International Journal of Approximate Reasoning 70 (2016) 13–35

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Improvements to Variable Elimination and Symbolic Probabilistic Inference for evaluating Influence Diagrams ☆

Rafael Cabañas [a,*], Andrés Cano [a], Manuel Gómez-Olmedo [a], Anders L. Madsen [b,c]

[a] Department of Computer Science and Artificial Intelligence, CITIC, University of Granada, Spain
[b] HUGIN EXPERT A/S, Aalborg, Denmark
[c] Department of Computer Science, Aalborg University, Denmark

ARTICLE INFO

Article history:
Received 13 May 2015
Received in revised form 27 November 2015
Accepted 30 November 2015
Available online 10 December 2015

Keywords:
Influence Diagrams
Probabilistic graphical models
Combinatorial optimization problem
Exact evaluation
Heuristic algorithm
Lazy Evaluation

ABSTRACT

An Influence Diagram is a probabilistic graphical model used to represent and solve decision problems under uncertainty. Its evaluation requires performing several combinations and marginalizations on the potentials attached to the Influence Diagram. Finding an optimal order for these operations, which is NP-hard, is an element of crucial importance for the efficiency of the evaluation. In this paper, two methods for optimizing this order are proposed. The first one is an improvement of the Variable Elimination algorithm while the second is the adaptation of the Symbolic Probabilistic Inference for evaluating Influence Diagrams. Both algorithms can be used for the direct evaluation of IDs but also for the computation of clique-to-clique messages in Lazy Evaluation of Influence Diagrams. In the experimental work, the efficiency of these algorithms is tested with several Influence Diagrams from the literature.

1. Introduction

Influence Diagrams (IDs) [3,4] are an effective modelling framework for analysis of Bayesian decision making under uncertainty. The goal of evaluating an ID is to obtain the best option for a single decision maker (*optimal policy*) and its utility.

The evaluation algorithms proposed [5–8] require performing several combinations and marginalizations on the potentials attached to the ID (probability and utility functions not necessarily normalized). Finding an optimal order for these operations is a NP-hard problem [9] and it is an element of crucial importance for the efficiency of the evaluation. The evaluation of an ID can be considered as a combinatorial optimization, that is the problem of finding an optimal order in which combinations are performed. This idea was already used to make inference in Bayesian Networks (BNs) with the first version of Symbolic Probabilistic Inference algorithm (SPI) [10] and with an improved algorithm in the SPI family called set-factoring [11]. In a related work [12] some experiments with SPI were performed to evaluate decision networks, however no details of the algorithm were provided.

☆ Preliminary versions of this paper were presented at IPMU'14 [1] and at the workshop PGM'14 [2].

* Corresponding author.
E-mail addresses: rcabanas@decsai.ugr.es (R. Cabañas), acu@decsai.ugr.es (A. Cano), mgomez@decsai.ugr.es (M. Gómez-Olmedo), madsen@hugin.con (A.L. Madsen).

http://dx.doi.org/10.1016/j.ijar.2015.11.011
0888-613X/

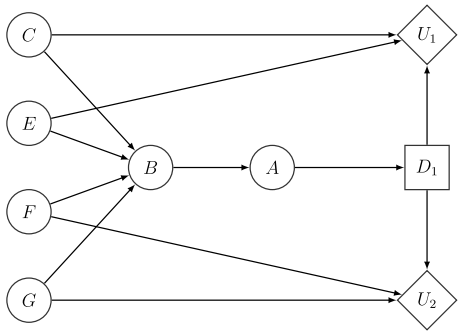

Fig. 1. An example of an ID with the partial order: $\{A\} \prec D_1 \prec \{B, C, E, F, G\}$.

In this paper different approaches for optimizing the order of the operations involved in the evaluation of IDs are considered. First, we describe the SPI algorithm for evaluating IDs taking into account the differences of an ID compared to a BN: two kinds of potentials, the temporal order of decisions, etc. Secondly, an optimization of Variable Elimination (VE) based on [13] is also proposed. This optimization consists of using a greedy algorithm for minimizing the cost of the combination of all the potentials involved in the removal of a variable. This optimization can be seen as an extension of the *binary join trees* of P.P. Shenoy [13] to IDs. Both algorithms are described for the direct evaluation of IDs (without using any auxiliary structure) and for the computation of clique-to-clique messages in Lazy Evaluation (LE) of IDs. In the experimental work, we analyze the behaviour of all these algorithms using a set of IDs from the literature. It is demonstrated that the algorithms proposed can improve the efficiency of the evaluation. Moreover, SPI outperforms VE in many instances. We also propose a pre-analysis algorithm based on the number of arithmetic operations that could help to predict which of the algorithms is the most appropriate one for evaluating each ID.

