Optimal Design in Collaborative Design Network

Y. Xiang
University of Guelph
Canada
yxiang@cis.uoguelph.ca

J. Chen
University of Guelph
Canada
junjiang@uoguelph.ca

W.S. Havens
Simon Fraser University
Canada
havens@cs.sfu.ca

ABSTRACT

We consider a multiagent system whose task is to aid component-centered design by collaborative designers in a supply chain. In the earlier work, collaborative design networks are proposed as a decision-theoretic framework for such a system. In this work, we analyzes how choice of agent interface affects the computational complexity of collaborative design. Based on the analysis, we proposes a set of algorithms that allow agents to produce an overall optimal design by autonomous local evaluation of local designs. We show that these algorithms reduce the complexity exponentially from that of an exhaustive centralized design.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; J.6 [**Computer-Aided Engineering**]: Computer-aided design (CAD)

General Terms

Algorithms, Design

Keywords

Collaborative Design, Optimal Design, Supply Chain, Graphical Models, MSBNs

1. INTRODUCTION

Most research on collaborative design focuses on information sharing mechanisms among distributed designers but not on making design choices, e.g., [3]. *Collaborative optimization* [1] decomposes a design domain into a number of subdomains. These design subsystems are organized into a star architecture and work cooperatively to provide design solutions. However, collaborative optimization only produces locally optimal solutions and does not guarantee globally optimal design.

We consider component-centered design in which a final product is designed as a set of components supplied by manufacturers in a supply chain. We interpret design under broad design-for-X (DFX) concepts including design for assembly, manufacture, disassembly, environment, recyclability, etc. The objective is to produce an overall optimal performance while taking into account diverse sources of uncertainty such as materials, manufacturing tolerance, operating environment, etc.

In the previous work [12], a decision-theoretic graphical model for collaborative design in a supply chain was proposed. The model encodes knowledge of distributed designers on the uncertain dependence among design choices and between design and performance. It represents preference of multiple manufacturers and end-users such that optimal decision-theoretic designs are well-defined. It shows that such distributed design knowledge can be represented as a multiagent multiply sectioned Bayesian network (MSBN), called a *collaborative design network*. This allows multiagent collaborative design to be investigated through rigorous algorithmic study. In the current work, we develop multiagent decision algorithms that yields optimal designs in collaborative design networks.

Our approach differs from multiagent influence diagrams (MAIDs) [10, 7, 4]. In MAIDs, each agent maintains its own representation on other agents. It infers about other agents in much the same way as single-agent reasoning. That is, it observes others' behavior and updates its own belief accordingly. While in an MSBN-based multiagent system, agents exchange beliefs on shared variables in a much more cooperative way.

Collaborative design can also be viewed as a type of distributed constraint optimization problem (DCOP). We are given a complex design problem with many variables and design constraints as welll as a global objective function - to maximize the expected utility of the design. What distinguishes DCOPs from traditional combinatorial optimization problems is that portions of the problem are distributed among multiple agents who must work together collaboratively to maximize the objective. Recent work in DCOPs has built upon earlier work in solving distributed constraint satisfaction problems (DCSPs) [15, 14]. There the problem is coordinating efforts of multiple agents to find a globally satisficing solution to a set of distributed constraints over shared variables. Research issues include defining effective coordination protocols among agents and maximizing the asynchrony of agents during search. Complete methods using asynchronous backtracking schemes typically exhibit significant idle times for agents higher in the backtrack ordering [8]. Distributed local search methods can better utilize agent resources during search but without any guarantees of finding a solution.

DCOP research extended DCSP work by adding a global objective function to be optimized by agents collectively. The ADOPT system of Modi et al. [6] is perhaps the most cited DCOP system which can find optimal solutions while attempting to maximize the utilization of individual agents within a branch-and-bound protocol among agents. Another recent optimal approach [5] is based

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

on a cooperative mediation protocol where agents find solutions to overlapping subproblems and recommend value changes to their neighbouring agents.

Collaborative design shares similarities with DCOP research as noted above. Both attempt to maximize an objective function in a distributed environment. But collaborative design has additional decision-theoretic complexity. Agents have local utility functions to optimize and only have uncertain knowledge of the states of their neighbouring agents. Each agent must make design decisions solely by evaluating a local design problem within the context of the probable design decisions of its neighbours.

The remainer of the paper is organized as follows: Section 2 introduces background knowledge from previous work on MSBNs and collaborative design networks. Section 3 considers optimal design with two agents and analyzes their effectiveness when alternative agent interfaces are used. Algorithm 1 uses performance-based interface, which produces optimal design but does not reduce computational complexity relative to exhaustive centralized design. Algorithm 2 uses instead partial design-based interface. It not only produces optimal design but also improves complexity significantly. This analysis suggests that effective general design algorithms should use agent interfaces that are based on partial designs. An example run of such algorithms with four agents is then presented in Section 4, followed in Section 5 by Algorithms 3, 4 and 5 for optimal design in general collaborative design networks. In Section 6, we draw conclusion and indicate our direction for further extension.

