A Decision-Theoretic Graphical Model For Collaborative Design On Supply Chains

Y. Xiang*, J. Chen*, and A. Deshmukh†

* University of Guelph, Canada
† University of Massachusetts at Amherst, USA

Abstract. We propose a decision-theoretic graphical model for collaborative design in a supply chain. The graphical model encodes the uncertain performance of a product resultant from an integrated design distributively. It represents preference of multiple manufacturers and end-users such that a decision-theoretic design is well-defined. We show that these distributed design information can be represented in a multiply sectioned Bayesian network. This result places collaborative design in a formal framework so that it can be subject to rigorous algorithmic study.

1 Introduction

Supply chain literature has focused mostly on supply chain management [4]. This work emphasizes collaborative design in a supply chain. We consider component-centered design in which a final product is designed as a set of components supplied by manufacturers in a supply chain. Hence, the manufacturers are collaborative designers. We interpret design under broad design-for-X (DFX) concepts including design for assembly, manufacture, disassembly, environment, recyclability, etc, with the objective of producing an overall optimal performance.

From the management perspective, a supply chain consists of business entities such as customers, retailers, manufacturers, and suppliers [4]. We adopt an abstraction of the supply chain from the design perspective: The supply chain includes only entities directly involved in the collaborative design and each is referred to as a *manufacturer*. In the simplest case, a supply chain produces a single final product t and we abstract all entities not directly involved in the design (e.g., retailers, distributors and customers) of t as the *end user*. We regard the end user as outside the supply chain.

Such a supply chain can be modeled by a directed acyclic graph (DAG). Each node corresponds to a manufacturer. Each arc corresponds to a supplying relation and is directed from a *consumer* to a *supplier* (both are manufacturers). The role of consumer or supplier is relative, depending on which supplying relation is in question. A manufacturer is a consumer relative to an outgoing arc and is a supplier relative to an incoming arc. The root (node without parent nodes) R of the supply chain is the manufacturer of the final product, which supplies the product to the end user. The leaves are suppliers who are not consumers in the supply chain.

Contemporary practice of design in a supply chain is either centralized or essentially top-down [4]. In a top-down design, R designs the product by decomposing it into components. For each component C to be supplied to R, a supplier S further decomposes C into subcomponents, and the process continues. In summary, consumers play dominant roles in the design and suppliers are more passive. Such consumer-dominant designs are unlikely to be optimal because consumers are unlikely to be in the best position to judge the options available to suppliers. We propose a computational framework for collaborative design where suppliers play equally active roles in shaping the final design.

Given a design, the actual performance of the final product is uncertain due to the uncertainties in raw materials used, in the actual manufacturing process, and in the operating environment. Conventional approach of deterministic design is undesirable because it assumes representative maximum loads and minimum material property, which often leads to overdesign and inability to risk evaluation. Probabilistic approach optimizes design in the face of these uncertainties [2, 1]. We extend the probabilistic approach to a decision-theoretic approach which incorporates explicitly the preference of manufactures and end users, and to collaborative design which requires distributed reasoning.

Many of the objectives of this work have been articulated in the literature informally, e.g., [7, 4]. The key contribution of this work is the proposal of a formal decision-theoretic graphical model that explicitly encodes the information on supply chain organization, design constraints, performance measures, utilities of manufactures and end users. Such model provides a rigorous multiagent framework for computational study of collaborative design and for distributed decision aids to collaborative designers.

In Section 2, we define the problem of decision-theoretic design in a centralized context, which is expressed as a graphical model in Section 3. In Section 4, we cover the background on multiply sectioned Bayesian networks (MSBNs) as a graphical model for multiagent inference. We extend MSBNs into a knowledge representation formalism for multiagent collaborative design in Section 5. We outline the research issues and future work in Section 6.

2 Decision-Theoretic Design

A product has a *design space* described by a set D of variables. Each variable in D is called a *design parameter*. For example, the type of CPU used in a smart appliance is a design parameter and the height of the case is another. As mentioned in Section 1, we interpret design under broad concepts of DFX and hence design parameters are interpreted broadly as well. In this paper, we assume that each design parameter is associated with a discrete domain of possible values and a naturally continuous parameter is discretized. A *(complete) design* is an assignment of values to all variables in D and a *partial design* is an assignment of values to variables in a proper subset of D.

An assignment of values to variables in D must satisfy a set of constraints. Otherwise, the design cannot be realized. For instance, a computer design with a case of length L and a motherboard of length $L' > L$ cannot be realized. A constraint involves a subset $S \subset D$ of variables and specifies the allowable combinations of values for S [10]. A design (partial or complete) is *invalid* if it violates one or more constraints. Otherwise, the design is *valid*.

Different designs result in products with different performances. Given a design with a given performance, it may or may not be desirable to different people. We use the term *performance* to refer to *objective* measures of the functionalities of a product resultant from a design. That is, the value of a performance measure is independent of who is making the measurement. For example, the maximum speed of a car is a performance measure. For simplicity, we refer to the performance of a product resultant from a design as the performance of the design. The *performance space* of a product is described by a set M of variables. Each variable in M is called a *performance measure*. In this paper, we assume that each performance measure is associated with a discrete domain and a naturally continuous measure is discretized.

