Representing and Solving Decision Problems with Limited Information*

Steffen L. Lauritzen† Dennis Nilsson†

June 7, 2001

Abstract

We introduce the notion of **LI**mited **M**emory **I**nfluence **D**iagram (LIMID) to describe multi-stage decision problems where the traditional assumption of no forgetting is relaxed. This can be relevant in situations with multiple decision makers or when decisions must be prescribed under memory constraints, such as e.g. in partially observed Markov decision processes (POMDPs). We give an algorithm for improving any given strategy by local computation of single policy updates and investigate conditions for the resulting strategy to be optimal.

Key words: Local computation; message passing; optimal strategies; partially observed Markov decision process, single policy updating.

---

*To appear in *Management Science*.

†Department of Mathematical Sciences, Aalborg University, Fredrik Bajers Vej 7G, DK-9220 Aalborg, Denmark.

1 Introduction

Influence diagrams (Howard and Matheson 1984) are compact representations of decision problems under uncertainty, and local computation algorithms for solving these have been developed, for example, by Olmsted (1983), Shachter (1986), Shenoy (1992) and Jensen *et al*. (1994).

In the present article we relax the standard assumption in an influence diagram of 'no forgetting', i.e. that values of observed variables and decisions that have been taken are remembered at all later times. We denote these more general diagrams by LIMIDs (**LI**mited **M**emory **I**nfluence **D**iagrams). Such diagrams have also been studied by Zhang *et al*. (1994) with a similar motivation as ours under the name of *decision networks*. We have chosen to use a less general term.

Partially observed Markov decision processes, also known as POMDPs, can be seen as special types of influence diagrams that develop over time. As opposed to fully observed Markov decision processes (Howard 1960), the complexity of POMDP algorithms grows quickly with time and they fail therefore to provide the optimal solutions desired, see Lovejoy (1991) and White (1991) for surveys. Formulating finite POMDPs as LIMIDs allows handling explicit memory constraints, as described in the example below.

Example 1 **(**PIGS**)** Although this example is fictitious, more realistic variants have served as motivation for the theoretical developments in this paper.

> A pig breeder is growing pigs for a period of four months and subsequently selling them. During this period the pig may or may not develop a certain disease. If the pig has the disease at the time when it must be sold, the pig must be sold for slaughtering and its expected market price is then 300 DKK (Danish kroner). If it is disease free, its expected market price as a breeding animal is 1000 DKK.

Once a month, a veterinary doctor sees the pig and makes a test for presence of the disease. If the pig is ill, the test will indicate this with probability .80, and if the pig is healthy, the test will indicate this with probability .90. At each monthly visit, the doctor may or may not treat the pig for the disease by injecting a certain drug. The cost of an injection is 100 DKK.

A pig has the disease in the first month with probability .10. A healthy pig develops the disease in the subsequent month with probability .20 without injection, whereas a healthy and treated pig develops the disease with probability .10, so the injection has some preventive effect. An untreated pig which is unhealthy will remain so in the subsequent month with probability .90, whereas the similar probability is .50 for an unhealthy pig which is treated. Thus spontaneous cure is possible but treatment is beneficial on average.

The story now continues in two versions. In the traditional influence diagram (ID) version, the pig breeder will at all times know whether the pig has been treated earlier and also the previous test results. This story corresponds to a (finite) POMDP. If we extend the story to continue for many months or to have weekly or daily examinations with potential injections associated, the complexity of finding an optimal treatment strategy becomes forbidding.

In the LIMID version of the story, the pig breeder does not keep individual records for his pigs and has to make his decision knowing only the test result for the given month and the age of the pig. The memory has been limited to the extreme of only remembering the present. □

Situations in which multiple decision makers cooperate lend themselves naturally to situations with limited memory. One decision maker can only communicate partially what has previously happened, and therefore the decision makers may not have the full history available when a given decision must be taken. Also, multiple decision makers can sometimes make their decisions in parallel, without fully sharing information.

The LIMID representation of decision problems is described in Section 2. The complexity of finding fully optimal strategies within LIMIDs is in general prohibitive. In Section 3 we describe the procedure of *Single Policy Updating* which leads to strategies which are locally optimal, in the sense that no single policy modification can increase the expected utility of the strategy. We also show how to calculate this strategy by message passing in a suitable junction tree. In Section 4 we establish general conditions for local optimal strategies to be globally optimal and provide algorithms for identifying such cases and reducing computational complexity. These results extend and generalize those of Zhang *et al*. (1994).

2 Describing LIMIDs

2.1 Diagrams

LIMIDs are represented by directed acyclic graphs (DAGs) with three types of nodes. *Chance nodes*, displayed as circles, represent random variables. *Decision nodes*, displayed as squares, correspond to alternative choices available to the decision maker. Finally *value nodes*, displayed as diamonds, represent additive components of the joint utility function. We assume that the joint utility of a configuration of the chance and decision variables can be represented as the sum of the local utility functions associated with the value nodes.

A node n_2 is a *child* of node n_1 if there is an arc from n_1 to n_2. In this case n_1 is a *parent* of n_2. The set of parents of n is denoted by $\mathrm{pa}(n)$. The *family* of n is $\mathrm{fa}(n) = \mathrm{pa}(n) \cup \{n\}$ and for a subset A of nodes, we let $\mathrm{fa}(A) = \cup_{\alpha \in A} \mathrm{fa}(\alpha)$. If there is a directed path from n_1 to n_2, n_2 is a

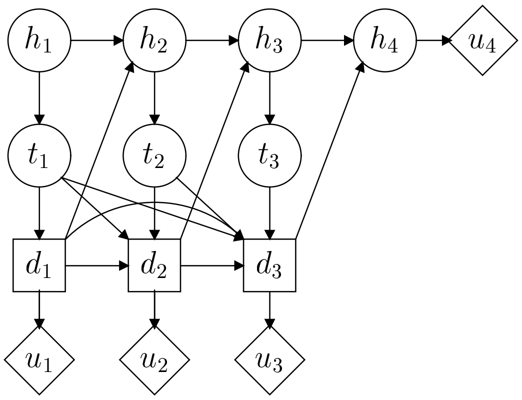

Figure 1: LIMID representation of the ID version of PIGS. The full previous treatment and test history is available when decisions must be taken.

descendant of n_1 and n_1 is an *ancestor* of n_2. The set of descendants and the set of ancestors of n is denoted $\mathrm{de}(n)$ and $\mathrm{an}(n)$ respectively.

The arcs in a LIMID have a different meaning depending on the type of node they go into. If chance node r_1 (connoting *r*andom variable) is a parent of chance node r_2, it indicates that the distribution of (the random variable) r_2 is specified conditionally on the value of r_1. A decision node d is a parent of chance node r if the distribution of r can depend on decision d. A decision node d_1 is a parent of decision node d_2 if the choice of alternative for decision d_1 is known to the decision maker when decision d_2 is taken and may influence that decision. When chance node r is a parent of a decision node d it indicates that the value of r will be known when decision d is taken and might influence that decision. Finally arcs into value nodes represent the decision maker's (expected) utility given the states of its parents. Value nodes cannot have children.

Example 2 (Diagrams for PIGS) To represent the ID version of PIGS by a LIMID, we let h_i, $(i = 1, \ldots, 4)$ denote the (chance) variables which indicate whether the pig is *healthy* or *unhealthy* in the ith month and t_i, $(i = 1, 2, 3)$ represent the corresponding test results, which are said to be *positive* if they indicate presence of the disease, and otherwise *negative*. The nodes d_i, $(i = 1, 2, 3)$

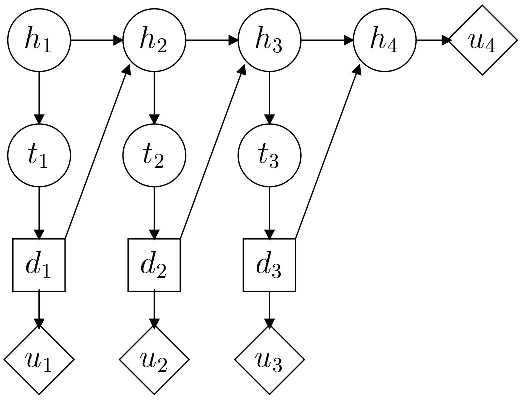

Figure 2: LIMID version of PIGS. Only current test result is available when decisions are taken.

correspond to the decisions *treat* or *leave*, the latter implying that no injection is made. The utility nodes u_1, u_2, u_3 represent the potential injection costs, whereas u_4 is the (expected) market price of the pig as determined by its health at the fourth month. The diagram is displayed in Figure 1. The assumption of 'no forgetting' is made explicit by arcs from predecessors of decision nodes. This is in contrast to the convention used by e.g. Jensen *et al*. (1994).

In the LIMID version of the story, the pig breeder does not keep individual records for his pigs and has to make his decision knowing only the test result for the given month. The corresponding diagram is displayed in Figure 2. □

A LIMID can be viewed as a special type of Bayesian network, where the state of each decision variable is to be imposed from the decision maker to meet an optimization objective, and where the variables at the value nodes are completely determined from its parent configurations. A LIMID differs from a traditional ID representation of a decision problem in two ways:

(i) The sequence in which decisions are to be taken is not specified other than through it being compatible with the partial order induced by the DAG, i.e. if d_2 is a descendant of d_1, the decision d_1 must be taken before d_2.

(ii) The parents of a decision node d represent exactly the variables whose values are known and taken into consideration when d is to be taken. In traditional IDs, this relation is more complicated and varies somewhat between authors.

Thus, LIMIDs allow for multiple or forgetful decision makers.

2.2 Specifications

For a given LIMID, we denote the sets of decision and chance nodes by Δ and Γ respectively and let $V = \Delta \cup \Gamma$. The set of value nodes is denoted by Υ.

Each node $n \in V$ is associated with a variable which takes values in a finite set $\mathcal{X}_n$ of states. Usually we use the label n interchangeably with the variable associated with n. For $W \subseteq V$ we write $\mathcal{X}_W = \times_{n \in W} \mathcal{X}_n$. Typical elements in $\mathcal{X}_W$ are called *configurations* and denoted by $x_W = (x_n)_{n \in W}$ and so on.

For every chance node r there is a non-negative real function p_r on $\mathcal{X}_r \times \mathcal{X}_{\mathrm{pa}(r)}$ such that for each configuration of $\mathrm{pa}(r)$, p_r adds to 1 when summing over the configurations in $\mathcal{X}_r$. The terms p_r are not necessarily conditional distributions but rather a family of distributions parametrized by the states of $\mathrm{pa}(r)$, since the distribution of $\mathrm{pa}(r)$ is generally unspecified. If node r has no parents then p_r represents the marginal distribution of r.

