MULTIAGENT DECISION MAKING IN COLLABORATIVE DECISION NETWORKS BY UTILITY CLUSTER BASED PARTIAL EVALUATION

YANG XIANG

School of Computer Science, University of Guelph, 50 Stone Road East
Guelph, Ontario, N1G 2W1, Canada

yxiang@uoguelph.ca

FRANK HANSHAR

Medicalis, 508 Riverbend Drive
Kitchener, Ontario, N2K 3S2, Canada

frank.hanshar@gridltd.ca

We consider optimal multiagent cooperative decision making in stochastic environments. The focus is on simultaneous decision making, during which agents cooperate by limited communication. We model the multiagent system as a collaborative decision network (CDN). Several techniques are developed to improve efficiency for decision making with CDNs. We present an equivalent transformation of CDN subnets to facilitate model manipulation. We propose partial evaluation to allow action profiles evaluated with reduced computation. We decompose a CDN subnet, based on clustering of utility variables. A general simultaneous decision making algorithm suite is developed that embeds these techniques. We show that the new algorithm suite improves efficiency by a combination of a linear factor and an exponential factor.

Keywords: Multiagent reasoning, simultaneous decision making, graphical models, collaborative decision networks

1. Introduction

A number of frameworks for cooperative multiagent decision making exist.[1–7] We consider multiagent, cooperative, simultaneous decision making in partially observable and stochastic environments. Here, *simultaneous* signifies that each agent decides on actions over multiple decision variables at once. One example problem is collaborative design in supply chains,[8] where multiple manufacturers cooperate to design a product, each manufacturer is responsible for designing one component, local designs are subject to local and inter-component constraints, and are to be globally optimized. Design must take into account uncertainty on materials, manufacturing processes, product deployment conditions, etc., and optimization must consider preferences of all manufacturers, as well as those of consumers.

In general, complexity of simultaneous decision making is exponential on the total number of decision variables. A general approach for tractability is to ex-

plore conditional independence and factorization through graphical models.[9] Multiply sectioned Bayesian networks (MSBNs)[10,11] are among the earliest multiagent graphical models for probabilistic reasoning. A key idea pioneered in MSBNs is to organize agents into a hypertree with running intersection, according to conditional independence. The hypertree decomposition is used subsequently in other probabilistic graphical models, e.g., OOBNs,[12] and some Dec-MDP frameworks.[13] CDNs[8a] are extensions of MSBNs, from probabilistic to decision theoretic, and also explore the hypertree. In addition to hypertree, MSBNs also pioneered inter-agent message decomposition, through *linkage trees*,[14] for a second level of factorization, resulting in further improvement of efficiency. Neither OOBNs, nor Dec-MDP frameworks,[13] nor existing CDN frameworks,[15] explore such message decomposition. One contribution of this work is to enable exploration of inter-agent message decomposition in CDNs, through utility clusters (Section 5).

We model simultaneous decision problems through CDNs, whose main assumptions are the following. The multiagent system (whose state is described by a collection of variables) is decomposed into overlapping subsystems, that are organized into a hypertree with running intersection, according to certain conditional independence (elaborated later). Each subsystem is hosted by an agent A_i, where i indexes agent, and consists of decision variables (action choices of A_i), chance variables (environmental conditions or consequences of actions), and utility variables (preference of A_i). Dependency among variables in each subsystem is modeled as a decision subnet.

In simultaneous decision making, each agent decides an action, for each local decision variable. The collection of local actions specifies a *local action profile* of the agent. The collection of actions over all decision variables of all agents specifies a *joint action profile*, which is equivalent to joining local action profiles of all agents. The objective of simultaneous decision making in CDN is to find a joint action profile, that maximizes the total expected utility for the agent team. Applying to collaborative design, a joint action profile represents a product design, consisting of a design for each component. A local action profile is the corresponding design of one component, to be supplied by one manufacturer.

A method for optimal decision making in CDNs was proposed earlier.[15] Its computational complexity is linear on the number of agents, but exponential on the number of decisions per subsystem. In this work, we propose novel techniques, that improve the decision making efficiency, while maintaining optimality. Our contributions include the following. First, existing CDN specification requires agent interfaces (variables shared between subsystems) to be made of decision variables only. It is motivated by efficiency consideration,[15] but it limits the expressiveness. We extend CDN agent interfaces to include additional chance variables. The extension widens applicability of CDNs, while maintaining the computational advantage

[a]CDN was proposed with the name *collaborative design network*. Due to its generality, it is renamed later, keeping the abbreviation.

of existing agent interfaces. Secondly, we propose a compact, equivalent transformation of CDNs, where each directed path starts from a decision variable, followed by a chance variable, and ends with a utility variable. The compact transformation facilitates development and execution of novel techniques described below. Third, we propose a technique, called *partial evaluation*, that replaces the normal evaluation of a local action profile by a more efficient computation for non-optimal action profiles, whenever possible. It improves decision making efficiency by a linear factor. Finally, we present a mechanism for inter-agent message decomposition, called *utility clusters*, which further improves decision making efficiency exponentially.

The remainder of the paper is organized as follows: Section 2 introduces and extends the CDN graphical models. The compact, equivalent transformation of CDNs is developed in Section 3. Section 4 introduces partial evaluation for local decision making by individual agents. The utility cluster based partial evaluation is developed in Section 5, where local decision making is integrated into multiagent decision making. Experimental evaluation of the new algorithm suite is reported in Section 6. Section 7 relates the current work to the literature. To ease reader's burden on formal notations, they are summarized in Appendix.

2. Collaborative Decision Networks

This section introduces CDNs, while extending the earlier representation.[8,15] Problem definition is given in Section 2.1. Decomposition of a cooperative multiagent system is considered in Section 2.2. Graphical modeling of subsystems is presented in Section 2.3. Existing decision method for (unextended) CDNs is overviewed in Section 2.4.

2.1. *Problem definition*

A cooperative multiagent system consists of an environment, populated by a set $\mathcal{A}$ of n agents. The *environment state* is described by a collection $\mathcal{E}$ of chance variables. Each $e_i \in \mathcal{E}$ has a finite domain $Ef_i = \{e_{i1}, e_{i2}, ...\}$. Denote the maximum domain cardinality by $\kappa = \max_i |Ef_i|$. A proper subset of $\mathcal{E}$ may be observed by agents.

Let $\mathcal{D}$ be a collection of decision variables. Each $d_i \in \mathcal{D}$ has a finite domain of options or actions $Op_i = \{d_{i1}, d_{i2}, ...\}$. Denote the maximum domain cardinality by $\sigma = \max_i |Op_i|$. If a decision is made by a single agent, it is a *private* decision. Otherwise, it is a *shared* decision, made cooperatively by two or more agents. Each configuration of $\mathcal{D}$ describes a *joint action profile* of agents.

Let $\mathcal{U}$ be a collection of utility variables, such that each is associated with a single agent. Each $u_i \in \mathcal{U}$ is dependent on a set $\pi_i \subset \mathcal{E}$ of chance variables, through a utility function $u_i(\pi_i) \in [0, 1]$. Denote the maximum cardinality of π_i by $m = \max_i |\pi_i|$. Utilities in $\mathcal{U}$ are assumed additively independent. Each u_i is assigned a weight $w_i \in (0, 1)$, and weights of utility variables for the same agent sum to one. The weighted sum $\sum_i w_i\ u_i(\pi_i)$ over all utility variables of the same agent encodes preference of the agent over environment states. Each $A_j \in \mathcal{A}$ is

associated with an agent weight $aw_j \in (0,1)$, such that $\sum_j aw_j = 1$, which encodes relative significance of preference among agents. $\mathcal{U}$, together with agent weights and utility weights, describe *preference state* of the system.

The multiagent system state is described by the collection $\mathcal{V} = \mathcal{D} \cup \mathcal{E} \cup \mathcal{U}$ of variables. The system $\mathcal{V}$ is decomposed into overlapping subsystems V_1, ..., V_n, where each V_i is hosted by agent $A_i \in \mathcal{A}$. Just as a decision variable may be shared or private, a chance variable may be shared or private. When a private variable is observed (chance) or instantiated (decision), the observation or instantiation is also private. On the other hand, every utility variable is private to a particular agent.

For any agent A_i with subsystem $V_i = D_i \cup E_i \cup U_i$, a *local action profile* is a configuration over any subset $SD \subseteq D_i$ of decision variables. We refer to an action profile that is neither local nor joint as a *partial action profile*.

Let $\overline{jd}$ be a joint action profile, and $\overline{d}_i$ be the projection of $\overline{jd}$ over D_i, denoted $\overline{d}_i = proj(\overline{jd}, D_i)$. In other words, $\overline{d}_i$ is the local action profile over D_i that is consistent with $\overline{jd}$. The expected utility of $\overline{jd}$ is

$$eu(\overline{jd}|obs) = \sum_i aw_i \ (\sum_j w_{ij} \ P(\pi_{ij}|\overline{d}_i, obs) \ u_{ij}(\pi_{ij})),$$

where i indexes subsystem, j indexes utility variable in ith subsystem, π_{ij} denotes the parent set of utility u_{ij}, and *obs* denotes the set of observed chance variable values by all agents. The objective of CDN decision making is to determine an optimal joint action profile

$$\overline{jd}^* = \arg \max_{\overline{jd}} \ eu(\overline{jd}|obs),$$

which we refer to as *simultaneous* decision making.

Example 1. (Collaborative design on supply chain) The set $\mathcal{E}$ consists of environment variables, that describe properties of raw materials, manufacturing processes, and product deployment conditions, as well as those that describe objective performance measures (e.g., cost of a material, max speed of a car) of the product under design. The set $\mathcal{D}$ consists of design parameters (e.g., material of casing, max memory capacity of a laptop) of the product. The set $\mathcal{U}$ consists of utility variables that quantify subjective preference of stakeholders over product performance measures.

Although the joint action profile obtained by simultaneous decision making is often executed at the same time, as in Example 1, it does not have to be the case.

Example 2. (Multiagent expedition - MAE[16]) A team of mobile agents explore a unknown open area (a grid), with objects of various types. Each agent can sense a small radius, but not agents and objects beyond. It can move to a cell, and manipulate the object there. Successful manipulation may involve cooperation, which requires agents to meet at the cell. Effects of actions (movement and manipulation) are uncertain. An agent receives a reward at a cell, whose value depends on the

object at the cell (encoded by cell type), the action performed, and degree of success in manipulation. To maximize team reward, agents need to choose behavior wisely, among weak avoidance of unproductive cells, strong avoidance of harmful cells, independent operation, or cooperation.

$\mathcal{E}$ consists of agent positions (ps), position cell types (ct), and effects of manipulation (ef). $\mathcal{D}$ consists of movement decisions (mv) of each agent, and private object manipulation decisions (pd). $\mathcal{U}$ consists of rewards (rw). Each mv action moves an agent from the current cell to an adjacent cell. More than one movement may be needed for nearby agents to meet and cooperate. Hence, a joint action profile needs to specify a short sequence of movements for each agents. To this end, variables may be temporally indexed. We index agents by subscripts, and temporal steps $t = 0, ..., T$ by superscripts, where T is the number of steps (a small integer, e.g., $T = 2$). For instance, after observing current positions ps_i^0 $(i = 1, ..., n)$, decision on mv_i^1 will be made, which leads to new positions ps_i^1.

We consider alternative executions of joint action profile from simultaneous decision making in MAE: (1) interleaving decision making for $T = 1$ with executing the resultant joint action; (2) interleaving decision making for $T = 2$ with execution; (3) interleaving decision making for $T = 2$ with executing the resultant joint action for $t = 1$ only.

The joint action in case (1) is optimal for $T = 1$, but may not be optimal from the perspective of $T = 2$. The joint action for $t = 1$ in case (2) is optimal for $T = 2$, but the joint action for $t = 2$ may not be optimal. This is because the joint action for $t = 2$ is conditioned on observation at $t = 0$, but not on observation available at $t = 1$. Hence, case (3) is superior than both (1) and (2). It executes the joint action for $t = 1$, which is optimal from the perspective of $T = 2$.

We assume that when simultaneous decision making with CDN is applied to problems that involve multiple temporal steps, it is used as case (3), and T is a small integer.

2.2. *System decomposition*

We assume that system decomposition satisfies a hypertree condition, defined below. In the definition, a *junction tree* is a cluster tree, where the intersection of any two clusters is contained in every cluster, on the path between the two.

Definition 1. (Hypertree) Decomposition of $\mathcal{V}$ into V_1, ..., V_n forms a hypertree, if the following holds.

(1) Intersection (if non-empty) between any V_j and V_k $(i \neq k)$ consists of only (shared) decision variables and chance variables.
(2) A junction tree exists with V_1, ..., V_n as its clusters.

We refer to any separator in the hypertree $\mathcal{H}$ as an *agent interface*. For disjoint sets X, Y, Z of variables, denote conditional independence (CI) of X and Y given Z

by $I(X, Z, Y)$.[9] System decomposition into hypertree is assumed to observe decision and observation induced CI (DOICI), defined below and illustrated in Fig. 1.

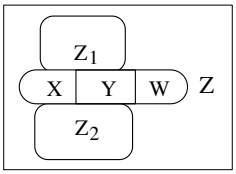

Fig. 1. Illustration of DOICI

Definition 2. (Decision and observation induced CI) Let $Z = X \cup Y \cup W$ be an agent interface, that separates hypertree $\mathcal{H}$ into subtrees $\mathcal{H}_1$ and $\mathcal{H}_2$, where $X \subset \mathcal{D}$ is a set of decision variables, Y is a set of observed chance variables, and W is a set of additional chance variables ($Y, W \subset \mathcal{E}$ and $Y \cap W = \emptyset$). Let Z_1 and Z_2 be collections of decision and chance variables, excluding Z, located in $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. Then interface Z has DOICI property, if the following holds.

(1) Y is observed before decision making;

(2) $I(Z_1, Z, Z_2)$; (3) $I(Z_1, X \cup Y, W)$; and (4) $I(Z_2, X \cup Y, W)$.

In earlier works in CDNs,[8,15] agent interfaces are assumed as being made of decisions only, namely, X above. It was shown that, although interfaces made of chance variables allow distributed optimal decision making, they cannot reduce computational complexity, relative to centralized decision making. On the other hand, interfaces made of decisions not only support distributed optimal decision making, but also allows significant efficiency improvement, over centralized decision making. Defs. 1 and 2 extend the interface composition to include chance variables, $Y \cap W$, subject to DOICI, as specified in Def. 2. The extension allows chance variables in the interface, if either they are observed, i.e., Y, or they are independent given shared decisions X and observed chances Y, i.e., W. The extension enhances expressiveness of CDNs, making them more widely applicable, while preserving the advantage analyzed earlier.[15] Additional illustration of Defs. 1 and 2 can be found in Examples 2 and 3, after introducing subsystem modeling.