The performance of a product also depends on the environment in which it operates, for instance, high level of humidity may cause a digital system to fail. We describe such environmental factors by a set T of discrete variables. For each $t \in T$, a probability distribution quantifies the uncertainty over its possible values. Hence, formally, each performance measure is a function of a subset of $D \cup T$.

A principal's *subjective* preference of a design is represented by a *utility function* [6]. We assume that the utility of a design is directly dependent on the performance of the corresponding product. That is, the utility does not depend on directly on design parameters. This is mainly for conceptual and representational clarity. For any practical situation where the utility directly depends on some design parameters, it is always possible to introduce one or more performance measures to mediate the dependency. Formally, we denote the utility function as $U(M)$.

For simplicity of representation and acquisition, we assume that the utility function $U(M)$ satisfies the *additive independence* condition [6] and can be decomposed as follows: First, the performance measures are partitioned into groups $M_0, M_1, ...$ Each group M_i is associated with a utility function $U_i(M_i)$ whose value is normalized to $[0, 1]$. The overall utility function $U(M)$ satisfies $U(M) = \sum_i k_i \; U_i(M_i)$, where each weight $k_i \in (0, 1)$ and $\sum_i k_i = 1$.

Due to the uncertainty in the performance of the product given its design, we can only evaluate the expected utility of a design. Denote a given design by $D = \mathbf{d}$. Denote one possible combination of performance values of the resultant product by $M = \mathbf{m}$. Then the probability $P(\mathbf{m}|\mathbf{d})$ expresses the likelihood of performance $\mathbf{m}$ of the product resultant from design $\mathbf{d}$. The expected utility of $\mathbf{d}$ relative to utility $U_i()$ is $EU_i(\mathbf{d}) = \sum_{\mathbf{m}} U_i(\mathbf{m}) \; P(\mathbf{m}|\mathbf{d})$, where $U_i(\mathbf{m})$ is evaluated according to $U_i(M_i)$ with the values in $\mathbf{m}$ for variables outside M_i dropped. The expected utility of design $\mathbf{d}$ is then $EU(\mathbf{d}) = \sum_i k_i \; (\sum_{\mathbf{m}} U_i(\mathbf{m}) \; P(\mathbf{m}|\mathbf{d}))$. The problem of a decision-theoretic design given (D, T, M, U) can be specified as to find a valid design $\mathbf{d}^*$ that maximizes $EU(\mathbf{d})$ among all valid designs.

3 Graphical Model Representation

In order to compute $EU(\mathbf{d})$ effectively, we represent the design problem as a graphical model, which we term as a *design network*. The structure of the network is a DAG $G = (V, E)$. The set of nodes is $V = D \cup T \cup M \cup U$, where, without confusion, we overload the notation U to denote a subset of nodes each corresponding to a utility function $U_i()$.

Each arc in E is denoted (p, c) directed from the parent node p to the child node c. As V consists of four types of nodes, there are 16 potential types of arcs. We define the following 5 types as *legal* arcs:

1. Arc (d, d') between design parameters $d \in D$ and $d' \in D$.

   The arc signifies that the two parameters are involved in a constraint.

2. Arc (d, m) from a design parameter $d \in D$ to a performance measure $m \in M$.

   The arc signifies that the performance measure m depends on the design parameter d.

3. Arc (t, m) from an environment variable $t \in T$ to $m \in M$.

4. Arc (m, m') between performance measures $m \in M$ and $m' \in M$.

   The arc signifies that m' is a composite performance measure function. Because m is also a performance measure function which may or may not be composite, all root ancestors of m must be members of D or T. Fig. 1 (a) illustrates such a case, where $m' = f(m(d_1, d_2), d_3, t)$.

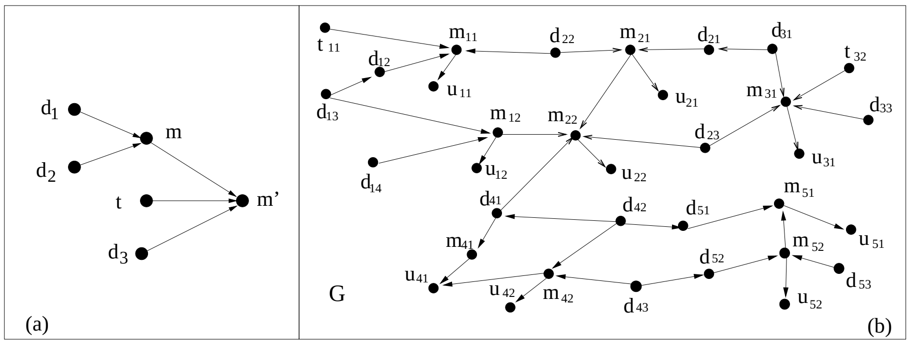

Fig. 1. (a) Performance measure m' as a composite function. (b) Graphical model of a design network.

5. Arc (m, u) from a performance measure $m \in M$ to a utility node $u \in U$.

   The arc signifies that the utility u depends on the performance measure m.