The paper is organized as follows: Section 2 introduces basic concepts about IDs; Section 3 describes two algorithms present in the literature for evaluating IDs (VE and LE); in Section 4 the motivation of this work is explained; the SPI algorithm for the direct evaluation of IDs is explained in Section 5 while the use of this algorithm for computing clique-to-clique messages in LE of IDs is described in Section 6; the description of the optimization of VE is given in Section 7; Section 8 includes the experimental work and results; finally Section 9 details our conclusions and lines for future work.

2. Influence diagrams

2.1. Definitions and notation

An ID [3] 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.

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 or never observed. That is, there is a partial order: $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$. For example, the ID shown in Fig. 1, contains a single decision D_1. The set of chance nodes $\mathcal{I}_0$ is $\{A\}$ and the set $\mathcal{I}_1$ is $\{B, C, E, F, G\}$.

Some evaluation algorithms require a regularity property. An ID is regular if several conditions are satisfied: the directed graph does not have directed cycles; the utility nodes have no successors (children), and there is a directed path traversing all the decision nodes. This last condition implies a total ordering for the decisions. The *non-forgetting assumption* is usually required as well: previous decisions and observations are known at each decision. Information arcs that satisfy this condition (*no-forgetting arcs*) are usually assumed implicit to reduce complexity of the graphical display. In this paper, we consider only regular, discrete IDs and we assume non-forgetting arcs to be present.

In the description of an ID, it is more convenient to think in terms of predecessors and successors. Thus, a node can belong to different sets in relation to another node:

- **Informational predecessors**: Let D_i be a decision node, then the direct predecessors of D_i are called *informational predecessors* or *informational parents*. The set of all informational predecessors of D_i is denoted by $pa(D_i)$. Arcs into decisions are called *informational arcs*.
- **Conditional predecessors**: Let X_i be a chance or utility node, then the direct predecessors of X_i are called *conditional predecessors* or *conditional parents*. The set of all conditional predecessors of X_i is denoted by $pa(X_i)$.
- **Direct successors**: Let X_i be a chance or decision node, then the set of all *direct successors* or *children* of X_i are denoted by $ch(X_i)$.

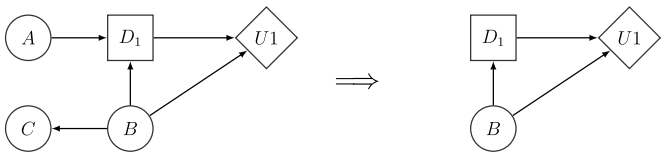

Fig. 2. Example of minimalization of an ID.

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

2.2. ID evaluation

The goal of evaluating an ID is to obtain an *optimal policy* δ_{D_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 $\Delta = \{\delta_{D_1}, \delta_{D_2}, \ldots, \delta_{D_n}\}$, including a policy for each decision variable. An optimal strategy $\widehat{\Delta}$ returns the optimal choice the decision maker should take for each decision.

Policy and Expected Utility: *Let ID be influence diagram over the universe $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$ and let $\mathcal{U}_V$ 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:*

(a) An optimal policy for D_i is

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

(b) The expected utility for D_i (and acting optimally in the future) is:

$$EU_{D_i}(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{i-1}) = \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{i-1}|D_1, \ldots, D_{i-1})}$$
$$\max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} \prod_{X \in \mathcal{U}_C} P(X|pa(X)) \left(\sum_{V \in \mathcal{U}_V} U(pa(V)) \right) \tag{2}$$

(c) *The strategy Δ for the ID, consisting of an optimal policy for each decision, yields the maximum expected utility:*

$$MEU(\Delta) = \sum_{\mathcal{I}_0} \max_{D_1} \cdots \max_{D_n} \sum_{\mathcal{I}_n} \prod_{X \in \mathcal{U}_C} P(X|pa(X)) \left(\sum_{V \in \mathcal{U}_V} U(pa(V)) \right) \tag{3}$$

2.3. Minimalization of an ID

Before the evaluation, an ID can be simplified (*minimalization*) by removing redundant informational arcs and barren nodes. Let X be an informational predecessor of a decision D_i, then the informational arc between X and D_i is redundant if X is d-separated [14,15] from the utility nodes that are descendant of D_i given the rest of informational predecessors and D_i. In other words, X is non-requisite for computing the optimal policy for D_i. Any redundant arc can be removed in reverse order [16,17]. Note that the d-separation property for IDs [15,18] is slightly different from the one defined for BNs [14] since, utility nodes and informational arcs are ignored.

A chance or decision node is a *barren node* if it is sink, in other words, it has no successors or only barren successors. For evaluating an ID, barren nodes have no impact on the decisions and therefore they can be directly removed without processing [6]. After the minimalization, the parents of a decision compose its *relevant past*. Fig. 2 shows an example of minimalization of an ID where the redundant informational arc (A, D_1) can be removed. Then, barren nodes A and C can also be removed (chance node A becomes a barren node after the removal of the redundant arc).