2. BACKGROUND

2.1 Multiply Sectioned Bayesian Networks

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.

To ensure exact, distributed 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. Note that if G_0 and G_1 are sectioned from a third graph, then G_0 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`.

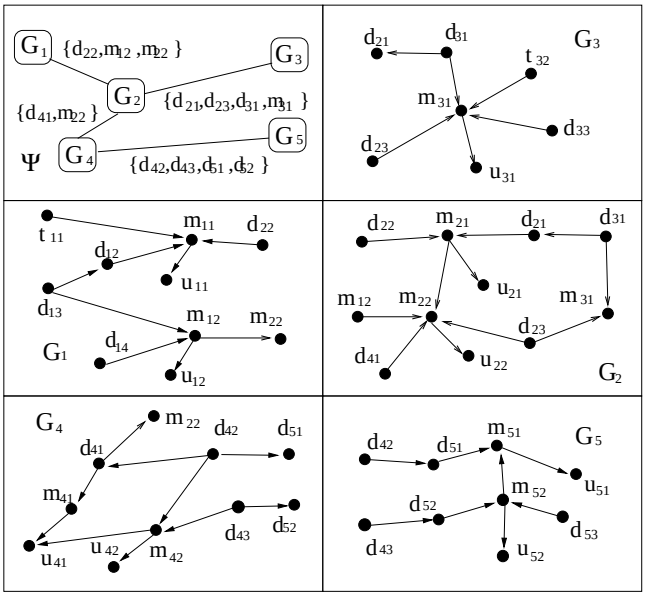

Figure 1: A trivial MSBN with subnets G_1 through G_5 organized into hypertree Ψ.

Fig. 1 shows a hypertree. 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 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 3 variables indicated in Fig. 1. 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.*

An MSBN is then defined as follows. Uniform potentials (constant distributions) are used to ensure that knowledge about the dependence strength are not doubly specified.

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 uncertain reasoning in cooperative multiagent systems. Each agent holds a partial perspective (a subnet) of the domain, reasons autonomously as well as through limited communication [11]. The framework is not accidental. From a few high level requirements, (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 [13] that the resultant representation of a cooperative multiagent system is an MSBN.

2.2 Collaborative Design Network

A Collaborative Design Network (CDN) is an MSBN syntactically, but semantically it represents uncertain design knowledge and preference of a set of designers in a supply chain, who collaboratively design a complex product made out of multiple components. Each distributed designer is equipped with a subnet. The domain V is the union of disjoint sets, D, M, T, U, of variables. The product to be designed is associated with an *overall design space* described by *design parameters* D. Objective functionalities of a designed product is described by *performance measures* M. The environment in which the product operates is described by *environmental factors* T. Subjective preferences of distributed designers are represented by *utility functions* U. We assume that variables in D, M and T are discrete.

Only five types of arcs are legal. Arcs among design parameters represent design constraints. Arcs from design parameters to a performance measure signify the dependence of the performance on these parameters. An arc from an environmental factor to a performance measure signifies the corresponding dependence. Arcs from several performance measures to still another means that the later is a composite performance. Finally, arcs from performance measures to a utility node signify that the utility function depends on these performances.

Conditional probability distribution at each node is assigned according to its semantics in a straightforward manner, except for utility nodes. A utility node u_i with parents M_i represents a utility function $U_i(M_i)$, which is encoded in CDN as follows: The domain of u_i is $\{y, n\}$, the distribution at u_i is $P(u_i = y|M_i) = U_i(M_i)$ and $P(u_i = n|M_i) = 1 - P(u_i = y|M_i)$. Under additive independence assumption [2], the overall utility function $U(M)$ is $U(M) = \sum_i k_i\ U_i(M_i)$, where each weight $k_i \in (0, 1)$ and $\sum_i k_i = 1$.

Given a valid overall design (violating no design constraints) $\mathbf{d}$ (a configuration of D), belief propagation produces, at node u_i, u_i-based expected utility [12]

$$EU_i(\mathbf{d}) = P(u_i = y|\mathbf{d}) = \sum_{\mathbf{m_i}} U_i(\mathbf{m_i}) P(\mathbf{m_i}|\mathbf{d}), \qquad (1)$$

where $\mathbf{m_i}$ is a configuration of M_i. The expected utility of overall design $\mathbf{d}$ is

$$EU(\mathbf{d}) = \sum_i k_i\ EU_i(\mathbf{d}).$$

More details on CDN can be found in [12]. We illustrate with a simple example for customerized PC design. Figure 2 shows the hypertree of a CDN of six distributed designers. Figure 3 and Figure 4 show subnets corresponding to cpu and motherboard designers.

