Optimal Collaborative Design in Supply Chains

Yang Xiang
School of Computer Science, University of Guelph, Canada

INTRODUCTION

Most modern industrial products are manufactured through supply chains. Competition, rapid product upgrading, and demand for customized products make design and redesign important activities in supply chains. From the manufacturing perspective, a supply chain consists of a set of *manufacturers* that cooperate in the design and production of a final product or multiple related products. Each pair of manufacturers directly involved in the supply chain form a supplying relation, with one of them being the *supplier* and the other being the *consumer*. In this chapter, we use the term *consumer* to refer to a manufacturer in the above sense, and we refer to people who purchase and consume the product as *end-users*, who are regarded as outside the supply chain. A manufacturer R_1 may be the consumer relative to another manufacturer R_2, but acts as the supplier to a third manufacturer R_3 in the same supply chain.

Product design in supply chains are dominated by *component-centered design*, in which a final product is designed as a set of components. In computer design, a processor chip is a component, and so is a hard-disk drive. Lower level components may be composed into a higher level component. For instance, a desktop system unit is a component assembled from month-board, processor chips, etc. Under component-centered design and production, each supplier supplies one or more components to one or more consumers.

Contemporary design in supply chains is essentially top-down (Huang et al., 2000). The manufacturer R of final product decomposes it into components. For each component C to be supplied to R, a supplier S further decomposes C into sub-components, and the process continues. With top-down design, consumers play dominant roles and suppliers are passive. Such designs are unlikely to be optimal, because consumers are usually not in the best position to judge options available to suppliers.

This chapter presents a computational framework for collaborative design where suppliers play equally active roles in shaping design. In particular, we describe a multiagent framework, where manufacturers are collaborative designers aided by intelligent agents. The objective is to produce an overall optimal design by distributed decision making. The multiagent system and decision algorithm are elaborated.

BACKGROUND

A product has a design space described by a set D of variables. Each variable in D is a *design parameter*. Type of processor used in a smart appliance is a design parameter. We assume that each parameter is associated with a discrete domain of possible values, and a naturally continuous parameter is discretized. A *partial design* is an assignment of values to variables in a proper subset of D, and a *complete design* assigns values to all variables in D.

A design is subject to a set of constraints. For instance, if length of a computer case is L and length of the motherboard is L', then L>L' should hold. A constraint involves a subset $S \subset D$ of variables and specifies allowable assignments for S. A design is *valid* if it satisfies all constraints.

Different valid designs result in products with different performances. Maximum speed is a

DOI: 10.4018/978-1-4666-5202-6.ch151

performance measure of a car. For simplicity, we refer to performance of a product resultant from a design as performance of the design. *Performance space* of a product is described by a set M of variables, each being a *performance measure*. We assume that each measure is associated with a discrete domain.

Performance of a product also depends on environment in which it operates. For instance, high level of humidity may cause a computer to fail. We describe such *environmental factors* by a set T of discrete variables.

People differ in preference over a given product performance. Subjective preference of stake-holders (manufacturer or end-user) over design is represented by utility functions (Keeney & Raiffa, 1976). For clarity, we assume that utility is directly dependent on performance of product, not directly on design parameters. Hence, we denote the utility function U(M). An overview of methods for utility function assessment is given in (Farquhar, 1984).

We assume that U(M) can be decomposed additively (Keeney & Raiffa, 1976) as follows: Partition performance measures into groups M_1, M_2, ... Each M_i is associated with a utility function $U_i(M_i) \in [0,1]$. The overall utility function satisfies

$$U(M) = \Sigma_i\ \omega_i\ U_i(M_i),$$

where each weight $\omega_i \in (0,1)$ such that $\Sigma_i\ \omega_i = 1$. As argued in (Keeney & Winterfeldt, 2007), when decision objectives are properly chosen, additive utility decompositions are widely applicable.

Design cannot ensure performance deterministically, due to uncertainty in product life-cycle. We evaluate expected utility of design instead. Denote a design by D = $\underline{d}$. Denote an assignment of performance measures of resultant product by M = $\underline{m}$. P($\underline{m}$|$\underline{d}$) is the probability of performance $\underline{m}$ of product resultant from design $\underline{d}$. Expected utility of $\underline{d}$ relative to $U_i(M_i)$ is

$$EU_i(\underline{d}) = \Sigma m\ U_i(proj(\underline{m}, M_i))\ P(\underline{m}|\underline{d}), \quad (1)$$

where proj($\underline{m}$, M_i) is projection of $\underline{m}$ to M_i. Expected utility of $\underline{d}$ is

$$EU(\underline{d}) = \Sigma_i\ \omega_i\ (\Sigma m\ U_i(proj(\underline{m}, M_i))\ P(\underline{m}|\underline{d})). \quad (2)$$

Given (D, T, M, U), the problem of decision-theoretic design is to find a valid design $\underline{d}$* that maximizes EU($\underline{d}$).

Deterministic design assumes typical maximum loads and minimum material property. It often leads to overdesign and inability to risk analysis. Probabilistic design optimizes in face of uncertainties (Batill et al., 2000). We extends probabilistic design to decision-theoretic, which incorporates stake-holder preference, and to collaborative design by distributed decision-making. Collaborative design may be viewed as distributed constraint satisfaction problems (DisCSPs), e.g., (Meisels & Zivan, 2007). However, DisCSPs involve finding constraint satisfying solutions, but not optimization among them. The limitation is overcome by distributed constraint optimization (DCOP). However, most research on DCOP, e.g., (Petcu & Faltings, 2005) is not decision-theoretic. Below, we present a multiagent framework for decision-theoretically optimal, collaborative design (Xiang et al., 2004).