2.3. *Subsystem graphical models*

For each subsystem $V = D \cup E \cup U$, where $D \subset \mathcal{D}$, $E \subset \mathcal{E}$, and $U \subset \mathcal{U}$, variables in an agent interface are *shared*, and the rest are *private*. Denote $\rho = |D|$ and $\eta = |U|$ (these notations are summarized in Appendix). The subsystem is modeled as a decision *subnet* $S = (D, E, U, G, P, T)$. G is a connected acyclic directed graph (DAG), whose nodes are labeled by elements of V. It encodes dependence and conditional independence in V, through four types of *legal* arcs: decision to decision

(for decision constraints), decision to chance (for effects of decisions), chance to chance, and chance to utility.

Because a utility is intended to evaluate desirability of actions, every parent (a chance) of every utility must have a decision ancestor. Because an action must be judged on its expected utility, each decision must have a utility descendant. Because a chance is intended to model the uncertain consequence of actions, each chance must be on an undirected path to a decision. These topological requirements are summarized below:

Definition 3. (Subnet topology) A subnet structure G satisfies the following:

(1) For every $u \in U$, each parent of u must have an ancestor $d \in D$.
(2) Every $d \in D$ must have a descendant $u \in U$.
(3) For every $e \in E$, there must be an undirected path from e to a $d \in D$.

Each chance node e_i is associated with a conditional probability table (CPT) $P(e_i|\delta_i)$, conditioned on its parent set δ_i. Denote the maximum size of such parent set by $m = \max_i |\delta_i|$. Each non-root decision node is associated with a CPT (made of extreme probabilities only), expressing a decision constraint (forbidding some action profiles). P is the set of CPTs, one for each node in $D \cup E$. T is the set of utility functions, one for each node in U.

A CDN is a tuple $(\mathcal{A}, \mathcal{V}, \mathcal{H}, \mathcal{S})$, where $\mathcal{S}$ is the set of decision subnets, one for each subsystem. Example 3 below illustrates a CDN. Before the example, we note several differences between CDN subnets and well known influence diagrams (IDs).

Due to the above restriction on legal arcs, a decision node in a subnet can only have decision parents. This differs from IDs. For sequential decision making, IDs allow chance parents for decision nodes, signifying observations available prior to making each decision. CDNs are intended for simultaneous decision making, and there is no essential need for such encoding. Any prior information that constrains the domain for any $d_i \in \mathcal{D}$ is assumed to be reflected in Op_i accordingly.

Furthermore, although both CDNs and IDs allow decision parents for decision nodes, their syntax and semantics are different. IDs allow a decision node to have a past decision parent. Since CDNs are for simultaneous decision making, such encoding of temporal sequence between decisions is not needed. Instead, arcs between decision nodes in CDNs express atemporal constraints. In addition, arcs between decision nodes in IDS are not associated with any numerical parameter, while in CDNs, they are associated with numerical CPTs.

Example 3. Decision making in MAE (Example 2) can be modeled with a CDN. Fig. 2 illustrates a CDN with $T = 2$. Physical formation of an agent team can be maintained, using techniques known as *cooperation frame* and *agent sphere*.[16] They influence agent movements, so that their interactions, e.g., meeting, are regulated. As the result, the hypertree in (c) is valid under DOICI. Details on these techniques can be found from the above reference.

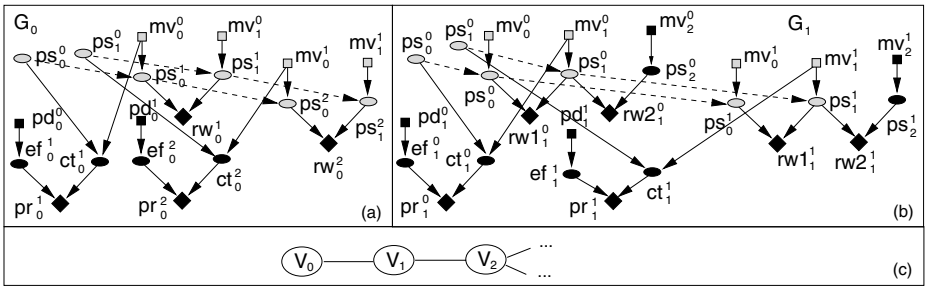

Fig. 2. (a) CDN subnet for agent A_0. (b) Subnet for A_1. (c) Hypertree.

Agent A_0 is adjacent to A_1 on hypertree, and Fig. 2 (a) shows its subnet. Nodes in V_0, E_0, U_0 are drawn as squares, ovals, diamonds, respectively. Subscripts index agents and superscripts index temporal steps. Movement decisions (mv) of both A_0 and A_1 are modeled, as well as two private object manipulation decisions (pd). Agent positions (ps) and position cell types (ct) are dependent on movement actions. Temporal dependency is shown by dashed arcs.

Variables shared by A_0 and A_1 are shaded. The interface is decomposed into $Z = X \cup Y \cup W$ according to Def. 2 as follows:

$$X = \{mv_0^0, mv_1^0, mv_0^1, mv_1^1\}, \quad Y = \{ps_0^0, ps_1^0\}, \quad W = \{ps_0^1, ps_1^1, ps_0^2, ps_1^2\},$$

where Y is observed prior to decision making. Note that agent interface cannot contain Y and W, according to.[8,15] However, they are necessary for A_0 and A_1 to reason about meeting and cooperation. They are enabled by Defs. 1 and 2.

2.4. *Decision making*

A distributed optimal decision method for CDNs was developed earlier.[15] First, the expected utility of each local action profile over D_i is evaluated by the corresponding agent A_i. Then, pairwise inter-agent communication occurs in two rounds, initiated by an arbitrary agent A_0. In round one, messages flow along hypertree towards A_0, starting from leaf agents. For each sending agent A_i, the message contains a utility evaluation of each local action profile over its interface with the receiving agent A_j, based on the subtree rooted at A_i. In the second round, messages flow away from A_0. For each sending agent A_j, the message contains an optimal local action profile, over its interface with the receiving agent A_i.

After communication, an optimal joint action profile is determined. By exploring conditional independence encoded in the hypertree, the decision method reduces the time complexity from being exponential in $|\mathcal{D}| \approx n\ \rho$, to being linear in n and exponential in ρ.[15]

CDNs are extensions of MSBNs,[14] a framework for multiagent probabilistic reasoning, to decision theoretic reasoning. In addition to hypertree decomposition, MSBNs also pioneered inter-agent message decomposition (using so-called *linkage*

trees), for a second level of factorization, resulting in further improvement of efficiency. Decision method in earlier work of CDNs[15] is unable to explore such message decomposition.

In this work, we propose a novel mechanism for message decomposition in CDNs, called *utility clusters*, that allows an exponential efficiency improvement. Utility clusters are closely related to linkage trees, but are significantly different. They are significantly different from the graph-theoretical perspective, but both improve efficiency by decomposing inter-agent message. Linkage trees do so by factorizing probabilistic messages multiplicatively in MSBNs, while utility clusters decompose utility messages in CDNs additively.

As introduced above, existing method for simultaneous decision making in CDNs starts by computing the expected utility of each local action profile. In this work, we propose another technique, called *partial evaluation*, to replace such an evaluation by a less expensive computation for non-optimal action profiles, whenever possible. Partial evaluation allows a linear efficiency improvement. We propose an algorithm suite to integrate partial evaluation and utility clustering, to enable combined efficiency gain, while maintaining optimality of simultaneous decision making.

3. Length-2 CDN Subnets

The new algorithm suite we propose is directly applicable to CDNs, whose subnets are in a regular form, or can be converted equivalently to such form. In Section 3.1, we restrict CDN subnets to a form where decision nodes must be roots. In Section 3.2, we further restrict subnets and define the regular form. We illustrate how a subnet that does not conform to the regular form is transformed into one in Section 3.3.

3.1. *Decision rooted subnets*

First, we regulate decision nodes in a subnet.

Definition 4. (Decision rooted subnet) A subnet is decision rooted, if all its decision nodes are root nodes.

For example, subnets in Fig. 2 are decision rooted. The only restriction of rooted subnet is that it disallows a decision node from having other decision parents, which is legal in a general CDN subnet. We show below that such syntactic restriction is not semantically limiting: An arc between a decision node and its decision parents signifies a constraint between these decisions. Decision nodes involved in such a constraint can be semantically equivalently combined.

Example 4. Let $d_1 \in \{d_{10}, d_{11}\}$ and $d_2 \in \{d_{20}, d_{21}\}$ be two decisions in a subnet, with an arc from d_2 to d_1. Decision d_2 can be freely made, denoted by a uniform CPT $P(d_2)$. The CPT $P(d_1|d_2)$ associated with d_1 is

$$P(d_{10}|d_{20}) = 0.5, \quad P(d_{10}|d_{21}) = 1.$$

That is, d_1 can be freely made when $d_2 = d_{20}$, but action d_{11} is prohibited when $d_2 = d_{21}$. The two decisions can be combined into a single decision

$$d \in \{(d_{10}, d_{20}), (d_{11}, d_{20}), (d_{10}, d_{21})\}.$$

Although the above combination may lose some representational clarity, it is semantically equivalent: specifying the same set of decision options. To facilitate development and implementation of the techniques presented below, we assume that such combination has been performed recursively for each internal decision node. Hence, we consider only decision rooted CDN subnets in the remainder.

3.2. *Regular length-2 subnets*

Next we define a regular form for CDN subnets.

Definition 5. (Regular length-2 subnet) A subnet is regular length-2, if the following holds.

(1) All decision nodes are root nodes and vice versa.
(2) All utility nodes are leaf nodes and vice versa.
(3) Every directed path from a root to a leaf has length 2.

Note that the first condition is stronger than decision rooted. It follows that a regular length-2 subnet must be decision rooted. The inverse, however, is not always true. Fig. 3 shows a regular length-2 subnet. Subnet in Fig. 2 (a) is not regular length-2, as it contains chance roots ps_0^0 and ps_1^0, and length-3 paths from mv_0^0 and mv_1^0 to rw_0^2. We will sometimes refer to regular length-2 subnets simply as being length-2.

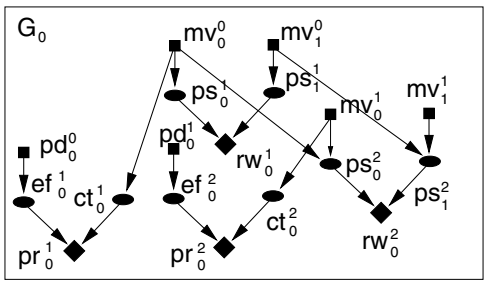

Fig. 3. A regular length-2 subnet

A subnet that is not length-2 may be transformed equivalently into length-2.

Definition 6. (Decision equivalent subnets) Two decision rooted subnets are decision equivalent, if the following holds:

(1) They have the same set of decision variables.

(2) They have the same set of utility variables.
(3) Expected utility of each local action profile computed from one subnet is identical to that from the other.

We illustrate below how a subnet that is not regular length-2 can be decision equivalently transformed into regular length-2, using the subnet in Fig. 2 (a) as an example. A general characterization of length-2 equivalent subnets and a general transformation procedure are beyond the scope of this work.

3.3. *Equivalent subnet transformation: illustration*

First, remove the arc from observed chance node ps_0^0 to ct_0^1 equivalently. The arc from ps_0^0 to ps_0^1 can be removed similarly. Then ps_0^0 is isolated without impact to local action profile evaluation, and can be removed equivalently. Chance ps_1^0 can be removed similarly.

To illustrate removal of arc from ps_0^0 to ct_0^1, the subnet fragment, including ps_0^0, ct_0^1, mv_0^0, and pr_0^1, is depicted in Fig. 4 (a) with variable names simplified. Encoding

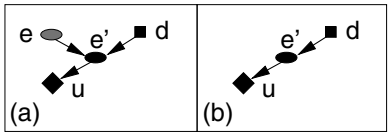

Fig. 4. (a) A subnet fragment that is not length-2. (b) Length-2 fragment after chance root removal.

observation on e by $P(e|obs)$, the arc from e to e' can be equivalently removed, by replacing CPT $P(e'|e,d)$ at node e' with

$$P(e'|d,obs) = \sum_e P(e'|d,e)P(e|obs).$$

Using the above technique, ps_0^0 and ps_1^0 (as well their outgoing arcs) can be removed equivalently from Fig. 2 (a). The resultant subnet is shown in Fig. 5 (a). It is still not length-2, due to length-3 paths from mv_0^0 and mv_1^0 to rw_0^2.

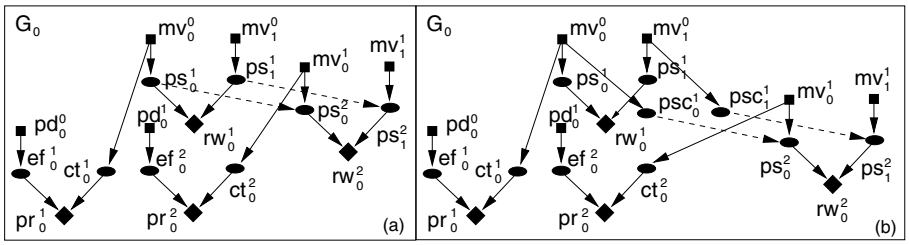

Fig. 5. (a) Subnet after removing chance roots. (b) Subnet after chance node duplication.

To reduce the two paths to length-2, first create duplicate copies of ps_0^1 and ps_1^1, shown as psc_0^1 and psc_1^1 in (b). It is equivalent as the expected utility of mv_0^0 and mv_1^0 through rw_0^1 and rw_0^2 are additively independent. Next, merge the chance nodes between mv_0^0 and rw_0^2, and do the same for the other length-3 path.

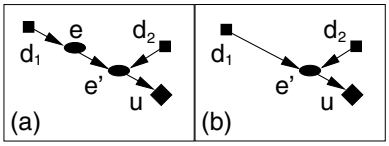

Fig. 6. (a) A subnet fragment that is not length-2. (b) Length-2 fragment after merging chance nodes.