Each value node $u \in \Upsilon$ has an associated real function U_u on $\mathcal{X}_{\mathrm{pa}(u)}$ that represents an additive component of the joint utility function.

2.3 Policies and strategies

A *pure policy* for $d \in \Delta$ prescribes an alternative in $\mathcal{X}_d$ for each possible configuration of its parents $\mathrm{pa}(d)$. To allow for possible randomization we consider more general policies represented by functions δ_d on $\mathcal{X}_d \times \mathcal{X}_{\mathrm{pa}(d)}$ which represent a probability distribution over alternative choices of d for each possible configuration of $\mathrm{pa}(d)$.

A *strategy* for a LIMID is a set $q = \{\delta_d : d \in \Delta\}$ of policies. A *pure strategy* is a set $q = \{\delta_d : d \in \Delta\}$ of pure policies. If a strategy is not pure it is called *random*.

A strategy $q = \{\delta_d : d \in \Delta\}$ induces a joint distribution of all the variables in V as

$$f_q = \prod_{r \in \Gamma} p_r \prod_{d \in \Delta} \delta_d. \tag{1}$$

For all such q, p_r represents the appropriate conditional distribution calculated with respect to f_q. If U is the joint utility, i.e. $U = \sum_{u \in \Upsilon} U_u$, then the expected utility of q is given by

$$\mathrm{EU}(q) = \mathbf{E}_q(U) = \sum_x f_q(x) U(x) = \sum_x f_q(x) \left\{ \sum_{u \in \Upsilon} U_u(x_{\mathrm{pa}(u)}) \right\}$$

and we are searching for a strategy q which maximizes the expected joint utility. We define:

Definition 1 A *global maximum strategy* $\hat{q}$ is a strategy which satisfies

$$\mathrm{EU}(\hat{q}) \geq \mathrm{EU}(q) \text{ for all strategies } q.$$

Given any strategy $q = \{\delta_d : d \in \Delta\}$ and any $d_0 \in \Delta$, we let $q_{-d_0} = \{\delta_d : d \in \Delta \setminus \{d_0\}\}$ be the partially specified strategy obtained by retracting the policy at d_0 and for any policy δ'_{d_0} we let

$$\delta'_{d_0} * q = \{\delta'_{d_0}\} \cup q_{-d_0}.$$

So $\delta'_{d_0} * q$ denotes the strategy obtained from q by replacing δ_{d_0} with δ'_{d_0} and leaving all other policies as in q. Next we let

Definition 2 A *local maximum policy* for a strategy q at d, is a policy $\tilde{\delta}_d$ which satisfies

$$\mathrm{EU}\left(\tilde{\delta}_d * q\right) = \sup_{\delta'_d} \mathrm{EU}\left(\delta'_d * q\right).$$

Definition 3 A strategy $\tilde{q}$ is said to be a *local maximum strategy* if all its policies are local maximum policies, i.e. if for all $d \in \Delta$ and all policies δ_d we have $\mathrm{EU}(\tilde{q}) \geq \mathrm{EU}(\delta_d * \tilde{q})$.

In other words, a strategy is a local maximum strategy if and only if the expected utility does not increase by changing only one of its policies.

For later use, we establish the following lemma, where $f_{q_{-d}}$ is defined through (1) and the partial strategy q_{-d}. Note that $f_{q_{-d}}$ is not a joint probability distribution.

Lemma 1 *A policy $\tilde{\delta}_d$ is a local maximum policy for a strategy q at d if and only if for all $x_{\mathrm{fa}(d)}$ with $\tilde{\delta}_d(x_d \mid x_{\mathrm{pa}(d)}) > 0$ we have*

$$x_d = \arg\max_{z_d} \sum_{x_{V\setminus \mathrm{fa}(d)}} f_{q_{-d}}(x_{V\setminus d}, z_d) U(x_{V\setminus d}, z_d). \tag{2}$$

Proof Let $\hat{U}(x_{\mathrm{pa}(d)}) = \max_{z_d} \sum_{x_{V\setminus \mathrm{fa}(d)}} f_{q_{-d}}(x_{V\setminus d}, z_d) U(x_{V\setminus d}, z_d)$. For all policies δ_d we have

$$\begin{aligned} \mathrm{EU}(\delta_d * q) &= \sum_x \delta_d(x) f_{q_{-d}}(x) U(x) \\ &= \sum_{x_{\mathrm{fa}(d)}} \delta_d(x_d \mid x_{\mathrm{pa}(d)}) \sum_{x_{V\setminus \mathrm{fa}(d)}} f_{q_{-d}}(x) U(x) \\ &\leq \sum_{x_{\mathrm{fa}(d)}} \delta_d(x_d \mid x_{\mathrm{pa}(d)}) \hat{U}(x_{\mathrm{pa}(d)}). \end{aligned}$$

As the inequality is sharp if and only if $\delta_d(x_d \mid x_{\mathrm{pa}(d)}) > 0$ for some $x_{\mathrm{fa}(d)}$ which does not satisfy (2), we obtain the desired result. □

3 Single Policy Updating by message passing

3.1 Single Policy Updating

This subsection describes an iterative procedure for improving strategies within LIMIDs termed *Single Policy Updating* (SPU). From an initial strategy q^0 it updates each policy in some order.

Assume that the current strategy is q^l and that the policy for d_0 is to be updated. Then the next strategy q^{l+1} only differs from q^l on the policy for d_0 and it is generated by finding a local maximum policy $\delta_{d_0}^{l+1}$ for q^l at d_0 and letting $q^{l+1} = \delta_{d_0}^{l+1} * q^l$.

When all the policies have been updated once we say that one *cycle* has been performed. The algorithm stops if the expected utility of strategies generated in two successive cycles is unaltered. A few comments are in place here:

- The initial strategy q^0 may be random and it is typically advantageous to choose it as such.
- Section 3.6 develops an efficient algorithm that finds a local maximum policy in each step.
- There is always a pure local maximum policy $\delta_{d_0}^{l+1}$; however it may not be unique.

If we always choose a pure local maximum policy in each step, the algorithm eventually reaches a local maximum strategy at which no progress is made. This holds because there is only a finite number of pure strategies and the expected utility increases at each cycle. These observations are stated formally below.

Iterative improvement *SPU is an iterative improvement algorithm: after each cycle, the expected utility of the current strategy has increased or is unaltered. In the latter case, the algorithm has reached a local maximum strategy.*

Convergence *SPU converges to a local maximum strategy if we always choose a pure policy in each updating step. In this case the algorithm converges in a finite number of cycles.*

We shall later discuss how to choose initial strategies, updating sequences, and give conditions ensuring local maximum strategies to be global maximum strategies.

SPU has a strong resemblance with the method of ‘policy iteration’ used in Markov Decision Processes (MDP) (Howard 1960). Indeed, consider a stationary MDP with infinite horizon and an initial stationary strategy that uses policy δ at all decisions $d_1, d_2, \ldots$. We apply SPU and first update the policy at d_1 to a local maximum policy δ^*, say. We next repeat this at d_2, but since the MDP has infinite horizon, the decision problem associated with $d_2, d_3, \ldots$ is identical to that of $d_1, d_2, \ldots$, so δ^* must also be local maximum for d_2. Continuing *ad infinitum,* all policies become updated to δ^*. Hence, with this little 'twist of infinity', one cycle of SPU specializes to one step of policy iteration, see also the comments at the end of Section 3.6.

3.2 Potentials and their operations

In our local computation algorithms we represent the quantitative elements of a LIMID through entities called potentials. Each such potential has two parts as detailed below.

Definition 4 Let $W \subseteq V$. A *potential* on W is a pair $\pi_W = (p_W, u_W)$ of real-valued functions on $\mathcal{X}_W$ where p_W is non-negative.

The first part p_W of the potential is called the *probability part*, and the second part u_W is called the *utility part*. We call the probability part *vacuous* if it is equal to unity, and the utility part is vacuous if it is identically equal to zero.

We identify two potentials $\pi_W^1 = (p_W^1, u_W^1)$ and $\pi_W^2 = (p_W^2, u_W^2)$ on W and write $\pi_W^1 = \pi_W^2$ if

$$p_W^1 = p_W^2 \text{ and } u_W^1(x_W) = u_W^2(x_W) \text{ whenever } p_W^1(x_W) = p_W^2(x_W) > 0,$$

i.e. two potentials are considered equal if they have identical probability parts and their utility parts agree almost surely with respect to the probability parts.

To represent and evaluate the decision problem in terms of potentials, we define *combination* and *marginalization* and verify the axioms of Shenoy and Shafer (1990).

Definition 5 [**Combination**]

Let $\pi_{W_1} = (p_{W_1}, u_{W_1})$ and $\pi_{W_2} = (p_{W_2}, u_{W_2})$ be two potentials on W_1 and W_2 respectively. The *combination* $\pi_{W_1} \otimes \pi_{W_2}$ of π_{W_1} and π_{W_2} is the potential on $W_1 \cup W_2$ given by

$$\pi_{W_1} \otimes \pi_{W_2} = (p_{W_1} p_{W_2}, u_{W_1} + u_{W_2}).$$

To define marginalization of potentials we first introduce the *sum-marginal* $\sum_{W \setminus W_1} \phi_W$ of a real-valued function ϕ_W on $\mathcal{X}_W$ for $W_1 \subseteq W$. This is simply the 'usual' marginal

$$\left(\sum_{W \setminus W_1} \phi_W \right) (x_{W_1}) = \sum_{y_W : y_{W_1} = x_{W_1}} \phi_W(y_W).$$

Then we define marginalization of potentials as follows:

Definition 6 [**Marginalization**]

Let $\pi_W = (p_W, u_W)$ be a potential on W, and let $W_1 \subseteq W$. The *marginalization* $\pi_W^{\downarrow W_1}$ of π_W onto W_1 is the potential on W_1 given by

$$\pi_W^{\downarrow W_1} = \left(\sum_{W \setminus W_1} p_W \ , \ \frac{\sum_{W \setminus W_1} p_W u_W}{\sum_{W \setminus W_1} p_W} \right).$$

The division operation in the utility part is necessary to preserve expected utilities. The convention $0/0 = 0$ has been used here and throughout.

The notion of potential and combination are similar to what is used in Shenoy (1992), Jensen *et al*. (1994) and Cowell *et al*. (1999). Marginalization is what these authors have termed sum-marginalization.