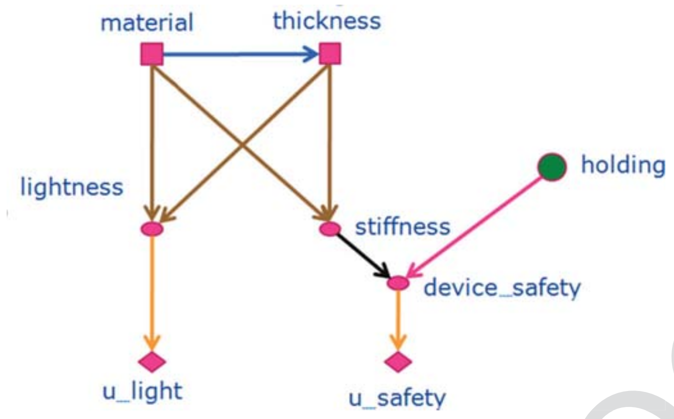
MAIN FOCUS

To compute EU($\underline{d}$) effectively, we represent a centralized design problem as a graphical model, called *design network (DN)*. We then represent a design problem on supply chain as a *collaborative design network (CDN)*. How to solve the design problem by distributed computation will be elaborated.

Design Networks

A DN consists of a set V of variables, a dependency structure G, and a set F of conditional probability tables (CPTs). $V = D \cup T \cup M \cup U$ is the set of

Figure 1. DAG of design network for mobile device casing



variables, where each $u_i \in U$ corresponds to a utility function $U_i()$. Each variable has a domain of possible values, which are naturally defined for elements of D, T, and M. Each $u_i \in U$ has a domain {true, false} or {t, f}.

G = (V, E) is an acyclic, directed graph (DAG), whose nodes are labeled by variables in V. Each arc in E is denoted (p,c) directed from parent p to child c. The following 5 types of arcs are *legal*:

1. Arc (d, d') between design parameters d, d' $\in$D signifies that they are constrained.
2. Arc (d, m) from design parameter d $\in$D to performance measure m $\in$ M signifies that m is dependent on d.
3. Arc (t, m) from environment factor t $\in$T to performance measure m $\in$M signifies that m is dependent on t.
4. Arc (m, m') between performance measures m, m' $\in$M signifies that m' is a composite performance measure.
5. Arc (m, u) from performance measure m $\in$M to utility u $\in$U signifies that u depends on performance measure m.

Figure 1 illustrates the DAG of a design network, where variables in D, T, M, V are shown as square, circle, oval, and diamond, respectively. G encodes conditional independence (Pearl, 1988).

F is s*yntactically* a set of CPTs, one for each node in G. Each t $\in$T is a root and F(t) = P(t). Each m $\in$M with parent set π(m) is associated with F(m, π(m)) = P(m|π(m)). For each d $\in$D, its CPT corresponds to a design constraint defined below:

If d is a root in G, it is unconstrained and F(d) = P(d) is uniform. If d is internal, its parent set $\pi(d) \subset D$, and F(d, π(d)) = P(d|π(d)) is made of 0 or 1. In particular, for each assignment $\underline{\pi(d)}$ of π(d) and each value d' of d, if assignment (d', $\underline{\pi(d)}$) is valid, P(d'| $\underline{\pi(d)}$) = 1. Otherwise, P(d'| $\underline{\pi(d)}$) = 0.

Some assignments of π(d) may also be invalid. In general, for any subset of D of size > 1, some assignments may be invalid, which can be encoded as follows: Consider 10 constrained binary variables d_0, …, d_9. Let $\pi(d_1) = \{d_0\}$ and define $P(d_1|d_0)$ as above. Then let $\pi(d_2) = \{ d_0, d_1\}$ and define $P(d_2|d_0, d_1)$. Repeat the process until $P(d_9|d_0, ..., d_8)$ is assigned. For each d_i, if no value is invalid for any valid assignment of $d_0, ..., d_{i-1}$, the corresponding step is skipped. This yields best case space complexity of $2*10+2^{10} = 1044$ (all steps skipped except the last) and worst case complexity of $2+2^2+2^3+...+2^{10} = 2046$ (no step is skipped).

Each $u_i \in U$ corresponds to a utility function $U_i(M_i)$. Its parent set in G is $\pi(u_i) = M_i$. In Figure 1, utility node u_safety has π(u_safety) = {device_safety}. The function associated with u_i is $F(u_i, \pi(u_i))$ such that $F(u_i = t, \pi(u_i)) = U_i(M_i)$ and $F(u_i = f, \pi(u_i)) = 1 - U_i(M_i)$. Thus, $F(u_i, \pi(u_i))$ syntactically is a CPT while semantically encodes $U_i(M_i)$.

Formally, denote a DN specified above as S = (V, G, F). It is syntactically a Bayesian network (Pearl, 1988), while semantically decision theoretic (with decisions and utilities). DN differs from influence diagrams (IDs) (Jensen & Nielsen, 2007). First, in IDs, arcs into a decision node can be directed from both chance and decision nodes, signifying what is known prior to the decision. DN is used for online decision making, arcs from chance nodes (performance and environment factor) to design parameters are disallowed. Second, arcs into a decision node in IDs are not associated with quantitative knowledge. In DNs, arcs into design parameters from other design parameters signify design constraints, and are associated with design constraints in terms of CPTs.