To illustrate merging of psc_0^1 and ps_0^2, the subnet fragment, including mv_0^0, psc_0^1, ps_0^2, mv_0^1, and rw_0^2, is depicted in Fig. 6 (a) with simplified variable names. Chance e can be merged into e' as shown in (b), by replacing CPT $P(e'|e, d_2)$ at e' with

$$P(e'|d_1, d_2) = \sum_e P(e'|e, d_2)P(e|d_1).$$

After merging chance nodes in the two length-3 paths, subnet in Fig. 5 (b) is decision equivalently transformed into the subnet in Fig. 3. In the remainder, we assume that subnets of CDNs are either regular length-2, or have been converted decision equivalently to length-2. We focus on developing a more efficient method for simultaneous decision making in such CDNs. Since every chance variable in a length-2 CDN has a non-empty subset of decision variables as its parents, we refer to chance variables in length-2 CDNs as *effect* variables.

4. Partial Evaluation in Local Decision Making

This section presents partial evaluation for more efficient action profile evaluation in regular length-2 subnets. A local action profile is normally evaluated by computing its expected utility. The basic idea of partial evaluation is to replace the evaluation by a less expensive computation for non-optimal action profiles, whenever possible. Section 4.1 introduces partial evaluation through a trivial decision subnet, with a single decision and a single utility. The method is generalized to subnets with multiple decisions in Section 4.3, and to subnets with multiple utilities in Section 4.4. The algorithms presented form the basis for local computation at individual agents, during simultaneous decision making.

4.1. *Single decision variable*

First, we introduce partial evaluation through a trivial decision subnet, where $\rho = 1$, $\eta = 1$, and $V = \{d_i, e_i, u_i\}$. That is, the DAG is $d_i \rightarrow e_i \rightarrow u_i$. The expected utility

of taking action $d_i = d_{ij}$ is

$$eu(d_{ij}) = P(e_{i1}|d_{ij})u_i(e_{i1}) + P(e_{i2}|d_{ij})u_i(e_{i2}) + ...,$$

which requires κ probability retrievals, κ utility retrievals, κ multiplications, and $\kappa-1$ additions. Action d_{ij} is *fully evaluated*, when $eu(d_{ij})$ is computed. Its complexity is $O(\kappa)$. To compute the optimal action $(d^*_{ik}, meuv)$, where

$$d^*_{ik} = \arg\max_j eu(d_{ij}) \quad and \quad meuv = eu(d^*_{ik}),$$

it generally involves full evaluation of each d_{ij}, with $O(\sigma\ \kappa)$ complexity.

When there is no confusion, we refer to an effect value $e_{ij} \in Ef_i$ simply as an effect. For every action, we refer to an effect value with the highest probability (indexed once and referenced repeatedly) as the *pivot effect* of the action.

Definition 7. (Pivot effect) Let d_{ik} be an action of d_i in the subnet $d_i \rightarrow e_i \rightarrow u_i$. Let e_{ik} be an effect value such that

$$P(e_{ik}|d_{ik}) = max_j P(e_{ij}|d_{ik}).$$

Then e_{ik} is the pivot effect of d_{ik}. If the condition holds for multiple effect values, one is selected arbitrarily.

Note that effect e_{ik} is obtained by varying j in e_{ij}, until $P(e_{ij}|d_{ik})$ is maximal. We denote the pivot effect of action d_{ik} by e_{ik}, and refer to $P(e_{ik}|d_{ik})$ as the *pivot probability* of d_{ik}. Consider MAE, for example, where decision *move* has alternative actions *m_n* (move north), *m_s*, *m_e*, *m_w*, and *halt*. Its effect *landing* has alternative values *l_n* (land north), *l_s*, *l_e*, *l_w*, and *s_p* (same place). Table 1 shows the CPT $P(landing|move)$. From the third row, the pivot effect of *m_n* is *l_n*, and the pivot probability of *m_n* is 0.90.

Table 1. Illustration of pivot assumption with CPT $P(landing|move)$

landing					
l_n	*l_s*	*l_e*	*l_w*	*s_p*	*move*
0.90	0.03	0.03	0.03	0.01	*m_n*
0.03	0.90	0.03	0.03	0.01	*m_s*
0.03	0.03	0.90	0.03	0.01	*m_e*
0.03	0.03	0.03	0.90	0.01	*m_w*
0.025	0.025	0.025	0.025	0.90	*halt*

If we denote the maximum utility of u_i as $u_i^{max} = \max_x u_i(e_{ix})$, it follows that

$$eu(d_{ik}) \leq P(e_{ik}|d_{ik})u_i(e_{ik}) + (1 - P(e_{ik}|d_{ik}))u_i^{max} \equiv Q_{ik}.$$

Note that utility function $u_i(.)$ normally spans the range $[0, 1]$ fully, in which case $u_i^{max} = 1$. Our method does not depend on this condition, and hence we keep u_i^{max} explicit.

Action d_{ij} is said to *dominate* d_{ik}, iff $eu(d_{ij}) > eu(d_{ik})$. If $Q_{ik} < eu(d_{ij})$, then $eu(d_{ik}) < eu(d_{ij})$ and d_{ik} is dominated by d_{ij}. Computing Q_{ik} requires one pivot probability retrieval, two utility retrievals, two multiplications, and two additions. Action d_{ik} is *partially evaluated*, when Q_{ik} is computed.

Suppose pivot probabilities of alternative actions of d_i are nearly identical, e.g., Table 1, where $P(l\_n|m\_n) = P(l\_e|m\_e) = P(s\_p|halt) = 0.90$. As another example, a message may be sent by email or post (alternative actions). Emails most likely arrive in seconds (pivot effect of action *emailing*), while post mails most likely arrive in days (pivot effect of *post_mailing*). $P(arrive\_in\_seconds|emailing)$ and $P(arrive\_in\_days|post\_mailing)$ (pivot probabilities) are similar.

Formally, we make the following *pivot assumption*:

$$\forall j, k \; P(e_{ij}|d_{ij}) = P(e_{ik}|d_{ik}) \equiv p, \quad (j \neq k). \tag{1}$$

That is, for any effect, its pivot probabilities for distinct actions are identical.

Partial evaluation of action d_{ik} amounts to computing

$$Q_{ik} = p \; u_i(e_{ik}) + (1 - p)u_i^{max}.$$

To determine dominance of d_{ij} over d_{ik} and, in particular, whether

$$Q_{ik} = p \; u_i(e_{ik}) + (1 - p)u_i^{max} < eu(d_{ij}) \tag{2}$$

holds, we check equivalently whether

$$u_i(e_{ik}) < \frac{eu(d_{ij})}{p} - \frac{1 - p}{p} u_i^{max}$$

holds. We assume that d_{ij} has been fully evaluated, and the right hand side (threshold) has been obtained before d_{ik} is evaluated. If the above inequality holds, d_{ik} is dominated by d_{ij}, it can be rejected with just one utility retrieval and one comparison, and the threshold can be reused for evaluating the next action. Hence, partial evaluation of d_{ik} has $O(1)$ complexity.

This leads to *partial evaluation based decision* for computing $(d_{ik}^*, meuv)$ as follows:

(1) Apply full evaluation to the first action and establish threshold.
(2) For each alternative action, apply partial evaluation, and reject it if so warranted.
(3) Otherwise, apply full evaluation to it and update threshold.
(4) Select the last accepted action as d_{ik}^*.

Using this procedure, efficiency is gained by evaluating a d_{ik} fully, only if the above inequality fails. Let $\theta \in (0, 1)$ be the percentage of d_{ik} fully evaluated. Then the complexity of partial evaluation based decision is $O(\theta \; \sigma \; \kappa + \sigma)$.

4.2. *Examples for single decision variable*

We illustrate partial evaluation based decision with examples.

Example 5. Consider a trivial decision subnet with DAG $d_i \rightarrow e_i \rightarrow u_i$, $d_i \in \{0, 1, 2, 3, 4\}$, $e_i \in \{0, 1, 2, 3, 4\}$, and parameters in Table 2. The optimal action

Table 2. CPT $P(e_i|d_i)$ (left) and utility function $u_i(e_i)$ (right) for Example 5

e_i					
0	1	2	3	4	d_i
0.09	0.07	0.01	0.80	0.03	0
0.09	0.04	0.80	0.05	0.02	1
0.80	0.02	0.03	0.08	0.07	2
0.06	0.04	0.03	0.80	0.07	3
0.80	0.04	0.02	0.01	0.13	4

d_i	$u_i(e_i)$
0	0.6
1	0.2
2	1.0
3	0.0
4	0.9

$(d^*_{ik}, meuv)$ from a full evaluation is $d^*_{ik} = 1$ and $meuv = 0.88$.

The pivot assumption (Eq. 1) holds for $P(e_i|d_i)$ and the pivot probability is $p = 0.8$. Applying partial evaluation, d_{i0} is fully evaluated to get $eu(d_i = 0) = 0.105$. For d_{i1}, apply Eq. (2) to obtain estimate $Q_{i1} = 0.8 * 1.0 + 0.2 * 1.0 = 1.0 > 0.105$. Hence, d_{i1} is fully evaluated to get $eu(d_i = 1) = 0.88$. For d_{i2}, the estimate is $Q_{i2} = 0.68 < 0.88$ and d_{i2} is rejected. Subsequently, $d_i = 3$ and $d_i = 4$ are all rejected. Hence, the resultant optimal action is $d^*_{ik} = 1$ and $meuv = 0.88$, and is identical to the result of full evaluation. Out of the five alternative actions of d_i, only two of them ($d_i = 0, 1$) are fully evaluated.

In practice, pivot assumption may not always hold. The example below shows that the soundness of partial evaluation is not sensitive to violation of the pivot assumption.

Example 6. Consider another subnet with parameters in Table. 3. The optimal action from a full evaluation is $d^*_{ik} = 1$ and $meuv = 0.888$.

Pivot assumption does not hold in $P(e_i|d_i)$, as pivot probabilities for distinct d_i values are 0.85, 0.82, 0.6, 0.89, 0.9, respectively. Suppose the first pivot probability $p = 0.85$ is used for partial evaluation. First, d_{i0} is fully evaluated to get $eu(d_i = 0) = 0.081$. For d_{i1}, estimate is $Q_{i1} = 0.85 * 1.0 + 0.15 * 1.0 = 1.0 > 0.081$. Hence, d_{i1} is fully evaluated to get $eu(d_i = 1) = 0.888$. Subsequently, $d_i = 2, 3, 4$ are all rejected. Hence, the same optimal action as full evaluation is obtained. This shows that violation of the pivot assumption does not necessarily prevent identification of the optimal action.

Although partial evaluation is not sensitive to violation of pivot assumtion, there is no guarantee that the result is truly optimal when pivot assumption does not hold,

Table 3. CPT $P(e_i|d_i)$ (left) and utility function $u_i(e_i)$ (right) for Example 6

e_i					
0	1	2	3	4	d_i
0.09	0.04	0.01	0.85	0.01	0
0.07	0.04	0.82	0.05	0.02	1
0.60	0.12	0.13	0.08	0.07	2
0.03	0.01	0.03	0.89	0.04	3
0.90	0.04	0.02	0.01	0.03	4

d_i	$u_i(e_i)$
0	0.6
1	0.2
2	1.0
3	0.0
4	0.9

as shown below.

Example 7. Consider another subnet with parameters in Table. 4. The optimal action from a full evaluation is $d^*_{ik} = 1$ and $meuv = 0.601$.

Table 4. CPT $P(e_i|d_i)$ (left) and utility function $u_i(e_i)$ (right) for Example 7

e_i					
0	1	2	3	4	d_i
0.07	0.04	0.82	0.05	0.02	0
0.30	0.22	0.13	0.08	0.27	1
0.09	0.04	0.01	0.85	0.01	2
0.03	0.01	0.03	0.89	0.04	3
0.90	0.04	0.02	0.01	0.03	4

d_i	$u_i(e_i)$
0	0.2
1	1.0
2	0.6
3	0.0
4	0.9

Pivot assumption does not hold in $P(e_i|d_i)$. Suppose the first pivot probability $p = 0.82$ is used for partial evaluation. First, d_{i0} is fully evaluated to get $eu(d_i = 0) = 0.564$. For d_{i1}, estimate $Q_{i1} = 0.82 * 0.2 + 0.18 * 1.0 = 0.344 < 0.564$ and d_{i1} is rejected. Similarly, the subsequent 3 actions are also rejected. The final decision is $d^*_{ik} = 0$ and $meuv = 0.564$, which is suboptimal. Note that $meuv = 0.564$ is fairly close to 0.601 from full evaluation.

To summarize, when pivot assumption holds, partial evaluation is guaranteed to be optimal while allowing computational savings. When the assumption does not hold, partial evaluation can still be optimal, although not guaranteed. In the experimental study (Section 6), we evaluate how likely optimal decisions can still be obtained when pivot assumption does not hold and how close the suboptimal decisions are relative to the optimal.

4.3. *Multiple decision variables*

Next, we generalize partial evaluation to the case where $\rho > 1$ and $\eta = 1$. That is, multiple decision variables collectively influence a utility u_i, e.g., subnet in Fig. 7. We first generalize pivot effect of action in Def. 7 to compound pivot effect of a local action profile.

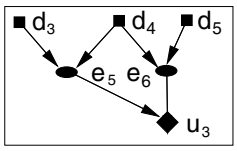

Fig. 7. A length-2 subnet where $\rho > 1$ and $\eta = 1$

Definition 8. (Compound pivot effect) Let $S = (D, E, U, G, P, T)$ be a regular length-2 subnet, where $\rho > 1$ and $\eta = 1$. Let $\overline{d}$ be a local action profile over D.

(1) A **compound effect** $\overline{e''}$ of $\overline{d}$ is a configuration over E.
(2) For each $e_i \in E$, parent set $\delta_i \subset D$ of e_i, and local action profile $proj(\overline{d}, \delta_i)$, the effect value e_{ij} is the **pivot effect** of local plan $proj(\overline{d}, \delta_i)$ if
$$e_{ij} = \arg \max_k P(e_{ik}|proj(\overline{d}, \delta_i)),$$
breaking ties arbitrarily.
(3) The **compound pivot effect** $\overline{e}$ of $\overline{d}$ is the configuration of all pivot effects over E.

Example 8. For the subnet in Fig. 7, $\overline{d} = (d_{31}, d_{42}, d_{51})$ is a local action profile, and so is $proj(\overline{d}, \delta_6) = (d_{42}, d_{51})$. Configuration (e_{51}, e_{62}) is a compound effect.