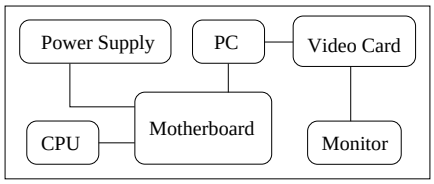

Figure 2: Hypertree of a simple CDN.

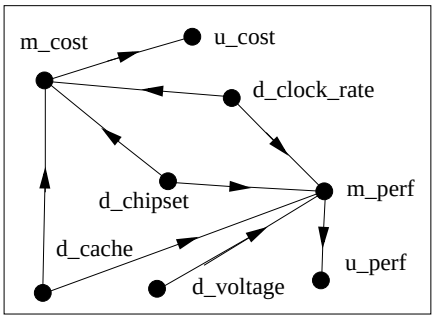

Figure 3: Subnet for cpu designer.

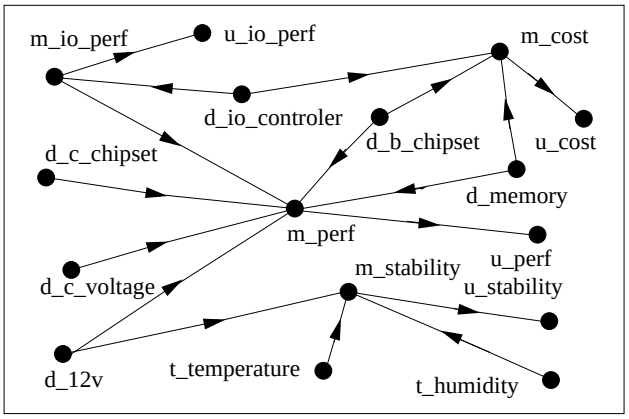

Figure 4: Subnet for motherboard designer.

3. OPTIMAL DESIGN WITH TWO AGENTS

CDNs provide a rigorous framework for optimal collaborative design. Given a CDN, optimal design amounts to finding an overall design $\mathbf{d}^*$ that maximizes $EU(\mathbf{d}) = \sum_i k_i\ EU_i(\mathbf{d})$.

Consider the simplest case of two agents A_0 and A_1, with corresponding subdomains V_i $(i = 0, 1)$. Denote the number of design parameters in V_i as n_i. Assume that all design parameters are binary. Hence, there are 2^{n_i} possible *local designs* in subdomain V_i and a total of $2^{n_0+n_1}$ possible *overall designs*. In A_i's design space, we denote jth local design as $\mathbf{d_i^j}$.

Through local evaluation, A_i can obtain *local expected utility* of $\mathbf{d_i^j}$, denoted as

$$EV_i^j = \sum_x k_{ix}\ EU_{ix}(\mathbf{d_i^j}),$$

where EU_{ix} is the local expected utility according to utility variable u_{ix}, u_{ix} is a utility variable in V_i, and k_{ix} is the weight asso-

ciated with u_{ix}. Similar to Eqn (1),

$$EU_{ix}(\mathbf{d_i^j}) = \sum_{\mathbf{m_{ix}}} U_{ix}(\mathbf{m_{ix}}) P_i(\mathbf{m_{ix}}|\mathbf{d_i^j}), \quad (2)$$

where P_i is the *local* distribution of A_i.

We develop a method for agents to obtain an optimal overall design by autonomous optimal local designs and their integration. The efficiency of such collaboration depends highly on what information they exchange. We analyze two extreme cases below.

3.1 Performance As Agent Interface

In the first case, the agent interface I_{01} consists of q performance measures. Consider the following agent actions:

ALGORITHM 1.

1. *For each local design $\mathbf{d_0^j}$, A_0 computes distribution $P_0^j = P_0(I_{01}|\mathbf{d_0^j})$ and sends P_0^j to A_1.*
2. *For each P_0^j received from A_0, A_1 does the following:*
   - *For each local design $\mathbf{d_1^m}$, update belief $P_1(V_1|P_0^j, \mathbf{d_1^m})$ and compute local expected utility $EV_1^{m|j}$.*
   - *Compute distribution $P_1^{m|j} = P_1(I_{01}|P_0^j, \mathbf{d_1^m})$ and sends $P_1^{m|j}$ and $EV_1^{m|j}$ to A_0.*
3. *After receiving $P_1^{m|j}$ from A_1, A_0 updates belief to $P_0(V_0|P_1^{m|j}, \mathbf{d_0^j})$ and computes local expected utility $EV_0^{j|m}$.*
4. *After receiving each $P_1^{m|j}$ from A_1 for each $\mathbf{d_0^j}$, A_0 does the following:*
   - *Compute $MEV = max_{j,m}\ (EV_0^{j|m} + EV_1^{m|j})$.*
   - *From indexes j and m that attain the value MEV (breaking ties arbitrarily), label local design $\mathbf{d_0^j}$ as $\mathbf{d_0^*}$ and send m to A_1.*
5. *A_1 labels $\mathbf{d_1^m}$ as $\mathbf{d_1^*}$.*