Since DN is a Bayesian network, for any valid design $\underline{d}$ (for detection of invalid design, see (Xiang, 2006)), conditional probability $P(x|\underline{d})$ for any $x \in V$ can be computed by probabilistic inference. Several algorithms exist and readers are referred to literature, e.g., (Pearl, 1988; Xiang, 2002; Jensen & Nielsen, 2007; Darwiche, 2009) for details. Through probabilistic inference, for each $u_i \in U$, we obtain

$$P\left(u_i = t \mid \underline{d}\right) = \sum_{\underline{m}_i} P(u_i = t \mid \underline{m}_i, \underline{d}) P(\underline{m}_i \mid \underline{d})$$
$$= \sum_{\underline{m}_i} P(u_i = t \mid \underline{m}_i) P(\underline{m}_i \mid \underline{d})$$
$$= \sum_{\underline{m}_i} U_i\left(\underline{m}_i\right) P(\underline{m}_i \mid \underline{d}) = EU_i\left(\underline{d}\right),$$

due to Equation (1), where $\underline{m}_i$ is an assignment of M_i. From Equation (2), we can obtain

$$EU\left(\underline{d}\right) = \sum_i \, _i P\left(u_i = t \mid \underline{d}\right). \tag{3}$$

Collaborative Design Networks

It is infeasible to conduct design on supply chain with centralized DNs, as design constraints and utility functions for each manufacturer are often proprietary. In the following, we present a forward looking multiagent framework, where a globally, decision-theoretically optimal design is obtained distributively.

Each supplier is the designer of its supplied component, and this design role is delegated to a computational agent (Russell & Norvig, 2010). These agents, one per supplier, form a multiagent system. The set V of system variables, is divided (with overlapping) among agents into subsystems. Denote the ith subsystem by V_i, we have $V = \cup_i V_i$. Each agent A_i embodies a DN over a subsystem, called a *design subnet,* denoted as $S_i = (V_i, G_i, F_i)$, where $V_i = D_i \cup T_i \cup M_i \cup U_i$.

Subsystems are organized into a *hypertree*: Each hypernode corresponds to a subsystem. Each hyperlink corresponds to a set I_{ij} of design parameters shared by subsystems V_i and V_j, where I_{ij} renders $V_i \backslash I_{ij}$ and $V_j \backslash I_{ij}$ conditionally independent. $I_{ij} = D_i \cap D_j$ is also called *agent interface*. Variables in V_i not contained in any agent interface are *private* to A_i. The hypertree satisfies *running intersection*: I_{ij} must be contained in each subsystem on the path between V_i and V_j in the hypertree. Hypertree specifies direct communicate links between agents. See (Xiang et al., 2004) for more on hypertree construction. Figure 2 illustrates hypertree and subnet of a CDN.

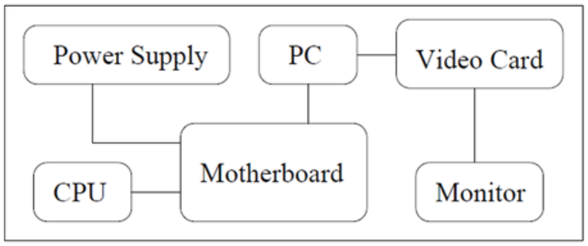
Each agent is assigned a weight $\alpha_i \in (0,1)$ where $\Sigma_i \alpha_i = 1$, representing compromise among agents over individual preferences. The collection of subnets $\{S_i \mid i = 1,2,\ldots\}$ forms a CDN.

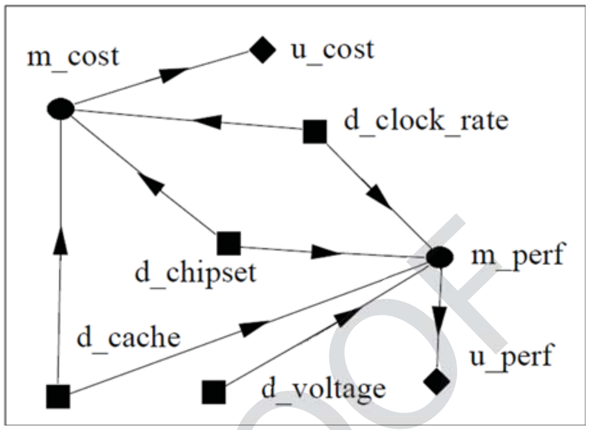
Let $\underline{d}$ be a complete design over $D = \cup_i D_i$ and $\underline{d}_i = proj(\underline{d}, D_i)$. If each A_i obtains $EU(\underline{d}_i)$ by local inference and Equation (3), then

$$EU\left(\underline{d}\right) = \sum_i \, _i EU\left(\underline{d}_i\right) \tag{4}$$

can be computed. The globally optimal design is a valid design $\underline{d}^*$ that maximizes $EU(\underline{d})$. Hence, given a CDN, decision-theoretically optimal design is well defined. On the other hand, exhaustively evaluating each $\underline{d}$ and selecting the

Figure 2. Left: Hypertree of a CDN for customized PC design; right: CPU design subnet.





optimal has complexity of O($\kappa^{|D|}$), where κ is the maximum domain size of design parameters, and is intractable. A more efficient method is presented below.

Optimal Design in CDNs

The algorithm suite consists of algorithms CollectUtil and DistributeUtil, executed by each agent, and algorithm CoDesign, executed by the system coordinator. When coordinator starts CoDesign, it designates one of the agents as the *root* agent. Suppose the PC agent in Figure 2 (left) is the root. Then the hypertree is effectively directed from hypernode PC outwards. Agents corresponding to hypernodes CPU, Monitor, and Power Supply become leaf agents.