If e_{51} is the pivot effect of local action profile (d_{31}, d_{42}), and e_{62} is the pivot effect of (d_{42}, d_{51}), then (e_{51}, e_{62}) is the compound pivot effect of $\overline{d}$.

Without confusion, we simply omit the word 'compound' in 'compound effect' and 'compound pivot effect'. Let $\overline{d}$ be a local action profile over D, $\overline{e}$ be its pivot effect, $\overline{e}''$ be any other effect, and $eu(\overline{d})$ be the expected utility of $\overline{d}$. Then
$$eu(\overline{d}) = P(\overline{e}|\overline{d})u_i(\overline{e}) + \sum_{\overline{e}''} P(\overline{e}''|\overline{d})u_i(\overline{e}''). \quad (3)$$

Below, we consider a decision subtask, which will be the basis for local computation by individual agents, during simultaneous decision making. Let β be a subset of D and $\gamma = D \setminus \beta$. Let $\overline{b}$ be an action profile over β, $\overline{y}$ be an action profile over γ, and $(\overline{b}, \overline{y})$ be a *join* of action profiles. The decision subtask is to obtain a pair of functions $(meu(\beta), peer(\beta))$, where $meu : \beta \to [0, 1]$ and $peer : \beta \to \gamma$, such that for each $\overline{b}$,
$$meu(\overline{b}) = \max_{\overline{y}} eu(\overline{b}, \overline{y}) \quad and \quad peer(\overline{b}) = \arg \max_{\overline{y}} eu(\overline{b}, \overline{y}).$$

Note that $peer(\overline{b})$ returns an optimal local action profile over γ, when β is constrained to local action profile $\overline{b}$. Hence, we refer to β as the *constraint scope*, and γ as the *optimization scope*. In other words, the function pair $(meu(\beta), peer(\beta))$ specifies, for each constraint $\overline{b}$ over β, the maximum expected utility (MEU) $meu(\overline{b})$, and the corresponding optimal action profile $(\overline{b}, peer(\overline{b}))$ over D.

Example 9. Let $\beta = \{d_3, d_4\}$ and $\gamma = \{d_5\}$ for the subnet in Fig. 7, assuming binary decision variables. A function pair $(meu(\beta), peer(\beta))$ takes the following form.

$\overline{b}$	$meu(\overline{b})$	$peer(\overline{b})$
(d_{31}, d_{41})	0.70	(d_{52})
(d_{31}, d_{42})	0.65	(d_{51})
(d_{32}, d_{41})	0.82	(d_{51})
(d_{32}, d_{42})	0.73	(d_{52})

For a given local action profile $\overline{d}$ over D, denote $\overline{b} = proj(\overline{d}, \beta)$, $\overline{y} = proj(\overline{d}, \gamma)$. Then, $\overline{d} = (\overline{b}, \overline{y})$. The decision subtask requires evaluating $eu(\overline{d}) = eu(\overline{b}, \overline{y})$ for each $\overline{d}$. To evaluate $eu(\overline{d})$ for a given $\overline{d}$ by Eq. (3), $P(\overline{e}''|\overline{d})$ must be computed for each $\overline{e}''$ over E.

Example 10. For subnet in Fig. 7, we have $P(\overline{e}''|\overline{d}) = P(e_5, e_6|d_3, d_4, d_5) = P(e_5|d_3, d_4)P(e_6|d_4, d_5)$.

Computing $P(\overline{e}''|\overline{d})$ for a particular $\overline{e}''$ involves m probability retrievals and $m-1$ multiplications. There are κ^m alternative $\overline{e}''$. Hence, a full evaluation of $\overline{d}$ by Eq. (3) takes $m\ \kappa^m$ probability retrievals, κ^m utility retrievals, $m\ \kappa^m$ multiplications, and $\kappa^m - 1$ additions. The complexity of a full evaluation of $\overline{d}$ is thus $O(m\ \kappa^m)$. To obtain $meu(\beta)$ by full evaluation, a total of σ^ρ alternative $\overline{d}$ must be evaluated, and the complexity is thus $O(\sigma^\rho\ m\ \kappa^m)$.

For more efficient computation, we explore partial evaluation introduced in Section 4.1. First, we extend pivot assumption on a single effect variable with a single decision parent, to multiple effect variables each with multiple decision parents.

Definition 9. (Pivot assumption) Let e_i be an effect variable, in a regular length-2 subnet S, with parent set δ_i. Let $\overline{\delta_{ik}}$ be kth configuration of δ_i, and e_{ik} be its pivot effect. Subnet S satisfies pivot assumption if, for every e_i, the following holds:

$$\forall\ j, k\ \ P(e_{ij}|\overline{\delta_{ij}}) = P(e_{ik}|\overline{\delta_{ik}}), \quad (j \neq k). \tag{4}$$

The practical interpretation is that, for each effect variable, its pivot probabilities for distinct action profiles are approximately identical.

Example 11. In MAE, the pivot effect of moving north, followed by moving east, is landing on the north east location. The pivot effect of moving south, followed by moving west, is landing on the south west location. The two corresponding pivot probabilities are approximately identical.

As shown later in our experiments, even though our decision method is derived from the pivot assumption, the method works well, when Eq. (4) holds only approximately. We observe

$$eu(\overline{d}) = P(\overline{e}|\overline{d})u_i(\overline{e}) + \sum_{\overline{e}''} P(\overline{e}''|\overline{d})u_i(\overline{e}'') \leq P(\overline{e}|\overline{d})u_i(\overline{e}) + (1 - P(\overline{e}|\overline{d}))u_i^{max}, \quad (5)$$
$$where \;\; u_i^{max} = \max_{\overline{e}''} u_i(\overline{e}'').$$

It then follows from the pivot assumption that

$$\forall\ \overline{d}, \overline{d}' \quad P(\overline{e}|\overline{d}) = P(\overline{e}'|\overline{d}') \equiv p,$$

where $\overline{d} \neq \overline{d}'$, and $\overline{e}$ ($\overline{e}'$) is the pivot effect of local action profile $\overline{d}$ ($\overline{d}'$).

Example 12. For the subnet in Fig. 7, suppose pivot probability of e_5 given action profile (d_3, d_4) is 0.7, and that of e_6 given (d_4, d_5) is 0.9. Then, from Example 10, it follows $p = 0.7 * 0.9 = 0.63$.

When pivot assumption holds, inequation (5) becomes

$$eu(\overline{d}) \leq p\ u_i(\overline{e}) + (1 - p)u_i^{max}.$$

Let $eu(\overline{d}')$ be given for $\overline{d}'$. For an alternative action profile $\overline{d}$, if

$$p\ u_i(\overline{e}) + (1 - p)u_i^{max} < eu(\overline{d}'), \quad (6)$$

it follows that $eu(\overline{d}) < eu(\overline{d}')$, and $\overline{d}$ is dominated by $\overline{d}'$. Partial evaluation of $\overline{d}$ by Eq. (6) takes only two utility retrievals, with complexity $O(1)$.

Extending partial evaluation based decision in Section 4.1 with the above operation, the decision subtask to obtain $(meu(\beta), peer(\beta))$ can be solved by PeDecSu. In the algorithm, the scope of each *for* or *if* statement is indicated by indentation.

Algorithm 1. PeDecSu
Input: subnet over $D \cup E \cup U$ where $U = \{u_i\}$, max utility u_i^{max}, pivot probability p, constraint scope $\beta \subset D$, and optimization scope $\gamma = D \setminus \beta$;
Output: function pair $(meu(\beta), peer(\beta))$;

for each constraint $\overline{b}$ over β,
  pick an action profile $\overline{y}'$ over γ to fully evaluate by Eq. (3) and get $eu(\overline{b}, \overline{y}')$;
  set $meu(\overline{b}) = eu(\overline{b}, \overline{y}')$ and $peer(\overline{b}) = \overline{y}'$;
  set threshold $th = (eu(\overline{b}, \overline{y}') - (1 - p)u_i^{max})/p$;
  for each action profile $\overline{y}$ over γ where $\overline{y} \neq \overline{y}'$,
    retrieve $u_i(\overline{e})$ for pivot effect $\overline{e}$ of $\overline{d} = (\overline{b}, \overline{y})$;
    if $u_i(\overline{e}) \geq th$,
      fully evaluate $\overline{d}$ by Eq. (3) to get $eu(\overline{d})$;
      $meu(\overline{b}) = eu(\overline{d})$, $peer(\overline{b}) = \overline{y}$, $th = (eu(\overline{d}) - (1 - p)u_i^{max})/p$;
return $(meu(\beta), peer(\beta))$;

Note that when $\beta = \emptyset$, the optimization scope $\gamma = D$. In the return value of PeDecSu, $meu(\beta)$ becomes a single value, and $peer(\beta)$ becomes a single optimal local action profile $\overline{d}^*$ over D.

Let $\theta \in (0, 1)$ be the percentage of local action profiles fully evaluated by PeDecSu. Its complexity is then $O(\theta\ \sigma^\rho\ m\ \kappa^m + \sigma^\rho) \approx O(\theta\ \sigma^\rho\ m\ \kappa^m)$. Proposition 1 summarizes key properties of PeDecSu, whose proof is straightforward, given the above analysis.

Proposition 1. *PeDecSu satisfies the following.*

(1) For each constraint $\overline{b}$ over β, $meu(\overline{b})$ is the MEU.
(2) For each $\overline{b}$, $peer(\overline{b})$ is the optimal action profile over γ.
(3) Its complexity is $O(\theta\ \sigma^\rho\ m\ \kappa^m)$.

4.4. *Multiple utility variables*

Next, we generalize partial evaluation to decision subnets where $\rho > 1$ and $\eta > 1$.

Example 13. Fig. 8 shows a subnet with $\eta = 2$ and a weight w_i is associated with

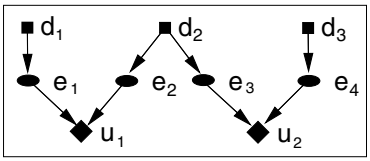

Fig. 8. A length-2 subnet where $\rho > 1$ and $\eta > 1$

each u_i $(i = 1, 2)$. The optimal local action profile cannot be obtained by solving the decision problem as two independent sub-problems, one over $\{d_1, d_2\}$ and the other over $\{d_2, d_3\}$. This is because the optimal action profile over $\{d_1, d_2, d_3\}$ may be incompatible with the optimal action profile over either $\{d_1, d_2\}$ or $\{d_2, d_3\}$. Hence, fully evaluating a local action profile over D amounts to compute

$$\begin{aligned} & eu(d_1, d_2, d_3) \\ = & w_1 \sum_{e_1, e_2} P(e_1|d_1)P(e_2|d_2)u_1(e_1, e_2) + w_2 \sum_{e_3, e_4} P(e_3|d_2)P(e_4|d_3)u_2(e_3, e_4). \end{aligned}$$

In general, let $\overline{d}$ be a local action profile over D, $\overline{e}$ be its pivot effect, $\overline{e}''$ be any alternative effect, and $eu(\overline{d})$ be expected utility of $\overline{d}$. For each utility u_i $(i = 1, ..., \eta)$ with parents π_i, let α_i be the set of decision ancestors of u_i (see Fig. 9). In Fig. 8, α_1 for u_1 is $\{d_1, d_2\}$. Define $\overline{e_i} = proj(\overline{e}, \pi_i)$, and $\overline{d_i} = proj(\overline{d}, \alpha_i)$. Then, a full evaluation of $\overline{d}$ computes

$$eu(\overline{d}) = \sum_{i=1}^{\eta} w_i \left[P(\overline{e_i}|\overline{d_i})\ u_i(\overline{e_i}) + \sum_{\overline{e}_i''} P(\overline{e}_i''|\overline{d_i})\ u_i(\overline{e}_i'') \right]. \quad (7)$$

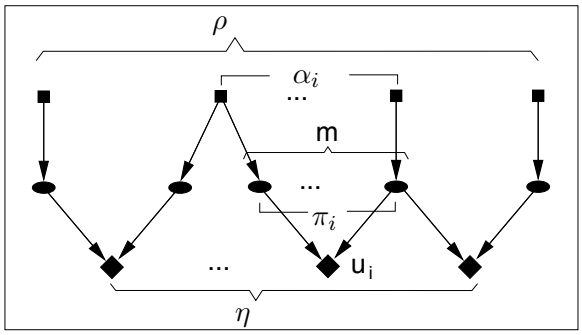

Fig. 9. Illustration of symbols

Below, we extend solution to the decision subtask in Section 4.3 to subnets where $\eta > 1$. As will be seen, simultaneous decision making involves such subtasks by individual agents. Let β be the given constraint scope, and $\gamma = D \setminus \beta$ be the optimization scope. The decision subtask is to obtain $(meu(\beta), peer(\beta))$ such that, for each action profile $\overline{b}$ over β,

$$meu(\overline{b}) = \max_{\overline{y}} eu(\overline{b}, \overline{y}) \quad and \quad peer(\overline{b}) = \arg \max_{\overline{y}} eu(\overline{b}, \overline{y}),$$

where $\overline{y}$ is any action profile over γ.

Applying an analysis to Eq. (7) similarly as in Section 4.3, the complexity to fully evaluate $\overline{d}$ is $O(\eta\ m\ \kappa^m)$. The complexity to obtain $meu(\beta)$ by full evaluation is $O(\sigma^\rho\ \eta\ m\ \kappa^m)$. We consider efficiency improvement with partial evaluation.

Example 14. Consider the subnet in Example 13. If e_j is the child node of d_i, we denote its pivot effect corresponding to d_{ik} by e_{jk}. We have

$$\begin{aligned} &eu(d_{1x}, d_{2y}, d_{3z}) \leq \\ &w_1 P(e_{1x}|d_{1x}) P(e_{2y}|d_{2y}) u_1(e_{1x}, e_{2y}) + w_2 P(e_{3y}|d_{2y}) P(e_{4z}|d_{3z}) u_2(e_{3y}, e_{4z}) \\ &+ w_1(1 - P(e_{1x}|d_{1x}) P(e_{2y}|d_{2y})) u_1^{max} + w_2(1 - P(e_{3y}|d_{2y}) P(e_{4z}|d_{3z})) u_2^{max}. \end{aligned}$$

If the pivot assumption (Def. 9) holds, denote the pivot probability of e_i by p_i. We have

$$\begin{aligned} eu(d_{1x}, d_{2y}, d_{3z}) \leq\ & w_1 p_1 p_2 u_1(e_{1x}, e_{2y}) + w_2 p_3 p_4 u_2(e_{3y}, e_{4z}) \\ & + w_1(1 - p_1 p_2) u_1^{max} + w_2(1 - p_3 p_4) u_2^{max} \equiv Q. \end{aligned}$$

Suppose, for action profile $(d'_{1x}, d'_{2y}, d'_{3z})$, $eu(d'_{1x}, d'_{2y}, d'_{3z})$ has been obtained. For action profile (d_{1x}, d_{2y}, d_{3z}), we compute the above value Q. If

$$Q < eu(d'_{1x}, d'_{2y}, d'_{3z}), \tag{8}$$

it follows that $eu(d_{1x}, d_{2y}, d_{3z}) < eu(d'_{1x}, d'_{2y}, d'_{3z})$, and (d_{1x}, d_{2y}, d_{3z}) is dominated by $(d'_{1x}, d'_{2y}, d'_{3z})$. Partial evaluation of (d_{1x}, d_{2y}, d_{3z}) by Eq. (8) takes only four utility retrievals.