The first axiom amounts to combination satisfying the properties of a commutative semigroup, i.e. being associative and commutative:

Lemma 2 (Commutativity and associativity of combination)

Suppose π_{W_1}, π_{W_2}, *and* π_{W_3} *are potentials. Then*

$$\pi_{W_1} \otimes \pi_{W_2} = \pi_{W_2} \otimes \pi_{W_1} \text{ and } (\pi_{W_1} \otimes \pi_{W_2}) \otimes \pi_{W_3} = \pi_{W_1} \otimes (\pi_{W_2} \otimes \pi_{W_3}).$$

Proof Follows directly from the definitions. □

The fundamental properties of marginalization corresponding to the last two axioms are established in the two lemmas below.

Lemma 3 (Consonance of marginalization)

Suppose π_W is a potential on W, and suppose $W \supseteq W_1 \supseteq W_2$. Then

$$(\pi_W^{\downarrow W_1})^{\downarrow W_2} = \pi_W^{\downarrow W_2}.$$

Proof Let $\pi_W = (p_W, u_W)$. The probability parts of the two potentials $(\pi_W^{\downarrow W_1})^{\downarrow W_2}$ and $\pi_W^{\downarrow W_2}$ are clearly the same. The utility part of $(\pi_W^{\downarrow W_1})^{\downarrow W_2}$ is given by

$$\frac{\sum_{W_1 \setminus W_2} \left\{ \left(\sum_{W \setminus W_1} p_W \right) \left(\frac{\sum_{W \setminus W_1} p_W u_W}{\sum_{W \setminus W_1} p_W} \right) \right\}}{\sum_{W_1 \setminus W_2} \left(\sum_{W \setminus W_1} p_W \right)}.$$

Since $(\sum_{W \setminus W_1} p_W)(x_{W_1}) = 0 \Rightarrow (\sum_{W \setminus W_1} p_W u_W)(x_{W_1}) = 0$, the above expression reduces to

$$\frac{\sum_{W \setminus W_2} p_W u_W}{\sum_{W \setminus W_2} p_W}$$

which is the utility part of $\pi_W^{\downarrow W_2}$. □

Lemma 4 (Distributivity and marginalization over combination)

Suppose π_{W_1} and π_{W_2} are potentials on W_1 and W_2 respectively. Then

$$(\pi_{W_1} \otimes \pi_{W_2})^{\downarrow W_1} = \pi_{W_1} \otimes \pi_{W_2}^{\downarrow W_1}.$$

Proof Let $\pi_{W_1} = (p_{W_1}, u_{W_1})$ and $\pi_{W_2} = (p_{W_2}, u_{W_2})$. The probability part for the potentials $\pi_{W_1} \otimes \pi_{W_2}^{\downarrow W_1}$ and $(\pi_{W_1} \otimes \pi_{W_2})^{\downarrow W_1}$ are easily seen to agree and be equal to $\sum_{W_2 \setminus W_1} p_{W_1} p_{W_2}$. If

$(\sum_{W_2\setminus W_1} p_{W_1}p_{W_2})(x_{W_1}) > 0$, the utility part of $(\pi_{W_1} \otimes \pi_{W_2})^{\downarrow W_1}$ is

$$\begin{aligned}\frac{\sum_{W_2\setminus W_1} p_{W_1}p_{W_2}(u_{W_1}+u_{W_2})}{\sum_{W_2\setminus W_1} p_{W_1}p_{W_2}} &= \frac{\sum_{W_2\setminus W_1} p_{W_1}p_{W_2}u_{W_1}}{\sum_{W_2\setminus W_1} p_{W_1}p_{W_2}} + \frac{\sum_{W_2\setminus W_1} p_{W_1}p_{W_2}u_{W_2}}{\sum_{W_2\setminus W_1} p_{W_1}p_{W_2}} \\ &= u_{W_1} + \frac{\sum_{W_2\setminus W_1} p_{W_2}u_{W_2}}{\sum_{W_2\setminus W_1} p_{W_2}}\end{aligned}$$

which is the utility part of $\pi_{W_1} \otimes \pi_{W_2}^{\downarrow W_1}$. □

Lemmas 2-4 together say that combination and marginalization satisfy the Shenoy-Shafer axioms, establishing correctness of the propagation scheme presented in Section 3.5.

3.3 From LIMIDs to junction trees

As mentioned, our algorithm proceeds by message passing in a suitable computational structure known as a junction tree. In the present subsection we describe how to construct this junction tree.

As for similar local computation algorithms the construction involves first a moralization process, in which an undirected graph is constructed, then a triangulation, where additional edges are added, and finally the organization of the cliques of the triangulated graph into a junction tree.

The transformation from the LIMID $\mathcal{L}$ to an undirected graph is made by first adding undirected edges between all nodes with a common child (including children that are value nodes). As value nodes do not have children, only edges between chance or decision nodes are added. Then we drop the directions on all arcs and finally remove all value nodes. The resulting 'moral' graph is denoted by $\mathcal{L}^m$.

Next, edges are added to the undirected graph $\mathcal{L}^m$ to form a triangulated graph $\bar{\mathcal{L}}^m$. It is important to note that, in contrast with the local computation methods described by Jensen *et al.* (1994) and others, the triangulation does not need to respect any specific partial or total ordering

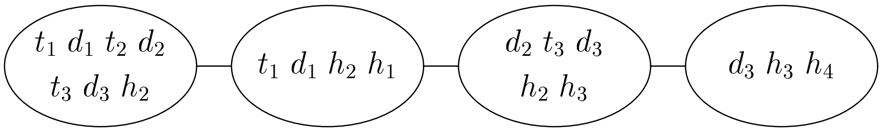

Figure 3: Junction tree for the ID version of PIGS from Figure 1.

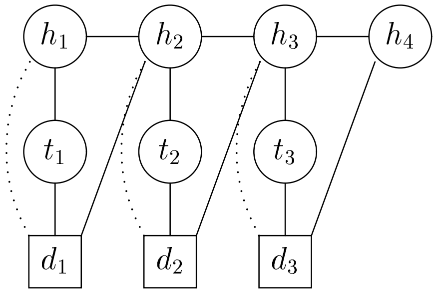

Figure 4: Moral graph of the LIMID version of PIGS from Figure 2. The moral graph is triangulated so no additional edges are needed.

of the nodes, but the triangulation can simply be chosen to minimize the computational costs, for example as described in Kjærulff (1992).

Finally the cliques $\mathcal{C}$ of $\bar{\mathcal{L}}^m$ are organized into a *junction tree* $\mathcal{T}$ having the property that for any $n \in V$, the collection of all cliques containing n correspond to a connected subtree of $\mathcal{T}$. This can be done in a number of ways (Cowell *et al*. 1999).

Example 3 (Junction trees for PIGS) The ID version of PIGS leads to the junction tree displayed in Figure 3. Note that the full set of decision variables and test variables are contained in a single clique. If the story is extended by more months, this clique will grow correspondingly and the complexity of finding an optimal solution becomes forbidding.

The LIMID version of PIGS has the moral graph displayed in Figure 4. This is triangulated and a junction tree is displayed in Figure 5. The maximal clique size does not increase with time. □

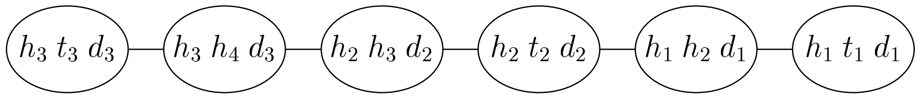

Figure 5: Junction tree for the LIMID version of PIGS from Figure 2.

3.4 Initialization

Suppose we are given a LIMID $\mathcal{L}$ and an initial strategy $q = \{\delta_d : d \in \Delta\}$. To initialize the junction tree $\mathcal{T}$ one first associates a vacuous potential to each clique $C \in \mathcal{C}$. Then for each chance node r in $\mathcal{L}$, p_r is multiplied onto the probability part of the potential of an arbitrary clique containing $\mathrm{fa}(r)$. When this has been done, one takes each value node $u \in \Upsilon$ and adds U_u to the utility part of the potential of any clique containing $\mathrm{pa}(u)$. The moralization process has ensured the existence of such cliques.

In principle we now continue by multiplying initial policies for each decision node d onto the probability part of a clique containing $\mathrm{fa}(d)$. However, as we later want to retract and change these policies, we store them separately from the probability part and only perform the necessary multiplication during message passing as described below in Section 3.5.

In a ‘lazy’ version of the algorithm (Madsen and Jensen 1998), the individual factors and terms in the probability and utility parts would also be stored separately until needed for message passing.

Let $\pi_C = (p_C, u_C)$ be the potential on clique C after these operations have been performed. The *joint potential* is equal to the combination of all the clique potentials and satisfies

$$\pi_V = \otimes\{\pi_C : C \in \mathcal{C}\} = (p_V, u_V) = \left(\prod_{r\in\Gamma} p_r \prod_{d\in\Delta} \delta_d, \sum_{u\in\Upsilon} U_u\right) = (f_q, U). \tag{3}$$

3.5 Collect propagation

Let $\{\pi_C : C \in \mathcal{C}\}$ be a collection of potentials on the junction tree $\mathcal{T}$. Let $\pi_V = \otimes\{\pi_C : C \in \mathcal{C}\}$ and suppose we wish to find the marginal $\pi_V^{\downarrow R}$ for some clique $R \in \mathcal{C}$.

To achieve our purpose we direct all the edges in $\mathcal{T}$ towards the 'root-clique' R. Then each clique passes a message to its child after having received messages from all its other neighbours. The structure of a *message* $\pi_{C_1 \to C_2}$ from clique C_1 to its neighbour C_2 is given by

$$\pi_{C_1 \to C_2} = \Big(\pi_{C_1} \otimes \Big(\otimes_{C \in \mathrm{ne}(C_1) \setminus \{C_2\}} \pi_{C \to C_1}\Big)\Big)^{\downarrow C_2},$$

where $\mathrm{ne}(C_1)$ are the neighbours of C_1 in $\mathcal{T}$ and $\pi_{C \to C_1}$ is the message from C to C_1.

In words, the message which C_1 sends to its neighbour is the combination of all the messages that C_1 receives from its other neighbours together with its own potential, suitably marginalized.

The result below now follows from the fact that the two mappings, combination ($\otimes$) and marginalization ($\downarrow$) obey the Shafer–Shenoy axioms, as shown in Lemma 2–4.