The other 11 types of arcs are (m, d), (u, d), (u, m), (d, t), (t, d), (t, t'), (m, t), (t, u), (u, t), (d, u) and (u, u'). The first 9 of them are inconsistent with the interpretation of environment factor, performance measure and utility. Thus they are illegal. Arc (d, u) is regarded illegal for reasons of clarity as explained in Section 2. Arc (u,u) allows arbitrary integration of utilities and enables specification of non-additive utilities. It is a source of potential violation of the additive independence assumption discussed in Section 2. We therefore regard such arcs as illegal at the current stage of this research. Fig. 1 (b) shows a graphical model for design.

Each node in the design network is associated with a numerical distribution of the corresponding variable conditioned on its parent variables. The distribution at a design parameter encodes a design constraint and that at an environment variable encodes uncertainty on operating condition. The distribution at a performance measure encodes a performance function and that at a utility node encodes a utility function. We propose the following representation such that each distribution is *syntactically* a conditional probability distribution. Such uniform representation allows design computation to be conducted using existing inference algorithms for probabilistic networks [9, 5, 8, 11].

Consider first a constraint over a subset $X \subset D$ of design parameters. Jensen [5] proposed a probabilistic encoding of a constraint over a set X of variables by introducing an additional binary variable $c \in \{y, n\}$ with its parent set X. For each combination $\mathbf{x}$ of values for variables in X, if it is allowable according to the constraint, then $P(c = y|\mathbf{x})$ is assigned the value 1. Otherwise, it is assigned the value 0. The distribution $P(c|X)$ has the space complexity of $O(2^{|X|+1})$.

We propose an alternative representation that is more compact. Consider a constraint over 10 binary variables $d_0, ..., d_9$. Suppose that all combinations of $d_0, ..., d_8$ are allowed. However, for some combinations of $d_0, ..., d_8$, some values of d_9 are not allowed. We represent this constraint by assigning $d_0, ..., d_8$ as the parents of d_9. For each combination of values of $d_0, ..., d_9$, if it is allowable, we assign $P(d_9|d_0, ..., d_8) = 1$. Otherwise, we assign $P(d_9|d_0, ..., d_8) = 0$. The size of $P(d_9|d_0, ..., d_8)$ is then $2^{10} = 1024$ instead of $2^{11} = 2048$ as the alternative representation.

It is possible that not all combinations of $d_0, ..., d_8$ are allowed either. In general, for any subset of two or more elements, certain combinations of values may be disallowed. The following general representation can be used: Let d_0 be the parent of d_1 and let $P(d_1|d_0)$ be assigned similarly as above. Then, Let d_0, d_1 be the parent of d_2 and let $P(d_2|d_0, d_1)$ be assigned. Repeat the process until $P(d_9|d_0, ..., d_8)$ is assigned. For each d_i above, if no value of d_i is disallowed for any allowable combination of $d_0, ..., d_{i-1}$, then the corresponding step can be skipped. This provides the best case space complexity of 1024 and the worst case complexity of $2^2 + 2^3 + ... + 2^{10} = 2044$. The alternative representation has the constant space complexity of 2048. Our method does not need to introduce additional nodes.

It follows from the types of legal arcs that all root nodes of G are elements of either D or T. Each root node is associated with a unconditional probability distribution. $P(d)$ for $d \in D$ suggests the most commonly used values of d, given no other information, and $P(t)$ for $t \in T$ reflects the uncertain environment condition.

For each node $m \in M$ whose parents $X \subset D \cup T$, $P(m|X)$ is a typical probability distribution representing the likelihood of performance values given partial designs over X. If m is a composite function, in which case, at least one of its parent m' is an element of M, $P(m|X)$ is interpreted similarly by taking into account $P(m'|X')$, where X' is the parents of m'. For the example in Fig. 1 (a), $P(m|d_1, d_2)$ and $P(m'|m, d_3, t)$ together define the likelihood of performance values of m' given partial designs over d_1, d_2, d_3 with environment condition t.

For each utility function $U_i(M_i)$, there is a utility node u_i in G with its parent set being M_i. We assign the corresponding variable u_i a binary domain $\{y, n\}$. We assign the distribution at u_i as $P(u_i = y|M_i) = U_i(M_i)$. We specify $P(u_i = n|M_i) = 1 - P(u_i = y|M_i)$. Thus, $P(u_i|M_i)$ is a syntactically valid probability distribution but semantically encodes the utility function $U_i(M_i)$.

From the above specification of the design network, it is a simple matter to verify that in its structure G, graphical separation (strictly speaking, d-separation [9]) corresponds to conditional independence. Since the design network encodes the probabilistic dependence between design and performance as well as the utility of a principal, it is semantically a *value network*. On the other hand, syntactically, the design network is a Bayesian network [9]. It is straightforward to show that $P(D)$ can be obtained by the product of distributions associated with nodes in D and it represents the likelihood of each potential design. $P(D, M)$ can be obtained by the product of distributions associated with nodes in $D \cup M$ and it represents the likelihood of each valid design and each possible performance of the design.