For CollectUtil and DistributeUtil, denote executing agent by A_0, whose execution is activated by a caller agent, denoted A_c. If A_0 is root, A_c is coordinator. Otherwise, A_c is an adjacent agent of A_0 on hypertree. If A_c is not coordinator, the interface between A_c and A_0 is denoted as I_{0c}. If A_0 has additional adjacent agents on hypertree, they are denoted as A_1, A_2, ..., A_w, and their interface with A_0 are denoted as I_{01}, I_{02}, ..., I_{0w}, respectively, where subscript identifies the two agents in arbitrary order. The kth partial design over I_{0i} is denoted as $\underline{e}_{0i}^{k}$, and that over I_{0c} is denoted as $\underline{e}_{0c}^{k}$.

First, we introduce CollectUtil. CollectUtil propagates utility evaluations of local designs from leaf agents towards the root. During execution, A_0 receives a vector message $\underline{MEU}_{i0}(I_{0i})$, from each adjacent agent A_i (i=1,…,w), where subscript of $\underline{MEU}_{i0}$ identifies sender and receiver in that order. Elements of the vector are indexed by partial designs over I_{0i}. The kth element of the vector, indexed by $\underline{e}_{0i}^{k}$, is denoted MEU_{i0}^{k}. When A_i is a leaf agent (whose only adjacent agent is A_0), MEU_{i0}^{k} represents the maximum expected utility of partial design $\underline{e}_{0i}^{k}$, as evaluated in subsystem V_i. Denote the vector A_0 sends to A_c as $\underline{MEU}_{0c}(I_{0c})$. Finally, let $\underline{d}_0^{j}$ be the jth local design over D_i.

If A_0 is a leaf agent, it executes lines 1 through 5, and lines 12 through 17. In lines 1 through 5, it evaluates each local design and computes its weighted expected utility EU'($\underline{d}_0^{j}$). In line 14, it compares EU'($\underline{d}_0^{j}$) for each $\underline{d}_0^{j}$ whose projection to I_{0c} matches $\underline{e}_{0c}^{k}$, and sets MEU_{0c}^{k} to the maximum. It also records the corresponding best $\underline{d}_0^{j}$ as $\underline{\delta}_{0c}^{k}$. It then sends $\underline{MEU}_{0c}(I_{0c})$ to A_c.

If A_0 is an internal agent (neither leaf nor root), it executes lines 1 and 2, and lines 6 through 17. In lines 7 through 9, it receives from each A_i utility evaluation $\underline{MEU}_{i0}(I_{0i})$. In lines 10 and 11, it updates EU'($\underline{d}_0^{j}$) with the utility evaluation received from A_i. The updating uses A_i's evaluation

Algorithm 1. [CollectUtil] When A_0 is called by A_c to CollectUtil, it does the following:

```
1   for each local design $\underline{d}_0^j$ over $D_0$,
2       compute EU($\underline{d}_0^j$) by Equation (2);
3   if $A_c$ is the only adjacent agent,
4       for each local design $\underline{d}_0^j$ over $D_0$,
5           $EU'(\underline{d}_0^j) = \alpha_0 * EU(\underline{d}_0^j)$;
6   else
7       for each adjacent agent $A_i$ (i=1,...,w),
8           call $A_i$ to CollectUtil;
9           receive $\underline{MEU}_{i0}(I_{0i})$ from $A_i$;
10      for each local design $\underline{d}_0^j$ over $D_0$,
11          $EU'(\underline{d}_0^j) = \alpha_0 * EU(\underline{d}_0^j) + \sum_{i=1, proj(\underline{d}_0^j, I_{oi}) = \underline{e}_{0i}^k}^{w} MEU_{i0}^k$;
12   if $A_c$ is an adjacent agent,
13      for each partial design $\underline{e}_{0c}^k$ over $I_{0c}$,
14          $MEU_{0c}^k = \max_{\underline{d}_0^j, proj(\underline{d}_0^j, I_{0c}) = \underline{e}_{0c}^k} EU'(\underline{d}_0^j)$;
15          $\underline{\delta}_{0c}^k = \arg\max_{\underline{d}_0^j, proj(\underline{d}_0^j, I_{0c}) = \underline{e}_{0c}^k} EU'(\underline{d}_0^j)$;
16      send $\underline{MEU}_{0c}(I_{0c})$ to $A_c$;
17   return;
```

MEU_{i0}^k relative to $\underline{e}_{0i}^k$, the projection of $\underline{d}_0^j$ to interface I_{0i}. If A_0 is the root agent, it executes lines 1 and 2, lines 6 through 11, and line 17.

Upon returning from CollectUtil, A_0 has computed EU'($\underline{d}_0^j$) for each local design $\underline{d}_0^j$. In addition, for each partial design $\underline{e}_{0c}^k$ over interface I_{0c}, a local design $\underline{\delta}_{0c}^k$ is recorded. EU'($\underline{d}_0^j$) and $\underline{\delta}_{0c}^k$ are used in algorithm DistributeUtil presented below:

The algorithm executed by the system coordinator combines CollectUtil and DistributeUtil, as specified below:

It has been shown (Xiang et al., 2005) that after CoDesign, a complete, globally optimal design $\underline{d}$* is defined by the collection of $\underline{d}_0^*$ at each agent. Note that $\underline{d}$* is not assembled anywhere in the multiagent system. Only $\underline{d}_0^*$ is recorded privately as the globally optimal, local design for each agent. The proof of optimality is through induction. Once agent A in CoDesign is selected, the hypertree is viewed as a tree rooted at A. The length of the longest path from the root to a leaf is the *depth* of the tree. The induction is performed on the depth. For the base case where depth = 1, the root hypernode has one or more child hypernodes, each being a leaf. It can be established that local computations at leafs and the root, coupled by messages between them will determine the optimal design. The base case is then extended to any hypertree with depth > 1. See the above reference for formal details of the proof.

