, no preferred distribution order for node C and $O_{out}(C) = (F, G)$ is arbitrarily chosen. The consequence is $\sigma_{R'}(A) = 16 + 9 + 14 + 16 + 7 + 9 + 14 = 85$. Here, the preferred distribution order for node C is $O_{out}(C) = (G, F)$. The consequence is $\sigma_{R'}(A) = 16 + 9 + 14 + 16 + 5 + 9 + 14 = 83$.

This comparison shows that an optimal schedule under the absolute off-line time criterion may not be optimal under the average off-line time criterion. However, for the absolute off-line time, both examples happen to have the identical value 16.

Example 35 Figure 16 shows two distribution schedules in an identical rooted tree (subject to a difference in left-right order of child nodes). The schedule shown in R is optimal under the absolute off-line time criterion, obtained according to Theorem 30. The distribution order is right-to-left. It has $\Delta_{1-2} = 14 - 0$ and $\sigma_R(A) = 9 + 14 + 12 + 14 + 12 + 8 = 69$. The schedule shown in R' is optimal under the average off-line time criterion, obtained according to Theorem 33. The distribution order is left-to-right. It has $\Delta_{1-2} = 15 - 0$ and $\sigma_R(A) = 9 + 8 + 15 + 8 + 11 + 15 = 56$.

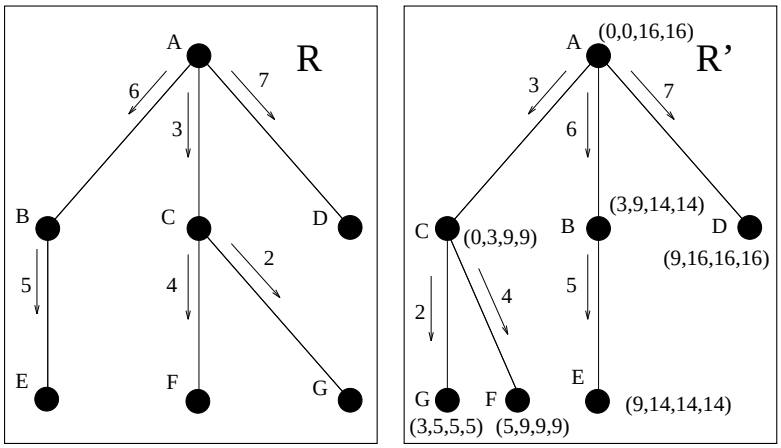

Figure 15: R: A rooted tree for distribution. R': R with the left-right order of nodes B, C, D rearranged according to Theorem 33. The distribution schedule is shown by the label $(t_{sel}(X), t_{cpt}(X), t_{on}(X), t_{on}(X) - t_d)$ at each node X. The distribution starts at $t_d = 0$.

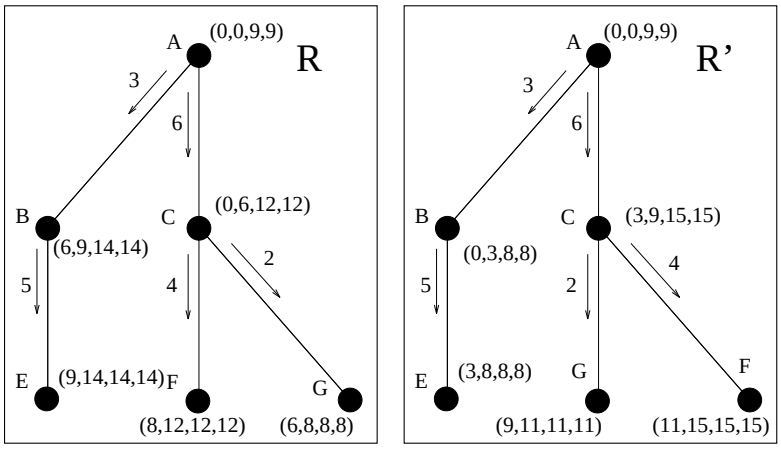

Figure 16: Comparison of the optimal distribution schedules under different criteria. An schedule with the minimum absolute off-line time is shown in R. An schedule with the minimum average off-line time is shown in R'. Each schedule is shown by the label $(t_{sel}(X), t_{cpt}(X), t_{on}(X), t_{on}(X) - t_d)$ at each node X in the corresponding rooted tree. The distribution starts at $t_d = 0$.