Theorem 1 *Consider a joint potential π_V on a junction tree $\mathcal{T}$ with cliques $\mathcal{C}$, and pass messages towards a 'root-clique' $R \in \mathcal{C}$ as described above. When R has received a message from each of its neighbours, the combination of all messages with its own potential is equal to the R-marginal of the joint potential π_V: $\pi_V^{\downarrow R} = (\otimes_{C \in \mathcal{C}} \pi_C)^{\downarrow R} = \pi_R \otimes \Big(\otimes_{C \in \mathrm{ne}(R)} \pi_{C \to R}\Big)$.*

3.6 Local optimization

This section is concerned with showing how to find a local maximum policy during SPU.

Definition 7 Let $\pi_W = (p_W, u_W)$ be a potential. The *contraction* of π_W, denoted $\mathrm{cont}(\pi_W)$, is defined as the real function on $\mathcal{X}_W$ given as $\mathrm{cont}(\pi_W) = p_W u_W$.

We shall need the following theorem.

Theorem 2 *For a potential $\pi_W = (p_W, u_W)$ on W and $W_1 \subseteq W$ we have*

$$\operatorname{cont}(\pi_W^{\downarrow W_1}) = \sum_{W \setminus W_1} \operatorname{cont}(\pi_W).$$

Proof By definition, the marginal $\pi_W^{\downarrow W_1}$ is given as

$$\pi_W^{\downarrow W_1} = \left(\sum_{W \setminus W_1} p_W, \frac{\sum_{W \setminus W_1} p_W u_W}{\sum_{W \setminus W_1} p_W} \right).$$

In the utility part of $\pi_W^{\downarrow W_1}$ we have for all x_{W_1} that the numerator is zero whenever the denominator is zero. This allows us to write

$$\operatorname{cont}(\pi_W^{\downarrow W_1}) = \sum_{W \setminus W_1} p_W u_W = \sum_{W \setminus W_1} \operatorname{cont}(\pi_W),$$

which was to be proved. □

Recall from (1) that f_q denotes the probability distribution obtained by effectuating the strategy q. Note that with the joint potential π_V defined by (3) it holds that $f_q(x)U(x) = \operatorname{cont}(\pi_V)$. We then further have:

Corollary 1 *Let the joint potential π_V on the junction tree be given by*

$$\pi_V = \left(\prod_{r \in \Gamma} p_r \prod_{d \in \Delta} \delta_d, U \right) = (f_q, U),$$

where $q = \{\delta_d : d \in \Delta\}$ is a strategy. Then the expected utility of q is $\mathrm{EU}(q) = \operatorname{cont}(\pi_V^{\downarrow \emptyset})$.

Proof By definition

$$\mathrm{EU}(q) = \sum_x f_q(x)U(x) = \sum_V \operatorname{cont}(\pi_V) = \operatorname{cont}(\pi_V^{\downarrow \emptyset}),$$

where the last equality follows from Theorem 2. □

Suppose we wish to compute a local maximum policy for q at d, and suppose the policy for d is assigned to clique R. Let $\tilde{\pi}_R$ be the potential on R obtained by retracting the policy for d from π_R. Then by Theorem 2

$$\begin{aligned} \sum_{x_{V\setminus \mathrm{fa}(d)}} f_{q_{-d}}(x)U(x) &= \sum_{V\setminus \mathrm{fa}(d)} \mathrm{cont}\left(\tilde{\pi}_R \otimes (\otimes_{C:C\neq R}\pi_C)\right) \\ &= \mathrm{cont}\left((\tilde{\pi}_R \otimes (\otimes_{C:C\neq R}\pi_C))^{\downarrow \mathrm{fa}(d)}\right). \end{aligned}$$

Combining with Lemma 1, a local maximum policy for strategy q at d can be found by carrying out the following steps:

1. **Retract:** Retract the policy for d from the potential on R to obtain $\tilde{\pi}_R$.

2. **Collect:** Collect to R to obtain $\pi_R^* = (\tilde{\pi}_R \otimes (\otimes_{C:C\neq R}\pi_C))^{\downarrow R}$ as in Theorem 1.

3. **Marginalize:** Compute $\pi_{\mathrm{fa}(d)}^* = (\pi_R^*)^{\downarrow \mathrm{fa}(d)}$.

4. **Contract:** Compute the contraction $c_{\mathrm{fa}(d)}$ of $\pi_{\mathrm{fa}(d)}^*$.

5. **Optimize:** For each $x_{\mathrm{pa}(d)}$, find a point x_d^* satisfying $x_d^* = \arg\max_{x_d} c_{\mathrm{fa}(d)}(x_d, x_{\mathrm{pa}(d)})$ and define $\tilde{\delta}_d(x_{\mathrm{pa}(d)})$ as the distribution degenerate at x_d^*. Add $\tilde{\delta}_d$ to the potential on R to get $\tilde{\pi}_R^*$.

Note that all the computations apart from the second step are local in the root clique R. Furthermore, after the above steps are carried out the joint potential on the junction tree is equal to

$$\tilde{\pi}_V^* = \tilde{\pi}_R^* \otimes (\otimes_{C:C\neq R}\pi_C) = \left(f_{\tilde{\delta}_d * q}, U\right),$$

where $\tilde{\delta}_d$ is a local maximum policy for q at d.

Steps 1–4 of this procedure is the analogue of the 'value determination' step of Howard's policy iteration and step 5 is the analogue of the 'policy improvement' step.

3.7 Partial collect propagation

The message passing algorithm presented in Section 3.5 can be used to update the policies during SPU: Suppose we begin with a potential π_V on the junction tree $\mathcal{T}$ and want to update the policies in the order $d_1, \ldots, d_k$. Assume the policy for d_i is assigned to clique R_i. Thus $R_i \supseteq \mathrm{fa}(d_i)$. As an initial step messages are collected towards R_1. Then we find a local maximum policy for d_1 and the obtained policy replaces the old policy for d_1 in R_1.

When the next policy needs to be updated we could apply the same algorithm; however each time a local maximum policy is to be computed we must repeat the algorithm using a new clique as root-clique. This usually involves a great deal of duplication. The propagation scheme presented below provides one way to eliminate much of this duplication. In this scheme messages are only passed from the previous root towards the new root.

We now explain the partial propagation scheme in detail: the messages are passed via a mailbox placed on each edge of the junction tree. If the edge connects C_1 and C_2, the mailbox can hold messages in the form of potentials on $C_1 \cap C_2$.

Updating the policy for d_1 is made exactly as in the procedure described above. As new root we choose the clique R_2. However, instead of collecting messages towards the new root as described above, we only pass messages along the (unique) path from the old root R_1 to R_2. This is done by first emptying the mailboxes on the path and then passing the messages. After the passage of these messages, R_2 has received messages from all its neighbours. Then a local maximum policy at d_2 can be computed and the potential on R_2 is changed appropriately. Next we choose R_3 as new root and pass messages on the path from R_2 to the new root in a similar manner.

Proceeding in this way we eventually have updated all the policies. When several cycles of SPU are to be performed we only need to collect messages to R_1 from the previous root R_k.

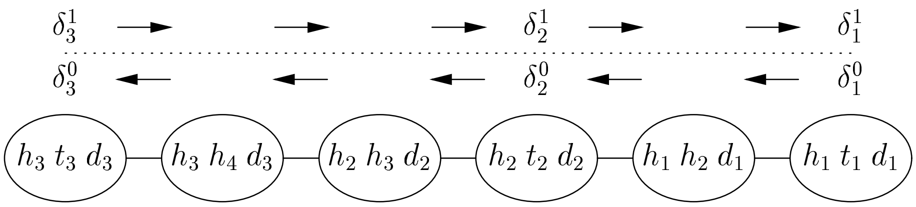

Figure 6: Flows of messages in PIGS during one cycle of SPU using partial collect propagation.

Example 4 (Improving strategies in PIGS**)** To illustrate the computational procedure we again consider PIGS, described in Example 1. We proceed by SPU and choose to update the policies in the order d_3, d_2, d_1. The initial policy can be chosen in a variety of ways. We choose the initial strategy as *uniform*, i.e. decisions to treat or not are made completely at random, independently of the test result. Alternatively, one may instead use an initial strategy chosen by the pig breeder based upon experience and common sense. Such strategies could for example be those of *never* or *always* treating. Or it could be the *direct* strategy, treating if and only if the test result is positive.

There are exactly four pure policies for each decision, denoted by *never*, *always*, *direct*, and *reverse*. The strategies just described use the same policy for every decision node. The reverse policy is the policy of treating if and only if the test result is negative. This policy is clearly unattractive, but mentioned for completeness.

Letting δ_i^l be short for $\delta_{d_i}^l$, the flow of messages in one cycle of SPU using partial collect propagation are illustrated in Figure 6. The corresponding policy modifications become:

1. The clique containing $\mathrm{fa}(d_3) = \{d_3, t_3\}$ is $\{h_3, t_3, d_3\}$. Collecting to this clique reveals that the expected utility of the uniform strategy is **644**. The local maximum policy at d_3 is $\delta_3^1 =$ *direct*, i.e. to treat if and only if the test result is positive. The utility of the improved strategy can be calculated locally in the same clique, yielding an expected utility of **667**.

Table 1: Expected utilities of pure strategies with $\delta_3 =$ *direct*.

	Policy for d_2			
Policy for d_1	*reverse*	*always*	*never*	*direct*
reverse	615	615	662	585
always	605	603	655	653
never	682	686	724	727
direct	673	674	716	718

2. To find the local maximum policy at d_2, we send messages to the clique $\{h_2, t_2, d_2\}$. The local maximum policy becomes $\delta_2^1 =$ *direct* with an expected utility of **690**.

3. The final step in the cycle is to send messages to $\{h_1, t_1, d_1\}$. The local maximum policy at d_1 becomes $\delta_1^1 =$ *never*. The expected utility has now increased to **727**.

4. Another cycle of SPU identifies the strategy as local maximum and the iteration stops.

The example described is sufficiently small for all pure LIMID strategies to be evaluated as there are only 64 such strategies. In this specific example, it turns out that $\delta_3 =$ *direct* is a local maximum policy for all these LIMID strategies so that only 16 pure strategies need to be evaluated. The result of this evaluation is displayed in Table 1. The table shows that the obtained LIMID strategy is indeed both local and global maximum.

With the story only covering 4 months, the ID version of PIGS can still be solved exactly. The optimal strategy has expected utility equal to **729** and is given as follows:

1. Do not treat in the first month: $\delta_1 =$ *never*.

Table 2: Expected utilities of selected strategies in PIGS.

always	*uniform*	*never*	*direct*	best LIMID	best ID
586	644	669	718	727	729

2. Treat in the second month if and only if test results are positive in the first month and in the second month.

3. Treat in the last month if and only if the test results are positive in the third month or in both of the first and second months.