Denote the number of subsystems in CDN by g. On average, each subsystem has |D|/g design parameters. During CollectUtility, each agent evaluates O($\kappa^{|D|/g}$) local designs. Hence, the computation complexity of CoDesign is O(g $\kappa^{|D|/g}$). In comparison with exhaustively evaluating each complete design directly, the complexity is reduced exponentially by a ratio of

Algorithm 2. [DistributeUtil] When A_0 is called by A_c to DistributeUtil, it does the following:

```
1    if A_c is the coordinator,
2        $\underline{d}_0^* = \arg\max_{\underline{d}_0^j} EU'\left(\underline{d}_0^j\right)$;
3    else
4        receive $\underline{e}_{0c}^k$ from A_c;
5        $\underline{d}_0^* = \underline{\delta}_{0c}^k$;
6    for each adjacent agent A_i (i=1,...,w),
7        call A_i to DistributeUtil with message Proj($\underline{d}_0^*$, I_0i);
8    return;
```

$$\frac{1}{g}\kappa^{\left(1-\frac{1}{g}\right)*|D|}.$$

Illustration of Collaborative Design

We illustrate execution of CoDesign below with a trivially-sized CDN, shown in Figure 3. The multiagent system consists of agents A_1 through A_4, for subsystems V_1 through V_4, respectively. Variables labeled d_i are private design parameters, e.g., d_1 is private to A_1. Variables labeled s_j are shared design parameters, e.g., s_2 is shared by A_2 and A_3. All design parameters are binary with the domain {0, 1}. Variables labeled m_k and u_l are performance measures and utilities, respectively. The hypertree, the subnets, and agent interfaces are shown in Figure 3.

Suppose A_4 is selected as the root in CoDesign. CollectUtil is then called on A_4. A_4 performs utility evaluation of local designs with EU(s_3,d_4) shown in Table 1 (top left). By recursive call of CollectUtil (line 8), each agent does the same with its utility evaluation shown in Table 1.

Algorithm 3 [CoDesign]

```
1    select an agent A arbitrarily;
2    call A to CollectUtil;
3    call A to DistributeUtil;
4    return;
```

Assume α_i = 1/4 for i=1,…,4. after local evaluation, leaf agent A_1 executes lines 4 and 5 to scale utility evaluation into EU'(s_1,d_1) (Table 2, left). It then executes lines 12 through 17, obtains $\underline{\text{MEU}}_{13}(s_1)$ in Table 2 (2nd to the left), and sends it to A_3. The local designs corresponding to $\underline{\text{MEU}}_{13}(s_1)$ are

$$\left\{\delta_{13}^0 = (s_1 = 0,\ d_1 = 0);\ \delta_{13}^1 = (s_1 = 1,\ d_1 = 0\right\}.$$

Similarly, A_2 scales utility evaluation into EU'(s_2,d_2) (Table 2, 3rd to the left), obtains $\underline{\text{MEU}}_{23}(s_2)$ (Table 2, right), and sends it to A_3. The local designs corresponding to $\underline{\text{MEU}}_{23}$ are

$$\left\{\delta_{23}^0 = (s_2 = 0,\ d_2 = 1);\ \delta_{23}^1 = (s_2 = 1,\ d_2 = 0\right\}.$$

After receiving the messages, A_3 executes lines 10 through 17. It updates EU(s_1, s_2, s_3,d_3) in Table 1 to EU'(s_1, s_2, s_3,d_3) in Table 3 (left). It obtains $\underline{\text{MEU}}_{34}(s_3)$ shown in Table 3 (middle) and sends it to A_4. The local designs corresponding to $\underline{\text{MEU}}_{34}(s_3)$ are