Example 35 shows that, given a rooted tree, an optimal schedule under the absolute off-line time criterion may not be the optimal under the average off-line time criterion and vice versa. Though the example involves distribution only, the conclusion is general due to the duality of distribution and collection.

10 Discussion

10.1 DAI, tree structure and conditional independence

In a MSBN, the intersection between each pair of subnets must satisfy the d-sepset condition such that the pair is conditionally independent given the d-sepset. It can be seen from Section 3, the semantics of joint probability distribution of a cooperative multi-agent system is undefined without this condition. With this condition, in order to bring two adjacent subnets up-to-date, it is sufficient to pass the new probability distribution on the d-sepset between them and nothing else.

One of the major concerns in DAI is how to balance the need to maintain as much as possible global consistency and the need to reduce the traffic of communication. As argued by Pearl [15], a tree structure makes use of conditional independence and allows coherent and the most efficient information passage among a group of elements. Our study on MSBNs highlights the MSBNs that are organized into a *hypertree* structure. This structural preference is a more general case of the exploration of the conditional independence in singly connected BNs [14] and in the JT representation of multiply connected BNs [9]. If we call an element which can render a pair of elements conditionally independent a 'dependency mediator', we see an increase of complexity of the internal structures of dependency mediators in the three cases. In singly connected BNs, a dependency mediator is an internal node in the network. In the JT representation of multiply connected BNs, a dependency mediator is an internal clique (a group of variables). In MSBNs/LJFs of a hypertree structure, a dependency mediator is a subnet/JT. As argued by Pearl [15], conditional independence should not be viewed as a restrictive assumption for mathematical convenience, nor as an occasional grace of nature for which we must passively wait, but rather as a mental construct that we should actively create. The progression of probabilistic reasoning techniques from singly connected BNs, to the JT representation of multiply connected BNs, and to MSBNs/LJFs is just one example of such endeavor.

10.2 Communication in MSBNs and belief propagation in a JT

Our `CommunicateBelief` operation in a multi-agent MSBN is a direct extension of `CollectEvidence` and `DistributeEvidence` in a single JT by Jensen et al.[9, 10] and the multi-linkage belief propagation in a single-agent MSBN by Xiang et al.[19].

If we concentrate on the overall pattern of information flow, we see that communication in a MSBN works in the same way (a semi-parallel inward movement followed by a semi-parallel outward movement in a tree structure) as belief propagation in a JT. As commonly applied, BNs are single-agent-oriented. Therefore, it is taken as granted that all nodes of a JT are centralized. The special pattern of information flow is used to support the object-oriented implementation as indicated in Jensen et al.[10]. However, concurrent processing is reasonably considered uninteresting.

In a multi-agent MSBN, the agents are very likely to be spatially distributed, the efficiency of communication becomes a necessary concern. Part of the effort in this work has thus been devoted to this issue.

Once the issue in concurrency control is resolved, however, the result can be applied to the belief propagation in a JT, if parallel processing of a JT is desired, i.e., when the JT is large and the computation is time-critical, and the hardware is available.

10.3 Future Research

In this pilot study, we proposed a probabilistic framework for cooperative multi-agent distributed interpretation systems. We showed that if agents are cooperative, conditionally independent and (initially) consistent, then they can be organized into a MSBN and the semantics of the joint probability distribution of the MSBN is well defined. Such an organization ensures the coherent probabilistic inference among multiple agents. We proposed new communication operations in addition to the belief propagation operations in single-agent-oriented MSBNs. We showed that these operations can maintain consistency among agents after evidence has been gathered by agents asynchronously in parallel. We proposed scheduling algorithms that optimize the communication operations and minimize the unavailability of agents for evidence processing.

There are many directions for future research. These include dynamic construction of a MSBN from a pool of potentially cooperative agents in a demand-driven fashion as mentioned in Section 3.3, formal justification of localized communication as outlined in Section 6.3, refinement of belief propagation operations and their implementation, application to specific problem structures, integration of interpretation and action and optimization of the communication root as indicated in Section 7.4.

