On SPI-Lazy Evaluation of Influence Diagrams

Rafael Cabañas[1], Andrés Cano[1], Manuel Gómez-Olmedo[1], and Anders L. Madsen[2,3]

[1] Department of Computer Science and Artificial Intelligence, CITIC, University of Granada, Spain
{rcabanas,acu,mgomez}@decsai.ugr.es
[2] HUGIN EXPERT A/S
Aalborg, Denmark
anders@hugin.com
[3] Department of Computer Science, Aalborg University, Denmark

Abstract. Influence Diagrams are an effective modelling framework for analysis of Bayesian decision making under uncertainty. Improving the performance of the evaluation is an element of crucial importance as real-world decision problems are more and more complex. Lazy Evaluation is an algorithm used to evaluate Influence Diagrams based on message passing in a strong junction tree. This paper proposes the use of Symbolic Probabilistic Inference as an alternative to Variable Elimination for computing the clique-to-clique messages in Lazy Evaluation of Influence Diagrams.

Keywords: Influence Diagrams, Combinatorial Factorization Problem, Exact Evaluation, Heuristic Algorithm, Lazy Evaluation, Junction Tree.

1 Introduction

Influence Diagrams (IDs) [1] are a tool to represent and evaluate decision problems under uncertainty. A technique used to evaluate IDs is Lazy Evaluation (LE) [2,3]. Its basic idea is to maintain a decomposition of the potentials and postpone computations for as long as possible. Thus it is possible to exploit barren variables and independence induced by evidence.

LE is based on message passing in a strong junction tree, which is a representation of a decision problem represented as an ID. Computing the messages involves the removal of variables. In the original proposal, the method used is *Variable Elimination* (VE) [3]. An alternative method for removing a set of variables from a set of potentials is *Symbolic Probabilistic Inference* algorithm (SPI) [4,5,6], which considers the removal as a combinatorial factorization problem. That is, SPI tries to find the optimal order for the combinations and marginalizations (i.e. max-marginalization and sum-marginalization). In a previous paper [7], the basic version of the SPI algorithm was described for the direct evaluation of IDs. This algorithm was also proposed as an alternative for computing clique-to-clique messages in LE of Bayesian Networks (BNs) [8]. Our contribution is to describe how the SPI algorithm can be used for computing the messages in LE of IDs. This new method for evaluating IDs is called *SPI-Lazy Evaluation (SPI-LE)*. The differences between BNs and IDs must be considered: two kinds of potentials, the temporal order between decisions, etc. The experimental work shows how SPI can improve the efficiency of LE. In the experimental work we use a set of IDs present in the literature.

The paper is organized as follows: Section 2 introduces basic concepts about IDs, LE and the motivation of this work; Section 3 describes how SPI can be used for computing the messages in LE of IDs; Section 4 includes the experimental work and results; finally Section 5 details our conclusions and lines for future work.

2 Preliminaries

2.1 Influence Diagrams

An ID [1] is a Probabilistic Graphical Model for decision analysis under uncertainty which contains three kinds of nodes: *decision nodes* (squares) that correspond with the actions which the decision maker can control; *chance nodes* (circles) representing random variables; and *utility nodes* (diamonds) representing the decision maker preferences. Fig. 1 shows an example of an ID.

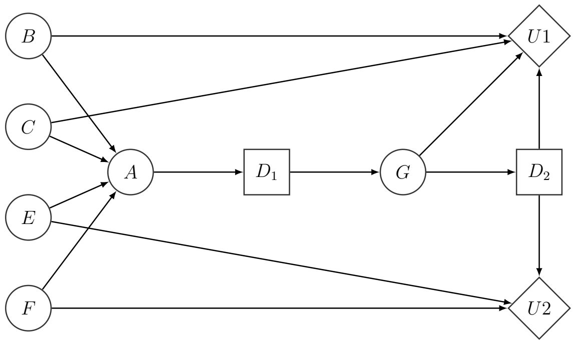

Fig. 1. An example of an ID with the partial order is $\{A\} \prec D_1 \prec \{G\} \prec D_2 \prec \{B, C, E, F\}$

We denote by $\mathcal{U}_C$ the set of chance nodes, by $\mathcal{U}_D$ the set of decision nodes, and by $\mathcal{U}_V$ the set of utility nodes. The decision nodes have a temporal order, $D_1, \ldots, D_n$, and the chance nodes are partitioned into a collection of disjoint sets according to when they are observed: $\mathcal{I}_0$ is the set of chance nodes observed before D_1, and $\mathcal{I}_i$ is the set of chance nodes observed after decision D_i is taken and before decision D_{i+1} is taken. Finally, $\mathcal{I}_n$ is the set of chance nodes observed after D_n. That is, there is a partial order: $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$.

In the description of an ID, it is more convenient to think in terms of predecessors: the parents of a chance node X_i, denoted $pa(X_i)$, are also called *conditional predecessors*. The parents of a utility node V_i, denoted $pa(V_i)$, are also called conditional predecessors. Similarly, the parents of a decision D_i are called *informational predecessors* and are denoted $pa(D_i)$. Informational predecessors of each decision D_i, must include previous decisions and their informational predecessors (*no-forgetting assumption*).