In general, each utility u_i has a set π_i of effect parents (see Fig. 9). Let the effect parents be indexed by j. Assuming pivot assumption, each effect parent of u_i is associated with a pivot probability p_{ij}. Given $eu(\overline{d'})$ from full evaluation of action profile $\overline{d'}$, and an alternative action profile $\overline{d}$, if

$$\sum_{i=1}^{\eta} w_i \left((\prod_j p_{ij})\ u_i(\overline{e_i}) + (1 - (\prod_j p_{ij}))\ u_i^{max} \right) < eu(\overline{d'}),$$

then $\overline{d}$ is dominated by $\overline{d'}$. This leads to threshold assignment and dominance test below:

$$th = eu(\overline{d'}) - \sum_{i=1}^{\eta} w_i (1 - (\prod_j p_{ij}))\ u_i^{max}, \tag{9}$$

$$\sum_{i=1}^{\eta} w_i (\prod_j p_{ij})\ u_i(\overline{e_i}) < th. \tag{10}$$

By pre-computing $w_i(\prod_j p_{ij})$ for each u_i, partial evaluation of $\overline{d}$ using Eq. (10) has a complexity of $O(\eta)$. Extending PeDecSu with the above operations, PeDecMu makes partial evaluation based decision, with multiple decision and utility variables.

Algorithm 2. PeDecMu
Input: subnet over $D \cup E \cup U$, max utility u_i^{max} for each u_i, a pivot probability p_{ij} for each u_i and its jth parent, $\beta \subset D$ and $\gamma = D \setminus \beta$;
Output: function pair $(meu(\beta), peer(\beta))$;

for each constraint $\overline{b}$ over β,
  pick an action profile $\overline{y'}$ over γ and denote $\overline{d'} = (\overline{b}, \overline{y'})$;
  fully evaluate $\overline{d'}$ by Eq. (7) to get $eu(\overline{d'})$;
  set $meu(\overline{b}) = eu(\overline{d'})$, $peer(\overline{b}) = \overline{y'}$, and threshold th by Eq. (9);
  for each action profile $\overline{y}$ over γ where $\overline{y} \neq \overline{y}'$,
    denote $\overline{d} = (\overline{b}, \overline{y})$;
    test dominance by Eq. (10);
    if test fails,
      fully evaluate $\overline{d}$ by Eq. (7) to get $eu(\overline{d})$;
      $meu(\overline{b}) = eu(\overline{d})$, $peer(\overline{b}) = \overline{y}$, and update th by Eq. (9);
return $(meu(\beta), peer(\beta))$;

When $\beta = \emptyset$, the return value of PeDecMu, $meu(\beta)$, becomes a single value, and $peer(\beta)$ becomes a single optimal action profile $\overline{d}^*$ over D. Proposition 2 summarizes key properties of PeDecMu.

Proposition 2. *PeDecMu satisfies the following.*

(1) For each constraint $\overline{b}$ over β, $meu(\overline{b})$ is the MEU.
(2) For each $\overline{b}$, $peer(\overline{b})$ is the optimal action profile over γ.

(3) Its complexity is $O(\theta\ \sigma^{\rho}\ \eta\ m\ \kappa^{m})$.

Comparing the complexity $O(\sigma^{\rho}\ \eta\ m\ \kappa^{m})$ by full evaluation, PeDecMu reduces the complexity by a factor of θ. If $\theta = 20\%$, then PeDecMu will be 5 times faster.

5. Utility Cluster Based Partial Evaluation in Simultaneous Decision Making

In this section, we present a novel technique, utility clusters, to decompose inter-agent messages, in the context of simultaneous decision making in regular length-2 CDNs. It is integrated seamlessly with partial evaluation. The technique decomposes messages additively, and improves efficiency exponentially from the existing method.[15]

Simultaneous decision making is carried out by two rounds of message passing, interleaved with local decision making by partial evaluation. The hypertree is viewed as directed from an arbitrary agent (as root), so that adjacent agents are referred to as parent and child according to the direction. In the first round, utility messages flow from leaf agents towards the root. Section 5.1 introduces utility clusters in the context of message computation by leaf agents. Message computation by general agents are considered in Section 5.2, where the clustering is extended to an arbitrary agent. In the second round, presented in Section 5.3, local action profiles over agent interfaces flow from the root agent towards leaf agents.

5.1. *Utility clusters and messages from leaf agents*

Let a leaf agent A be associated with the subnet over $D \cup E \cup U$. Let the set of decision variables in its interface with the adjacent agent B on hypertree be $SD \subset D$. Message $utm_0(SD)$ that A sends to B is a MEU function that, for each local action profile $\overline{sd}$ over SD, specifies

$$utm_0(\overline{sd}) = \max_{\overline{rd}} eu(\overline{sd}, \overline{rd}), \tag{11}$$

where $\overline{rd}$ is a local action profile over $RD = D \setminus SD$. In other words, we have

$$utm_0(SD) = meu(SD).$$

From Proposition 2, it follows that, if A applies PeDecMu with $\beta = SD$, the return value $meu(\beta)$ satisfies $meu(\beta) = utm_0(SD)$. Since complexity of PeDecMu is exponential on ρ, we seek to improve its efficiency by message decomposition below.

Generally speaking, additive contributions of utility variables to $utm_0(SD)$ may enable decomposition in evaluating $utm_0(SD)$. On the other hand, as demonstrated at the start of Section 4.4, decomposition at the level of individual utility variables is not always feasible. We explore the existence of private decisions, that is, variables in RD. In the following, we classify utility variables in relation to SD and RD, which provides a basis for sound decomposition.

For each utility u_i and its decision ancestors α_i, we define

$$\beta_i = \alpha_i \cap SD \quad and \quad \gamma_i = \alpha_i \setminus \beta_i = \alpha_i \cap RD.$$

That is, β_i is decision ancestors of u_i shared between A and B, and γ_i is decision ancestors of u_i private to A. Each utility variable can be classified, based on the relation between α_i, SD, and RD, into one of four cases:

Definition 10. Let SD be the set of decision variables in an agent interface, and α_i be the decision ancestor set of a utility variable u_i. Let $\beta_i = \alpha_i \cap SD$ and $\gamma_i = \alpha_i \setminus \beta_i$. Then u_i is in case k, if the kth condition below holds.

(1) $\gamma_i = \emptyset$.
(2) $\gamma_i \neq \emptyset$, $\beta_i \neq \emptyset$, and $\gamma_i \cap \gamma_j = \emptyset$ for all u_j where $j \neq i$.
(3) $\gamma_i \neq \emptyset$, $\beta_i \neq \emptyset$, and $\gamma_i \cap \gamma_j \neq \emptyset$ for some u_j where $j \neq i$.
(4) $\beta_i = \emptyset$.

The four cases are interpreted as follows.

Case 1: $\alpha_i = \beta_i$ is entirely shared.
Case 2: There exists no other utility variable u_j, such that u_i and u_j have a common private decision ancestor.
Case 3: There exists at least one utility variable u_j, such that u_i and u_j have a common private decision ancestor.
Case 4: $\alpha_i = \gamma_i$ is entirely private.

Example 15. Consider Fig. 10, where $SD = \{d_2, d_3\}$. Then u_2 is case 1, u_1 is case 2, u_3 is case 3, and u_4 is case 4.

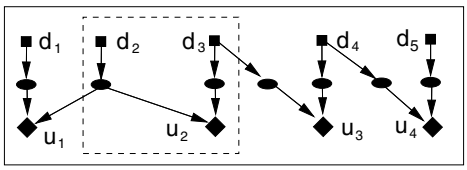

Fig. 10. Illustration of classification of utility variables

Proposition 3. *The four cases of utility variable in Def. 10 are mutually exclusive and exhaustive.*

Proof. The mutual exclusion is obvious. We show that they are exhaustive.

First, we have either $\beta_i = \emptyset$ (case 4) or $\beta_i \neq \emptyset$. Assuming $\beta_i \neq \emptyset$, we then have either $\gamma_i = \emptyset$ (case 1) or $\gamma_i \neq \emptyset$. Next, we assume $\beta_i \neq \emptyset$ and $\gamma_i \neq \emptyset$. Then either another utility variable u_j exists with a common private decision ancestor (case 3), or no such u_j exists (case 2). □

From Proposition 3, it suffices to analyze message $meu(SD)$ decomposition, enabled by u_i of each case. Denote

$$\overline{b_i} = proj(\overline{sd}, \beta_i) \quad and \quad \overline{y_i} = proj(\overline{sd}, \gamma_i).$$

[Case 1] For this case, $\overline{b_i} = proj(\overline{sd}, \alpha_i)$. We refer to the subnet segment that contains u_i, all its ancestors (including π_i and α_i), and arcs among them, as the *subnet segment of* u_i. In Fig. 10, subnet segment of u_2 is highlighted by the dashed box.

We observe that (a) contribution of u_i to $meu(SD)$ can be evaluated using its subnet segment, independently of other $u_j \in U$, and (b) the contribution is additive. To see (a), note that contribution of u_i to $meu(SD)$ is expected utility function $eu(\alpha_i)$, whose evaluation is completely determined by the subnet segment of u_i. To see (b), note that if $meu(\overline{sd}) = v$ is obtained using A's subnet, $meu(\overline{sd}) = v'$ is obtained using A's subnet with node u_i removed, and $eu(\overline{b_i}) = v''$ is obtained using the subnet segment of u_i, then

$$v = v' + v''. \tag{12}$$

Therefore, we can compute $eu(\alpha_i)$ using the subnet segment of u_i according to Eq. (3). As it involves full evaluation for each $\overline{b_i}$, the complexity is

$$O(\sigma^{|\alpha_i|}\ m\ \kappa^m). \tag{13}$$

[Case 2] For this case, the contribution of u_i to $meu(SD)$ is also additive, and obtainable from its subnet segment. Formally, in addition to a relation similar to Eq. (12) (except all terms are $meu()$ values), this can be seen from an alternative perspective. Let $(\overline{b_i}, \overline{y_i})$ be a local action profile over α_i, such that $eu(\overline{b_i}, \overline{y_i}) = \max_{\overline{y_i}''} eu(\overline{b_i}, \overline{y_i}'')$, where $eu(\alpha_i)$ is computed from the subnet segment of u_i. Let $\overline{z}$ over $D \setminus \gamma_i$ be a local action profile with $proj(\overline{z}, \beta_i) = \overline{b_i}$, and $\overline{y_i}'$ be a local action profile over γ_i, such that

$$eu(\overline{z}, \overline{y_i}') = \max_{\overline{y_i}''} eu(\overline{z}, \overline{y_i}''),$$

where $eu(D)$ is computed from A's subnet. Then, there must be

$$eu(\overline{z}, \overline{y_i}') = eu(\overline{z}, \overline{y_i}).$$

That is, if a local action profile over private decision ancestors of u_i is optimal, viewed from the subnet segment of u_i, then it is optimal viewed from the entire subnet as well.

Therefore, we can apply PeDecSu to the subnet segment of u_i. Set parameters of PeDecSu to $D = \alpha_i$, $\beta = \beta_i$, and $\gamma = \gamma_i$. The return value $meu(\beta_i)$ is the additive contribution of u_i to $meu(SD)$. The return value $peer(\beta_i)$ will be used later, and needs to be stored. Applying Proposition 1 to this case, the complexity to obtain $meu(\beta_i)$ is

$$O(\theta\ \sigma^{|\alpha_i|}\ m\ \kappa^m). \tag{14}$$

[Cases 3 and 4] Unlike utility variables in cases 1 and 2, contribution of a case 3 or case 4 utility variable to $meu(SD)$ cannot be individually and additively evaluated.

Example 16. The contribution of u_4 in Fig. 10 to $meu(d_2, d_3)$ cannot be evaluated independently using its subnet segment, because the segment contains neither d_2 nor d_3.

The contribution of u_3 to $meu(d_2, d_3)$ cannot be evaluated independently using its subnet segment either, because an optimal action profile over $\{d_3, d_4\}$ is not necessarily compatible with any optimal action profile over $\{d_3, d_4, d_5\}$.

We show below that by proper grouping of case 3 and case 4 utility variables into *correlated clusters*, we can evaluate contribution of each cluster independently and additively. Each cluster is obtained by starting with one variable of case 3, whose existence is established below.

Proposition 4. *Let SD be agent interface of a connected subnet, such that there exists a utility variable that is neither case 1 nor case 2, relative to SD. Then there exists at least one utility variable that is case 3.*

Proof. We prove by contradiction. Suppose not all utility variables are case 1 or case 2, and all utilities, that are not case 1 or case 2, are case 4. Let u_i be such a case 4 variable. By Def. 10, α_i is entirely private. Since there exists no case 3 utility variable that shares any decision ancestor in α_i, it follows that the subnet segment of u_i is disconnected from SD: a contradiction to that the subnet is connected. Therefore, from Proposition 3, there must be at least one utility variable that is case 3. □

Proposition 4 says whenever cases 1 and 2 do not cover all utilities, there exist some in case 3. Let a correlated cluster be initiated with a case 3 utility u_i. By Def. 10, there exists u_j such that u_i and u_j share a private decision ancestor. Note that u_j may be case 3 or case 4. Add u_j to the cluster, and continue until no such utility variable can be found. In Fig. 10, u_3 and u_4 form a correlated cluster. Formally, a correlated cluster is defined as follows.

Definition 11. Let $SU = \{u_1, ..., u_{\eta'}\} \subseteq U$ be a subset of utility variables in a subnet. SU is a correlated cluster if

(1) u_1 is case 3,
(2) for each u_i $(i = 2, ..., \eta')$, there exists $j < i$ with $\gamma_j \cap \gamma_i \neq \emptyset$, and
(3) no proper superset of SU satisfies the above two conditions.