PROPOSITION 5. *The overall design defined by $\mathbf{d_0^*}$ and $\mathbf{d_1^*}$ through Algorithm 1 is optimal.*

Proof: For each overall design $\mathbf{d} = (\mathbf{d_0^j}, \mathbf{d_1^m})$, we have

$$EU(\mathbf{d}) = \sum_i k_{0i}\ EU_{0i}(\mathbf{d}) + \sum_j k_{1j}\ EU_{1j}(\mathbf{d}).$$

First, we show that $\sum_j k_{1j}\ EU_{1j}(\mathbf{d}) = EV_1^{m|j}$ obtained by A_1. This amounts to show that for each utility node u_k in A_1, distribution $P_1(M_k|\mathbf{d_1^m})$ used in local evaluation (Eqn (2)) is identical to $P(M_k|\mathbf{d})$ in (Eqn (1)). This is true because I_{01} makes V_0 and V_1 conditionally independent and we have

$$P_1(V_1|P_0^j, \mathbf{d_1^m}) = P(V_1|\mathbf{d_0^j}, \mathbf{d_1^m}) = P(V_1|\mathbf{d}).$$

Second, we show that $\sum_i k_{0i}\ EU_{0i}(\mathbf{d})$ equals to $EV_0^{j|m}$ obtained by A_0. This is true by an argument similar to the above such that

$$P_0(V_0|P_1^{m|j}, \mathbf{d_0^j}) = P(V_0|\mathbf{d_0^j}, \mathbf{d_1^m}) = P(V_0|\mathbf{d}).$$

Therefore, MEV is the maximum expected utility overall all overall designs, and the overall design defined by $\mathbf{d_0^*}$ and $\mathbf{d_1^*}$ is optimal.

□

Note that agents exchange their beliefs and local evaluations over their interface but not the local designs. This helps protect privacy of each agent. However, the cardinality of the local design space is revealed.

The complexity for A_0 according to step 1 is $O(2^{n_0})$. The complexity for A_1 according to step 2 is $O(2^{n_0+n_1})$. That for A_0 according to steps 3 and 4 is also $O(2^{n_0+n_1})$. Hence, the overall complexity is $O(2^{n_0+n_1})$.

In summary, collaborative design using performance as agent interface allows optimal design, but does not reduce computational complexity relative to exhaustive centralized design.

3.2 Partial Design As Agent Interface

In the second case, agent interface I_{01} consists of q design parameters. In the design space of I_{01}, there are 2^q *partial designs*. We denote kth partial design as $\mathbf{e^k}$.

Given a partial design $\mathbf{e^k}$, there are 2^{n_i-q} corresponding local designs in the design space of A_i. All of them share the value over I_{01} with $\mathbf{e^k}$. These local designs are said to be *consistent* with $\mathbf{e^k}$. Given $\mathbf{e^k}$, remaining variables in V_0 and V_1 are conditionally independent. Hence, for each overall design $\mathbf{d} = (\mathbf{d_0^j}, \mathbf{d_1^m})$, where $\mathbf{d_0^j}$ is a local design in subdomain V_0 and $\mathbf{d_1^m}$ is a local design in V_1, such that both are consistent with a partial design over I_{01}, we have

$$EU(\mathbf{d}) = \sum_x k_{0x}\ EU_{0x}(\mathbf{d_0^j}) + \sum_y k_{1y}\ EU_{1y}(\mathbf{d_1^m})$$

$$= EV_0^j + EV_1^m.$$

For a given $\mathbf{e^k}$, denote the maximum local expected utility over the 2^{n_i-q} consistent local designs in V_i as MEV_i^k. A local design with its expected utility equal to MEV_i^k is selected, breaking ties arbitrarily, and denoted as $\mathbf{d_i^{k*}}$. Consider the following agent actions:

ALGORITHM 2.

1. *For each local design $\mathbf{d_i^j}$, A_i computes EV_i^j. For each partial design $\mathbf{e^k}$, A_i obtains MEV_i^k and $\mathbf{d_i^{k*}}$.*
2. *For each partial design $\mathbf{e^k}$, A_0 sends MEV_0^k to A_1.*
3. *A_1 computes*

   $$MEV = max_k\ (MEV_0^k + MEV_1^k),$$

   where maximization is over each partial design $\mathbf{e^k}$, labels a local design corresponding to MEV as $\mathbf{d_1^*}$, and sends the partial design $\mathbf{e^{k*}}$ that is consistent with $\mathbf{d_1^*}$ to A_0.
4. *A_0 labels local design corresponding to MEV_0^{k*} as $\mathbf{d_0^*}$.*

The following proposition establishes optimality of Algorithm 2. Given the descriptions before the algorithm and step 3 in the algorithm, its proof is trivial.