The *universe* of the ID is $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D = \{X_1, \dots, X_m\}$. Let us suppose that each variable X_i is discrete and it takes values on a finite set $\Omega_{X_i} = \{x_1, \dots, x_{|\Omega_{X_i}|}\}$. Each chance node X_i has a conditional probability distribution $P(X_i|pa(X_i))$ associated. In the same way, each utility node V_i has a utility function $U(pa(V_i))$ associated. In general, we will talk about potentials (not necessarily normalized). The set of all variables involved in a potential ϕ is denoted $dom(\phi)$, defined on $\Omega_{dom(\phi)} = \times\{\Omega_{X_i}|X_i \in dom(\phi)\}$. The elements of $\Omega_{dom(\phi)}$ are called configurations of ϕ. Therefore, a *probability potential* denoted by ϕ is a mapping $\phi : \Omega_{dom(\phi)} \rightarrow [0, 1]$. A *utility potential* denoted by ψ is a mapping $\psi : \Omega_{dom(\psi)} \rightarrow \mathbb{R}$. The set of probability potentials is denoted by Φ while the set of utility potentials is denoted by Ψ.

An arc between an informational predecessor and a decision is redundant if it is d-separated from the utility nodes given the rest of informational predecessors. Any redundant arc can be removed [9]. If no-forgetting arcs have been added and redundant arcs have been removed, the parents of a decision compose its *relevant past*. A chance or decision node is a *barren node* if it is a sink, in other words, it has no children or only barren descendants. Any barren node can be directly removed from the ID.

The goal of evaluating an ID is to obtain an *optimal policy* δ_i for each decision D_i, that is a function of a subset of its informational predecessors. The optimal policy maximizes the *expected utility* for the decision. A strategy is an ordered set of policies $\Delta = \{\delta_1, \dots, \delta_n\}$, including a policy for each decision variable. An optimal strategy $\widehat{\Delta}$ returns the optimal choice the decision maker should take for each decision.

Optimal policy: *Let ID be an influence diagram over the universe $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$ and let $\mathcal{U}_V$ be the set of utility nodes. Let the temporal order of the variables be described as $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$. Then, an optimal policy for D_i is*

$$\delta_{D_i}(\mathcal{I}_0, D_1, \dots, \mathcal{I}_{i-1}) =$$
$$= \arg\max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} \prod_{X \in \mathcal{U}_C} P(X|pa(X)) \left(\sum_{V \in \mathcal{U}_V} U(pa(V)) \right) \qquad (1)$$

2.2 Lazy Evaluation

Lazy Evaluation (LE) was already used for making inference in BNs [10], so it can be adapted for evaluating IDs [2,3]. The basic idea of this method is to maintain the decomposition of the potentials for as long as possible and to postpone computations for as long as possible, as well as to exploit barren variables. LE is based on message passing in a *strong junction tree*, which is a representation of an ID built by moralization and by triangulating the graph using a strong elimination order [11].

Nodes in the strong junction trees correspond to *cliques* (maximal complete subgraphs) of the triangulated graph. Each clique is denoted by C_i where i is the index of the clique. The root of the strong junction tree is denoted by C_1. Two neighbour cliques are connected by a separator which contains the intersection of the variables in both cliques. The size of a clique C_i, denoted $|C_i|$, is the number of variables. The weight of a clique C_i, denoted $w(C_i)$, can be defined as $\prod_{X \in C_i} |\Omega_X|$. An example of a strong junction tree is shown in Fig. 2.

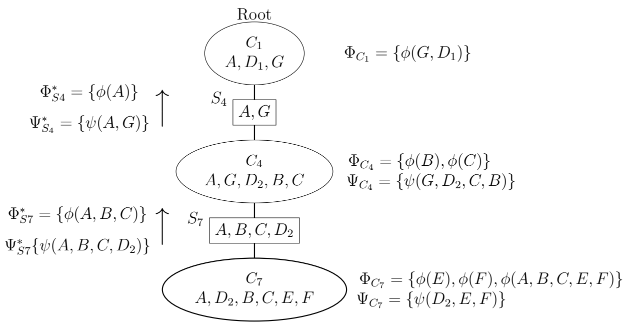
Root
C_1 A, D_1, G
$\Phi_{C_1} = \{\phi(G, D_1)\}$
$\Phi^*_{S4} = \{\phi(A)\}$
$\Psi^*_{S_4} = \{\psi(A, G)\}$
S_4 A, G
C_4 A, G, D_2, B, C
$\Phi_{C_4} = \{\phi(B), \phi(C)\}$
$\Psi_{C_4} = \{\psi(G, D_2, C, B)\}$
$\Phi^*_{S7} = \{\phi(A, B, C)\}$
$\Psi^*_{S7}\{\psi(A, B, C, D_2)\}$
S_7 A, B, C, D_2
C_7 A, D_2, B, C, E, F
$\Phi_{C_7} = \{\phi(E), \phi(F), \phi(A, B, C, E, F)\}$
$\Psi_{C_7} = \{\psi(D_2, E, F)\}$

Fig. 2. Strong junction tree for the ID shown in Fig. 1 with the potentials associated to each clique (right) and messages stored at each separator (left)

Propagation is performed by message-passing. Initially, each potential is associated to the closest clique to the root containing all its variables. These potentials are not combined, so during propagation each clique and separator keeps two sets of potentials (one for probabilities and another for utilities). Sets of potentials stored in a clique C_j are denoted Φ_{C_j} and Ψ_{C_j}. Similarly, sets of potentials (or messages) stored in a separator S_j are denoted $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$. Message propagation starts by invoking the *Collect Message* algorithm in the root (Algorithm 1).

Algorithm 1 (Collect Message) *Let C_j be a clique where Collect Message is invoked, then:*

1. *C_j invokes Collect Message in all its children.*
2. *The message to the clique parent of C_j is built and sent by absorption (Algorithm 2).*

A clique can send the message to its parent (*Absorption*) if it has received all the messages from its children. Consider a clique C_j and its parent separator S_j.