Appendix

Theorem 24 **(Duality)**

> Let R be a weighted tree rooted at A with the in-weight and out-weight of each link identical. Let S_d be a distribution schedule which starts from A at t_d and terminates at τ_d. Let the distribution order of a parent node Y be denoted as $O_d(Y)$ in this schedule. A schedule S_c for collection, that starts at

t_c and terminates at A at τ_c, can be obtained by requiring each parent node Y to follow a collection order that is opposite to $O_d(Y)$ such that the followings hold:

1. $\tau_c - t_c = \tau_d - t_d$, and
2. for every node X, $\tau_c - t_{off}(X)$ in S_c is identical to $t_{on}(X) - t_d$ in S_d.

The converse (obtaining S_d from S_c) is also true.

Proof:

We start with a S_d and construct a S_c such that the above two conditions are satisfied. Once the mapping is created, the converse of the theorem is trivially true.

We assume that for each parent node Y with m child nodes, the child nodes are arbitrarily indexed as $X_1, \ldots, X_m$. Without losing generality, we denote the distribution order of Y in S_d by $O_d(Y) = (X_1, \ldots, X_m)$. Using the notation defined in Section 7.4, we can then characterize S_d as follows:

$$\begin{cases} t_{cpt}(Y) = t_d & \text{if } Y \text{ is the root} \\ t_{cpt}(Y) = t_{sel}(Y) + w(Y) & \text{if } Y \text{ is not the root} \\ t_{on}(Y) = t_{cpt}(Y) + \sum_{k=1}^{m} w(X_k) & \text{if } Y \text{ is not a leaf} \\ t_{sel}(X_i) = t_{cpt}(Y) + \sum_{k=1}^{i-1} w(X_k) & \text{if } X_i \text{ is not the root} \\ t_{on}(X) = t_{cpt}(X) & \text{if } X \text{ is a leaf} \\ \tau_d = max(t_{on}(Z)) & max() \text{ over every node } Z \end{cases} \tag{7}$$

We construct a collection schedule S_c from S_d by requiring each parent node Y to follow the collection order $O_c(Y) = (X_m, \ldots, X_1)$ which is opposite to $O_d(Y)$. The resultant schedule S_c can be characterized as follows:

$$\begin{cases} t_{off}(X) = \tau_d - t_{on}(X) + t_c & \text{if } X \text{ is a leaf} \\ t_{off}(Y) = t_{off}(X_m) & \text{if } Y \text{ is not a leaf} \\ t_{rdy}(X) = t_c & \text{if } X \text{ is a leaf} \\ t_{rdy}(Y) = t_{off}(Y) + \sum_{k=1}^{m} w(X_k) & \text{if } Y \text{ is not a leaf} \\ t_{wat}(Y) = t_{rdy}(Y) + w(Y) & \text{if } Y \text{ is neither the root, nor a leaf} \\ t_{wat}(X) = t_{off}(X) + w(X) & \text{if } X \text{ is a leaf} \\ t_{wat}(Y) = t_{rdy}(Y) & \text{if } Y \text{ is the root} \\ \tau_c = t_{rdy}(Y) & \text{if } Y \text{ is the root} \end{cases} \tag{8}$$

We show inductively that S_c and S_d satisfy the two conditions stated in the theorem.

Consider a R with depth $D = 1$. That is, R has a root node A with leaf children $X_1, \ldots, X_m$. For S_d, using Equation 7, we obtain $t_{cpt}(A) = t_d$ and $t_{on}(A) = t_d + \sum_{i=1}^{m} w(X_i)$ which implies

$$t_{on}(A) - t_d = \sum_{i=1}^{m} w(X_i). \tag{9}$$

For child nodes, we obtain $t_{on}(X_i) = t_{cpt}(X_i) = t_d + \sum_{k=1}^{i} w(X_k)$ which implies

$$t_{on}(X_i) - t_d = \sum_{k=1}^{i} w(X_k). \tag{10}$$

Maximization over t_{on}s results in $\tau_d = t_d + \sum_{i=1}^{m} w(X_i)$ and

$$\tau_d - t_d = \sum_{i=1}^{m} w(X_i). \tag{11}$$

From Equation 8 and S_d specified by Equation 9, 10, and 11, we derive S_c as follows:

For each leaf, we have $t_{off}(X_i) = \tau_d - t_{on}(X_i) + t_c = t_c + \sum_{k=i+1}^{m} w(X_k)$. For the root, we obtain $t_{off}(A) = t_{off}(X_m) = t_c$ and $\tau_c = t_{rdy}(A) = t_{off}(X_m) + \sum_{k=1}^{m} w(X_k) = t_c + \sum_{k=1}^{m} w(X_k)$. The above implies

$$\begin{aligned}
\tau_c - t_c &= \sum_{i=1}^{m} w(X_i) = \tau_d - t_d \\
\tau_c - t_{off}(A) &= \sum_{i=1}^{m} w(X_k) = t_{on}(A) - t_d \\
\tau_c - t_{off}(X_i) &= \sum_{k=1}^{i} w(X_k) = t_{on}(X_i) - t_d.
\end{aligned}$$

Therefore, the statement is true when $D = 1$.