Suppose we perform a standard probabilistic inference in the above specified network with a valid design $\mathbf{d}$ entered into the corresponding nodes. After belief propagation, $P(M_i|\mathbf{d})$ can be obtained. At node u_i, we have

$$\begin{aligned} P(u_i = y|\mathbf{d}) &= \textstyle\sum_{\mathbf{m_i}} P(u_i = y|\mathbf{m_i}, \mathbf{d})P(\mathbf{m_i}|\mathbf{d}) = \sum_{\mathbf{m_i}} P(u_i = y|\mathbf{m_i})P(\mathbf{m_i}|\mathbf{d}) \\ &= \textstyle\sum_{\mathbf{m_i}} U_i(\mathbf{m_i})P(\mathbf{m_i}|\mathbf{d}) = EU_i(\mathbf{d}). \end{aligned}$$

The first equation above is the probabilistic inference known as *reasoning by case*. The second equation is due to the independence encoded in the network. In this case, u_i is graphically separated from design parameters by the parent nodes of u_i, namely, M_i. This equation represents the normal inference computation at node u_i during belief propagation. The third equation holds due to the assignment of the distribution to u_i. Hence, $EU_i(\mathbf{d})$ can be obtained at the node u_i after standard belief propagation. Similar idea is explored by Cooper [3] in the context of decision making in influence diagrams by using probabilistic inference.

In addition to the distribution $P(u_i|M_i)$ associated with each utility node u_i, we also associate u_i with the weight k_i (see Section 2). Hence, by integrating $EU_i(\mathbf{d})$ for all i, $EU(\mathbf{d})$ can be obtained for a given design $\mathbf{d}$ after standard probabilistic inference in the design network.

4 Overview of Multiply Sectioned Bayesian Networks

In this section, we briefly review the background on multiagent distributed probabilistic reasoning using a representation known as Multiply Sectioned Bayesian Networks (MSBNs) [11]. In the next section, we show how to represent knowledge for collaborative design in a supply chain as an MSBN. Our choice using

MSBNs is deeply rooted: From a few high level requirements, namely, (1) exact probabilistic measure of agent belief, (2) agent communication by belief over small sets of shared variables, (3) a simpler agent organization, (4) DAG domain structuring, and (5) joint belief admitting agents' belief on private variables and combining their beliefs on shared variables, it has been shown [12] that the resultant representation of a cooperative multiagent system is an MSBN or some equivalent.

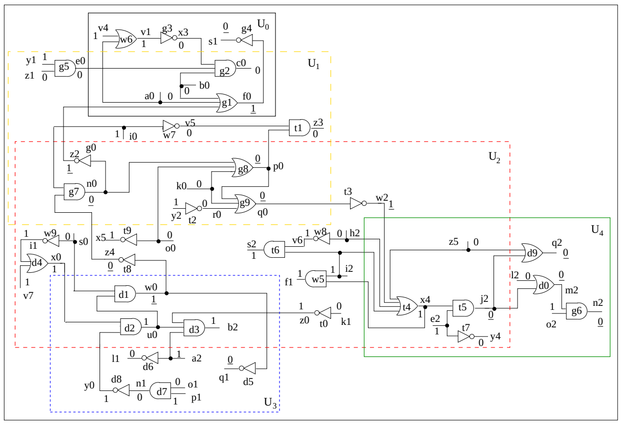

Fig. 2. A digital system consisting of 5 components.

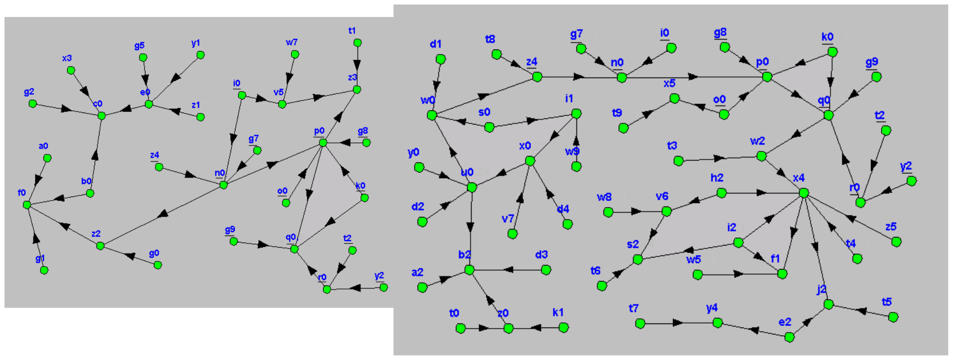

Fig. 3. Left: The subnet G_1 for U_1. Right: The subnet G_2 for U_2.

A Bayesian Network (BN) [9] S is a triplet (V, G, P) where V is a set of domain variables, G is a DAG whose nodes are labeled by elements of V, and P is a joint probability distribution (jpd) over V, specified in terms of a distribution for each variable $x \in V$ conditioned on the parents $\pi(x)$ of x in G. An MSBN M is a collection of Bayesian subnets that together define a BN. The most well-studied application of MSBNs is diagnosis and monitoring and we will use such an example to illustrate: Fig. 2 shows a piece of digital equipment made out of five components U_i $(i = 0, ..., 4)$.

Each box in the figure corresponds to a component and contains the logical gates and their connections with the input/output signals of each gate labeled.