Absorption in C_j amounts to eliminating the variables of $C_j \backslash S_j$ from the list of probability and utility potentials associated with C_j and the separators of $ch(C_j)$ and then associating the obtained potentials with S_j. The original proposal [3] uses VE for removing the variables. Thus we will refer to this method as VE-Lazy Evaluation (VE-LE)

Algorithm 2 (Absorption) *Let C_j be a clique, S_j be the parent separator and $S' \in ch(C_j)$ be the child separators. If Absorption is invoked on C_j, then:*

1. *Let $\mathcal{R}_{S_j} = \Phi_{C_j} \cup \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} (\Phi^*_{S'} \cup \Psi^*_{S'})$.*
2. *Let $\mathbf{X} = \{X | X \in C_j, X \notin S_j\}$ the variables to be removed.*
3. *Choose an order to remove the variables in $\mathbf{X}$.*
4. *Marginalize out all variables in $\mathbf{X}$ from $\mathcal{R}_{S_j}$. Let $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$ be the set of probability and utility potentials obtained.*
5. *Associate $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$ to the parent separator S_j.*

The propagation finishes when the root clique has received all the messages. The utility potential from which each variable D_i is eliminated during the evaluation should be recorded as the *expected utility* for the decision D_i. The values of the decision that maximizes the expected utility is the policy for D_i. In case of decisions that are attached to the root node, the expected utility and policy is calculated by marginalizing out all variables in the root clique that do not belong to the relevant past of the decision.

2.3 Motivation

When Absorption is invoked on a clique, all variables in the clique not present in the parent separator are marginalized out from the set of relevant potentials. The original definition of LE proposes using *Variable Elimination* (VE) [3]. This algorithm chooses at each step a variable to remove based on any criteria or heuristic. This removal involves combining all potentials containing the chosen variable. Let us consider the strong junction tree shown in Fig. 2. If Absorption is invoked on C_7, variables E and F are marginalized out in order to compute the messages to S_7, denoted $\Phi^*_{S_7}$ and $\Psi^*_{S_7}$. The computations performed using VE are:

$$\Phi^*_{S_7} = \left\{ \sum_E P(E) \sum_F P(F) P(A|B,C,E,F) \right\} \tag{2}$$

$$\Psi^*_{S_7} = \left\{ \frac{\sum_E P(E) P(A|B,C,E) \frac{\sum_F P(F)P(A|B,C,E,F)U(D_2,E,F)}{P(A|B,C,E)}}{P(A|B,C)} \right\} \tag{3}$$

Assuming that all the variables are binary, the computation of $\Phi^*_{S_7}$ and $\Psi^*_{S_7}$ requires 144 multiplications and 72 additions and 48 divisions. Independently of the elimination ordering used to remove E and F, VE will always have to combine the marginal potentials with a large potential such as $P(A|B,C,E,F)$. However, with a re-order of the operations this situation can be avoided:

$$\Phi^*_{S_7} = \left\{ \sum_E \sum_F \left(P(A|B,C,E,F) \left(P(E)P(F) \right) \right) \right\} \tag{4}$$

$$\Psi^*_{S_7} = \left\{ \frac{\sum_E \sum_F \left(P(A|B,C,E,F) \left(P(E)P(F) \right) U(D_2,E,F) \right)}{P(A|B,C)} \right\} \tag{5}$$

Using Eq. (4) and (5) the computation of the messages requires 100 multiplications, 72 additions and 16 divisions. In some cases it could be better to combine small potentials even if they do not share any variable (e.g., $P(E)$ and $P(F)$). This combination will never be performed using VE since it is guided by the elimination ordering. Thus the efficiency of the message computation can be improved if an optimal ordering for the operations of marginalization and combination is found [5]. To overcome this, we propose for computing the clique-to-clique messages the SPI algorithm , which is more flexible than VE.

3 SPI Lazy Evaluation

3.1 Overview

SPI Lazy Evaluation (SPI-LE) uses SPI instead of VE in order to compute the messages. Thus the process for building the strong junction tree and *Collect Message* algorithm are the same. The general scheme of the *Absorption* algorithm is slightly different (see Algorithm 3). The main difference is that the set of variables to remove is partitioned into disjoint subsets of chance variables or single sets of decisions. Then, the removal of these subsets of variables is invoked in an order that respects the the temporal constraints. Notice that the removal of a subset of variables $\mathbf{X}_k$ is invoked only on the set of potentials containing any variable in $\mathbf{X}_k$.

Algorithm 3 (Absorption SPI) *Let C_j be a clique, S_j be the parent separator and $S' \in ch(C_j)$ be the child separators. If Absorption is invoked on C_j, then:*

1. *Set the relevant potential sets:*
   $\Phi^*_{S_j} := \Phi_{C_j} \cup \bigcup_{S' \in ch(C_j)} \Phi^*_{S'}$ $\Psi^*_{S_j} := \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} \Psi^*_{S'}$
2. *Let $\mathbf{X} := \{X | X \in C_j, X \notin S_j\}$ the variables to remove.*
3. *Partition $\mathbf{X}$ into disjoint subsets of chance variables or single sets of decisions. Determine a partial order of the subsets that respects the temporal constraints: $\{\mathbf{X}_1 \prec \mathbf{X}_2 \prec \cdots \prec \mathbf{X}_n\}$*
4. *For $k := n$ to 1 do:*
   (a) *Set $\Phi_{\mathbf{X}_k} := \{\phi \in \Phi^*_S | \mathbf{X}_k \cap dom(\phi) \neq \emptyset\}$ and $\Psi_{\mathbf{X}_k} := \{\psi \in \Psi^*_S | \mathbf{X}_k \cap dom(\psi) \neq \emptyset\}$.*