Table 2 displays the expected utility of a number of interesting strategies. Note how close the best LIMID strategy is to the best ID strategy for the problem. □

4 Optimal strategies within LIMIDs

This section is concerned with developing methods for reducing LIMIDs to obtain lower complexity of computations during SPU and identifying conditions for LIMIDs which ensure that local maximum strategies obtained by SPU are also global maximum strategies.

4.1 Separation within LIMIDs

The key to simplification of computational problems for LIMIDs is the notion of *irrelevance* as expressed through d-separation (Pearl 1986), see Verma and Pearl (1990) for a formal treatment.

A *trail* τ (a sequence of nodes that are connected by arcs, not necessarily following the directions) from node a to node b in a DAG $\mathcal{D}$ is said to be *blocked* by S if it contains a node $n \in \tau$ such that either $n \in S$ and arcs of τ do not meet head-to-head at n, or n and all its descendants are not in S, and arcs of τ meet head-to-head at n. A trail that is not blocked by S is said to be *active.* Two subsets A and B of nodes are *d-separated* by S if all trails from A to B are blocked by S.

In the following we use the symbolic expression $A \perp_{\mathcal{L}} B \,|\, S$ to denote that A and B are d-separated by S in the DAG formed by all nodes of the LIMID $\mathcal{L}$, i.e. including the utility nodes.

Verma and Pearl (1990) show that d-separation satisfies the so-called *graphoid axioms:* For any disjoint subsets A, B, C, and D of nodes of $\mathcal{L}$ we have:

(C1) if $A \perp_{\mathcal{L}} B \,|\, C$ then $B \perp_{\mathcal{L}} A \,|\, C$;

(C2) if $A \perp_{\mathcal{L}} B \,|\, C$ and $D \subseteq B$, then $A \perp_{\mathcal{L}} D \,|\, C$;

(C3) if $A \perp_{\mathcal{L}} B \,|\, C$ and $D \subseteq B$, then $A \perp_{\mathcal{L}} (B \setminus D) \,|\, (C \cup D)$;

(C4) if $A \perp_{\mathcal{L}} B \,|\, C$ and $A \perp_{\mathcal{L}} D \,|\, (B \cup C)$, then $A \perp_{\mathcal{L}} (B \cup D) \,|\, C$;

(C5) if $A \perp_{\mathcal{L}} B \,|\, (C \cup D)$ and $A \perp_{\mathcal{L}} C \,|\, (B \cup D)$ then $A \perp_{\mathcal{L}} (B \cup C) \,|\, D$.

Indeed (C1)–(C4) also hold for subsets that are not necessarily disjoint whereas B and C must be disjoint for (C5) to hold.

Any distribution P which factorizes over a DAG satisfies the *global directed Markov property*

$$A \perp_{\mathcal{L}} B \,|\, S \Rightarrow A \perp\!\!\!\perp B \,|\, S, \tag{4}$$

where $\perp\!\!\!\perp$ denotes probabilistic conditional independence with respect to P. In particular this holds for the joint distribution f_q associated with a given strategy q for the LIMID $\mathcal{L}$.

4.2 Requisite information within LIMIDs

In this subsection we determine and exploit requisite information for decision problems represented by LIMIDs.

Definition 8 A node $a \in \mathrm{pa}(d)$ is said to be *non-requisite* for the decision node d in $\mathcal{L}$ if

$$(\Upsilon \cap \mathrm{de}(d)) \perp_{\mathcal{L}} a \mid (\mathrm{fa}(d) \setminus \{a\}). \tag{5}$$

We then also say that the arc from a to d is *non-requisite*. If (5) is not satisfied then a is said to be *requisite for* d and the arc from a to d is said to be *requisite*.

Intuitively the condition expresses that a has no influence on the utilities that d can affect, once the states of the remaining parents are known.

For the special case of IDs with a single utility node, our non-requisiteness coincides with the similar concept of Fagiuoli and Zaffalon (1998). The relation to those of Nielsen and Jensen (1999) and Shachter (1999) for more general IDs is more complex, see the end of Section 4.3.

If we introduce the *uniform policy* at d as the policy $\bar{\delta}_d(x_d \mid x_{\mathrm{pa}(d)}) = 1/|\mathcal{X}_d|$, we have the following alternative version of Lemma 1:

Lemma 5 *A policy $\tilde{\delta}_d$ is a local maximum policy for a strategy q at d if and only if for all $x_{\mathrm{fa}(d)}$ with $\tilde{\delta}_d(x_d \mid x_{\mathrm{pa}(d)}) > 0$ we have*

$$x_d = \arg\max_{z_d} \sum_{u \in \Upsilon} \sum_{x_{\mathrm{pa}(u)}} f_{\bar{\delta}_d * q}(x_{\mathrm{pa}(u)} \mid z_d, x_{\mathrm{pa}(d)}) U_u(x_{\mathrm{pa}(u)}). \tag{6}$$

Proof Lemma 1 states that $\tilde{\delta}_d$ is a local maximum policy for q if and only if for all $x_{\mathrm{fa}(d)}$ with $\tilde{\delta}_d(x_d \mid x_{\mathrm{pa}(d)}) > 0$ we have $x_d = \arg\max_{z_d} \bar{c}(x_{\mathrm{pa}(d)}, z_d)$, where $\bar{c}(x_{\mathrm{fa}(d)}) = \sum_{x_{V \setminus \mathrm{fa}(d)}} f_{q-d}(x) U(x)$. If we let $\bar{f} = f_{\bar{\delta}_d * q}$ then we clearly have $\bar{f}(x) \propto f_{q-d}(x)$ and thus

$$\begin{aligned}
\bar{c}(x_{\mathrm{fa}(d)}) &\propto \sum_{x_{V \setminus \mathrm{fa}(d)}} \bar{f}(x) U(x) = \sum_{x_{V \setminus \mathrm{fa}(d)}} \bar{f}(x) \sum_{u \in \Upsilon} U_u(x_{\mathrm{pa}(u)}) \\
&= \sum_{u \in \Upsilon} \sum_{x_{\mathrm{pa}(u)}} U_u(x_{\mathrm{pa}(u)}) \sum_{x_{V \setminus (\mathrm{fa}(d) \cup \mathrm{pa}(u))}} \bar{f}(x) \\
&= \sum_{u \in \Upsilon} \sum_{x_{\mathrm{pa}(u)}} U_u(x_{\mathrm{pa}(u)}) \bar{f}(x_{\mathrm{fa}(d)}) \bar{f}(x_{\mathrm{pa}(u)} \mid x_{\mathrm{fa}(d)}).
\end{aligned}$$

Because in fact $\bar{f}(x_{\mathrm{fa}(d)}) = 1/|\mathcal{X}_d| \; \bar{f}(x_{\mathrm{pa}(d)})$, so when $\bar{c}$ is to be maximized for fixed $x_{\mathrm{pa}(d)}$, $\bar{f}(x_{\mathrm{fa}(d)})$ is constant and can be ignored. The lemma follows. $\square$

The definition of non-requisite nodes and arcs is motivated by the following theorem, implying that the computations during SPU can be performed with reduced complexity.

Theorem 3 *If $a \in \mathrm{pa}(d)$ is non-requisite for decision node d in LIMID $\mathcal{L}$, then for all strategies q in $\mathcal{L}$ there is a local maximum policy for q at d which does not depend on a.*

Proof Suppose a is non-requisite for d in $\mathcal{L}$. Then for any utility node $u \in \mathrm{de}(d)$ it must hold that $\mathrm{pa}(u) \perp_{\mathcal{L}} a \mid (\mathrm{fa}(d) \setminus \{a\})$, since if otherwise, an active trail from a to $\mathrm{pa}(u)$ would induce an active trail from a to u, implying that a would be requisite. From the global directed Markov property (4) it thus follows that for all $u \in \mathrm{de}(d)$, where as before $\bar{f} = f_{\bar{\delta}_d * q}$, we have

$$\bar{f}(x_{\mathrm{pa}(u)} \mid z_d, x_{\mathrm{pa}(d)}) = \bar{f}(x_{\mathrm{pa}(u)} \mid z_d, x_{\mathrm{pa}(d)\setminus\{a\}}). \tag{7}$$

For $u \notin \mathrm{de}(d)$ we clearly have $d \perp_{\mathcal{L}} \mathrm{pa}(u) \mid \mathrm{pa}(d)$, implying that

$$\bar{f}(x_{\mathrm{pa}(u)} \mid z_d, x_{\mathrm{pa}(d)}) = \bar{f}(x_{\mathrm{pa}(u)} \mid x_{\mathrm{pa}(d)}). \tag{8}$$

If we now let $\Upsilon_1 = \Upsilon \cap \mathrm{de}(d)$ and $\Upsilon_2 = \Upsilon \setminus \mathrm{de}(d)$ and insert (7) and (8) into the expression to be maximized in Lemma 5, we get

$$\begin{aligned}&\sum_{u \in \Upsilon} \sum_{x_{\mathrm{pa}(u)}} \bar{f}(x_{\mathrm{pa}(u)} \mid z_d, x_{\mathrm{pa}(d)}) U_u(x_{\mathrm{pa}(u)}) \;= \\ &\sum_{u \in \Upsilon_1} \sum_{x_{\mathrm{pa}(u)}} \bar{f}(x_{\mathrm{pa}(u)} \mid z_d, x_{\mathrm{pa}(d)\setminus\{a\}}) U_u(x_{\mathrm{pa}(u)}) \;+\; \sum_{u \in \Upsilon_2} \sum_{x_{\mathrm{pa}(u)}} \bar{f}(x_{\mathrm{pa}(u)} \mid x_{\mathrm{pa}(d)}) U_u(x_{\mathrm{pa}(u)}).\end{aligned}$$

As the first of these sums does not depend on x_a, and the second does not depend on z_d, it follows that $\tilde{\delta}_d$ can be chosen not to depend on x_a. $\square$

4.3 Reduction of LIMIDs

The importance of the notion of non-requisite arcs is that these can be removed from a LIMID without loss of utility but with reduction of computational complexity. We define:

Definition 9 A LIMID $\mathcal{L}'$ is said to be a (stepwise) *reduction* of $\mathcal{L}$ if it is obtained by successive removals of non-requisite arcs.