Assume that the two conditions in the theorem hold for any R of depth $D = d \geq 1$.

Suppose we add some child nodes to the leaves of R at depth d to form a new tree R'. The depth of R' is $D = d + 1$. We show that the two conditions hold for R' as well. We will add a prime mark (') to a quantity (e.g., t_d, τ_c, and t_{off}) or an object (e.g., S_d and S_c) associated with R' to distinguish it from that associated with R.

Let the starting time t'_d of S'_d be the same as t_d of S_d, i.e., $t'_d = t_d$. Then, for each node Y of depth $\leq d - 1$ and each leaf Y of depth d, we have $t'_{on}(Y) = t_{on}(Y)$ which implies

$$t'_{on}(Y) - t'_d = t_{on}(Y) - t_d. \tag{12}$$

Let X be a node of depth d in R' with m leaf children: $Z_1, \ldots, Z_m$. Based on Equation 7, we have $t'_{on}(X) = t_{on}(X) + \sum_{j=1}^{m} w(Z_j)$ which implies

$$t'_{on}(X) - t'_d = t_{on}(X) - t_d + \sum_{j=1}^{m} w(Z_j). \tag{13}$$

For each leaf Z_j of depth $d+1$ with parent X, we obtain $t'_{on}(Z_j) = t_{on}(X) + \sum_{i=1}^{j} w(Z_i)$ which implies

$$t'_{on}(Z_j) - t'_d = t_{on}(X) - t_d + \sum_{i=1}^{j} w(Z_i) \tag{14}$$

The termination time is $\tau'_d = max(t'_{on}(X)) \geq \tau_d$ where maximization is over all nodes of R'. The above completely specifies S'_d.

We construct S'_c from S'_d based on Equation 8. Let us assign

$$t'_c = t_c - (\tau'_d - \tau_d) \tag{15}$$

which means that S'_c starts earlier than S_c by an amount of time by which S'_d is longer than S_d.

For each leaf Z_j of depth $d+1$ with parent X, we obtain

$$t'_{off}(Z_j) = \tau'_d - t'_{on}(Z_j) + t'_c = t_c + \tau_d - t_{on}(X) - \sum_{i=1}^{j} w(Z_i), \tag{16}$$

where the first equality is due to Equation 8 and the second equality is due to Equations 15 and 14. We also obtain

$$t'_{rdy}(X) = t'_{off}(Z_m) + \sum_{i=1}^{m} w(Zi) = t_c + \tau_d - t_{on}(X), \tag{17}$$

where the first equality is due to Equation 8 and the second equality is due to Equation 16. Equations 17 and 8 imply $t'_{rdy}(X) = t_{off}(X)$ for each X that is a leaf in R but a parent in R'.

For each leaf at depth $\leq d$,

$$t'_{off}(X) = \tau'_d - t'_{on}(X) + t'_c = t_c + \tau_d - t'_{on}(X) = t_c + \tau_d - t_{on}(X) = t_{off}(X), \tag{18}$$

where the second equality is due to Equation 15 and the third equality is due to Equation 12.

Note that $t'_{rdy}(X)$ of nodes of depth d and $t'_{off}(X)$ of leaves of depth $\leq d$ form a boundary condition for the timing of activities of all nodes above them. Therefore, Equations 17 and 18 together with the inductive assumption imply that, for each node of depth $\leq d$, the timing of its activity in S'_c (excluding the inward movement of in-agents from children of a node at depth d) is exactly the same as in S_c. We thus obtain $\tau'_c = \tau_c$ which implies the first condition in the theorem:

$$\tau'_c - t'_c = \tau_c - t_c + \tau'_d - \tau_d = \tau_d - t_d + \tau'_d - \tau_d = \tau'_d - t_d = \tau'_d - t'_d, \tag{19}$$

where the first equality is due to Equation 15 and the second equality is derived by the inductive assumption.