Figure 3. A trivially-sized example CDN

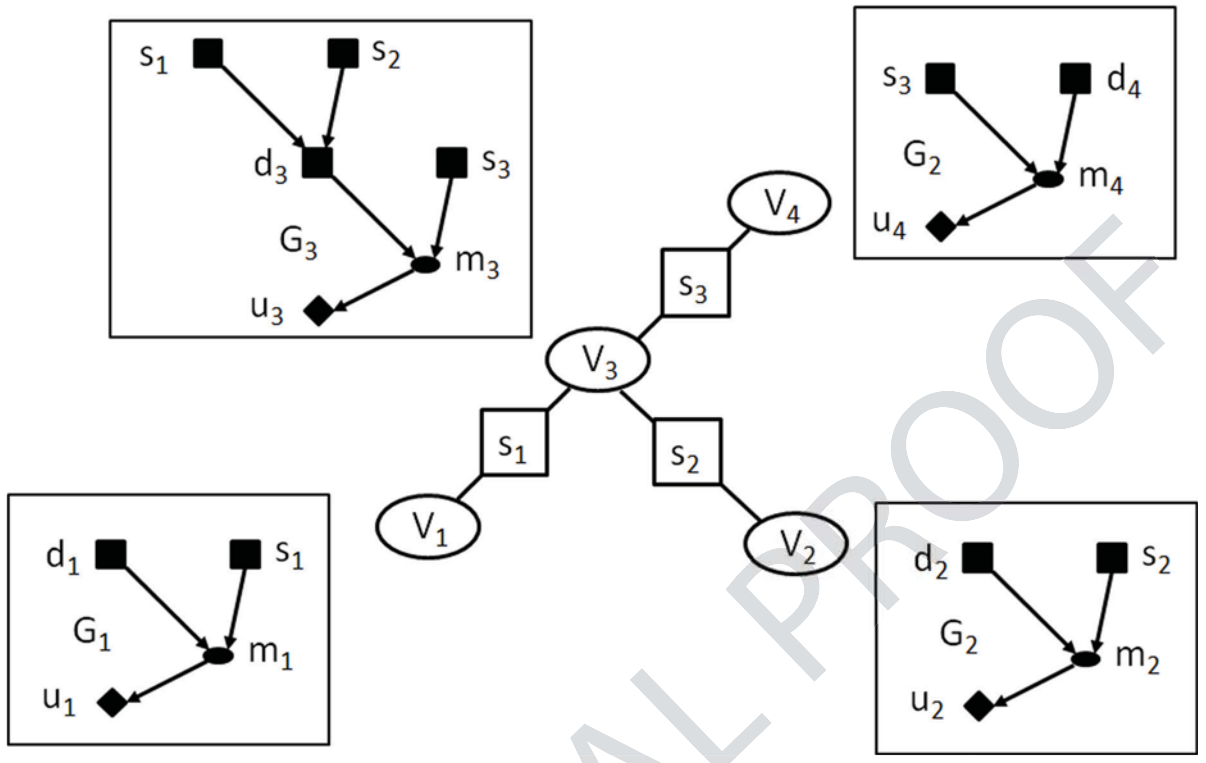


Table 1. Utility evaluation of local designs by each agent

s_3	d_4	EU(s_3,d_4)
0	0	0.05
0	1	0.06
1	0	0.08
1	1	0.09

s_2	d_2	EU(s_2,d_2)
0	0	0.06
0	1	0.08
1	0	0.10
1	1	0.07

s_1	d_1	EU(s_1,d_1)
0	0	0.15
0	1	0.14
1	0	0.17
1	1	0.16

s_1	s_2	s_3	d_3	EU(s_1, s_2, s_3,d_3)
0	0	0	0	0.15
0	0	0	1	0.13
0	0	1	0	0.10
0	0	1	1	0.09
0	1	0	0	0.11
0	1	0	1	0.17
0	1	1	0	0.09
0	1	1	1	0.08
1	0	0	0	0.12
1	0	0	1	0.09
1	0	1	0	0.15
1	0	1	1	0.13
1	1	0	0	0.08
1	1	0	1	0.10
1	1	1	0	0.11
1	1	1	1	0.14

Table 2. Utility weighting and maximization by A_1 (first two) and A_2 (last two)

s_1	d_1	EU'(s_1,d_1)		s_1	$\underline{\text{MEU}}_{13}(s_1)$		s_2	d_2	EU'(s_2,d_2)		s_2	$\underline{\text{MEU}}_{23}(s_2)$
0	0	0.15/4		0	0.15 / 4		0	0	0.06/4		0	0.08 / 4
0	1	0.14/4		1	0.17 / 4		0	1	0.08/4		1	0.10 / 4
1	0	0.17/4					1	0	0.10/4			
1	1	0.16/4					1	1	0.07/4			

$$\begin{Bmatrix} \delta_{34}^{0} = (s_1 = 0,\ s_2 = 1,\ s_3 = 0,\ d_3 = 1); \\ \delta_{34}^{1} = (s_1 = 1,\ s_2 = 1,\ s_3 = 1,\ d_3 = 1) \end{Bmatrix}.$$

After receiving the message from A_3, agent A_4 executes lines 10 and 11. It updates EU(s_3,d_4) in Table 1 (top left) to EU'(s_3,d_4) in Table 3 (right). CollectUtil is now finished.

When DistributeUtil is called on A_4, it determines that complete, globally optimal design has expected utility of 0.50/4 = 0.125, and a globally optimal local design is $\underline{d}_4^* = (s_3{=}1, d_4{=}1)$. It calls A_3 to DistributeUtil with message Proj($\underline{d}_4^*$, s_3) = (s_3=1). From message (s_3=1), A_3 determines its globally optimal local design

$$\underline{d}_3^* = \delta_{34}^{1} = (s_1{=}1,\ s_2{=}1,\ s_3{=}1,\ d_3{=}1).$$

Table 3. Left: Utility update by agent A_3. Middle: Message from A_3. Right: Utility update by A_4.