PROPOSITION 6. *The overall design defined by $\mathbf{d_0^*}$ and $\mathbf{d_1^*}$ through Algorithm 2 is optimal.*

Note that local designs in each agent beyond the partial design over shared variables are not disclosed. Furthermore, unlike Algorithm 1, the cardinality of the local design space is not revealed.

The complexity of step 1 in Algorithm 2 is $O(2^{n_i})$ for agent A_i. The complexity of message passing in step 2 is $O(2^q)$ and so is that of step 3. Hence, the overall complexity of Algorithm 2 is $O(2^{n_0} + 2^{n_1})$. This is a significant reduction from $O(2^{n_0+n_1})$, the complexity of an exhaustive, centralized design (also that of

Algorithm 1). For instance, if $n_0 = n_1 = 20$, then the complexity reduction ratio is $(2^{n_0+n_1})/(2^{n_0} + 2^{n_1}) = 2^{19}$.

In summary, collaborative design using partial design as agent interface allows optimal design with significant complexity improvement over its counterpart based on performance interface.

4. EXAMPLE: DESIGN IN GENERAL CDNS

Before presenting general algorithms for optimal design using CDNs, we illustrate the operations involved with a trivial example. The example CDN consists of 4 agents A_1 through A_4. Their corresponding subnets are shown in Figure 5. Variables labeled d_i are

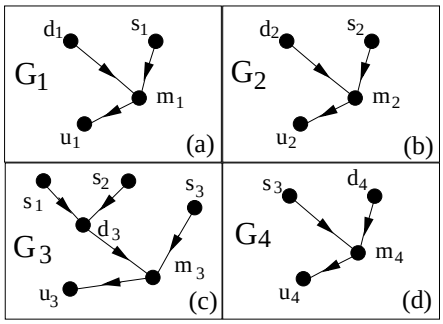

Figure 5: A trivial CDN.

private design parameters known only to the corresponding agent. For instance, d_1 is only known to A_1. Variables labeled s_j are shared design parameters. For instance, s_2 is known to both A_2 and A_3. We assume that all design parameters are binary with the domain $\{0, 1\}$. Variables labeled m_k are performance measures and are private. Variables labeled u_l are utility nodes and are also private.

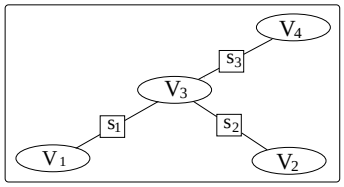

Figure 6: The hypertree of trivial CDN.

Figure 6 shows the hypertree and agent interfaces. Based on analysis in Section 3.2, each interface consists of design parameters only.

After local evaluation at each agent A_i, the local expected utility EV_i^j for each local design $\mathbf{d_i^j}$ is obtained. For readability, we have scaled each EV_i^j up to an integer and we label them as $EVlu_i^j$ to avoid confusion. Tables 1 and 2 show local expected utilities.

s_1	d_1	$EVlu_1$
0	0	15
0	1	14
1	0	17
1	1	16

s_2	d_2	$EVlu_2$
0	0	6
0	1	8
1	0	10
1	1	7

Table 1: Local expected utilities in V_1 and V_2.

s_1	s_2	s_3	d_3	$EVlu_3$
0	0	0	0	15
0	0	0	1	13
0	0	1	0	10
0	0	1	1	9
0	1	0	0	11
0	1	0	1	17
0	1	1	0	9
0	1	1	1	8
1	0	0	0	12
1	0	0	1	9
1	0	1	0	15
1	0	1	1	13
1	1	0	0	8
1	1	0	1	10
1	1	1	0	11
1	1	1	1	14

s_3	d_4	$EVlu_4$
0	0	5
0	1	6
1	0	8
1	1	9

Table 2: Local expected utilities in V_3 and V_4.

An agent on the hypertree is arbitrarily selected as a communication *root*. We assume that it is A_4. The operations then starts from the leaf agents, A_1 and A_2.

From local evaluation result (Table 1, left), A_1 determines the maximum expected utility relative to each partial design over the variables it shares with the neighbor agent A_3. In this example, the only shared variable is s_1. The result is shown in the following table (left) and is sent to A_3 as a message.

s_1	MEVlu
0	15
1	17

s_2	MEVlu
0	8
1	10

Similarly, A_2 determines the maximum expected utility relative to each partial design over variable s_2. The result is shown in the above table (right) and is sent to A_3 as a message.