Finally, we consider $\tau'_c - t'_{off}()$ in terms of four exclusive and exhaustive cases of the node involved:

For each leaf Z_j of depth $d+1$ with parent X, we have

$$\begin{aligned} \tau'_c - t'_{off}(Z_j) &= (t'_c + \tau'_d - t_d) - (t_c + \tau_d - t_{on}(X) - \sum_{i=1}^{j} w(Z_i)) \\ &= -t_d + t_{on}(X) + \sum_{i=1}^{j} w(Z_i) = t'_{on}(Z_j) - t'_d, \end{aligned} \quad (20)$$

where the first equality is due to Equations 19 and 16, the second equality is due to Equation 15 and the third equality is due to Equation 14.

For each leaf X of depth $\leq d$, we obtain

$$\tau'_c - t'_{off}(X) = \tau'_c - (\tau'_d - t'_{on}(X) + t'_c) = \tau'_d - t'_d - (\tau'_d - t'_{on}(X)) = t'_{on}(X) - t'_d, \quad (21)$$

where the second equality is obtained using Equation 19.

For each non-leaf node X of depth d, we derive

$$\tau'_c - t'_{off}(X) = \tau_c - t'_{off}(Z_m) = \tau_c - (t_c + \tau_d - t_{on}(X) - \sum_{i=1}^{m} w(Z_i)) = t'_{on}(X) - t'_d, \quad (22)$$

where the second equality is due to Equation 16 and the third equality is due to Equation 13.

For each non-leaf node X of depth $< d$, since $\tau'_c = \tau_c$, $t'_{off}(X) = t_{off}(X)$, $t'_{on}(X) = t_{on}(X)$ and $t'_d = t_d$, we have

$$\tau'_c - t'_{off}(X) = t'_{on}(X) - t'_d. \quad (23)$$

The second condition of the theorem is proven. □

Proposition 26 **(Optimal distribution schedule in trees of depth 2)**

> Let R be a tree rooted at A for distribution. Let the depth of R be 2. Let the children of root be $X_1, \ldots, X_n$. Let the sum of out-weights of children of X_i be v_i such that $v_1 \leq v_2 \leq \ldots \leq v_n$. The absolute distribution off-line time Δ_{1-2} in R is minimized if the distribution order of A is $O_{out}(A) = (X_n, \ldots, X_1)$.

Proof:

According to the order $O^{(1)}_{out}(A) = (X_n, \ldots, X_1)$, the distribution time is

$$\begin{aligned} \Delta^{(1)}_{1-2} = max(\quad & w(X_n) + v_n, \\ & w(X_n) + w(X_{n-1}) + v_{n-1}, \end{aligned}$$

$$
\begin{aligned}
&\ldots,\\
&w(X_n) + \ldots + w(X_i) + v_i,\\
&\ldots,\\
&w(X_n) + \ldots + w(X_1) + v_1)
\end{aligned}
$$

where we have written $w(X_i)$ instead of $w_{out}(X_i)$ for simplicity. Note we have listed the entries in $max()$ in the order consistent with the order in which X_i appears in $O^{(1)}_{out}(A)$. In this order, if $w(X_i)$ appears in an entry, it appears in every entry to its right. Note also that each entry has exactly one v among its addends.

Let $O^{(2)}_{out}(A) = (X'_n, \ldots, X'_1)$ be any order distinct from $O^{(1)}_{out}(A)$ where X'_i is some X_k ($k \in \{1 \ldots n\}$) and $X'_i \neq X'_j$ for $i \neq j$. We denote the weight associated with X'_i by $w(X'_i)$, and the sum of weights of children of X'_i by v'_i. The distribution time under $O^{(2)}_{out}(A)$ is

$$\Delta^{(2)}_{1-2} = max(e'_n, e'_{n-1}, \ldots, e'_1) \text{ where } e'_i = (\sum_{k=i}^{n} w(X'_k)) + v'_i.$$

First we simplify $\Delta^{(2)}_{1-2}$ by removing superfluous entries. Suppose v_n is an addend of e'_d and $d < n$, or equivalently, X_n is not the first in $O^{(2)}_{out}(A)$. Then $e'_n, \ldots, e'_{d+1}$ can be eliminated from $max()$ without altering the value of $\Delta^{(2)}_{1-2}$, since $v_n = max^n_{i=1}(v_i)$ and therefore $e'_i < e'_d$ when $i > d$. After the elimination, we have $\Delta^{(2)}_{1-2} = max(e'_d, e'_{d-1}, \ldots, e'_1)$.