We extend subnet segment of a single utility variable to *subnet segment of a correlated cluster*, which contains utility nodes in the cluster, ancestors of each utility node in the cluster, and arcs among them. The contribution to $meu(SD)$ by utility variables in a correlated cluster can be evaluated independently (of evaluations of

other clusters) and additively (e.g., Eq. (12)), using its subnet segment. This can be seen by applying the same argument for case 2 to the cluster.

From counter-examples in Example 16, we have Proposition 5 below.

Proposition 5. *A correlated utility cluster and its subnet segment is the minimum unit, where contribution of utility variables to meu(SD) can be independently and additively evaluated.*

For each correlated cluster with its utility variables indexed as $u_1, ..., u_{\eta'}$, we apply PeDecMu to its subnet segment. Denote $\alpha' = \cup_{i=1}^{\eta'} \alpha_i$, $\beta' = \cup_{i=1}^{\eta'} \beta_i$, $\gamma' = \cup_{i=1}^{\eta'} \gamma_i$. Set parameters of PeDecMu to $D = \alpha'$, $\beta = \beta'$, and $\gamma = \gamma'$. The return value $meu(\beta')$ is the additive contribution of the cluster to $meu(SD)$. Return value $peer(\beta')$ is needed later and is to be stored. Applying Proposition 2, the complexity to obtain $meu(\beta')$ is

$$O(\theta\ \sigma^{|\alpha'|}\ \eta'\ m\ \kappa^m). \tag{15}$$

Let $SU_1, SU_2, ...$ be subsets of U, where $\cup_i SU_i = U$, and each SU_i is either a singleton under case 1, or a singleton under case 2, or a correlated cluster from cases 3 or 4. Let α'_i, β'_i and γ'_i denote the sets of decision ancestor variables for SU_i. Then Eq. (11) can be computed as

$$utm_0(\overline{sd}) = (\sum_i eu_i(proj(\overline{sd}, \beta'_i))) + \sum_j meu_j(proj(\overline{sd}, \beta'_j)), \tag{16}$$

where each $eu_i()$ is the contribution from a SU_i under case 1, and each $meu_j()$ is the contribution either from a SU_j under case 2 or from a SU_j under case 3 or 4.

When $|\alpha'_i|$ for each u_i under case 1 is small, and there are enough private decision variables in the subnet, the second term in Eq. (16) (involving utility variables of cases 2, 3, 4) dominates the computation. From Eqs. (14) and (15), the complexity to obtain $utm_0(\overline{sd})$ is the following, where $|\alpha^*| = \max_i |\alpha'_i|$,

$$O(\theta\ \sigma^{|\alpha^*|}\ \eta\ m\ \kappa^m). \tag{17}$$

It is significantly more efficient than $O(\theta\ \sigma^{\rho}\ \eta\ m\ \kappa^m)$ (Proposition 2) as the complexity would be, if PeDecMu is directly applied to the subnet. It is only exponential on cardinality of the largest cluster decision ancestor set, while the latter is exponential on $\rho = |D|$.

5.2. *Utility clusters and messages from general agents*

Before generalizing to an arbitrary agent, we consider non-leaf agents in the first round of message passing. Let D be the set of decision variables of a non-leaf agent A. A receives utility messages from child agents $A_1, ..., A_k$, over interfaces $SD_1, ..., SD_k$, respectively, and then computes and sends a utility message over interface SD with parent agent B (see Fig. 11).

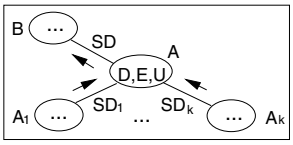

Fig. 11. First round message passing by non-leaf agent A

We denote message from A_j by $utm_j(SD_j)$, which specifies $utm_j(\overline{sd_j})$ for each local action profile $\overline{sd_j}$ over SD_j. To absorb incoming $utm_j(SD_j)$ in computing outgoing message $utm_0(SD)$ to B, we let A modifies its subnet as follows. The modification is intended to maintain the subnet as regular length-2.

For each decision variable $d_i \in SD_j$ with domain Op_i, add a new child node e_i with domain $Ef_i = Op_i$. Associate CPT $P(e_i|d_i)$ with it, such that $P(e_i|d_i) = 1$ whenever $e_i = d_i$ and $P(e_i|d_i) = 0$ otherwise. Hence, e_i is deterministically dependent on d_i. Denote the set of new child nodes added relative to SD_j as SE_j. Add a new utility node utm_j with SE_j as its parents, associate it with the function $utm_j(SE_j)$, such that $utm_j(SE_j) = utm_j(SD_j)$, and assign it weight $w_j = 1$. Note that the weight assignment does not follow sum-to-one stated in Section 2.2. The modification to A's subnet is summarized in algorithm UpdateSubnet.

After UpdateSubnet, A's subnet is still regular length-2. Message $utm_0(SD)$ outgoing to B can be computed, using the method in Section 5.1 and Eq. (16). For each $\overline{sd}$ over SD, $utm_0(\overline{sd})$ is the MEU, according to all subnets on hypertree rooted at A's subnet, as will be seen below.

Algorithm 3. UpdateSubnet
Input: decision subnet S over $D \cup E \cup U$, incoming messages $utm_1(SD_1)$, ..., $utm_k(SD_k)$ over agent interfaces $SD_1, ..., SD_k$, and agent interface SD for outgoing message;

for each SD_j,
  initialize $SE_j = \emptyset$;
  for each $d_i \in SD_j$,
    add child node e_i to d_i in S;
    set domain of e_i to $Ef_i = Op_i$;
    set CPT of e_i to deterministic $P(e_i|d_i)$;
    $SE_j = SE_j \cup \{e_i\}$;
  add utility node utm_j in S;
  for each $e \in SE_j$, connect e as a parent of utm_j;
  for each $d_i \in SD_j$, rename variable d_i in $utm_j(SD_j)$ as e_i;
  associate function $utm_j(SE_j)$ from last step with node utm_j;
  set weight for node utm_j to $w_j = 1$;
return;

For the root agent, after UpdateSubnet, it performs PeDecMu with $\beta = \emptyset$ to

obtain an optimal local action profile $\overline{d}^*$. The first round of message passing now ends.

Algorithm 4. CollectUtilPeUc
Input: decision subnet of agent A;

1 if A is not a leaf agent,
2 for each child agent A_i, receive $utm_j(SD_j)$;
3 call UpdateSubnet;
4 if A is root agent,
5 call PeDecMu with $\beta = \emptyset$ and get return value $\overline{d}^*$;
6 return $\overline{d}^*$;
7 denote the set of utility variables in current subnet by U;
8 $U' = U$;
9 classify variables in U' into 4 cases;
10 while $U' \neq \emptyset$,
11 remove $u_i \in U'$ from U';
12 if u_i is case 1, compute $eu(\alpha_i)$ using subnet segment of u_i and Eq. (3);
13 else if u_i is case 2,
14 call PeDecSu with subnet segment of u_i and parameters α_i, β_i, γ_i;
15 get return value $(meu(\beta_i), peer(\beta_i))$ and save $peer(\beta_i)$;
16 else if u_i is case 3,
17 for each $u_j \in U'$ in the same correlated cluster with u_i, remove u_j from U';
18 call PeDecMu with subnet segment of the cluster and α', β', γ';
19 get return value $(meu(\beta'), peer(\beta'))$ and save $peer(\beta')$;
20 compute $utm_0(SD)$ by Eq. (16) from $eu(\alpha_i)$, $meu(\beta_i)$ and $meu(\beta')$ above;
21 send $utm_0(SD)$ to agent B;

Algorithm 4 specifies the overall operation of a general agent A, during the first round of message passing, whose parent agent, if any, is B, and whose child agents, if any, are $A_1, ..., A_k$. The interface of A with B is SD, and that with A_i is SD_i.

A leaf agent executes lines 7 through 21. The root agent executes lines 1 through 6. Each other agent executes all lines except 4 through 6. For such an agent, the subnet at line 7 is the updated one and so is the set U.

Proposition 6 establishes the key property of CollectUtilPeUc, when executed by non-root agents.

Proposition 6. *For each non-root agent A, after running CollectUtilPeUc, its message $utm_0(SD)$ is the MEU function with respect to partial action profiles, over union of subsystems on the sub-hypertree rooted at A.*

Proof. We prove by induction on depth $dep \geq 1$ of the hypertree (length of the longest path from the root to a leaf). When $dep = 1$, A is a leaf, and $utm_0(SD)$ is obtained by Eq. (16). From Propositions 1 and 2, the statement holds.

Assume that the statement holds for $dep \leq k$, where $k \geq 1$. Consider $dep = k+1$. A is non-leaf, and lines 2 and 3 are run. Due to the inductive assumption, during UpdateSubnet, adding subnet segment for each incoming message $utm_j(SD_j)$ is equivalent to including in Eq. (16) a MEU function for the sub-hypertree rooted at child agent A_j. Hence, $utm_0(SD)$ is the MEU function for the sub-hypertree rooted at A. □

Proposition 7 establishes the key property of CollectUtilPeUc, when executed by the root agent.

Proposition 7. *For the root agent A, after running CollectUtilPeUc, the return local action profile* $\overline{d}^*$ *is globally optimal.*

Proof. Agent A runs only lines 1 through 6. By an inductive argument similar to the proof of Proposition 6, note that during UpdateSubnet, adding subnet segment for each incoming message $utm_j(SD_j)$ is equivalent to including in Eq. (7) a MEU function for the sub-hypertree rooted at child agent A_j. From Proposition 2, the statement holds. □

5.3. *Decision message distribution*

The second round of message passing starts at the root agent. It projects the optimal local action profile to interface with each adjacent agent on hypertree, and sends the projected action profile to the agent. When a non-root agent A receives the local action profile $\overline{sd}^*$ over its interface SD with the parent agent B, it uses the message to compute its optimal local plan. The computation is organized based on the four case classification of its utility variables (Section 5.1).

If u_i is case 1, then $\alpha_i \subset SD$, and the optimal local action profile over α_i is

$$\overline{sd_i}^* = proj(\overline{sd}^*, \alpha_i). \tag{18}$$

If u_i is case 2, A obtains $\overline{b_i}^* = proj(\overline{sd}^*, \beta_i)$, and retrieves $\overline{y_i}^* = peer(\overline{b_i}^*)$ using the peer function, stored during CollectUtilPeUc. The optimal local action profile over α_i is

$$\overline{sd_i}^* = (\overline{b_i}^*, \overline{y_i}^*). \tag{19}$$

For case 3 and 4 utility variables in the same correlated cluster (which we index by i), the optimal plan over the cluster's decision ancestor set α' is obtained. A obtains $\overline{b}'^* = proj(\overline{sd}^*, \beta')$, and retrieves $\overline{y}'^* = peer(\overline{b}'^*)$ using the peer function, stored during CollectUtilPeUc. The optimal local action profile over α' is

$$\overline{sd_i}^* = (\overline{b}'^*, \overline{y}'^*). \tag{20}$$

After the optimal local action profile over each α_i (cases 1 and 2) or α' (cases 3 and 4) is specified, the optimal local action profile over D (decision nodes in A) is the action profile join

$$\overline{d}^* = (\overline{sd_1}^*, \overline{sd_2}^*, ...). \tag{21}$$

Algorithm DistributeActionPeUc specifies operations of a general agent A.

Algorithm 5. DistributeActionPeUc
Input: decision subnet over $D \cup E \cup U$;

1 if A is root with $\overline{d}^*$ from CollectUtilPeUc,
2 send $proj(\overline{d}^*, SD_j)$ to each child agent A_j;
3 return;
4 receive message $\overline{sd}^*$ over interface SD with agent B;
5 $U' = U$;
6 while $U' \neq \emptyset$,
7 remove $u_i \in U'$ from U';
8 if u_i is case 1, obtain $\overline{sd_i}^*$ over α_i by Eq. (18);
9 else if u_i is case 2, obtain $\overline{sd_i^*}$ over α_i by Eq. (19);
10 else if u_i is case 3,
11 for each $u_j \in U'$ in the same correlated cluster with u_i, remove u_j from U';
12 obtain $\overline{sd_i}^*$ over α' by Eq. (20);
13 compute $\overline{d}^*$ by Eq. (21);
14 if A is non-leaf,
15 for each child agent A_j, send message $proj(\overline{d}^*, SD_j)$ to A_j;

The root agent executes lines 1 through 3. A leaf agent executes lines 4 through 13. Each other agent executes lines 4 through 15. The second round of message passing ends at leaf agents. The system coordinator executes DecisionPeUc, which causes distributed execution of the above algorithms.

Algorithm 6. DecisionPeUc

select an agent A arbitrarily;
call CollectUtilPeUc in A;
call DistributeActionPeUc in A;

Theorem 1 establishes the optimality of DecisionPeUc. The joint action profile in the theorem is a virtual object, and is never physically constructed.

Theorem 1. *After DecisionPeUc, the joint action profile formed by joining the local action profile $\overline{d}^*$ at each agent is optimal.*

Proof. By Proposition 7, after the root agent A completes CollectUtilPeUc, $\overline{d}^*$ obtained is the projection of a globally optimal joint action profile to the root subsystem.

The optimal local action profile under each interface constraint is stored as a $peer()$ function, in CollectUtilPeUc. It is retrieved in DistributeActionPeUc by each child agent A_j of root. Hence, $\overline{d}^*$ obtained by A_j, during DistributeActionPeUc, is the projection of the globally optimal joint action profile to the subsystem at A_j. Applying this argument recursively for each non-root agent down the hypertree, $\overline{d}^*$

obtained by the agent during DistributeActionPeUc is the projection of the globally optimal joint action profile to the subsystem of the agent. □

From Eq. (17), the total complexity of DecisionPeUc is approximately

$$O(n\ \theta\ \sigma^{|\alpha^*|}\ \eta\ m\ \kappa^m). \tag{22}$$

This represents a complexity reduction by a ratio of $\theta/\sigma^{\rho-|\alpha^*|}$, relative to the earlier algorithm.[15] It consists of an exponential factor $\sigma^{\rho-|\alpha^*|}$ due to utility clustering, as well as a linear factor θ due to partial evaluation.