A set of five agents, A_i $(i = 0, ..., 4)$, cooperate to monitor the system and trouble-shoot it when necessary. Each agent A_i is responsible for a particular component U_i. The knowledge of an agent about its assigned component can be represented as a BN, called a *subnet*. The subnet for agent A_1 is shown in Fig. 3 (left) and that for A_2 is shown in the right. Each node is labeled with a variable name. Only the DAG of the subnet is shown in the figure with the conditional probability distribution for each variable omitted. The five subnets (one for each component) collectively define an MSBN, which form the core knowledge of the multiagent system. Based on this knowledge and limited observations, agents can cooperate to reason about whether the system is functioning normally, and if not, which devices are likely to be responsible.

To ensure correct, distributed probabilistic inference, subnets in an MSBN are required to satisfy certain conditions. To describe these conditions, we introduce the terminologies first. Let $G_i = (V_i, E_i)$ $(i = 0, 1)$ be two graphs (directed or undirected). G_0 and G_1 are said to be *graph-consistent* if the subgraphs of G_0 and G_1 spanned by $V_0 \cap V_1$ are identical. Given two graph-consistent graphs $G_i = (V_i, E_i)$ $(i = 0, 1)$, the graph $G = (V_0 \cup V_1, E_0 \cup E_1)$ is called the *union* of G_0 and G_1, denoted by $G = G_0 \cup G_1$. Given a graph $G = (V, E)$, a partition of V into V_0 and V_1 such that $V_0 \cup V_1 = V$ and $V_0 \cap V_1 \neq \emptyset$, and subgraphs G_i of G spanned by V_i $(i = 0, 1)$, G is said to be *sectioned* into G_0 and G_1. See Fig. 4 for an example. Note that if G_0 and G_1 are sectioned from a third graph, then G_0

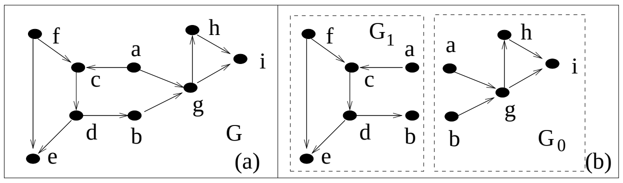

Fig. 4. G in (a) is sectioned into G_0 and G_1 in (b). G is the union of G_0 and G_1.

and G_1 are graph-consistent. The union of multiple graphs and the sectioning of a graph into multiple graphs can be similarly defined.

Graph sectioning is useful in defining the dependence relation between variables shared by agents. It is used to specify the following hypertree condition which must be satisfied by subnets in an MSBN:

Definition 1 *Let $G = (V, E)$ be a connected graph sectioned into subgraphs $\{G_i = (V_i, E_i)\}$. Let the subgraphs be organized into an undirected tree Ψ where each node is uniquely labeled by a G_i and each link between G_k and G_m is labeled by the non-empty* `interface` *$V_k \cap V_m$ such that for each i and j, $V_i \cap V_j$ is contained in each subgraph on the path between G_i and G_j in Ψ. Then Ψ is a* `hypertree` *over G. Each G_i is a* `hypernode` *and each interface is a* `hyperlink`*.*

Fig. 5 illustrates a hypertree for the digital system, where G_1 and G_2 are shown in Fig. 3.

The hypertree represents an organization of agent communication, where variables in each hypernode are local to an agent and variables in each hyperlink are shared by agents. Agents communicate in an MSBN by exchanging their beliefs over shared variables. We use *nodes* and variables interchangeably when

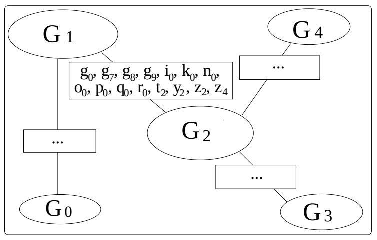

Fig. 5. The hypertree for the digital equipment monitoring system.

there is no confusion. Nodes shared by subnets in an MSBN must form a *d-sepset*, as defined below:

Definition 2 *Let G be a directed graph such that a hypertree over G exists. A node x contained in more than one subgraph with its parents $\pi(x)$ in G is a* `d-sepnode` *if there exists at least one subgraph that contains $\pi(x)$. An interface I is a* `d-sepset` *if every $x \in I$ is a d-sepnode.*

The interface between G_1 and G_2 contains 13 variables indicated in Fig. 5. The corresponding nodes in Fig. 3 are underlined. It is a d-sepset because these variables are only shared by G_1 and G_2, and each variable has all its parents contained in one of them. For instance, the parents of z_4 (t_8 and w_0) are all contained in G_2, while those of n_0 (i_0, g_7 and z_4) are contained in both G_1 and G_2 (see Fig. 3). The structure of an MSBN is a multiply sectioned DAG (MSDAG) with a hypertree organization:

Definition 3 *A* `hypertree MSDAG` *$G = \bigcup_i G_i$, where each G_i is a DAG, is a connected DAG such that (1) there exists a hypertree Ψ over G, and (2) each hyperlink in Ψ is a d-sepset.*

Note that although DAGs in a hypertree MSDAG form a tree, each DAG may be multiply connected: A *loop* in a graph is a sequence of nodes $a, b, c, ..., a$ such that the first node is identical to the last node and there is a link (not necessarily in the same direction) between each pair of nodes adjacent in the sequence. Such a loop is also referred to as an *undirected loop*. A DAG is *multiply connected* if