Suppose now v_m ($m \in \{1 \ldots n-1\}$) is an addend of e'_c, and m is the highest index value among vs contained in e'_{d-1} through e'_1. With the same argument as above, $e'_{d-1}, \ldots, e'_{c+1}$ can all be eliminated from $max()$.

Repeating this process, we end up with $\Delta^{(2)}_{1-2} = max(e'_d, e'_c, \ldots, e'_a)$ where v_n is an addend of e'_d, v_m is an addend of e'_c, $\ldots$, v_l is an addend of e'_a, such that $n > m > \ldots > l$. We will refer to the expression of $\Delta^{(2)}_{1-2}$ before simplification as the *expanded* form, and refer to the expression after simplification as the *simplified* form.

(Example) suppose $O^{(2)}_{out}(A) = (X_4, X_5, X_2, X_3, X_1)$. The expanded form of $\Delta^{(2)}_{1-2}$ is

$$
\begin{aligned}
\Delta^{(2)}_{1-2} \;=\; & max(w(X_4) + v_4, w(X_4) + w(X_5) + v_5, w(X_4) + w(X_5) + w(X_2) + v_2,\\
& w(X_4) + w(X_5) + w(X_2) + w(X_3) + v_3,\\
& w(X_4) + w(X_5) + w(X_2) + w(X_3) + w(X_1) + v_1).
\end{aligned}
$$

After removing superfluous entries, the simplified form is

$$
\begin{aligned}
\Delta^{(2)}_{1-2} \;=\; & max(w(X_4) + w(X_5) + v_5, w(X_4) + w(X_5) + w(X_2) + w(X_3) + v_3,\\
& w(X_4) + w(X_5) + w(X_2) + w(X_3) + w(X_1) + v_1).
\end{aligned}
$$

Next, we prove a lemma which states the following:

(Lemma) After $\Delta^{(2)}_{1-2}$ is simplified into $\Delta^{(2)}_{1-2} = max(e'_d, \ldots, e'_c, \ldots, e'_a)$, if $e'_c = (\sum w(X_i)) + w(X_m) + v_m$, then all of $w(X_n), \ldots, w(X_{m+1})$ are addends in $\sum w(X_i)$.

It suffices to show that e'_c has all of $w(X_n), \ldots, w(X_{m+1})$ as its addends.

Consider v_i $(n \geq i \geq m+1)$. In the expanded form of $\Delta^{(2)}_{1-2}$, v_i appears as an addend in an entry that is either before (to the left of) e'_c or after e'_c. In the following, we simply say that v_i appears before (after) e'_c.

If v_i appears before e'_c, then e'_c must contain the addend $w(X_i)$. It can not appear after e'_c, since entries in the simplified $\Delta^{(2)}_{1-2}$ has a decreasing order for the index of addend v. If v_i had appeared after e'_c in the expanded form of $\Delta^{(2)}_{1-2}$, then e'_c would have been eliminated in the simplification process. The lemma is then proven.

Finally, we show that $\Delta^{(1)}_{1-2} \leq \Delta^{(2)}_{1-2}$ by identifying one entry e'_c in the simplified $\Delta^{(2)}_{1-2}$ such that $\Delta^{(1)}_{1-2} \leq e'_c$.

Suppose $\Delta^{(1)}_{1-2} = (\sum_{i=k}^{n} w(X_i)) + v_k$ $(k \in \{1, \ldots, n\})$. We search for an entry e'_c in the simplified $\Delta^{(2)}_{1-2}$ such that e'_c has v_k as an addend. We consider the following three exclusive and exhaustive cases:

(Case 1) If such an e'_c is found, by the above lemma, we have $\Delta^{(1)}_{1-2} \leq e'_c$.

If an entry with the addend v_k is not found, we search for an entry e'_c with an entry e'_b next to its right such that e'_c has an addend v_l $(l > k)$ and e'_b has an addend v_j $(k > j)$.

(Case 2) If such e'_c and e'_b are found, we show that e'_c has all of $w(X_n), \ldots, w(X_k)$ as its addends.

By the lemma above, e'_c has all of $w(X_n), \ldots, w(X_l)$ as its addends. We need to show that e'_c also has all of $w(X_{l-1}), \ldots, w(X_k)$ as its addends.