Clearly, the reduction relation is transitive: If $\mathcal{L}_2$ is a reduction of $\mathcal{L}_1$, and $\mathcal{L}_3$ is a reduction of $\mathcal{L}_2$, then $\mathcal{L}_3$ is a reduction of $\mathcal{L}_1$. From Theorem 3 we then obtain the following corollary:

Corollary 2 *If $\mathcal{L}'$ is a reduction of $\mathcal{L}$, then any strategy q' which is local or global maximum for $\mathcal{L}'$, will also be so for $\mathcal{L}$.*

We will need that d-separation is preserved under arc-removal (and therefore under reduction):

Lemma 6 *If $\mathcal{L}'$ is obtained from $\mathcal{L}$ by removing arcs, then $A \perp_{\mathcal{L}} B \mid S \Rightarrow A \perp_{\mathcal{L}'} B \mid S$.*

Proof This holds because every trail in $\mathcal{L}'$ is a trail in $\mathcal{L}$ and if a trail is blocked by S in $\mathcal{L}'$ it is also blocked by S in $\mathcal{L}$. □

We further need to establish that non-requisiteness of arcs is preserved under reduction:

Lemma 7 *If $\mathcal{L}'$ is a reduction of $\mathcal{L}$ and an arc in $\mathcal{L}'$ from a to d is non-requisite in $\mathcal{L}$, it is also non-requisite in $\mathcal{L}'$.*

Proof It is sufficient to consider the case where $\mathcal{L}'$ has one arc less than $\mathcal{L}$. We assume that

$$(\Upsilon \cap \mathrm{de}(d)) \perp_{\mathcal{L}} a \mid (\mathrm{fa}(d) \setminus \{a\}) \tag{9}$$

and need to show that $(\Upsilon \cap \mathrm{de}'(d)) \perp_{\mathcal{L}'} a \mid (\mathrm{fa}'(d) \setminus \{a\})$, where $\mathrm{de}'(d)$ and $\mathrm{fa}'(d)$ refer to the reduced LIMID $\mathcal{L}'$.

If $\mathrm{fa}'(d) = \mathrm{fa}(d)$, the result follows from Lemma 6 and (C2) of the graphoid axioms because $\mathrm{de}'(d) \subseteq \mathrm{de}(d)$. Next consider the case with $\mathrm{fa}'(d) = \mathrm{fa}(d) \setminus \{a^*\}$. Since a^* is non-requisite for d in $\mathcal{L}$ we have $(\Upsilon \cap \mathrm{de}(d)) \perp_{\mathcal{L}} a^* \mid \mathrm{fa}'(d)$. Combining with (9) we now get from (C5) of the graphoid axioms that $(\Upsilon \cap \mathrm{de}'(d)) \perp_{\mathcal{L}} \{a, a^*\} \mid (\mathrm{fa}'(d) \setminus \{a\})$, since in this case $\mathrm{de}'(d) = \mathrm{de}(d)$. The conclusion now follows from (C2) and Lemma 6. $\square$

For any two LIMIDs $\mathcal{L}_1$ and $\mathcal{L}_2$ with identical node-set, we let $\mathcal{L}_1 \cap \mathcal{L}_2$ denote the LIMID with the same node-set and arc-set equal to the intersection of the arc-sets of $\mathcal{L}_1$ and $\mathcal{L}_2$. We then have

Lemma 8 *If $\mathcal{L}_1$ and $\mathcal{L}_2$ are reductions of $\mathcal{L}$, then $\mathcal{L}_1 \cap \mathcal{L}_2$ is a reduction of both $\mathcal{L}_1$ and $\mathcal{L}_2$.*

Proof Let $m_i, i = 1, 2$ be the number of non-requisite arcs removed from $\mathcal{L}$ to obtain $\mathcal{L}_i$. We will show the result by induction after $m = m_1 + m_2$.

For $m = 2$, the result follows directly from Lemma 7. Suppose the result holds for $m \leq k$, where $k \geq 2$ and consider the case $m = k + 1$. So $\max\{m_1, m_2\} > 1$, say $m_2 > 1$. Thus $\mathcal{L}_2$ is obtained by successively removing m_2 non-requisite arcs from $\mathcal{L}$. Let $\mathcal{L}_2'$ be the LIMID obtained by removing the first $m_2 - 1$ of these. By the induction assumption, $\mathcal{L}_1 \cap \mathcal{L}_2'$ is a reduction of $\mathcal{L}_2'$ obtained by removing at most m_1 non-requisite arcs from $\mathcal{L}_2'$. Furthermore, $\mathcal{L}_2$ is also a reduction of $\mathcal{L}_2'$ obtained by removing exactly one non-requisite arc. Since $(\mathcal{L}_1 \cap \mathcal{L}_2') \cap \mathcal{L}_2 = \mathcal{L}_1 \cap \mathcal{L}_2$ and $m_1 + 1 \leq k$, the induction assumption yields that $\mathcal{L}_1 \cap \mathcal{L}_2$ is a reduction of $\mathcal{L}_2$.

Similarly the induction assumption gives that $\mathcal{L}_1 \cap \mathcal{L}_2$ is a reduction of $\mathcal{L}_1 \cap \mathcal{L}_2'$ and also that $\mathcal{L}_1 \cap \mathcal{L}_2'$ is a reduction of $\mathcal{L}_1$. Transitivity of the reduction relation now yields that $\mathcal{L}_1 \cap \mathcal{L}_2$ is a reduction of $\mathcal{L}_1$ and the proof is complete. $\square$

Corollary 3 *If $\mathcal{L}_1$ and $\mathcal{L}_2$ are reductions of $\mathcal{L}$, then $\mathcal{L}_1 \cap \mathcal{L}_2$ is a reduction of $\mathcal{L}$.*

Proof Follows directly from Lemma 8 and transitivity of the reduction relation. □

It is clearly always advantageous to compute local or global maximum strategies in a LIMID which is reduced as much as possible. We define:

Definition 10 A LIMID $\mathcal{L}$ is *minimal* if all arcs in $\mathcal{L}$ are requisite.

We use the term *minimal reduction* for a reduction which is minimal. Corollary 3 now yields:

Theorem 4 *Any LIMID $\mathcal{L}$ has a unique minimal reduction.*

The minimal reduction of the LIMID $\mathcal{L}$ is denoted $\mathcal{L}_{\min}$. In Section 4.5 below we present a general algorithm for reducing an arbitrary LIMID $\mathcal{L}$ to $\mathcal{L}_{\min}$.

In the special case of an ID, the arcs removed from $\mathcal{L}$ to obtain $\mathcal{L}_{\min}$ are precisely those which originate from variables which are not *required* or *requisite* as defined by Nielsen and Jensen (1999) and Shachter (1999) respectively.

4.4 Extremal decision nodes and optimum policies

Sometimes a LIMID has a decision node d with a policy $\hat{\delta}_d$ which maximizes the expected utility among all policies for d whatever the other policies in $\mathcal{L}$ are. As we shall see, every extremal decision node, as defined below, has such a policy.

Definition 11 A decision node d is *extremal* in $\mathcal{L}$ if $(\Upsilon \cap \mathrm{de}(d)) \perp_{\mathcal{L}} \mathrm{fa}(\Delta \setminus \{d\}) \mid \mathrm{fa}(d)$.

The notion of extremal decision node is slightly different from what Zhang *et al*. (1994) term 'stepwise-decomposability candidate node' (SDCN). Any SDCN is extremal but not conversely.

Definition 12 An *optimum policy* for d in a LIMID $\mathcal{L}$ is a policy which is a local maximum policy at d for all strategies q in $\mathcal{L}$.

To establish the connection between optimum policies and extremal decision nodes we introduce the *uniform strategy* as the strategy where all policies are uniform. Then we have:

Theorem 5 *If decision node* d *is extremal in a LIMID* $\mathcal{L}$*, then it has an optimum policy. In this case, any local maximum policy at* d *for the uniform strategy is an optimum policy for* d *in* $\mathcal{L}$.

Proof Let q be an arbitrary strategy and let $\bar{\delta}_d$ denote the uniform policy for the extremal decision node d. We will show that whenever a policy $\tilde{\delta}_d$ is a local maximum policy for the uniform strategy $\bar{q}$, $\tilde{\delta}_d$ is also a local maximum policy for q.

First note that because $\mathrm{pa}(d)$ d-separates d from its non-descendants, we have for every utility node $u \notin \mathrm{de}(d)$ that

$$f_q(x_{\mathrm{pa}(u)} \,|\, x_d, x_{\mathrm{pa}(d)}) = f_q(x_{\mathrm{pa}(u)} \,|\, x_{\mathrm{pa}(d)}). \tag{10}$$

Next, note that for all x it holds that $f_{\bar{\delta}_d * q}(x \,|\, x_{\mathrm{fa}(\Delta)}) = f_{\bar{q}}(x \,|\, x_{\mathrm{fa}(\Delta)})$ whenever $f_{\bar{\delta}_d * q}(x_{\mathrm{fa}(\Delta)}) > 0$. Thus, if we let $Z = \mathrm{fa}(\Delta) \setminus \mathrm{fa}(d)$ we get for all utility nodes u and $x_{\mathrm{fa}(d)}$ with $f_{\bar{\delta}_d * q}(x_{\mathrm{fa}(d)}) > 0$

$$\begin{aligned} f_{\bar{\delta}_d * q}(x_{\mathrm{pa}(u)} \,|\, x_{\mathrm{fa}(d)}) &= \sum_{x_Z} f_{\bar{\delta}_d * q}(x_{\mathrm{pa}(u)} \,|\, x_{\mathrm{fa}(\Delta)}) f_{\bar{\delta}_d * q}(x_Z \,|\, x_{\mathrm{fa}(d)}) \\ &= \sum_{x_Z} f_{\bar{q}}(x_{\mathrm{pa}(u)} \mid x_{\mathrm{fa}(\Delta)}) f_{\bar{\delta}_d * q}(x_Z \,|\, x_{\mathrm{fa}(d)}). \end{aligned} \tag{11}$$

Because d is extremal, $u \perp_{\mathcal{L}} Z \mid \mathrm{fa}(d)$ for every utility node $u \in \mathrm{de}(d)$, so it must hold that $\mathrm{pa}(u) \perp_{\mathcal{L}} Z \mid \mathrm{fa}(d)$ since otherwise an active trail from Z to $\mathrm{pa}(u)$ would induce an active trail from Z to u. So (11) becomes

$$f_{\bar{\delta}_d * q}(x_{\mathrm{pa}(u)} \,|\, x_{\mathrm{fa}(d)}) = f_{\bar{q}}(x_{\mathrm{pa}(u)} \,|\, x_{\mathrm{fa}(d)}) \sum_{x_Z} f_{\bar{\delta}_d * q}(x_Z \,|\, x_{\mathrm{fa}(d)}) = f_{\bar{q}}(x_{\mathrm{pa}(u)} \,|\, x_{\mathrm{fa}(d)}). \tag{12}$$