6. Experimental Evaluation

To evaluate the effectiveness of DecisionPeUc empirically, multiple sets of experiments were performed. We report two sets of them that are representative. The first set evaluates efficiency gain due to partial evaluation and utility clustering, both individually and in combination. The experiments confirm the linear efficiency gain due to partial evaluation, and the exponential improvement due to utility clustering. Section 6.1 reports setup and result for this set of experiments.

The second set assesses the robustness of the pivot assumption (Def. 9). The pivot assumption states that, for each effect variable, probabilities of pivot effects for different action profiles are identical. The first set of experiments above also confirms optimal decision making under pivot assumption. The second set of experiments shows that optimal or near-optimal decision making can be achieved when the assumption is relaxed. Setup and result of this set of experiments are reported in Section 6.2.

6.1. *Efficiency gain due to partial evaluation and utility clustering*

MAE is used as the testbed (Example 3 with $T = 1$). The multiagent system, referred to by 4A6D, consists of $n = 4$ agents, with the hypertree organization $A_1 - A_2 - A_3 - A_4$. Agent subnets are structured similarly to those in Example 3, and are converted equivalently to regular length-2. Each agent has between two to three movement decisions that are shared with agents adjacent on the hypertree. In addition, each agent has 6 private decisions. Each decision variable has domain cardinality $\sigma = 5$. Each effect variable has domain cardinality $\kappa = 5$.

The 4A6D agent team is placed at 30 distinct team locations, and simultaneous decision making is performed for each team location, to determine the optimal joint action profile. For each team location, uncertainty on movements and other actions is represented in agent subnet CPTs, and reward distribution at neighborhood cells is encoded by utility functions. For all CPTs, the pivot assumption holds exactly, with the pivot probability value being 0.9. All utility functions are such that the density of high utility cells (of utility value > 0.5, where each utility $\in [0, 1]$) is about 5%.

The maximum number of shared decisions per agent is 3, and the number of private decisions per agent is 6. Hence, the maximum number of decisions per agent is 9. The corresponding number of local action profiles per agent is 1,953,125. The number of joint action profiles for 4A6D team is 1.455×10^{25}.

Four alternative methods for simultaneous decision making in CDNs are implemented distributively in Java:

FeNoUc The existing method[15] that is based on full evaluation
FeUc The existing method enhanced by utility clustering
PeNoUc The existing method enhanced by partial evaluation
PeUc The algorithm DecisionPeUc

Simultaneous decision making is run, distributively using a dual-core 2.9 GHz laptop, for each team location, by each method, with A_4 as the root agent.

For each team location, the decision made by FeNoUc is used as the golden standard, since it is proven to be optimal, is based on full evaluation (not influenced by pivot assumption), and is not influenced by utility clustering. The expected utility of the joint action profile obtained by FeNoUc is then compared against those obtained by alternative methods. For each team location, each of the other three methods returned joint action profiles with the expected utility identical to that of FeNoUc. Hence, optimal simultaneous decision making is achieved by all methods in each case. This confirms the optimality of DecisionPeUc, under the pivot assumption.

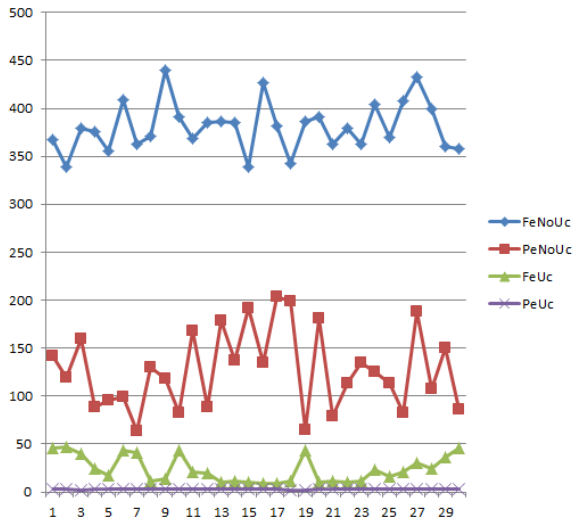

Fig. 12. 4A6D team runtimes by four alternative methods. The x-axis is indexed by team location, and the y-axis measures runtime in seconds.

Fig. 12 depicts team runtime for each team location by each method. The mean team runtime for each method across all team locations, and the corresponding standard deviation are summarized in Table 5.

Table 5. Summary of runtimes (in seconds) by alternative methods

	FeNoUc	PeNoUc	FeUc	PeUc
mean	380.8	128.6	23.8	2.8
stdev	25.6	41.4	13.9	0.3

The impact of partial evaluation on efficiency can be observed by comparing runtimes between FeNoUc and PeNoUc for each team location, and averaging the ratios. The comparison can also be performed between FeUc and PeUc. The result from data plotted in Fig. 12 shows that PeNoUc runs about 3.3 times as fast as FeNoUc, and PeUc runs about 8.7 times as fast as FeUc. In both pairs of comparisons, the runtime improvement reflects the percentage of action profiles that are fully evaluated: a linear factor. We observe that partial evaluation works more effectively, by a ratio of 2.6 (8.7/3.3), when coupled with utility clustering.

The impact of utility clustering on efficiency can be observed by a similar averaging of runtime ratio, between FeNoUc and FeUc, and between PeNoUc and PeUc. The result from data plotted in Fig. 12 shows that, FeUc runs about 22.4 times as fast as FeNoUc, and PeUc runs about 46.5 times as fast as PeNoUc. The largest utility cluster in a 4A6D agent contains 6 decision variables, and the corresponding subnet has 9 decision variables, each of which has 5 possible actions. Hence, the theoretical exponential factor of efficiency gain by utility clustering is $5^{9-6} = 125$ times (see end of Section 5.3). We observe that utility clustering works more effectively, by a factor of 2.1 (46.5/22.4), when it is coupled with partial evaluation.

The combined impact of partial evaluation and utility clustering on efficiency can be evaluated by comparing runtimes between FeNoUc and PeUc. The data shows that PeUc runs about 138.7 times as fast as FeNoUc: combining a linear factor and an exponential.

The statistical significance of the above comparisons are verified with Friedman test. The rank sums of FeNoUc, PeNoUc, FeUc, and PeUc are 120, 90 60, 30, respectively, resulting in the test statistics 90. For $\alpha = 0.005$, the critical value is 12.84. Hence, null hypothesis is rejected at the $\alpha = 0.005$ level of significance. For post-hoc analysis, 3 pairwise comparisons are performed, FeNoUc versus PeNoUc, PeNoUc versus FeUc, and FeUc versus PeUc. The rank sum difference between each pair is 30, that is larger than the critical value 29.35 for $\alpha = 0.005$. Hence, the null hypothesis in the one-sided test is rejected for each pair. In other words, at the $\alpha = 0.005$ level, PeUc is significantly faster than FeUc, FeUc is significantly faster than PeNoUc, and PeNoUc is significantly faster than FeNoUc.

Beside the above reported, we have run several additional sets of experiments, including teams 4A3D, 4A4D, and 4A5D (with 3, 4 and 5 private decisions per agent, respectively). The results generally demonstrate the similar trends, as reported above. Due to difference in numbers of decision variables, runtimes by FeNoUc increase by roughly 5 times among 4A3D, 4A4D, 4A5D, 4A6D, in that order. As decisions by 4A3D, 4A4D, 4A5D teams are less expensive, runtime differences between FeUc and PeUc are not as pronounced as 4A6D reported above (PeUc already takes only 2.8s), due to computation overhead not counted in the complexity analysis. Since results from teams such as 4A3D, 4A4D, 4A5D do not reveal as well, as the results from 4A6D, the general trends for scaling up, they are not included in our report. On the other hand, more expensive teams, such as 4A7D, would take significantly longer to run (at least $380s \times 5 = 1900s$ per execution of FeNoUc), but are expected to show only similar trends as 4A6D, they are not tested in large scale.

To evaluate how well PeUc scales up, we extended $4A6D$ agent team into additional agent teams 6A6D, 8A6D, 10A6D, 12A6D, 14A6D, 16A6D, 18A6D, 20A6D and have run PeUc on each. That is, each team has $n = 4, 6, 8, 10, 12, 14, 16, 18, 20$ agents. Each agent has up to 3 shared decisions and 6 private decisions. Recall that the number of joint action profiles for 4A6D team is 1.455×10^{25}. Each additional agent increases that number by 78,125 times. Fig. 13 depicts their runtime in seconds. As n scales up from 4 to 20, the runtime grows slightly more than linear. Although hyperchain agent organizations are used, the result is representative for hypertrees of radius n (with the root agent at the center). This demonstrates that PeUc scales up well. For 20 agents, it takes about 1 minute and hence near real-time performance is achieved by 20A6D team.

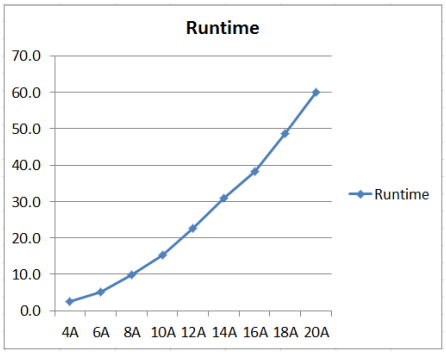

Fig. 13. PeUc runtime for teams 4A6D, 6A6D, 8A6D, 10A6D, 12A6D, 14A6D, 16A6D, 18A6D, and 20A6D

6.2. *Sensitivity of optimality to pivot assumption*

The pivot assumption requires that, for each effect variable, probabilities of pivot effects for different action profiles of its decision parents are identical. The first set of experiments reported above confirmed optimality of DecisionPeUc, when the assumption holds exactly. Since the assumption cannot be expected to hold for every application, we report below a second set of experiments, to assess robustness of the pivot assumption.

Let e_i be an effect variable of 5 possible values (the case in our testbed), with decision parent set δ_i. The pivot probability for any action profile over δ_i must be in the range $[0.2, 1)$. The lower bound 0.2 corresponds to a uniform distribution, and the upper bound to the case that all possible values of e_i are very unlikely, except the pivot effect. This range constrains our setup on pivot probabilities.

The set of experiments are divided into three groups. Each group consists of 30 4A3D team locations. For the first group, all CPTs in agent subnets have pivot probabilities in the range of $[0.8, 0.9]$. The upper bound 0.9 is identical to the pivot probability in the first set of experiments. For instance, if pivot probabilities in Table 1 were in the range of $[0.8, 0.9]$, then at least one row would have the pivot probability 0.9, at least one row would have 0.8, and other rows would have pivot probabilities in between. With pivot probabilities scattered between 0.8 and 0.9 for different action profiles, the pivot assumption no longer hold. The range for the second group is $[0.7, 0.9]$, which is twice the width as the first group. The range for the third group is $[0.3, 0.9]$, which covers almost the whole constraint range $[0.2, 1)$. A total of 90 team locations are used in the experiments.

To adopt PeUc to these environments, the method works as if the pivot assumption holds, but the pivot probability value used in decision making is either the maximum or the minimum pivot probability value in the CPT. We refer to the modified methods as PeUcMax and PeUcMin, respectively. For example, for the first group above, PeUcMax uses 0.9 as the pivot probability, and PeUcMin uses 0.8.

For each of the 90 team locations, we run FeUc to obtain the optimal expected utility over joint action profiles, as it does not depend on partial evaluation, is exact, and is more efficient than FeNoUc. We then run PeUcMax and PeUcMin, and calculate the ratio between the expected utility of the decision they obtain, and that of the decision obtained by FeUc. We will refer to this ratio as the *optimality ratio* (OR). If an execution of PeUcMax or PeUcMin obtains the identical expected utility as FeUc, the optimality ratio is 100 percent. A suboptimal result is indicated by a less than 100 percent ratio.

Fig. 14 (left) depicts the optimality ratios from PeUcMax, and Fig. 14 (right) depicts those from PeUcMin. The number of suboptimal runs, the mean optimality ratio, and the standard deviation, for each group and each method, are summarized in Table 6.

As shown by Table 6 (2nd and 3rd column), for pivot probability (PP) range

$[0.8, 0.9]$, both PeUcMax and PeUcMin made optimal decisions, for each of the 30 cases in the first group. It demonstrates that optimal decision is still achievable, when the pivot assumption does not hold, as long as pivot probability values are reasonably close.

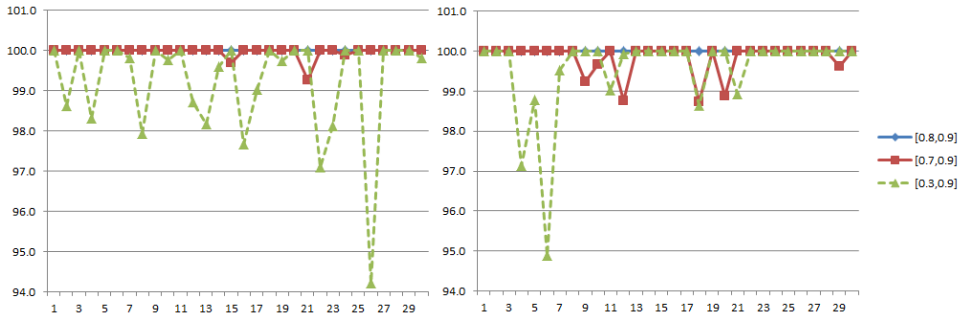

Fig. 14. Left: Optimality ratios from PeUcMax. Right: Optimality ratios from PeUcMin.

Table 6. Summary on numbers of suboptimal runs and optimality ratios by PeUcMax and PeUcMin

PP range	$[0.8, 0.9]$		$[0.7, 0.9]$		$[0.3, 0.9]$	
	PeUcMax	PeUcMin	PeUcMax	PeUcMin	PeUcMax	PeUcMin
subop runs	0	0	3	6	15	8
mean OR	100	100	99.96	99.8	99.2	99.6
stdev OR	0	0	0.1	0.4	1.3	1.1

As shown by 4th and 5th column, when pivot probability range widens to $[0.7, 0.9]$, between 10% (3 out of 30 by PeUcMax) and 20% (6 out of 30 by PeUcMin) of decisions are suboptimal. However, these suboptimal decisions yield no less than 98.5% of the optimal expected utility (see Fig. 14). It demonstrates that near optimal decision is achievable, when the pivot assumption is clearly violated, and pivot probability values are fairly different.

When the pivot probability range widens to $[0.3, 0.9]$ in the third group (last two columns), it represents the worst condition, as far as the violation of the pivot assumption is concerned. The percentages of suboptimal decisions reach 50% by PeUcMax and 27% by PeUcMin. Still, no less than 94% of the optimal expected utility is obtained.