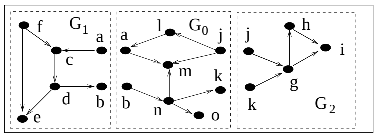

Fig. 6. A MSDAG with multiple paths across local DAGs.

it contains at least one (undirected) loop. Otherwise, it is singly connected. For example, G_1 in Fig. 3 has two loops. One of them is $(i_0, v_5, z_3, p_0, n_0, i_0)$. Hence,

G_1 is multiply connected. G_2 has several loops and is also multiply connected. Moreover, multiple paths may exist from a node in one DAG to another node in a different DAG after the DAGs are unioned. For instance, in Fig. 6, there are several (undirected) paths from node c in G_1 to node g in G_2. There is one path going through nodes a, l and j and another path goes through d, b, n and k. Each path goes across all three DAGs.

An MSBN is then defined as follows. Uniform potentials (constant distributions) are used to ensure that quantitative knowledge about the strength of dependence of a variable on its parent variables will not be doubly specified for the same variable.

Definition 4 *An MSBN M is a triplet $(V, G, \mathcal{P})$. $V = \bigcup_i V_i$ is the* `domain` *where each V_i is a set of variables. $G = \bigcup_i G_i$ (a hypertree MSDAG) is the* `structure` *where nodes of each DAG G_i are labeled by elements of V_i. Let x be a variable and $\pi(x)$ be all the parents of x in G. For each x, exactly one of its occurrences (in a G_i containing $\{x\} \cup \pi(x)$) is assigned $P(x|\pi(x))$, and each occurrence in other DAGs is assigned a uniform potential. $\mathcal{P} = \prod_i P_i$ is the* `jpd`, *where each P_i is the product of the potentials associated with nodes in G_i. A triplet $S_i = (V_i, G_i, P_i)$ is called a* `subnet` *of M. Two subnets S_i and S_j are said to be adjacent if G_i and G_j are adjacent on the hypertree MSDAG.*

MSBNs provide a framework for reasoning about an uncertain domain in cooperative multiagent systems. Each agent holds its partial perspective (a subnet) of a domain, reasons about the state of its subdomain with local observations and through limited communication with other agents. Each agent may be developed by an independent developer and the internals of an agent (agent privacy) are protected. Using the inference algorithms of MSBNs [11], agents can acquire observations in parallel and reason distributively, while their beliefs are exact relative to an equivalent centralized system.

5 Knowledge Representation for Collaborative Design

In this section, we extend the design network for decision-theoretic design presented in Section 3 to a *collaborative design network* for design in a supply chain. Recall that a design network is semantically a value network and syntactically a Bayesian network. It will be seen that a collaborative design network is semantically a distributed value network and syntactically an MSBN.

Using the MSBN representation, design knowledge is distributed among subnets. Except for public nodes (shared by two or more subnets), the knowledge about each subdomain is specified by the corresponding manufacturer and is owned privately. Therefore, a collaborative design network forms a cooperative multiagent system. Below, we consider knowledge representation issues in this social context.

5.1 How To Generate the Hypertree Structure

We assume that there are n manufacturers in the supply chain and the supply chain has a topology of a DAG where each arc is directed from a consumer to a supplier.

A directed acyclic graph may be singly connected (a unique path exists between any two nodes) or multiply connected. If it is multiply connected, there must be a undirected cycle and a manufacturer R on the cycle such that R is the supplier of two consumers C_1 and C_2 both of which are on the cycle and are adjacent to R. This dual-supplying relation has two possible cases: either R supplies two distinct components one to each consumer or R supplies identical components to both. If the components are distinct, they usually require different design and production processes and involve different departments within the organization of R. We can then treat each department as a separate supplier. That is, we replace R and represent the two departments as R_1 who supplies to C_1 and R_2 who supplies to C_2. This breaks the cycle at R.

It is also possible that R supplies the same component to each consumer. In this case, we treat the number of components as a design parameter (with a preferred value of 2) and pretend that R supplies to only one of the two consumers. The consequence is that the cycle at R is broken by deleting the arc going from one of the consumers into R.

If we apply the above technique to each cycle in the supply chain graph, it will eventually become singly connected. Fig. 7 shows an example. The singly

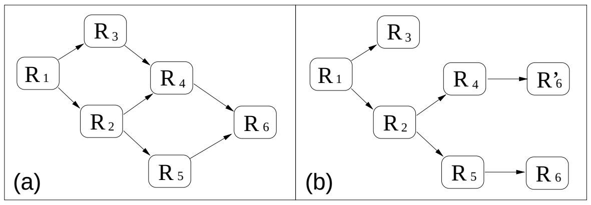

Fig. 7. (a) Supply chain graph where R_4 supplies the same type of components to R_2 and R_3, and R_6 supplies different components to R_4 and R_5. (b) After removing cycles.

connected supply chain graph becomes the basis for the hypertree of the MSBN. Each manufacturer in the supply chain is in charge of the representation of a subnet, the local computation within the subnet, and the necessary communication with the adjacent subnets. Without confusion, we will use the terms *agent* and *manufacturer* interchangeably.