   (b) *Remove $\mathbf{X}_k$ from $\Phi_{\mathbf{X}_k}$ and $\Psi_{\mathbf{X}_k}$. If $\mathbf{X}_k$ is a subset of chance variables use Algorithm 4, otherwise use Algorithm 6. The inputs to these algorithms are $\mathbf{X}_k$, $\Phi_{\mathbf{X}_k}$ and $\Psi_{\mathbf{X}_k}$ while $\Phi^*_{\mathbf{X}_k}$ and $\Psi^*_{\mathbf{X}_k}$ are the sets of potentials obtained.*
   (c) *Update the relevant potential sets:*

$$\Phi^*_{S_j} := (\Phi^*_{S_j} \backslash \Phi_{\mathbf{X}_k}) \cup \Phi^*_{\mathbf{X}_k} \quad \Psi^*_{S_j} := (\Psi^*_{S_j} \backslash \Psi_{\mathbf{X}_k}) \cup \Psi^*_{\mathbf{X}_k}$$

5. *Associate $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$ to the parent separator S_j.*

3.2 Removal of Chance Variables

In order to remove a subset of chance variables $\mathbf{X}$ from $\Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$, SPI considers probability and utility potentials separately: first, SPI tries to find the best order for combining all potentials in $\Phi_{\mathbf{X}}$. For that purpose, all possible pairwise combinations between the probability potentials are stored in the set combination candidate set B. Besides, B also contains those probability potentials that contain any variable of $\mathbf{X}$ which is not present in any other potential of $\Phi_{\mathbf{X}}$, that is a variable that can be directly removed (without performing any combination). At each iteration, an element of B is selected. If this element is a pair, both potentials are combined. The procedure stops when all variables have been removed. A variable can be removed in the moment it only appears in a single probability potential. Notice that this algorithm produces a factorization of potentials. This procedure is shown in Algorithm 4.

Algorithm 4 (Removal of a subset of chance variables) *Let $\mathbf{X}$ be a set of chance variables, $\Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$ be sets of probability and utility potentials relevant for removing $\mathbf{X}$. If the removal of $\mathbf{X}$ is invoked on $\Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$, then:*

1. *Initialize the combination candidate set $B := \emptyset$.*
2. *Repeat:*
   (a) *Add all pairwise combinations of elements of $\Phi_{\mathbf{X}}$ to B which are not already in B.*
   (b) *Add to B all potentials in $\Phi_{\mathbf{X}}$ which are not already in B and that contain any variable of $\mathbf{X}$ which is not present in any other potential of $\Phi_{\mathbf{X}}$, that is a variable that can be removed.*
   (c) *Select a pair $p := \{\phi_i, \phi_j\}$ or a singleton $p := \{\phi_i\}$ from B according to some heuristic.*
   (d) *If p is a pair, then $\phi_{ij} := \phi_i \otimes \phi_j$. Otherwise, $\phi_{ij} := \phi_i$*
   (e) *Determine the set $\mathbf{W}$ of variables that can be sum-marginalized:*

$$\mathbf{W} := \{W \in dom(\phi_{ij}) \cap \mathbf{X} | \forall \phi \in \Phi_{\mathbf{X}} \backslash p : W \notin dom(\phi)\}$$

   (f) *Select the utility potentials relevant for removing $\mathbf{W}$:*

$$\Psi_{\mathbf{W}} := \{\psi \in \Psi_{\mathbf{X}} | \mathbf{W} \cap dom(\psi) \neq \emptyset\}$$

```
    (g) If $\mathbf{W} \neq \emptyset$, sum-marginalize variables in $\mathbf{W}$ from $\phi_{ij}$ and $\Psi_{\mathbf{W}}$. A probability potential $\phi_{ij}^{\downarrow\mathbf{W}}$ and a set of utility potentials $\Psi^{\downarrow\mathbf{W}}$ are obtained as a result (Algorithm 5).
    (h) Update:
        – $\mathbf{X} := \mathbf{X} \backslash \mathbf{W}$
        – If $p$ is a pair, $\Phi_{\mathbf{X}} := \Phi_{\mathbf{X}} \backslash \{\phi_i, \phi_j\}$ and remove any element in $B$ containing $\phi_i$ or $\phi_j$. Otherwise, $\Phi_{\mathbf{X}} := \Phi_{\mathbf{X}} \backslash \{\phi_i\}$ and remove any element in $B$ containing $\phi_i$.
        – $\Phi_{\mathbf{X}} := \Phi_{\mathbf{X}} \cup \{\phi_{ij}^{\downarrow\mathbf{W}}\}$   $\Psi_{\mathbf{X}} := (\Psi_{\mathbf{X}} \backslash \Psi^{\mathbf{W}}) \cup \Psi^{\downarrow\mathbf{W}}$
  Until $\mathbf{X} = \emptyset$:
3. Return $\Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$.
```

In Algorithm 4 only probability potentials are combined while utility potentials are not. The utility potentials must be combined with ϕ_{ij} which is the resulting potential of combining all potentials containing X. For that reason, the utilities can only be combined when a variable can be removed. That is the moment when ϕ_{ij} has been calculated. The procedure for sum-marginalizing a set of variables (Algorithm 5) involves finding good order for summing the utility potentials. The procedure for that is quite similar to the procedure for combining probabilities, the main difference is that in the moment a variable can be removed, the probability and utility potentials resulting of the marginalization are computed. Notice that this procedure is invoked on $\Psi_{\mathbf{W}} \subseteq \Psi_{\mathbf{X}}$.