As in the proof of Theorem 3 we now let $\Upsilon_1 = \Upsilon \cap \mathrm{de}(d)$, and $\Upsilon_2 = \Upsilon \setminus \mathrm{de}(d)$, insert (10) and (12) into the expression to be maximized in Lemma 5, and get

$$\sum_{u \in \Upsilon} \sum_{x_{\mathrm{pa}(u)}} f_{\bar{\delta}_d * q}(x_{\mathrm{pa}(u)} \,|\, z_d, x_{\mathrm{pa}(d)}) U_u(x_{\mathrm{pa}(u)}) \;=$$

$$\sum_{u\in\Upsilon_1}\sum_{x_{\mathrm{pa}(u)}} f_{\bar{q}}(x_{\mathrm{pa}(u)} \mid z_d, x_{\mathrm{pa}(d)})U_u(x_{\mathrm{pa}(u)}) \ + \sum_{u\in\Upsilon_2}\sum_{x_{\mathrm{pa}(u)}} f_{\bar{\delta}_d * q}(x_{\mathrm{pa}(u)} \mid x_{\mathrm{pa}(d)})U_u(x_{\mathrm{pa}(u)}).$$

As the terms in the second sum do not depend on z_d, the joint expression is maximized if and only if the first sum is maximized, showing that a policy which is local maximum for $\bar{q}$ is local maximum for any q. This completes the proof. □

The conclusion in Theorem 5 still holds if the uniform strategy is replaced by any strategy with every policy giving positive probability to all decision alternatives. With deterministic policies in q, an extremal decision node d may have a local maximum policy which is not an optimum policy. This renders Proposition 3.3 and Theorem 4.1 of Zhang *et al*. (1994) inaccurate.

A special instance of the result in Theorem 5 is an important element in the method described by Cooper (1988) for solving IDs with a single utility node.

4.5 Soluble LIMIDs

Suppose d_0 is extremal in the LIMID $\mathcal{L}$. Theorem 5 ensures that we can find an optimum policy for d_0 by SPU, starting from the uniform strategy $\bar{q}$. When this has been done, we can implement this policy $\tilde{\delta}_{d_0}$ and convert $\mathcal{L}$ to $\mathcal{L}^*$, where d_0 now becomes a chance node in $\mathcal{L}^*$ with $\tilde{\delta}_{d_0}$ as the associated conditional probability distribution. Clearly, every local or global maximum strategy q^* for $\mathcal{L}^*$ then generates a local or global maximum strategy for $\mathcal{L}$ as $q = q^* \cup \{\tilde{\delta}_{d_0}\}$. If $\mathcal{L}^*$ again has an extremal decision node, we can yet again find an optimum policy and convert $\mathcal{L}^*$ as above. If the process can continue until all decision nodes have become chance nodes, we have obtained a global maximum strategy for $\mathcal{L}$. We thus define:

Definition 13 An *exact solution ordering* $d_1, \ldots, d_k$ of the decision nodes in $\mathcal{L}$ is an ordering with the property that for all i, d_i is extremal in $\mathcal{L}$ when $d_{i+1}, \ldots, d_k$ are converted into chance nodes.

Note that if $d_1, \ldots, d_k$ is an exact solution ordering then optimum policies for the decisions are identified in the reverse order, i.e. first d_k, then d_{k-1} and so on.

Definition 14 A LIMID $\mathcal{L}$ is said to be *soluble* if it admits an exact solution ordering.

Solubility is similar to 'stepwise decomposability' in Zhang *et al*. (1994) but stronger, as a LIMID can be soluble without being stepwise decomposable, see the remark after Definition 11.

Even if a LIMID $\mathcal{L}$ is not soluble, its minimal reduction $\mathcal{L}_{\min}$ may be. To show that the converse cannot happen, we need to establish that extremality is preserved under reduction:

Lemma 9 *If $\mathcal{L}'$ is a reduction of $\mathcal{L}$ and d^* is extremal in $\mathcal{L}$, then d^* is extremal in $\mathcal{L}'$.*

Proof Suppose d^* is extremal in $\mathcal{L}$. It suffices to consider the case where $\mathcal{L}'$ is obtained by removing a single arc from a non-requisite parent a into a decision node d. We must show that $(\Upsilon \cap \mathrm{de}'(d^*)) \perp_{\mathcal{L}'} \mathrm{fa}'(\Delta \setminus \{d^*\}) \mid \mathrm{fa}'(d^*)$, where de' and fa' refer to $\mathcal{L}'$.

We consider two cases. If $d \neq d^*$, the result follows directly from Lemma 6 and (C2) of the graphoid axioms as $\mathrm{fa}'(d^*) = \mathrm{fa}(d^*)$, $\Upsilon \cap \mathrm{de}'(d^*) \subseteq \Upsilon \cap \mathrm{de}(d^*)$, and $\mathrm{fa}'(\Delta \setminus \{d\}) \subseteq \mathrm{fa}(\Delta \setminus \{d\})$.

If $d = d^*$, then because d^* is extremal in $\mathcal{L}$, $(\Upsilon \cap \mathrm{de}(d^*)) \perp_{\mathcal{L}} \mathrm{fa}(\Delta \setminus \{d^*\}) \mid \mathrm{fa}(d^*)$ and because a is non-requisite for d^*, $(\Upsilon \cap \mathrm{de}(d^*)) \perp_{\mathcal{L}} a \mid (\mathrm{fa}(d^*) \setminus \{a\})$. Thus (C4) of the graphoid axioms implies that $(\Upsilon \cap \mathrm{de}(d^*)) \perp_{\mathcal{L}} (\{a\} \cup \mathrm{fa}(\Delta \setminus \{d^*\})) \mid (\mathrm{fa}(d^*) \setminus \{a\})$. By Lemma 6 this d-separation also holds in $\mathcal{L}'$ and since $\mathrm{de}'(d^*) = \mathrm{de}(d^*)$, $\mathrm{fa}'(d^*) = \mathrm{fa}(d^*) \setminus \{a\}$, and $\mathrm{fa}'(\Delta \setminus \{d^*\}) = \mathrm{fa}(\Delta \setminus \{d^*\})$, the desired conclusion follows from (C2). □

Next we show that solubility is preserved under reduction:

Theorem 6 *If $\mathcal{L}'$ is a reduction of a soluble LIMID $\mathcal{L}$, then $\mathcal{L}'$ is soluble.*

Proof Suppose $\mathcal{L}$ is soluble with exact solution ordering $d_1, \dots, d_k$. Let $\mathcal{L}'$ be a reduction of $\mathcal{L}$ obtained by removing a non-requisite arc into some decision node d_j. We want to show that $\mathcal{L}'$ is soluble with exact solution ordering $d_1, \dots, d_k$, i.e. we will show that d_i is extremal in $\mathcal{L}'$ where $d_{i+1}, \dots, d_k$ are converted into chance nodes. We again consider two cases. If $i \geq j$, the result follows from Lemma 9. If $i < j$, the result follows from Lemma 6 and the fact that $\Upsilon \cap \mathrm{de}'(d_i) \subseteq \Upsilon \cap \mathrm{de}(d_i)$, where as usual $\mathrm{de}'(d_i)$ refers to $\mathcal{L}'$. This completes the proof. □

As a consequence, if $\mathcal{L}$ is soluble, so is its minimal reduction $\mathcal{L}_{\min}$.

Requisiteness of arcs is not generally preserved under reduction, but for arcs into extremal decision nodes it is:

Lemma 10 *If d_0 is extremal in $\mathcal{L}$, $\mathcal{L}'$ is a reduction of $\mathcal{L}$, and a is a requisite parent for d_0 in $\mathcal{L}$, then a is also requisite for d_0 in $\mathcal{L}'$.*

Proof It is sufficient to consider the case where $\mathcal{L}'$ is obtained by removing an arc into d^* from some non-requisite parent a^*. We verify the statement by contraposition: We suppose that d_0 is extremal in $\mathcal{L}$ and a is non-requisite for d_0 in $\mathcal{L}'$, and show that a must also be non-requisite in $\mathcal{L}$.

Since d_0 is extremal in $\mathcal{L}$, we have $\Upsilon \cap \mathrm{de}(d_0) = \Upsilon \cap \mathrm{de}'(d_0)$, so we consider a utility node $u \in \mathrm{de}(d_0)$. If $d_0 \neq d^*$, we have $\mathrm{fa}'(d_0) = \mathrm{fa}(d_0)$ and non-requisiteness of a yields

$$a \perp_{\mathcal{L}'} u \mid (\mathrm{fa}(d_0) \setminus \{a\}). \tag{13}$$

Because d_0 is extremal in $\mathcal{L}$ and d-separation is preserved under arc-removal (Lemma 6), we have $\{a^*, d^*\} \perp_{\mathcal{L}'} u \mid \mathrm{fa}(d_0)$. Using (C4) on this and (13) yields $\{a, a^*, d^*\} \perp_{\mathcal{L}'} u \mid (\mathrm{fa}(d_0) \setminus \{a\})$. This d-separation also holds if we insert an arc from a^* into d^*, i.e. $\{a, a^*, d^*\} \perp_{\mathcal{L}} u \mid (\mathrm{fa}(d_0) \setminus \{a\})$, whereby (C2) yields $a \perp_{\mathcal{L}} u \mid (\mathrm{fa}(d_0) \setminus \{a\})$, so a is non-requisite in $\mathcal{L}$.