In particular, each agent A_i is associated with a subnet $S_i = (G_i, P_i)$, where $G_i = (V_i, E_i)$ is the graphical subnet structure and P_i is the set of conditional probability distributions defined over V_i. Note that in general, each V_i contains a subset of D, a subset of M, and a subset of U, as defined in a design network.

We emphasize that after cycle removal the resultant supply chain graph does not have to be a directed tree such as the one in Fig. 7 (b). A directed singly connected graph can have multiple root nodes. Each root node represents a manufacturer of a final product. A supply chain graph with multiple root nodes thus represents a supply chain that produce multiple related final products. Our representation is sufficiently general to accommodate such practice.

Our key observation is that with the modification discussed above, a supply chain graph is isomorphic with a hypertree, and the dependence structure of a supply chain can be modeled as an MSDAG. Fig. 8 illustrates a collaborative design network structure which is equivalent to the centralized design network in Fig. 1 (b).

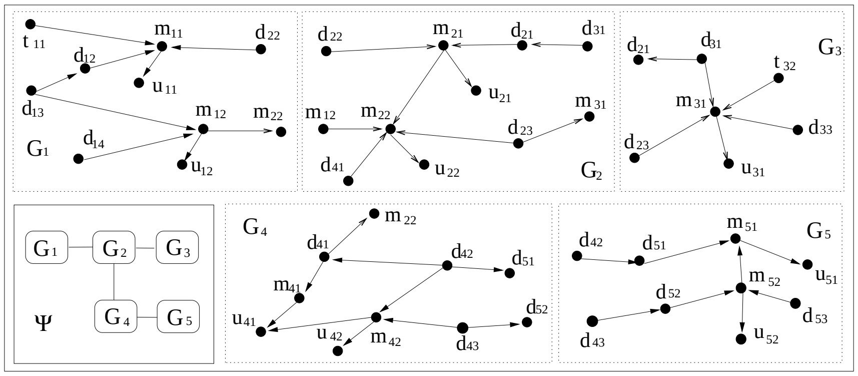

Fig. 8. Graphical model of a collaborative design network, where Ψ is the hypertree.

5.2 Partition of Design Responsibility

In a supply chain, each manufacturer is responsible to design a component to be supplied. As components must interfact with each other, certain design parameters will affect multiple components. For example, if an AC adaptor provides the power for several electronic devices, then the output voltage of the adaptor will affect the design of these devices. In general, we assume design parameters such as this are public variables and will be included in the agent interface.

When a design parameter is public, it would cause conflict if each agent that shares the parameter assigns to it a distinct value. To avoid such conflict, the value assignment authority should be partitioned among agents. That is, each agent A_i will be assigned a subset $D_i \subset D$ and A_i will be the authority to determine the partial design over D_i. The subsets satisfies $D_i \cap D_j = \emptyset$ for any distinct i and j and $\cup_i D_i = D$. In other words, $\{D_0, D_1, ...\}$ is a partition of D. We will refer to it as the *partition of design authority.* The partition can be specified permanently or can be formulated dynamically over the lifetime of the supply chain. For now, we assume a permanent partition.

In particular, we define the partition of design authority as follows:

1. For each private variable (contained in a single subnet) $d \in V_i$, agent A_i has the design authority over d, i.e., $d \in D_i$.
2. For each public variable d', one agent, say A_j, that shares d' is arbitrarily selected such that $d' \in D_j$.

Note that in practice, for each public variable d, there often exists an agent that is natural to assume the responsibility to design d. For example, the agent who designs the AC adaptor is a natural candidate to determine the value for its output voltage. In such a case, the natural candidate should be given the design authority. We stipulated above that A_j is selected arbitrarily. It simply means that the proposed framework does not dictate this choice.

5.3 How To Obtain Numerical Distributions

Numerical information encoded in a design network includes the following:

1. Unconditional probability distribution for each root design parameter node.
2. Conditional probability distribution for each non-root design parameter node.
3. Unconditional probability distribution for each root environment node.
4. Conditional probability distribution for each performance measure node.
5. Utility distribution for each utility node.
6. Weight for each utility node.

Using the MSBN representation, these distributions are distributed in subnets. Except for public nodes, these distributions are specified by the corresponding manufacturers and owned privately. Below, we consider each type of distributions in this social context.

For distributions associated with root design nodes, they provide guidance to manufacturers on frequently used values for these design parameters. If a root node is private, the distribution is assigned by the corresponding manufacturer and reflects its past design experience. When a root node is public, the distribution can be assigned by combining the past design experience of all manufacturers who share the root node. See [11] for details on how to combine multiple agents' experience in assigning a distribution. Distributions associated with root environment nodes are handled similarly.

Distributions associated with non-root design parameter nodes represent design constraints between the nodes and their parents. If the node is private within agent A_i's subdomain, then the distribution is assigned by the corresponding manufacturer. Otherwise, manufacturers who share the node must combine their opinion in deciding the constraint and assigning the distribution.

Probability distributions at performance measure nodes encode the likelihood of potential performances given particular designs. Significant expertise and past experience are needed to assign these distributions. Often, specialized CAD software and historical databases are needed in order to assess these probabilities. If the performance measure is private, a corresponding manufacturer is responsible to assign the distribution. Otherwise, expertises from all manufacturers that share the measure can be combined.