Algorithm 5 (Sum-marginalization) *Let ϕ be a probability potential and $\Psi_{\mathbf{W}}$ a set of utility potentials relevant for removing the chance variables in $\mathbf{W}$. Then, the procedure for sum-marginalizating $\mathbf{W}$ from ϕ and $\Psi_{\mathbf{W}}$ is:*

```
1. Initialize the combination candidate set $B' := \emptyset$.
2. if $\Psi_{\mathbf{W}} = \emptyset$, then return $\sum_{\mathbf{W}} \phi$
3. Repeat:
    (a) Add all pairwise combinations of elements of $\Psi_{\mathbf{W}}$ to $B'$ which are not already in $B'$.
    (b) Add to $B'$ all potentials in $\Psi_{\mathbf{W}}$ which are not already in $B'$ that contains any variable of $\mathbf{W}$ which is not present in any other potential of $\Psi_{\mathbf{W}}$, that is a variable that can be removed.
    (c) Select a pair $q := \{\psi_i, \psi_j\}$ or a singleton $q := \{\psi_i\}$ from $B'$ according to some heuristic.
    (d) If $q$ is a pair, then $\psi_{ij} := \psi_i + \psi_j$. Otherwise, $\psi_{ij} := \psi_i$
    (e) Determine the set $\mathbf{V}$ of variables that can be sum-marginalized:

            $$\mathbf{V} := \{V \in dom(\psi_{ij}) \cap \mathbf{W} | \forall \psi \in \Psi_{\mathbf{W}} \backslash q : V \notin dom(\psi)\}$$

    (f) If, $\mathbf{V} \neq \emptyset$, sum-marginalize $\mathbf{V}$, giving as a result:

            $$\phi^{\downarrow\mathbf{V}} := \sum_{\mathbf{V}} \phi \qquad \psi^{\downarrow\mathbf{V}} := \sum_{\mathbf{V}} (\phi \otimes \psi_{ij}) / \phi^{\downarrow\mathbf{V}}$$
```

```
  (g) Update:
      – $\mathbf{W} := \mathbf{W} \backslash \mathbf{V}$
      – If $q$ is a pair, $\Psi_{\mathbf{W}} := \Psi_{\mathbf{W}} \backslash \{\psi_i, \psi_j\}$ and remove any element in $B'$
        containing $\psi_i$ or $\psi_j$. Otherwise, $\Psi_{\mathbf{W}} := \Psi_{\mathbf{W}} \backslash \{\psi_i\}$ and remove any
        element in $B'$ containing $\psi_i$.
      – $\phi := \phi^{\downarrow \mathbf{V}}$ and $\Psi_{\mathbf{W}} := \Psi_{\mathbf{W}} \cup \{\psi^{\downarrow \mathbf{V}}\}$
  Until $\mathbf{W} = \emptyset$
4. Return $\phi$ and $\Psi^{\mathbf{W}}$.
```

3.3 Removal of Decision Variables

The removal of a decision variable does not imply the combination of any probability potential since any decision is d-separated from its predecessors [12] and any successor has already been removed (the removal order of the disjoint subsets of variables must respect the temporal constraints). Thus, any probability potential $\phi(D_k, \mathbf{X})$ must be directly transform into $\phi(\mathbf{X})$ if D_k is a decision and $\mathbf{X}$ is a set of chance variables that belong to $\mathcal{I}_i$ with $i < k$. This property is used at step 2 of Algorithm 6.

Algorithm 6 (Removal of a decision) *Let D be a decision variable, Φ_D and Ψ_D be sets of probability and utility potentials relevant for removing D. If the removal of D is invoked on Φ_D and Ψ_D, then:*

```
1. For each $\phi \in \Phi_D$, remove $D$ by restricting $\phi$ to any of the values of $D$. The
   set of potentials $\Phi^{\downarrow D}$ is given as a result.
2. Max-marginalize variable $D$ from $\Psi_D$. A new potential $\psi^{\downarrow D}$ is obtained as a
   result (Algorithm 7).
3. Return $\Phi^{\downarrow D}$ and $\psi^{\downarrow D}$
```

Algorithm 7 shows the procedure for finding the best order for summing all utility potentials containing a decision D. Notice that the pairwise candidate set does not contain singletons and the sum-marginalization is performed once all utility potentials have been summed.

Algorithm 7 (Max-marginalization) *Let D be a decision variable and Ψ_D be a set of utility potentials containing D. Then, the procedure for max-marginalizating D from Ψ_D is:*

```
1. Initialize the combination candidate set $B' := \emptyset$.
2. While $|\Psi_D| > 1$:
   (a) Add all pairwise combinations of elements of $\Psi_D$ to $B'$ which are not
       already in $B'$.
   (b) Select a pair $q := \{\psi_i, \psi_j\}$ according to some heuristic and sum both
       potentials giving as a result $\psi_{ij}$.
   (c) Update:
```

- *Delete all pairs p of B' where $\psi_i \in p$ or $\psi_j \in p$.*
- $\Psi_D := \Psi_D \backslash \{\psi_i, \psi_j\} \cup \{\psi_{ij}\}$.

3. *Let ψ^D be the single potential in Ψ.*
4. *Max-marginalize D, giving as a result $\psi^{\downarrow D} := \max_D \psi^D$ and record the policy for D.*
5. *Return $\psi^{\downarrow D}$.*

3.4 Heuristics

During the removal of the chance variables, at each iteration a pair of probability potentials is selected to be combined (Definition 4, step 2.c). For that, any heuristic used with VE can be adapted for selecting a pair. Let $p := \{\phi_i, \phi_j\}$ be a candidate pair to be combined, let $\phi_{ij} = \phi_i \otimes \phi_j$ be the resulting potential of the combination. Then the heuristic *minimum size* [13] will select a pair minimizing Eq. (6). Thus *minimum size* heuristic chooses a pair that minimizes the number of variables in the resulting potential. This heuristic can also be used for selecting a pair of utility potentials at steps 3.c and 2.b of Definitions 5 and 7 respectively.

$$min\_size(p) = |dom(\phi_i) \cup dom(\phi_j)| = |dom(\phi_{ij})| \tag{6}$$

Let $\mathbf{W}$ be the set of variables that can be removed after combining potentials in p. Then the algorithm should check if any variable in $\mathbf{W}$ is a *probabilistic barren*, that is a barren node if only the set of probability potentials are considered. The removal of a probabilistic barren from a probability potential leads to an unity-potential. During the calculation of messages unity-potentials are not calculated and if the denominator of a division is a unity-potential, then the division is no not performed.