By the lemma, e'_b must have all of $w(X_{l-1}), \ldots, w(X_k)$ as its addends. Therefore, in the expanded form of $\Delta^{(2)}_{1-2}$, all of $v_{l-1}, \ldots, v_k$ must have appeared as addends in entries to the left of e'_b. The question is whether they appear before e'_c or after e'_c.

If any v_i $(i \in \{l-1, \ldots, k\})$ had appeared in an entry after e'_c in the expanded form of $\Delta^{(2)}_{1-2}$, since $i > j$, at least one such entry appeared would have been included in the simplified $\Delta^{(2)}_{1-2}$. Therefore, each v_i $(i \in \{l-1, \ldots, k\})$ must have appeared before e'_c, and e'_c has $w(X_i)$ for $i = l-1, \ldots, k$ as addends.

(Case 3) If the above specified e'_c is found but there exists no e'_b as specified above, i.e., e'_c is the right-most entry in the simplified form of $\Delta^{(2)}_{1-2}$, we show that e'_c contains addends $w(X_n), \ldots, w(X_k)$.

Consider the left-most entry e'_1 in the expanded form of $\Delta^{(2)}_{1-2}$. This entry contains all of $w(X_n), \ldots, w(X_1)$. The simplification process does not eliminate the right-most entry of the expanded $\Delta^{(2)}_{1-2}$. Therefore, the simplified form has the identical right-most entry as the expanded form: $e'_c = e'_1$ which contains addends $w(X_n), \ldots, w(X_k)$.

We have shown that $\Delta^{(1)}_{1-2} \leq \Delta^{(2)}_{1-2}$ holds for an arbitrary order $O^{(2)}_{out}(A)$. Therefore, $O^{(1)}_{out}(A)$ minimizes Δ_{1-2}. $\square$

Lemma 32

Let Y be a non-leaf node in a rooted tree. Let the child nodes of Y be $X_1, \ldots, X_m$ $(m > 0)$ where X_k has n_k descendants. The summation $\psi = \sum_{k=1}^{m}(n_k + 1)(t_{cpt}(X_k) - t_{cpt}(Y))$ is minimized if

$$w_{out}(X_1)/(n_1 + 1) \leq w_{out}(X_2)/(n_2 + 1) \leq \ldots \leq w_{out}(X_m)/(n_m + 1)$$

and distribution order of Y is $O_{out}(Y) = (X_1, \ldots, X_m)$.

Proof:

When $O_{out}(Y)$ is followed, we have $t_{cpt}(X_k) - t_{cpt}(Y) = \sum_{i=1}^{k} w(X_i)$, and therefore

$$\begin{aligned}
\psi &= \sum_{k=1}^{m}(n_k + 1)\sum_{i=1}^{k} w(X_i) \\
&= \sum_{i=1}^{1}(n_1 + 1)w(X_i) + \sum_{i=1}^{2}(n_2 + 1)w(X_i) + \ldots + \sum_{i=1}^{m}(n_m + 1)w(X_i) \\
&= w(X_1)\sum_{i=1}^{m}(n_i + 1) + w(X_2)\sum_{i=2}^{m}(n_i + 1) + \ldots + w(X_m)\sum_{i=m}^{m}(n_i + 1)
\end{aligned}$$

where we have written $w(X_i)$ instead of $w_{out}(X_i)$ for simplicity.

Assume that the following condition holds:

$$w(X_1)/(n_1 + 1) \leq w(X_2)/(n_2 + 1) \leq \ldots \leq w(X_m)/(n_m + 1)$$

When $m = 2$, $\psi = w(X_1)(n_1 + 1 + n_2 + 1) + w(X_2)(n_2 + 1)$ if $O_{out}(Y)$ is followed. If instead the other possible order $O'_{out}(Y) = (X_2, X_1)$ is followed, using the similar derivation, we have $\psi' = w(X_2)(n_1 + 1 + n_2 + 1) + w(X_1)(n_1 + 1)$. The difference $\psi' - \psi = w(X_2)(n_1 + 1) - w(X_1)(n_2 + 1) \geq 0$ since $w(X_1)/(n_1 + 1) \leq w(X_2)/(n_2 + 1)$ by assumption.