Overall, the experiments demonstrate that, although pivot assumption is sufficient for optimality of partial evaluation, it is not a necessary condition. Partial evaluation is fairly robust with respect to the pivot assumption. Hence, combined with utility clustering, DecisionPeUc is widely applicable, where pivot assumption does not hold exactly, and it enables optimal or near-optimal simultaneous decision

making with significant efficiency gain.

7. Related Work

We discuss several frameworks for cooperative multiagent decision making. It is not an exhaustive review, but to note literature that are most closely related to CDNs. Where a framework is primarily for competitive multiagent, it is indicated explicitly.

7.1. *Decision-theoretic frameworks*

A number of frameworks for sequential decision making have been developed based on decentralized partially observable Markov decision process (DEC-POMDP) and its fully observable counterpart (DEC-MDP). The problem of generating optimal policies for DEC-POMDPs is known to be NEXP-complete.[3] CDNs focus on simultaneous decision making. The complexity of optimal decision making in general CDNs is exponential on the number of decision variables over all agents.[15] Hence, optimal solution methods are intractable for general DEC-MDPs[2] and DEC-POMDPs,[17] as well as for general CDNs.

Transition and observation independence (TOI) has been exploited for more tractable optimal policy generation[18–20] in TOI DEC-POMDPs. TOI implies that actions taken by one agent cannot affect any other agent's observation or local state. Because decision variables in CDN agent interfaces affect local states at other agents, systems encoded as CDNs are not subject to TOI restriction.

Policies for DEC-MDPs or DEC-POMDPs can be generated offline at planning time. Generation can be conducted centrally[17,18,21,22] or distributively.[23] Alternatively, planning can be performed online[24,25] to determine the next joint action profile. Since only the actual observation history, rather than all possible histories, needs to be considered, the planning can be more efficient. Between the two alternatives, execution-centric frameworks[21,26,27] combine simplified offline planning with execution-time communication of agent observation-action sequence to achieve coordination. CDNs focus primarily on simultaneous decision making. However, they are applicable to some Dec-POMDPs such as MAE (see[16] for relation between MAE and Dec-POMDP), where computation in CDNs is similar to online planning (see discussion after Example 2). DEC-POMDPs online planning methods[24,25] are approximate, while decision making methods for CDNs ([15] and this work) are optimal.

A DEC-POMDP environment is described by a set of states.[3,17,21,23,26] A factored DEC-POMDP has a state space spanned by a set of state variables,[2,22] and allows conditional independence among variables to be exploited by encoding transition and observation models through graphical models. It also allows decomposition of reward model by exploiting additive or other independence.

MSBNs, evolved from their modular forms[10,14] in early 1990s to multiagent,[11,28] are among the earliest multiagent probabilistic graphical models. CDNs are extensions of MSBNs from probabilistic frameworks to decision theoretic ones. Both

MSBNs and CDNs employ the hypertree decomposition (Def. 1). The hypertree decomposition is explored subsequently in other probabilistic graphical models, e.g., OOBNs[12] and some Dec-MDP frameworks.[13]

In addition to the hypertree decomposition, MSBNs also pioneer in inter-agent message decomposition, through *linkage trees*.[14] Neither OOBNs, nor Dec-MDP frameworks,[13] nor the earlier CDN framework[15] explore such message decomposition. A main contribution of the current work is to enable exploration of inter-agent message decomposition in CDNs, through utility clusters (Section 5). Note also that planning in some Dec-MDP frameworks[13] is approximate, while decision making in CDNs is optimal.

Some frameworks for DEC-MDP,[2] or DEC-POMDP,[26] assume that models are common knowledge. Other frameworks use centralized offline planning.[17,18,21,22,27] These frameworks are not concerned with agent privacy. In CDN decision making, identities of private variables, their associated numerical information, and decisions made over private decision variables are neither centralized nor communicated. Hence, simultaneous decision making in CDNs preserves agent privacy. Such privacy preservation is necessary for certain applications, e.g., collaborative industrial design,[8] and CDNs are well positioned to support them.

IDs[29] can be viewed as a subclass of factored POMDPs. IDs support sequential decision making, while CDNs focus on simultaneous decision making. CDN subnets differ from IDs as detailed in Section 2.3. IDs allow chance parents for decision nodes. Such is not needed, and is disallowed in CDN subnets. Arcs between decision nodes in IDs are not associated with quantitative knowledge. In CDN subnets, such arcs are associated with decision constraints, specified in terms of CPTs.

CDNs differ significantly from MAIDs.[4] An MAID is a centralized representation of a game among competitive agents. The developer of an MAID oversees the payoff structures of all agents, has the access of the entire MAID, and can analyze it by centralized computation. A CDN is a distributed representation of cooperative agents (with privacy) for optimal decision making. Each CDN agent has the access of its own subnet, without any knowledge about private variables of other subnets, nor their internal structures and numerical parameters. There is no agent in the system, who has the knowledge over all subnets. An optimal joint action profile emerges from distributed simultaneous decision making (Theorem 1). It cannot be derived by centralized computation, since no agent has all the knowledge necessary, nor is it physically represented centrally anywhere.

7.2. *Other related frameworks*

A widely applied multiagent framework is BDI,[1,30] which models the informational, motivational, and deliberative states of an agent in terms of belief, desire, and intention, respectively. The framework is not limited to cooperative agents. BDI is primarily a logic framework, and it has no quantitative evaluation of degree of optimality. On the other hand, CDN is decision-theoretic with probabilistic belief

representation, and utility based preference quantification.

Distributed constraint optimization (DCOP) is another area of multiagent cooperation. DCOP problems do not typically involve stochastic environments and hence do not consider uncertainty. A number of frameworks for solving DCOP problems exist, and an extensive review of them is beyond scope of this work. One particular framework, however, based on DPOP[5] and its various extensions,[31] is closely related to CDN decision making methods. Computation in both DPOP and CDNs follow a dynamic-programming style. However, CDN models uncertainty, while DPOP does not. DPOP uses a pseudo-tree organization, while CDN uses a junction tree organization. An organizational unit in DPOP is a single-variable-agent, while a unit in CDN is a multi-variable-subsystem.

8. Conclusion

This work makes a number of contributions to simultaneous decision making for cooperative multiagent systems in stochastic environment.

First, the agent interface in CDNs is extended to include certain chance variables while maintaining its support to more efficient decision making. Second, equivalent representation and transformation of CDNs into regular length-2 CDNs are developed, which facilitates model manipulation during decision making. The third contribution is partial evaluation, sanctioned by the pivot assumption, which ensures optimal decision making while improving efficiency by a linear factor. The experimental evaluation shows that partial evaluation is robust with respect to the pivot assumption, and is thus widely applicable. In particular, a sufficiently wide range of pivot probabilities enables the optimal or near optimal partial evaluation. Fourth, subnet decomposition by utility clusters is developed, sanctioned by existence of private decisions in each agent. Utility clustering improves efficiency further by an exponential factor. Finally, a new, general simultaneous decision making algorithm suite is formulated, that integrates the above techniques.

A number of directions deserve further investigation, including a characterization of length-2 equivalent CDNs, a general algorithm of decision equivalent transformation of CDNs into regular length-2, a necessary and sufficient condition for optimal partial evaluation, and a time-bounded partial evaluation for time-critical applications.

Acknowledgements

Financial support from NSERC, Canada through Discovery Grant is acknowledged.

References

1. A. Rao and M. Georgeff, "BDI agents: from theory to practice", *Proc. 1st Inter. Conf. on Multi-Agent Systems*, 1995, pp. 312-319.
2. C. Boutilier, "Planning, learning and coordination in multiagent decision processes", *Proc. Conf. on Theoretical Aspects of Rationality and Knowledge*, 1996, pp. 195-210.

3. D.S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of Markov decision processes", *Proc. 16th Conf. on Uncertainty in Artificial Intelligence*, 2000, pp. 32-37.
4. D. Koller and B. Milch, "Multi-agent influence diagrams for representing and solving games", *Proc. 17th Inter. Joint Conf. on Artificial Intelligence*, 2001, pp. 1027-1034.
5. A. Petcu and B. Faltings, "A scalable method for multiagent constraint optimization", *Proc. 19th Inter. Joint Conf. on Artificial Intelligence*, 2005, pp. 266-271.
6. T. Leaute and B. Faltings, "Protecting privacy through fistributed vomputation in multi-agent decision making", *J. Artificial Intelligence Research*. 47 (2013) 649-695.
7. Y. Xiang, Y. Mohamed, and W. Zhang, "Distributed constraint satisfaction with multiply sectioned constraint networks", *International J. Information and Decision Sciences*. 6(2) (2014) 127-152.
8. Y. Xiang, J. Chen, and A. Deshmukh, "A decision-theoretic graphical model for collaborative design on supply chains", in *Advances in Artificial Intelligence, LNAI 3060*, eds. A.Y. Tawfik and S.D. Goodwin (Springer, 2004) pp. 355-369.
9. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, 1988).
10. Y. Xiang, D. Poole, and M. Beddoes, "Exploring localization in Bayesian networks for large expert systems", *Proc. 8th Conf. on Uncertainty in Artificial Intelligence*, 1992, pp. 344-351.
11. Y. Xiang, "Distributed multi-agent probabilistic reasoning with Bayesian networks", in *Methodologies for Intelligent Systems*, eds. Z.W. Ras and M. Zemankova (Springer-Verlag, 1994) pp. 285-294.
12. D. Koller and A. Pfeffer, "Object-oriented Bayesian networks", *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, 1997, pp. 302-313.
13. C. Guestrin and G. Gordon, "Distributed planning in hierarchical factored MDPs", *Proc. 18th Conf. on Uncertainty in Artificial Intelligence*, 2002, pp. 197-206.
14. Y. Xiang, D. Poole, and M. Beddoes, "Multiply sectioned Bayesian networks and junction forests for large knowledge based systems", *Computational Intelligence*. 9(2) (1993) 171-220.
15. Y. Xiang, J. Chen, and W.S. Havens, "Optimal design in collaborative design network", *Proc. 4th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'05)*, 2005, pp. 241-248.
16. Y. Xiang and F. Hanshar, "Multiagent expedition with graphical models", *Inter. J. Uncertainty, Fuzziness and Knowledge-Based Systems*. 19(6) (2011) 939-976.
17. M.T.J. Spaan, F.A. Oliehoek, and C. Amato, "Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion", *Proc. 6th Workshop on Multiagent Sequential Decision-Making in Uncertain Domains*, 2011, pp. 63-70.
18. R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman, "Solving transition independent decentralized Markov decision processes", *J. Artificial Intelligence Research*. 22 (2004) 423-455.
19. R. Nair, P. Varakantham, M. Tambe, and M. Yokoo, "Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs", *Proc. 20th National Conference on Artificial Intelligence*, 2005, pp. 133-139.
20. P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo, "Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies", *Proc. Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, 2007, pp. 817-824.
21. M. Roth, R. Simmons, and M. Veloso, "What to communicate? execution-time decision in multi-agent POMDPs", *Proc. 8th Inter. Symp. on Distributed Autonomous Robotic Systems*, 2006, pp. 1-10.

22. F. Oliehoek, Matthijs Spaan, S. Whiteson, and N. Vlassis, "Exploiting locality of interaction in factored Dec-POMDPs", *Proc. 7th Inter. Conf. on Autonomous Agents and Multiagent Systems*, 2008, pp. 517-524.
23. P. Velagapudi, P. Varakantham, K. Sycara, and P. Scerri, "Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents", *Proc. Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, 2011, pp. 955-962.
24. C. Besse and B. Chaib-draa, "Parallel rollout for online solution of Dec-POMDPs", *Proc. of 21st Int. FLAIRS Conference (FLAIRS-21)*, 2008, pp. 619-624.
25. F. Wu, S. Zilberstein, and X. Chen, "Multi-agent online planning with communication", *Proc. Inter. Conf. on Automated Planning and Scheduling*, 2009, pp. 321-329.
26. D.V. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models", *J. Artificial Intelligence Research*. 16 (2002) 389-423.
27. J. Kwak, R. Yang, Z. Yin, M.E. Taylor, and M. Tambe, "Teamwork in distributed POMDPs: Execution-time coordination under model uncertainty", *Proc. 10th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2011, pp. 1261-1262.
28. Y. Xiang, "A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication", *Artificial Intelligence*. 87(1-2) (1996) 295-342.
29. R.A. Howard and J.E. Matheson, "Influence diagrams", in *Readings on the Principles and Applications of Decision Analysis*, eds. R.A. Howard and J.E. Matheson (Strategic Decisions Group, Menlo Park, CA, 1984) pp. 721-762.
30. L. de Silva, S. Sardina, and L. Padgham, "First principles planning in BDI systems", *Proc. 8th Inter. Conf. on Autonomous Agents and Multiagent Systems*, 2009, pp. 1105-1112.
31. A. Kumar, B. Faltings, and A. Petcu, "Distributed constraint optimization with structured resource constraints", *Proc. 8th Inter. Conf. on Autonomous Agents and Multiagent Systems*, 2009, pp. 923-930.

Appendix: Notation

$\mathcal{A}$:	Set of agents
n :	Number of agents
A, B, A_i :	Agent
$\mathcal{E}$:	Collection of chance variables
κ :	Max domain cardinality of chance variables
$\mathcal{D}$:	Collection of decision variables
σ :	Max domain cardinality of decision variables
$\mathcal{U}$:	Collection of utility variables
m :	Max parent set cardinality for utility variables
w_i :	Weight of utility variable u_i
aw_j :	Weight of agent A_j
S, S_i :	Subnet
D, D_i :	Set of decision variables in a subnet
E, E_i :	Set of chance variables in a subnet
U, U_i :	Set of utility variables in a subnet
V, V_i :	Collection of all variables in a subnet
ρ :	Number of decision variables in a subnet
η :	Number of utility variables in a subnet
d_i :	Decision variable
Op_i :	Domain of d_i
e_i :	Effect variable
Ef_i :	Domain of e_i
δ_i :	Parent set of e_i
u_i :	Utility variable
π_i :	Parent set of u_i
α_i :	Decision ancestors of u_i
$\overline{jd}$:	Joint action profile
$\overline{d}, \overline{d}_i$:	Local action profile
β :	Constraint scope
γ :	Optimization scope
$\overline{b}$:	Local action profile over constraint scope
$\overline{y}$:	Local action profile over optimization scope
SD :	Agent interface
$\overline{sd}$:	Local action profile over SD
θ :	Percentage of local action profiles fully evaluated