3.5 Example

To illustrate the computations of the messages using SPI-LE, let us consider the strong junction tree shown in Fig. 2 representing the ID in Fig. 1 with binary variables. To simplify the notation, $\phi(X_1, \ldots, X_n)$ will be denoted $\phi_{X_1,\ldots,X_n}$.

Initially, *Collect Message* (Algorithm 1) is invoked on the root clique C_1 and recursively invoked on its children. Once *Collect Message* is invoked on the leaf clique C_7, the messages for the parent separator S_7 are computed (Algorithm 3). The relevant potentials are:

$$\Phi_S^* := \{\phi_E, \phi_F, \phi_{ABCEF}\} \qquad \Psi_S^* := \{\psi_{D_2EF}\}$$

Variables $\{E, F\}$ must be removed for computing the messages. Both of them belong to $\mathcal{I}_2$, so they can be removed in any order. Then, the removal of $\{E, F\}$ is invoked on $\{\phi_E, \phi_F, \phi_{ABCEF}\} \cup \{\psi_{D_2EF}\}$ (Algorithm 4). The initial combination candidate set B is:

$$B := \{\{\phi_E, \phi_F\}, \{\phi_E, \phi_{ABCEF}\}, \{\phi_F, \phi_{ABCEF}\}\}$$

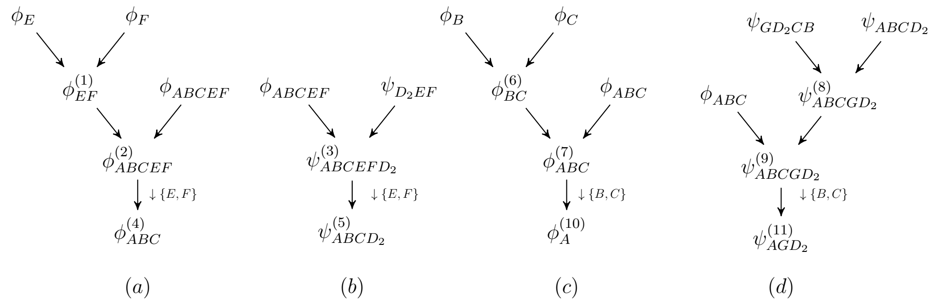

Fig. 3. Combination order of the probability and utility potentials obtained using SPI for removing chance variables when computing the messages sent from C_7 (a and b) and from C_4 (c and d) to their respective parent separators in the strong junction tree shown in Fig.2.

Notice that the set B does not contain any singleton because none of the variables appear only in one potential. If *minimum size* is the heuristic used, the element of B chosen is the pair $\{\phi_E, \phi_F\}$. Both potentials are combined giving as a result a new potential ϕ_{EF}. None of the variables can be marginalized as they are also contained in ϕ_{ABCEF}. In the second iteration, the combination candidate set is updated as follows:

$$B := \{\{\phi_{EF}, \phi_{ABCEF}\}\}$$

Now, the single pair of B is chosen and both potentials are combined giving as a result a new potential ϕ_{ABCEF} from which variables E and F can be sum-marginalized out. Using Algorithm 5 these variables are removed from the resulting probability potential and also from the utility potential ψ_{D_2EF}. For that, a combination candidate set B' with the relevant utility potentials is generated:

$$B' := \{\{\psi_{D_2EF}\}\}$$

Since B' contains only one element which is a singleton, the algorithm performs directly removal of variables E and F from ϕ_{ABCEF} and ψ_{D_2EF} (Algorithm 5 step 3.f). The resulting potentials are ϕ_{ABC} and ψ_{ABCD_2} which are, in this case, the messages for the parent separator.

The whole process for computing the messages from C_7 is shown in Fig. 3.a and 3.b in two factor graphs [14]. Nodes without any parent correspond to initial potentials while child nodes correspond to the resulting potentials of a combination. The numbers above each potentials indicate the combination ordering and arcs labels indicate the variables that are sum-marginalized.

When absorption is invoked on C_4, a similar procedure is followed for removing variables B, C and D_2. Now, these variables are partitioned into the disjoint subsets $\{\{D_2\} \prec \{B, C\}\}$. First, variables B and C are removed using Algorithm 4 giving as a result the potentials ϕ_A and ψ_{AGD_2}. The process for this removal is shown in Fig. 3.c and 3.d. The removal of D_2 is now performed using Algorithm 6 and it is almost directly since it only involves the utility potential ψ_{AGD_2}. Messages computed to the parent separator are ϕ_A and ψ_{AG}.

4 Experimental Work

4.1 Procedure

For testing purposes, a set of 10 IDs from the literature are used: NHL is a real world IDs used for medical purposes [15]; an ID used to evaluate the population population viability of wildlife species [16]; the oil wildcatter's problem[17]; the Chest Clinic ID [18] obtained from the Asia BN; an ID representing the decision problem in the poker game [12]; an ID used at agriculture for treating mildew [12]; an ID to model a simplified version of the dice game called *Think-box*[1]; finally, three synthetic IDs are used: the motivation example shown in Fig. 1 with binary variables and the ID used by Jensen et al. in [2]. The details of these IDs are shown in Table 1, which contains the number of nodes of each kind.