Based on the two messages, A_3 re-evaluates each local design, taking into account the messages. For each local design, it selects a consistent partial design from each message and adds the corresponding utility in the message to its local expected utility. For instance, the local design

$$(s_1 = 0, s_2 = 1, s_3 = 0, d_3 = 1)$$

has local expected utility 17 (Table 2, left). It is consistent with partial design $s_1 = 0$ from A_1. Hence, the corresponding utility 15 in the message is added to 17. It is also consistent with partial design $s_2 = 1$ from A_2. Hence, the corresponding utility 10 in the message is added. This gives the updated expected utility value 42, as shown in the following table.

s_1	s_2	s_3	d_3	$EVlu'_3$
0	0	0	0	15 + 15 + 8 = 38
0	0	0	1	13 + 15 + 8 = 36
0	0	1	0	10 + 15 + 8 = 33
0	0	1	1	9 + 15 + 8 = 32
0	1	0	0	11 + 15 + 10 = 36
0	1	0	1	17 + 15 + 10 = 42
0	1	1	0	9 + 15 + 10 = 34
0	1	1	1	8 + 15 + 10 = 33
1	0	0	0	12 + 17 + 8 = 37
1	0	0	1	9 + 17 + 8 = 34
1	0	1	0	15 + 17 + 8 = 40
1	0	1	1	13 + 17 + 8 = 38
1	1	0	0	8 + 17 + 10 = 35
1	1	0	1	10 + 17 + 10 = 37
1	1	1	0	11 + 17 + 10 = 38
1	1	1	1	14 + 17 + 10 = 41

From the updated local evaluation, A_3 determines the maximum expected utility relative to each partial design over variable s_3 (see table below) and sends to A_4.

s_3	MEVlu
0	42
1	41

A_4 re-evaluates each local design, taking into account the message:

s_3	d_4	$EVlu'$
0	0	5 + 42 = 47
0	1	6 + 42 = 48
1	0	8 + 41 = 49
1	1	9 + 41 = 50

It determines that the maximum expected utility is 50 and the optimal local design is

$$(s_3 = 1, d_4 = 1).$$

It sends $s_3 = 1$ to A_3.

From the message, A_3 determines its optimal local design

$$(s_1 = 1, s_2 = 1, s_3 = 1, d_3 = 1).$$

A_3 then sends $s_1 = 1$ to A_1 and $s_2 = 1$ to A_2.

From the message, A_1 determines its optimal local design

$$(s_1 = 1, d_1 = 0).$$

Similarly, A_2 determines its optimal local design

$$(s_2 = 1, d_2 = 0).$$

As a result, the optimal overall design is

$$(s_1 = 1, s_2 = 1, s_3 = 1, d_1 = 0, d_2 = 0, d_3 = 1).$$

Note that no single agent needs to know what the overall optimal design is.

5. OPTIMAL DESIGN IN GENERAL COLLABORATIVE DESIGN NETWORK

We present two recursive algorithms executed by each agent and one algorithm executed by the system coordinator. The example presented above is a trace of these algorithms. Without losing generality, we denote the agent executing the algorithms as A_0. The execution is activated by a caller, denoted as A_c, which is either an adjacent agent of A_0 or the system coordinator. Exactly one agent will be called by the coordinator. The interface between A_c (if an agent) and A_0 is denoted as I_c. If A_0 has additional adjacent agents, they are denoted as $A_1, A_2, ..., A_w$ and their interface with A_0 are denoted as $I_1, I_2, ..., I_w$, respectively. Based on the analysis in Section 3.2, we assume each I_i consists of only design parameters. The kth partial design in the design space of I_i is denoted as e_i^k and that relative to I_c is denoted as e_c^k.

The following algorithm, when executed by each agent, propagates utility evaluation of local designs inwards on the hypertree. During execution, A_0 will receive a vector message from each adjacent agent A_i. Elements of the vector are indexed by partial designs over I_i. The kth element of the vector, indexed by e_i^k, is denoted as MEV_i^k. When A_i is a leaf agent on the hypertree (whose only adjacent agent is A_0), MEV_i^k corresponds to the maximum local expected utility in Section 3.2, but otherwise its interpretation is different as is seen below. The vector message sent from A_i to A_0 is denoted as MEV_i and that sent from A_0 to A_c is denoted as MEV_c.

ALGORITHM 3 (COLLECTUTILITY). *When agent A_0 is called by A_c to CollectUtility, if A_c is the only adjacent agent, it does the following:*

1. *For each local design $\mathbf{d_0^j}$, compute EV_0^j.*
2. *For each partial design $\mathbf{e_c^k}$, compute*

$$MEV_c^k = max_{j_k}\ EV_0^{j_k},$$

where maximization is over each local design $\mathbf{d_0^{j_k}}$ that is consistent with $\mathbf{e_c^k}$, and label a local design that reaches the value MEV_c^k by $\mathbf{d_c^{k*}}$, breaking ties arbitrarily.