Next we consider the case where $d_0 = d^*$. Here we have $\mathrm{fa}'(d_0) = \mathrm{fa}(d_0) \setminus \{a^*\}$ and thus

$$a \perp_{\mathcal{L}'} u \mid (\mathrm{fa}(d_0) \setminus \{a, a^*\}). \tag{14}$$

As a^* is non-requisite in $\mathcal{L}$ and d-separation is preserved under arc-removal (Lemma 6), we have $a^* \perp_{\mathcal{L}'} u \mid (\mathrm{fa}(d_0) \setminus \{a^*\})$ and thus $\{a^*, d_0\} \perp_{\mathcal{L}'} u \mid (\mathrm{fa}(d_0) \setminus \{a^*\})$. Using (C4) on this and (14) gives $\{a, a^*, d_0\} \perp_{\mathcal{L}'} u \mid (\mathrm{fa}(d_0) \setminus \{a, a^*\})$. Again, this d-separation also holds in $\mathcal{L}$. Using (C3) gives $\{a, d_0\} \perp_{\mathcal{L}} u \mid (\mathrm{fa}(d_0) \setminus \{a\})$ and (C2) yields that a is non-requisite in $\mathcal{L}$, as desired. □

The following algorithm thus finds the minimal reduction $\mathcal{L}_{\min}$ of a LIMID $\mathcal{L}$ with decision nodes $\Delta \neq \emptyset$ and identifies whether $\mathcal{L}_{\min}$ is soluble:

1. $D := \Delta$;
2. **Search** for an extremal decision node $d \in D$; **if** none exists, **go to** 3; **else**
   (a) **remove** arcs from non-requisite parents of d; **convert** d into a chance node, and let $D := D \setminus \{d\}$;
   (b) if $D = \emptyset$, **return** '$\mathcal{L}_{\min}$ is soluble' and **go to** 5; else **go to** 2.
3. **Visit** each $d \in D$ in any order, and **remove** arcs from non-requisite parents of d.
4. If no arcs are removed in 3, **return** '$\mathcal{L}_{\min}$ is not soluble' and **go to** 5; else **go to** 2.
5. **Return** $\mathcal{L}_{\min}$ by **re-converting** all nodes in $\Delta \setminus D$ into decision nodes.

Example 5 (A soluble modification of PIGS**)** As illustration we use the LIMID $\mathcal{L}^*$ in Figure 7 (a) which is a modification of PIGS obtained by adding arcs into each decision d_i from the previous states $(h_{i-1}, t_{i-1}, d_{i-1})$. The original LIMID $\mathcal{L}$ in Figure 2 is not soluble but we will show that

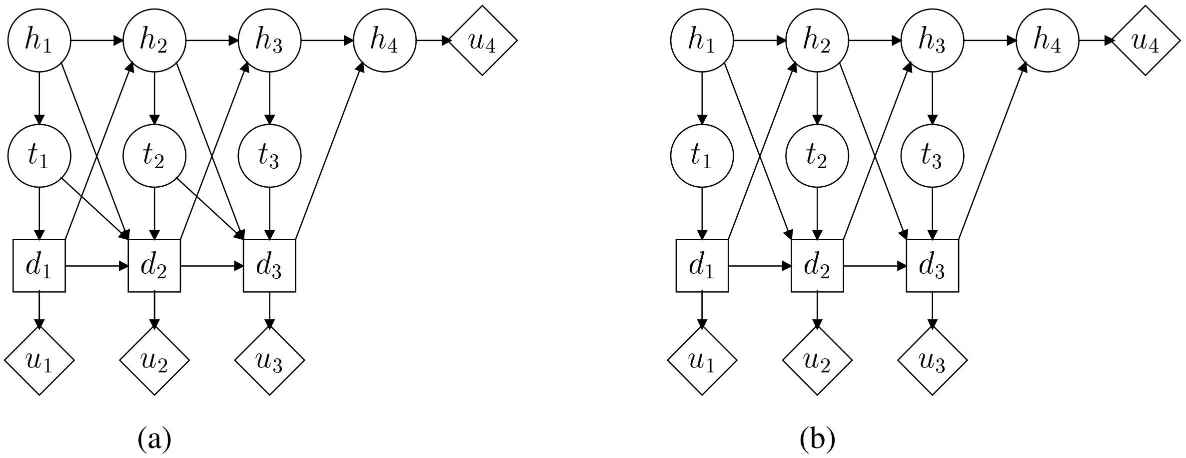

Figure 7: (a) A soluble modification $\mathcal{L}^*$ of PIGS with exact solution ordering d_1, d_2, d_3. (b) The minimal reduction $\mathcal{L}^*_{\min}$ of $\mathcal{L}^*$.

$\mathcal{L}^*_{\min}$ is soluble with exact solution ordering d_1, d_2, d_3. Although $\mathcal{L}^*$ is not a realistic model since the h_i's are unobservable, this soluble modification can be used to give an upper bound on the expected utility of the global maximum strategy of $\mathcal{L}$, since the soluble modification uses more information and therefore must have a strategy that performs at least as well. This can then in turn be used to assess the strategy found by SPU on $\mathcal{L}$.

To reduce $\mathcal{L}^*$ we apply the above algorithm, and start by searching for an extremal decision node in $D = \{d_1, d_2, d_3\}$. Clearly d_3 is the only extremal decision node: u_3 and u_4 are the only utility nodes that are descendants of d_3, and these are d-separated from $\mathrm{fa}(\{d_1, d_2\}) = \{d_1, d_2, t_1, t_2, h_1\}$ by $\mathrm{fa}(d_3) = \{d_2, d_3, t_2, t_3, h_2\}$. Next we remove non-requisite arcs into d_3: the arc from t_2 into d_3 is the only non-requisite arc and is thus removed. Then d_3 is converted into a chance node, and we let $D = \{d_1, d_2\}$.

After this conversion, we search in D for an extremal decision node: d_2 is extremal, and since the arc from t_1 into d_2 is the only non-requisite arc into d_2, it is removed. As above, d_2 is converted into a chance node, and we let $D = \{d_1\}$.

The resulting LIMID has only one decision node d_1, which trivially is extremal. There is only a single arc into d_1 and since it is requisite it cannot be removed. Now we let $D = \emptyset$, and thus the obtained minimal reduction $\mathcal{L}^*_{\min}$, shown in Figure 7 (b), is soluble. Note that $\mathcal{L}^*_{\min}$ has exact solution ordering d_1, d_2, d_3, and it follows that SPU achieves a global maximum strategy if it updates the policies, starting from the uniform strategy, in the order d_3, d_2, and finally d_1.

The global maximum strategy in $\mathcal{L}^*_{\min}$ has an expected utility of **732**, compared to an expected utility of **727** of the local maximum strategy in the original LIMID $\mathcal{L}$ found by SPU. Thus a decision maker will know that the global maximum strategy of $\mathcal{L}$ can at most increase the expected utility by **5** compared to the local maximum strategy found by SPU.

In this case the LIMID $\mathcal{L}^*$ was itself soluble. However, if the arc from t_1 into d_2 is replaced by an arc from t_1 into d_3, the corresponding LIMID is not soluble but its minimal reduction is. The reader may easily check that the above algorithm will indeed detect this. $\square$

We conclude by mentioning that all IDs are soluble. Hence a combination of the reduction algorithm and SPU yields an efficient algorithm for solving IDs (Nilsson and Lauritzen 2000).

Acknowledgements

This research was supported by DINA (Danish Informatics Network in the Agricultural Sciences), funded by the Danish Research Councils through their PIFT programme. The authors are indebted to Erik Jørgensen, Foulum, for initiating the research and adapting the pig breeding example to have some resemblance with reality.

A significant part of this article was written while the authors were visiting the Fields Institute for Research in Mathematical Sciences and we are grateful for financial support and the excellent working conditions provided.

We are grateful to the Editor James E. Smith and two anonymous referees for valuable comments and references leading to a much improved presentation.

References

Cooper, G. F. (1988). A method for using belief networks as influence diagrams. In *Proceedings of the 4th workshop on uncertainty in artificial intelligence*, pp. 55–63. Minneapolis, MN.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

Fagiuoli, E. and Zaffalon, M. (1998). A note about redundancy in influence diagrams. *International Journal of Approximate Reasoning*, **19**, 351–65.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Massachusetts.

Howard, R. A. and Matheson, J. E. (1984). Influence diagrams. In *Readings in the principles and applications of decision analysis*, (ed. R. A. Howard and J. E. Matheson). Strategic Decisions Group, Menlo Park, CA.

Jensen, F., Jensen, F. V., and Dittmer, S. L. (1994). From influence diagrams to junction trees. In *Proceedings of the 10th conference on uncertainty in artificial intelligence*, (ed. R. L. de Mantaras and D. Poole), pp. 367–73. Morgan Kaufmann Publishers, San Francisco, CA.

Kjærulff, U. (1992). Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, **2**, 7–17.

Lovejoy, W. S. (1991). A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, **28**, 47–66.

Madsen, A. L. and Jensen, F. V. (1998). Lazy propagation in junction trees. In *Proceedings of the 14th annual conference on uncertainty in artificial intelligence*, (ed. G. F. Cooper and S. Moral), pp. 362–9. Morgan Kaufmann Publishers, San Francisco, CA.

Nielsen, T. D. and Jensen, F. V. (1999). Welldefined decision scenarios. In *Proceedings of the 15th annual conference on uncertainty in artificial intelligence*, (ed. K. Laskey and H. Prade), pp. 502–11. Morgan Kaufmann Publishers, San Francisco, CA.

Nilsson, D. and Lauritzen, S. L. (2000). Evaluating influence diagrams using LIMIDs. In *Proceedings of the 16th conference on uncertainty and artificial intelligence*, (ed. C. Boutilier and M. Goldszmidt), pp. 436–45. Morgan Kaufmann Publishers, San Francisco, CA.

Olmsted, S. M. (1983). *On Representing and Solving Decision Problems*. Ph.D. Thesis, Department of Engineering–Economic Systems, Stanford University, Stanford, CA.

Pearl, J. (1986). A constraint–propagation approach to probabilistic reasoning. In *Uncertainty in artificial intelligence*, (ed. L. N. Kanal and J. F. Lemmer), pp. 357–70. North-Holland, Amsterdam, The Netherlands.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, **34**, 871–82.

Shachter, R. D. (1999). Efficient value of information computation. In *Proceedings of the 15th annual conference on uncertainty in artificial intelligence*, (ed. K. Laskey and H. Prade), pp. 594–601. Morgan Kaufmann Publishers, San Francisco, CA.

Shenoy, P. P. (1992). Valuation-based systems for Bayesian decision analysis. *Operations Research*, **40**, 463–84.

Shenoy, P. P. and Shafer, G. (1990). Axioms for probability and belief–function propagation. In *Uncertainty in artificial intelligence 4*, (ed. R. D. Shachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer), pp. 169–98. North-Holland, Amsterdam, The Netherlands.

Verma, T. and Pearl, J. (1990). Causal networks: Semantics and expressiveness. In *Uncertainty in artificial intelligence 4*, (ed. R. D. Shachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer), pp. 69–76. North-Holland, Amsterdam, The Netherlands.

White, C. C. (1991). A survey of solution techniques for the partially observed Markov decision process. *Annals of Operations Research*, **32**, 215–30.

Zhang, N. L. (1998). Probabilistic inference in influence diagrams. *Computational Intelligence*, **14**, 475–97.

Zhang, N. L., Qi, R., and Poole, D. (1994). A computational theory of decision networks. *International Journal of Approximate Reasoning*, **11**, 83–158.