Table 1. Features of the IDs used for the experimental work

	Number of nodes		
ID	Chance	Decisions	Utility
NHL	17	3	1
Wildlife	9	1	1
Oil Wildcatter	2	2	2
ChestClinic	8	2	2
Poker	7	1	1
Mildew	7	2	2
Motivation ID	6	2	2
Jensen et al.	12	4	4
Thinkbox	5	2	4
Car buyer	3	3	1

From each ID a strong junction tree is built using the *minimum size* heuristic [13] for triangulating the graph. Table 2 shows, for each tree, the number of cliques, the minimum and maximum clique size $|C|$ and the minimum and maximum clique weight $w(C)$.

The message passing is performed using SPI-LE and VE-LE. In the first case, the heuristic used for selecting the next pair to combine is *minimum size* as described in Section 3.4. For the VE-LE the order used for removing the variables is the same than the one used during the triangulation. It must be noticed that in both cases a similar heuristic is used in order to obtained comparable results. For each evaluation, the number of operations involved is measured. The ratio

[1] `http://www.hugin.com/technology/samples/think-box`

Table 2. Features of the strong junction trees used for the experimental work obtained with *minimum size* heuristic

		$\lvert C\rvert$		$w(C)$	
	Cliques	min	max	min	max
NHL	8	5	12	32	$5.530\cdot10^5$
Wildlife	7	3	4	8	16
Oil Wildcatter	1	4	4	36	36
ChestClinic	5	3	6	8	64
Poker	5	3	3	32	324
Mildew	4	4	6	256	9408
Motivation ID	3	3	6	8	64
Jensen et al.	9	3	5	8	32
Thinkbox	2	4	6	8	384
Car buyer	1	6	6	384	384

between the number of operations using both methods can be computed using Eq. (7). A value lower than 1 means a better performance of SPI-LE.

$$ratio = \frac{SPI\text{-}LE \text{ operations}}{VE\text{-}LE \text{ operations}} \tag{7}$$

In order to check that SPI-LE does not have a large overhead, the CPU time needed for the evaluation of each ID is also measured. All the algorithms are implemented in Java with the Elvira Software[2]. The tests are run on a Intel Core i7-2600 (8 cores, 3.4 GHz). Each evaluation is repeated 10 times to avoid the effects of outliers. To analyze the reduction in the evaluation time, the *speed-up* can be computed using Eq. (8). In this case, a value higher than 1 means a better performance of SPI-LE.

$$speed\text{-}up = \frac{VE\text{-}LE \text{ CPU time}}{SPI\text{-}LE \text{ CPU time}} \tag{8}$$

4.2 Results

Table 3 shows the total number of operations needed for each evaluation and the ratio of the number of operations using SPI-LE to the number of operations using VE-LE. It can be observed that in most of the cases the number of operations required using SPI-LE is lower than using VE-LE. The ID representing the wildcatter's problem requires the same number of operations with both algorithms. The reason for that is that this ID is too simple to obtain any gain. By contrast the higher reduction (lower ratio) is obtained when evaluating the NHL ID, which is the largest ID used in the experimentation.

Table 4 shows the number of operations of each kind used for the evaluation using each method. That is the number of multiplications, divisions, additions

[2] `http://leo.ugr.es/~elvira`

Table 3. Total number of operations used for evaluating each ID using SPI-LE and VE-LE

ID	SPI-LE	VE-LE	ratio
NHL	**$2.848 \cdot 10^6$**	$4.660 \cdot 10^6$	0.611
Wildlife	**163**	172	0.948
Oil Wildcatter	121	121	1
ChestClinic	**534**	576	0.927
Poker	**1781**	1916	0.93
Mildew	**$3.133 \cdot 10^4$**	$3.218 \cdot 10^4$	0.974
Motivation ID	**320**	428	0.748
Jensen et al.	**270**	280	0.964
Thinkbox	**1872**	2308	0.811
Car buyer	**1518**	1535	0.989

and max-comparisons. It can be observed that the gain is mainly obtained due to the reduction in the number of multiplications. The number of multiplications required using SPI-LE is always lower or equal than using VE-LE. By contrast, in some cases, the rest of operations is higher if SPI-LE is used.

Table 4. Number of each kind operation used for evaluating each ID using SPI-LE and VE-LE

	Multiplications		Divisions		Additions		Max comparisons	
ID	SPI-LE	VE-LE	SPI-LE	VE-LE	SPI-LE	VE-LE	SPI-LE	VE-LE
NHL	**$1.566 \cdot 10^6$**	$2.674 \cdot 10^6$	**$1.248 \cdot 10^4$**	$5.794 \cdot 10^5$	**$1.258 \cdot 10^6$**	$1.386 \cdot 10^6$	**$1.104 \cdot 10^4$**	$2.120 \cdot 10^4$
Wildlife	**104**	110	8	8	**50**	52	**1**	2
Oil Wildcatter	60	60	12	12	42	42	7	7
ChestClinic	**268**	280	**80**	96	**168**	180	**18**	20
Poker	**970**	1186	18	**0**	792	**729**	1	1
Mildew	**$1.552 \cdot 10^4$**	$1.626 \cdot 10^4$	**1408**	1472	$1.346 \cdot 10^4$	$1.346 \cdot 10^4$	**944**	992
Motivation ID	**152**	212	**24**	72	138	138	6	6
Jensen et al.	**152**	156	16	16	**92**	96	**10**	12
Thinkbox	**592**	624	**224**	256	**936**	1296	**120**	132
Car buyer	**736**	864	224	**192**	440	**364**	118	**115**