3. *Send MEV_c to A_c.*

Otherwise (A_0 has more adjacent agents), for each adjacent agent A_i ($i = 1, ..., w$), A_0 calls CollectUtility in A_i. After each A_i has returned with MEV_i, A_0 does the following:

4. *For each local design $\mathbf{d_0^j}$, compute*

$$EV_0^{'j} = EV_0^j + \sum_i MEV_i^{k_j},$$

where $MEV_i^{k_j}$ is indexed by partial design $\mathbf{e_i^{k_j}}$ and $\mathbf{e_i^{k_j}}$ is consistent with $\mathbf{d_0^j}$.

5. *If A_c is an adjacent agent, for each partial design $\mathbf{e_c^k}$, A_0 computes*

$$MEV_c^k = max_{j_k}\ EV_c^{'j_k},$$

where maximization is over each local design $\mathbf{d_0^{j_k}}$ that is consistent with $\mathbf{e_c^k}$, labels a local design that reaches the value MEV_c^k by $\mathbf{d_c^{k*}}$, breaking ties arbitrarily, and sends MEV_c to A_c.

Note that in the computation of $EV_0^{'j}$ at step 4, a unique $\mathbf{e_i^{k_j}}$ exists that is consistent with $\mathbf{d_0^j}$. Note also that when A_c is the coordinator, only step 4 of the algorithm will be executed.

The next algorithm, when executed by each agent, propagates utility evaluation MEV of an optimal overall design outwards on the hypertree. As the propagation progresses, each agent identifies its local design which corresponds to the optimal overall design. This is achieved by propagating an optimal partial design over the agent interface.

ALGORITHM 4 (DISTRIBUTEUTILITY). *When agent A_0 is called by A_c to DistributeUtility, if A_c is the coordinator, it does the following:*

1. *Compute*

$$MEV = max_j EV_0^{'j},$$

where $EV_0^{'j}$ is obtained during CollectUtility (step 4), and label a local design corresponding to MEV as $\mathbf{d_0^*}$, breaking ties arbitrarily.

2. *For each adjacent agent A_i ($i = 1, ..., w$), call DistributeUtility in A_i and send MEV and partial design $\mathbf{e_i^k}$ that is consistent with $\mathbf{d_0^*}$ to A_i.*

Otherwise (A_c is an adjacent agent), A_0 does the following:

3. *Receive MEV and partial design $\mathbf{e_c^k}$ from A_c.*
4. *Label local design corresponding to MEV_c^k as $\mathbf{d_0^*}$.*
5. *For each adjacent agent A_i ($i = 1, ..., w$), call DistributeUtility in A_i and send MEV and partial design $\mathbf{e_i^k}$ that is consistent with $\mathbf{d_0^*}$ to A_i.*

The following algorithm combines the above two algorithms and is executed by the system coordinator.

ALGORITHM 5 (COMMUNICATEUTILITY).

1. *Select an agent A arbitrarily.*
2. *Call CollectUtility in A.*
3. *Call DistributeUtility in A.*

THEOREM 7. *After CommunicateUtility, the overall design defined by local design $\mathbf{d}^*$ at each agent is optimal.*

Proof: We refer to the agent selected in CommunicateUtility as the root agent. Given the root, the hypertree can be effectively viewed as a rooted tree. We define its *depth* as the length of the longest path from root to a leaf.

It suffices to show that MEV obtained by the root agent from DistributeUtility is the maximum expected utility over all possible overall designs. Once this is established, it follows that the restriction of an overall design, that attains this maximum expected utility, to each subdomain is $\mathbf{d}^*$ labelled by the corresponding agent during DistributeUtility. We prove by induction on the depth of the rooted hypertree.

When $depth = 1$, the root has one or more child nodes, each of them is a leaf. Let the root agent be A_0 and its adjacent agents be A_i ($i = 1, ..., w$). From step 1 of DistributeUtility executed by root A_0, we have

$$MEV = max_j \, EV_0^{'j},$$

where maximization is over each local design $\mathbf{d_0^j}$. From step 4 of CollectUtility executed by root A_0, we have

$$MEV = max_j \, (EV_0^j + \sum_i MEV_i^{k_j}).$$

From step 5 of CollectUtility executed by each leaf A_i, each $MEV_i^{k_j}$ above is the result of maximization over all local designs in subdomain V_i that are consistent with $\mathbf{d_0^j}$ relative to the interface between A_0 and A_i. Therefore, as j runs through possible values, the above maximization is performed effectively over all possible overall designs.

Next, we assume that the theorem is true when $depth \leq m$. Consider the case where $depth = m + 1$. Let the root agent be A_0 and its adjacent agents be A_i ($i = 1, ..., w$). The subtree rooted at each A_i has a depth $\leq m$. By assumption, if CollectUtility is called on each A_i by the coordinator, followed by a call of DistributeUtility on A_i, the design defined by $\mathbf{d}^*$ at each agent in the subtree is optimal.