For utility distributions, the matter is different from the above. A supply chain is a cooperative multiagent system and multiple stakeholders exist. Each manufacturer has its own short term and long term interest. The short term spans, for example, the period during which the final products of the supply chain are designed and manufactured. The long term spans, for example, the period during which the equipment needed to manufacture the final products of the current supply chain must be reused for purposes beyond what are intended by the current supply chain. Even if the short term interests of all manufacturers can be assumed the same (e.g., to design and manufacture the products with the lowest cost subject to end-user's satisfaction), their long term interests may not be common. For instance, a particular design may require manufacturer A to deploy equipment e which fits A's long term interest, but requires manufacturer B to deploy equipment e' which is against B's long term interest.

We assume that manufacturers in a supply chain are cooperative and are willing to strike a compromise. Earlier, we have assumed that the overall utility of the supply chain can be expressed as a weighted sum of multiple utility functions (the additive independence assumption). This assumption also allows a simple representation of a compromise of interests of agents in a supply chain: We require that utility nodes be private. That is, each utility node encodes the utility of the corresponding manufacturer. Some nodes may correspond to its short term interest and others correspond to its long term interest. The compromise among agents is represented by the weights k_i for summing individual utilities into the overall utility. Note that association of a weight k_i to each utility node in each subnet is a minor extension to the standard MSBN subnet representation.

Early in the paper, we excluded the end-users from the supply chain graph. However, their interest must be explicitly represented as well. We assume that the interest of each end-user is delegated by the manufacturer of the corresponding final product. This is reasonable since performance measures of the final product must be either private to the corresponding agent or shared by this agent. Therefore, the end-user's utility functions can be encoded within the agent as utility nodes that depend on the relevant performance measures.

6 Conclusion

With the hypertree structure, the subnets and its associated distributions thus defined, we term the resultant representation a *collaborative design network*. Semantically, it is a distributed value network, which encodes collaborative design space, design constraints, uncertain dependency between performance and design, and manufacturer utilities of a collaborative design process for a supply chain. Syntactically, the network is an MSBN. The design space is defined by $D = \cup_i D_i$, where nodes in each D_i are contained in one agent. The likelihood and validity of each potential design is specified by $P(D)$ - the product of distributions associated with these distributed nodes. Similarly, the likelihood of each valid design, each possible environment condition, and each possible performance of the design is specified by $P(D, T, M)$ - the product of distributions associated with nodes in $D \cup T \cup M$ distributed among agents. Using distributed probabilistic reasoning [11], $P(M_i|\mathbf{d})$ for each valid design $\mathbf{d}$ can be obtained, where M_i is a set of performance measures that share a common child utility node u_i (and hence contained in a single agent's subdomain). Using a derivation similar to that in Section 3, the overall expected utility of each design $\mathbf{d}$ reflecting the collective preference of all manufacturers and end-users of the supply chain is well-defined. In summary, a collaborative decision-theoretic design on a supply chain can be adequately modeled by a collaborative design network. Due to space limit, we are unable to include a concrete case study. We intend to do so in our future work.

To solve the collaborative decision-theoretic design problem similar to what is defined in Section 2, it amounts to the following: For each valid design, compute expected utilities of each corresponding partial design by an agent, and integrate the local results by agent communication to obtain the collective expected utility of the design. Repeat the above and determine the best design. Clearly, exhaustive search is intractable. Our ongoing research is investigating more efficient distributed algorithms.

Acknowledgement This research is supported by NSERC Research Grant OGP0155425 and NASA Grant NCC 2-1348 and NCC 2-1353. Assistance from Zoe Jingyu Zhu is acknowledged.

References

1. S.M. Batill, J.E. Renaud, and X. Gu. Modeling and simulation uncertainty in multidisciplinary design optimization. In *The 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 5–8, 2000.
2. K.V. Bury. On probabilistic design. *J. of Engineering for Industry, Trans of the ASME*, pages 1291–1295, Nov. 1974.
3. G.F. Cooper. A method for using belief networks as influence diagrams. In R.D. Shachter, T.S. Levitt, L.N. Kanal, and J.F. Lemmer, editors, *Proc. 4th Workshop on Uncertainty in Artificial Intelligence*, pages 55–63, 1988.
4. S.H. Huang, G. Wang, and J.P. Dismukes. A manufacturing engineering perspective on supply chain integration. In *Proc. 10th Inter. Conf. on Flexible Automation and Intelligent Manufacturing*, volume 1, pages 204–214, 2000.
5. F.V. Jensen. *An Introduction To Bayesian Networks.* UCL Press, 1996.
6. R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives.* Cambridge, 1976.
7. M. Klein, H. Sayama, P. Faratin, and Y. Bar-Yam. The dynamics of collaborative design: insights from complex systems and negotiation research. *Concurrent Engineering Research and Applications J.*, 12(3), 2003.
8. R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* John Wiley and Sons, 1990.
9. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.
10. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.
11. Y. Xiang. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach.* Cambridge University Press, 2002.
12. Y. Xiang and V. Lesser. On the role of multiply sectioned Bayesian networks to cooperative multiagent systems. *IEEE Trans. Systems, Man, and Cybernetics-Part A*, 33(4):489–501, 2003.