The reduction in the number of operations needed for evaluating an ID should lead more efficient algorithms. Table 5 shows the CPU time needed for evaluating each ID and the speed-up obtained using SPI-LE with respect to VE-LE. For all the IDs, the CPU time is lower if SPI-LE is used instead for VE-LE. The speed-up obtained when evaluating NHL ID is not really high though there is a great difference in the number of operations. The reason for that is that SPI-LE is likely to have a larger overhead than VE-LE, specially with large ID where the algorithm considers a large number of pairwise combinations. However, the

Table 5. CPU time in milliseconds and speed-up needed for evaluating each ID using SPI-LE and VE-LE

ID	SPI-LE	VE-LE	speed-up
NHL	**7944.2**	$1.461 \cdot 10^4$	1.839
Wildlife	**167.7**	188	1.121
Oil Wildcatter	**78.6**	89.1	1.134
ChestClinic	**42.7**	204.3	4.785
Poker	**16.1**	187.6	11.652
Mildew	**81.2**	204.7	2.521
Motivation ID	**63.5**	159.2	2.507
Jensen et al.	**54.3**	311.2	5.731
Thinkbox	**50.3**	131.8	2.62
Car buyer	**63.6**	159.8	2.513

gain obtained with the reduction in the number of operations compensates this large overhead.

5 Conclusions and Future Work

In the present paper we have described how the SPI algorithm can be used for computing clique-to-clique messages in LE of IDs. This algorithm considers the removal of a set of variables as a combinatorial factorization problem. Thus SPI is more fine-grained than VE as it considers the order in which potentials are combined. Detailed methods for computing the messages using SPI-LE are given.

The experimentation have shown that using SPI for computing the messages is more efficient than using VE. The number of operations, particularly the number of multiplications, is reduced for most of the IDs. Even though SPI-LE usually have a larger overhead than VE-LE, experiments have shown that the CPU time needed for the evaluation is reduced as well: the reduction in the number of operations compensates this larger overhead. The experimentation has been performed using one of the heuristics *minimum size*. It could be interesting looking for alternative heuristics since the efficiency of the evaluation depends on this heuristic.

The SPI algorithm used only considers next pair of potentials to combine. Thus another line of future research could be studying the behaviour of the algorithm using a higher neighbourhood degree.

Acknowledgments. This research was supported by the Spanish Ministry of Economy and Competitiveness under project TIN2010-20900-C04-01, the European Regional Development Fund (FEDER), the FPI scholarship programme (BES-2011-050604). The authors have also been partially supported by “Junta de Andalucía” under projects TIC-06016 and P08-TIC-03717.

References

1. Howard, R., Matheson, J.: Influence diagram retrospective. Decision Analysis 2(3), 144–147 (2005)
2. Jensen, F., Jensen, F.V., Dittmer, S.L.: From Influence Diagrams to junction trees. In: Proceedings of the 10th International Conference on Uncertainty in AI, pp. 367–373. Morgan Kaufmann Publishers Inc. (1994)
3. Madsen, A., Jensen, F.: Lazy evaluation of symmetric Bayesian decision problems. In: Proceedings of the 15th Conference on Uncertainty in AI, pp. 382–390. Morgan Kaufmann Publishers Inc. (1999)
4. Shachter, R.D., D'Ambrosio, B., Del Favero, B.: Symbolic probabilistic inference in belief networks. In: AAAI, vol. 90, pp. 126–131 (1990)
5. Li, Z., d'Ambrosio, B.: Efficient inference in Bayes networks as a combinatorial optimization problem. IJAR 11(1), 55–81 (1994)
6. D'Ambrosio, B., Burgess, S.: Some experiments with real-time decision algorithms. In: Proceedings of the 12th International Conference on Uncertainty in AI, pp. 194–202. Morgan Kaufmann Publishers Inc. (1996)
7. Cabañas, R., Madsen, A.L., Cano, A., Gómez-Olmedo, M.: On SPI for evaluating influence diagrams. In: Laurent, A., Strauss, O., Bouchon-Meunier, B., Yager, R.R. (eds.) IPMU 2014, Part I. CCIS, vol. 442, pp. 506–516. Springer, Heidelberg (2014)
8. Madsen, A.: Variations over the message computation algorithm of lazy propagation. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36(3), 636–648 (2005)
9. Fagiuoli, E., Zaffalon, M.: A note about redundancy in influence diagrams. International Journal of Approximate Reasoning 19(3), 351–365 (1998)
10. Madsen, A., Jensen, F.: Lazy propagation: a junction tree inference algorithm based on lazy evaluation. Artificial Intelligence 113(1-2), 203–245 (2004)
11. Kjærulff, U.: Triangulation of graphs – algorithms giving small total state space. Research Report R-90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark (1990)
12. Jensen, F., Nielsen, T.: Bayesian networks and decision graphs. Springer (2007)
13. Rose, D.: A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. Graph Theory and Computing 183, 217 (1972)
14. Bloemeke, M., Valtorta, M.: A hybrid algorithm to compute marginal and joint beliefs in Bayesian networks and its complexity. In: Proceedings of the 14th Conference on Uncertainty in AI, pp. 16–23. Morgan Kaufmann Publishers Inc. (1998)
15. Lucas, P., Taal, B.: Computer-based decision support in the management of primary gastric non-Hodgkin lymphoma. UU-CS (1998-33) (1998)
16. Marcot, B., Holthausen, R., Raphael, M., Rowland, M., Wisdom, M.: Using Bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement. Forest Ecology and Management 153(1), 29–42 (2001)
17. Raiffa, H.: Decision analysis: Introductory lectures on choices under uncertainty (1968)
18. Goutis, C.: A graphical method for solving a decision analysis problem. IEEE Transactions on Systems, Man and Cybernetics 25(8), 1181–1193 (1995)