s_1	s_2	s_3	d_3	EU'(s_1, s_2, s_3,d_3)		s_3	$\underline{\text{MEU}}_{34}$ (s_3)		s_3	d_4	EU'(s_3,d_4)
0	0	0	0	(0.15+0.15+0.08)/4 = 0.38/4		0	0.42 / 4		0	0	(0.05+0.42)/4 = 0.47/4
0	0	0	1	(0.13+0.15+0.08)/4 = 0.36/4		1	0.41 / 4		0	1	(0.06+0.42)/4 = 0.48/4
0	0	1	0	(0.10+0.15+0.08)/4 = 0.33/4					1	0	(0.08+0.41)/4 = 0.49/4
0	0	1	1	(0.09+0.15+0.08)/4 = 0.32/4					1	1	(0.09+0.41)/4 = 0.50/4
0	1	0	0	(0.11+0.15+0.10)/4 = 0.36/4							
0	1	0	1	(0.17+0.15+0.10)/4 = 0.42/4							
0	1	1	0	(0.09+0.15+0.10)/4 = 0.34/4							
0	1	1	1	(0.08+0.15+0.10)/4 = 0.33/4							
1	0	0	0	(0.12+0.17+0.08)/4 = 0.37/4							
1	0	0	1	(0.09+0.17+0.08)/4 = 0.34/4							
1	0	1	0	(0.15+0.17+0.08)/4 = 0.40/4							
1	0	1	1	(0.13+0.17+0.08)/4 = 0.38/4							
1	1	0	0	(0.08+0.17+0.10)/4 = 0.35/4							
1	1	0	1	(0.10+0.17+0.10)/4 = 0.37/4							
1	1	1	0	(0.11+0.17+0.10)/4 = 0.38/4							
1	1	1	1	(0.14+0.17+0.10)/4 = 0.41/4							

It calls A_1 to DistributeUtil with message Proj($\underline{d}_3^*$, s_1) = (s_1=1), and calls A_2 to DistributeUtil with message Proj($\underline{d}_3^*$, s_2) = (s_2=1). From the message, A_1 determines its globally optimal local design

$$\underline{d}_1^* = \delta_{13}^1 = (s_1=1,\ d_1=0),$$

and A_2 determines $\underline{d}_2^* = \delta_{23}^1 = (s_2=1,\ d_2=0)$.

As the result, the complete, optimal design is distributively defined as

$$\underline{d}^* = (s_1=1,\ s_2=1,\ s_3=1,\ d_1=0,\ d_2=0,\ d_3=1).$$

FUTURE RESEARCH DIRECTIONS

Future research for collaborative design will follow several directions: CDN knowledge representation will be refined to enhance its expressiveness for design problems in various industries. New techniques will be developed to enhance hypertree subsystem organization, including algorithms for distributed hypertree existence detection and distributed hypertree construction. Decision algorithms to further improve computational complexity will be developed by exploring properties existing in collaborative design problems.

CONCLUSION

Most modern industrial products are manufactured through supply chains due to product complexity and manufacturing specialization. Competition, rapid product upgrading, and demand for customized products will drive design and redesign into more frequent supply chain activities. Distributed, collaborative, decision theoretically optimal design by multiagent systems offers a futuristic framework to satisfy the need of rapid design and redesign in supply chains. Prototypes of the framework already exist in university labs and are ready to be tested and deployed in industry.

REFERENCES

Batill, S., Renaud, J., & Gu, X. (2000). Modeling and simulation uncertainty in multidisciplinary design optimization. In The 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, pages 5-8.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press. doi:10.1017/CBO9780511811357

Farquhar, P. H. (1984). Utility assessment methods. *Management Science*, *30*, 1283–1300. doi:10.1287/mnsc.30.11.1283

Huang, S., Wang, G., & Dismukes, J. (2000). A manufacturing engineering perspective on supply chain integration. In Proc. 10th Inter. Conf. on Flexible Automation and Intelligent Manufacturing, volume 1, pages 204-214.

Jensen, F., & Nielsen, T. (2007). *Bayesian Networks and Decision Graphs* (2nd ed.). Springer. doi:10.1007/978-0-387-68282-2

Keeney, R., & Raiffa, H. (1976). *Decisions with Multiple Objectives*. Cambridge.

Keeney, R., & Winterfeldt, D. (2007). Practical value models. In W. Edwards, R. F. Miles, & D. Winterfeldt (Eds.), *Advances in Decision Analysis: From Foundations to Applications* (pp. 232–252). Cambridge. doi:10.1017/CBO9780511611308.014

Meisels, A., & Zivan, R. (2007). Asynchronous forward-checking for DisCSPs. *Constraints*, *12*(1), 131–150. doi:10.1007/s10601-006-9013-5

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Petcu, A., & Faltings, B. (2005). A scalable method for multiagent constraint optimization. In Proc. 19th Inter. Joint Conf. on Artificial Intelligence, pages 266-271.

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Xiang, Y. (2002). *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge, UK: Cambridge University Press. doi:10.1017/CBO9780511546938

Xiang, Y. (2006). Optimal design with design networks. In Proc. 3rd European Workshop on Probabilistic Graphical Models, pages 309-316, Prague, Czech.

Xiang, Y., Chen, J., & Deshmukh, A. (2004). A decision-theoretic graphical model for collaborative design on supply chains. In A. Tawfik, & S. Goodwin (Eds.), *Advances in Artificial Intelligence, LNAI 3060* (pp. 355–369). Springer. doi:10.1007/978-3-540-24840-8_25

Xiang, Y., Chen, J., & Havens, W. (2005). Optimal design in collaborative design network. In Proc. 4th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems, pages 241-248.

ADDITIONAL READING

Bernstein, D., Zilberstein, S., & Immerman, N. (2000). The complexity of decentralized control of Markov decision processes. In Proc. 16th Conf. on Uncertainty in Artificial Intelligence, pages 32-37, Stanford.

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In Proc. Conf. on Theoretical Aspects of Rationality and Knowledge, pages 195-210.

Burke, D. (2008). Exploiting Problem Structure in Distributed Constraint Optimization with Complex Local Problems. PhD thesis, U. College Cork, Ireland.