The actual execution of CommunicateUtility differs from this scenario as follows: Instead of performing step 1 of DistributeUtility, A_i performs step 5 of CollectUtility. In other words, instead of maximization over all designs over the subtree, A_i performs maximization over all designs that are consistent with a partial design on its interface with A_0, and it does this for each such partial design. If we regard the union of all subdomains on the subtree rooted at A_i as a single subdomain, what A_i performed is maximization over all local designs in this subdomain that are consistent with a partial design over its interface with A_0.

From this perspective, operations performed by A_0 and A_i ($i = 1, ..., w$) are equivalent to the case where $depth = 1$. Using the argument on that case, the theorem follows. □

Let the total number of agents in a CDN be g and the total number of design parameters be n. Then on average, an agent has n/g design parameters in its subdomain. Let q be the maximum number of design parameters in an agent interface. During CollectUtility, each agent evaluates $O(2^{n/g})$ local designs and sends a message of size $O(2^q)$ to the caller agent. Hence, the computational complexity of optimal design using CDN based on CommunicateUtility is

$$O(g \, 2^{n/g} + (g-1) \, 2^q) = O(g \, (2^{n/g} + 2^q)).$$

Normally, q is much smaller than n/g and hence the complexity $O(g \, 2^{n/g})$. This result can be compared with a centralized optimal design that evaluates all overall designs exhaustively. The complexity will be $O(2^n)$. Using multiagent CommunicateUtility, the complexity is reduced exponentially by a ratio of

$$\frac{1}{g} \, 2^{(1-\frac{1}{g}) \, n}.$$

Let $n = 200$ and $g = 10$, we have $2^n = 1.61 \times 10^{60}$ and $g \, 2^{n/g} = 1.05 \times 10^7$.

6. CONCLUSION

In the precursor [12], collaborative design networks were presented as a decision-theoretic framework to represent collaborative design knowledge as multiagent graphical models. In this work, we analyze the impact of choice of agent interfaces on the computational complexity of collaborative design. The analysis shows that interfaces made of design parameters allow significant reduction of complexity relative to centralized design, while interfaces made of performance measures do not reduce complexity at all.

Based on this analysis, we present algorithms that allow agents in a collaborative design network to obtain an overall design by local evaluations of local designs and by exchanging only evaluations of partial designs on their interfaces. We show that the computation is autonomous and the resultant overall design is globally optimal. The computational complexity is reduced exponentially from that of an equivalent centralized design.

Our current effort is on identification of conditions that allow further reduction of the complexity in evaluating local designs at each agent. The goal is to provide multiagent algorithms that achieve

global design optimality and are efficient on such well behaved design cases.

7. ACKNOWLEDGMENTS

Financial support to this research is provided by National Sciences and Engineering Research Council (NSERC) of Canada. We acknowledge anonymous reviewers for their comments and encouragement.

8. REFERENCES

[1] R.D. Braun, I.M. Kroo, and A.A. Moore. Use of the collaborative optimization architecture for launch vehicle design. In *Proc. 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 306–318, 1996.

[2] R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives*. Cambridge, 1976.

[3] G. Konduri and A. Chandrakasan. A framework for collaborative and distributed web-based design. In *Proc. 36th Design Automation Conference*, pages 898–903, 1999.

[4] S. Maes, K. Tuyls, and B. Manderick. Modeling a multi-agent environment combining influence diagrams. In *Proc. Inter. Conf. on Intelligent Agents, Web Technology and Internet Commerce*, pages 379–384, 2001.

[5] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proc. 3rd Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 04)*, pages 438–445. IEEE Computer Society, 2004.

[6] P.J. Modi, W. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proc. 2nd Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 03)*, pages 161–168, Melbourne, Australia, 2003. ACM Press.

[7] C. Mudgal and J. Vassileva. An influence diagram model for multi-agent negotiation. In *Proc. 4th Inter. Conf. on MultiAgent Systems (ICMAS 00)*, pages 451–452, 2000.

[8] R. Mueller and W.S. Havens. Queuing local solutions in distributed constraint satisfaction problems. In *Proc. 18th Canadian Conf. on Artificial Intelligence (to appear)*. Springer, 2005.

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

[10] D. Suryadi and P.J. Gmytrasiewicz. Learning models of other agents using influence diagrams. In *User Modeling: Proc. 7th Inter. Conf. (UM99)*, pages 223–232, 1999.

[11] Y. Xiang. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.

[12] Y. Xiang, J. Chen, and A. Deshmukh. A decision-theoretic graphical model for collaborative design on supply chains. In A.Y. Tawfik and S.D. Goodwin, editors, *Advances in Artificial Intelligence, LNAI 3060*, pages 355–369. Springer, 2004.

[13] 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.

[14] M. Yokoo. *Distributed Constraint Satisfaction*. Springer, 2001.

[15] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.