In general, for each X_k, ψ contains exactly m X_k-related addends, we denote their sum by P:

$$\begin{aligned}
P &= (n_k + 1)w(X_1) + \ldots + (n_k + 1)w(X_{k-1}) + (n_k + 1)w(X_k) \\
&+ (n_{k+1} + 1)w(X_k) + \ldots + (n_m + 1)w(X_k)
\end{aligned}$$

Note that each addend in P (in the form $(n_i + 1)w(X_j)$) is unique.

Given an arbitrary distribution order $O'_{out}(Y) = (X_{1'}, \ldots, X_{m'})$ where $i' \in \{1, \ldots, m\}$ and where $l' = k$, i.e., X_k appears at lth location in $O'_{out}(Y)$, we also have exactly m X_k-related addends in ψ'. we denote their sum by P':

$$\begin{aligned}
P' &= (n_k + 1)w(X_{1'}) + \ldots + (n_k + 1)w(X_{l-1'}) + (n_k + 1)w(X_k) \\
&+ (n_{l+1'} + 1)w(X_k) + \ldots + (n_{m'} + 1)w(X_k)
\end{aligned}$$

Note that each addend in P' is also unique.

We show the $P' - P \geq 0$ for every X_k. To show that, it is sufficient to create an one-to-one correspondence f from the set of addends of P' to the set of addends of P such that $p' - p \geq 0$ where p' is an addend of P' and $p = f(p')$.

A typical addend of P' is either in the form $(n_k+1)w(X_{i'})$ or in the form $(n_{i'}+1)w(X_k)$.

Suppose $p' = (n_k + 1)w(X_{i'})$. If $i' < k$, there exists a unique $p = (n_k + 1)w(X_{i'})$ in P and $p' - p = 0$. If $i' \geq k$, there exists a unique $p = (n_{i'} + 1)w(X_k)$ in P and $p' - p \geq 0$ since $w(X_{i'})/(n_{i'} + 1) \geq w(X_k)/(n_k + 1)$ by assumption.

Suppose $p' = (n_{i'} + 1)w(X_k)$. If $i' < k$, there exists a unique $p = (n_k + 1)w(X_{i'})$ in P and $p' - p \geq 0$ since $w(X_{i'})/(n_{i'} + 1) \leq w(X_k)/(n_k + 1)$ by assumption. If $i' \geq k$, there exists a unique $p = (n_{i'} + 1)w(X_k)$ in P and $p' - p = 0$.

Therefore, $P' - P \geq 0$ for every X_k, and the order $O_{out}(Y)$ minimize ψ. □

Acknowledgement

This work is supported by the Dean's Research Funding from Faculty of Science, University of Regina, the General NSERC Grant from University of Regina, and Research Grant OGP0155425 from NSERC.

References

[1] A.H. Bond and L. Gasser. An analysis of problems and research in DAI. In A.H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 3–35. Morgan Kaufmann, 1988.

[2] A.H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.

[3] R. Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[4] A.P. Dawid and S.L. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21(3):1272–1317, 1993.

[5] L.D. Erman, F A. Hayes-Roth, V.R. Lesser, and D.R. Reddy. The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, 1980.

[6] L. Gasser and M.N. Huhns, editors. *Distributed Artificial Intelligence, Volume II*. Morgan Kaufmann, 1989.

[7] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, 1991.

[8] C.E. Hewitt. Offices are open systems. *ACM Trans. on Office Information Systems*, 4(3):271–287, 1986.

[9] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, (4):269–282, 1990.

[10] F.V. Jensen, K.G. Olesen, and S.K. Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20:637–659, 1990.

[11] S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, (50):157–244, 1988.

[12] V.R. Lesser and L.D. Erman. Distributed interpretation: a model and experiment. *IEEE Trans. on Computers*, C-29(12):1144–1163, 1980.

[13] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems*. John Wiley and Sons, 1990.

[14] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, (29):241–288, 1986.

[15] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[16] W.A. Shay. *Understanding Data Communications and Networks*. PWS Publishing, 1995.

[17] A. Silberschatz and P.B. Galvin. *Operating System Concepts*. Addison Wesley, 1994.

[18] Y. Xiang, B. Pant, A. Eisen, M. P. Beddoes, and D. Poole. Multiply sectioned Bayesian networks for neuromuscular diagnosis. *Artificial Intelligence in Medicine*, 5:293–314, 1993.

[19] Y. Xiang, D. Poole, and M. P. Beddoes. Multiply sectioned Bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, 9(2):171–220, 1993.