Castillo, E., Gutierrez, J., & Hadi, A. (1997). *Expert Systems and Probabilistic Network Models*. Springer. doi:10.1007/978-1-4612-2270-5

Hirayama, K., & Yokoo, M. (2005). The distributed breakout algorithms. *Artificial Intelligence*, *161*(1-2), 89–116. doi:10.1016/j.artint.2004.08.004

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT.

Koller, D., & Milch, B. (2001). Multi-agent influence diagrams for representing and solving games. In Proc. 17th Inter. Joint Conf. on Artificial Intelligence, pages 1027-1034.

Kwak, J., Yang, R., Yin, Z., Taylor, M., & Tambe, M. (2011). Teamwork in distributed POMDPs: Execution-time coordination under model uncertainty. In Proc. 10th Int. Conf. on Autonomous Agents and Multiagent Systems, pages 1261-1262.

Modi, P., Shen, W., Tambe, M., & Yokoo, M. (2005). Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, *161*(1-2), 149–180. doi:10.1016/j.artint.2004.09.003

Neapolitan, R. (1990). *Probabilistic Reasoning in Expert Systems*. John Wiley and Sons.

Pynadath, D., & Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, *16*, 389–423.

Rao, A., & Georgeff, M. (1995). Bdi agents: from theory to practice. In Proc. 1st Inter. Conf. on Multi-Agent Systems, pages 312-319.

Silaghi, M., & Faltings, B. (2005). Asynchronous aggregation and consistency in distributed constraint satisfaction. *Artificial Intelligence*, *161*(1-2), 25–54. doi:10.1016/j.artint.2004.10.003

Velagapudi, P., Varakantham, P., Sycara, K., & Scerri, P. (2011). Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In Proc. Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, pages 955-962.

Vinyals, M., Rodriguez-Aguilar, J., & Cerquides, J. (2010). Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *J. Autonomous Agents and Multi-Agent Systems*, *22*(3), 439–464. doi:10.1007/s10458-010-9132-7

Xiang, Y. (2007). Tractable optimal multiagent collaborative design. In Proc. IEEE/WIC/ACM Inter. Conf. on Intelligent Agent Technology, pages 257-260.

Xiang, Y., & Hanshar, F. (2010). Comparison of tightly and loosely coupled decision paradigms in multiagent expedition. *International Journal of Approximate Reasoning*, *51*, 600–613. doi:10.1016/j.ijar.2010.01.016

Xiang, Y., & Hanshar, F. (2011). Partial evaluation for planning in multiagent expedition. In C. Butz, & P. Lingras (Eds.), *Advances in Artificial Intelligence, LNAI 6657* (pp. 420–432). Springer. doi:10.1007/978-3-642-21043-3_50

Xiang, Y., & Hanshar, F. (2012). Multiagent decision by partial evaluation. In L. Kosseim, & D. Inkpen (Eds.), *Advances in Artificial Intelligence, LNAI 7310* (pp. 242–254). Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-642-30353-1_21

Xiang, Y., & Janzen, M. (2006). A computational framework for package planning. *Inter. J. Knowledge-Based and Intelligent Engineering Systems*, *10*(2), 93–104.

Xiang, Y., Jensen, F., & Chen, X. (2006). Inference in multiply sectioned Bayesian networks: Methods and performance comparison. *IEEE Trans. Systems, Man, and Cybernetics-Part B*, *36*(3), 546–558. doi:10.1109/TSMCB.2005.861862

Xiang, Y., & Lesser, V. (2003). On the role of multiply sectioned Bayesian networks to cooperative multiagent systems. *IEEE Trans. Systems, Man, and Cybernetics-Part A*, *33*(4), 489–501. doi:10.1109/TSMCA.2003.809220

Xiang, Y., Smith, J., & Kroes, J. (2011). Multiagent Bayesian forecasting of structural time-invariant dynamic systems with graphical models. *International Journal of Approximate Reasoning*, *52*, 960–977. doi:10.1016/j.ijar.2010.07.004

Xiang, Y., Ye, C., & Stacey, D. (2002). Application of Bayesian networks to shopping assistance. In R. Cohen, & B. Spencer (Eds.), *Advances in Artificial Intelligence, LNAI 2338* (pp. 344–348). doi:10.1007/3-540-47922-8_31

Zhang, W., Wang, G., Xing, Z., & Wittenburg, L. (2005). Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, *161*(1-2), 55–87. doi:10.1016/j.artint.2004.10.004

KEY TERMS AND DEFINITIONS

Collaborative Design Network: Distributed graphical representation of industrial design problems, including design parameters, product working conditions, product performance measures, and subjective measures of stake-holders.

Conditional Independence: Involve generally disjoint sets of variables, X, Z, and Y, such that dependence between X and Y is completely mediated by Z. It is the fundamental condition that allows graphical models to be applied successfully to decision theoretical reasoning in large applications.

Decision Theoretic Optimal Design: Take into account both uncertainty in the life-cycle of product under design and desirability of stakeholders, and optimize according to the maximum expected utility principle.

Design Network: Centralized graphical representation of industrial design problems, including design parameters, product working conditions, product performance measures, and subjective measures of stake-holders.

Multiagent System: Computational paradigm where distributed intelligent programs access local sensors, make decisions, and take actions autonomously.

Supply Chain: A set of industrial manufacturers collectively involved in design and production of a set of related products. Each pair of directly interacting manufacturers is related by supplying relationship.

Utility Function: Numerical function defined over a set of variables, e.g., product performance measures, that specifies a stake-holder's subjective measure of desirability for each assignment of the variables.