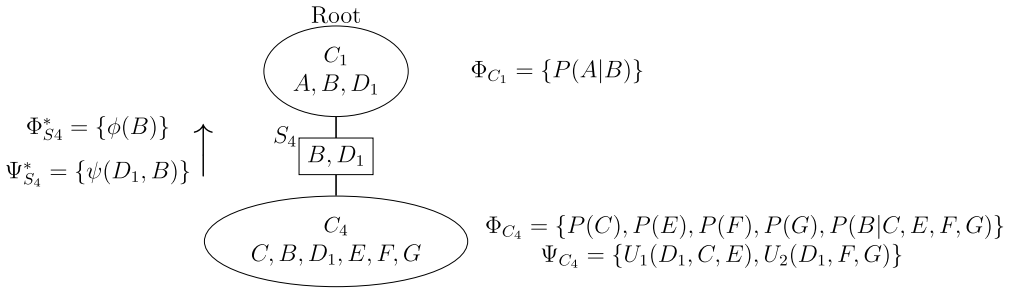

Fig. 3. Strong junction tree for the ID shown in Fig. 1 with the potentials associated to each clique (right) and messages stored at each separator (left).

3. Review of evaluation algorithms

3.1. Variable Elimination

The *Variable Elimination* algorithm (VE) [18] is one of the most common algorithms used for evaluating IDs. This algorithm has many similarities with the corresponding one for BNs [19]: it starts with a set of potentials and it eliminates all the variables one by one. There are however some differences compared to the VE algorithm for BNs. First, all the variables must be removed in reverse order of information precedence given by $\prec$. Secondly, chance variables are removed using *sum-marginalization* whereas for decisions *max-marginalization* is used. That is, it first sum-marginalizes $\mathcal{I}_n$, then max-marginalizes D_n, sum-marginalizes $\mathcal{I}_{i-1}$, etc. This type of elimination order is called a *strong elimination order* [20]. The procedure for removing a variable X_i is shown in Algorithm 1. It can be observed that the sets of potentials Φ and Ψ change along the evaluation process. Initially, Φ and Ψ contain the potentials present in the specification of the ID.

Algorithm 1: RemoveVariable.

1 $(\Phi_{X_i},\ \Psi_{X_i}) \leftarrow (\{\phi \in \Phi | X_i \in dom(\phi)\},\ \{\psi \in \Psi | X_i \in dom(\psi)\})$; // Relevant potentials
2 $(\phi_{X_i},\ \psi_{X_i}) \leftarrow (\prod \Phi_{X_i},\ \sum \Psi_{X_i})$; // Combine
3 **if** $X_i \in \mathcal{U}_C$ **then**
4 $\quad (\phi'_{X_i},\ \psi'_{X_i}) \leftarrow (\sum_{X_i} \phi_{X_i},\ \frac{\sum_{X_i} \phi_{X_i} \psi_{X_i}}{\phi'_{X_i}})$; // sum-marginalize
5 **else**
6 $\quad (\phi'_{X_i},\ \psi'_{X_i}) \leftarrow (\max_{X_i} \phi_{X_i},\ \max_{X_i} \phi_{X_i} \psi_{X_i})$; // max-marginalize
7 $(\Phi,\ \Psi) \leftarrow ((\Phi \backslash \Phi_{X_i}) \cup \{\phi'_{X_i}\},\ (\Psi \backslash \Psi_{X_i}) \cup \{\psi'_{X_i}\})$; // Update

In the original proposal of VE for evaluating IDs [18], the removal of a decision D_i involves the max-marginalization of D_i from the probability potentials (line 6). However, when removing D_i, often there is not any probability potential containing D_i and if any, it is a vacuous potential. Any decision is d-separated from its predecessors [18] and any successor has already been removed (the removal order of the disjoint subsets of variables must respect the temporal constraints). Thus, D_i can be directly removed from probability potentials without any computation.

The complexity of VE is linear in the size of the largest potential generated during the evaluation [21], which will be the result of the combination of $\phi_{X_i} \psi_{X_i}$ in lines 4 and 6 of Algorithm 1. That is, the complexity of VE for evaluating and ID with n variables (chance or decision) is $\mathcal{O}(n \cdot N_{max})$ where N_{max} is the largest potential ever created during the evaluation. However, the size of a potential is exponential in the number of variables in its domain. Thus, the computational cost of the VE algorithm depends on the sizes of the intermediate potentials generated. Note that the complexity can be related to the notion of *treewidth* [22,23] which is a measure of connectivity of the graph. Suppose we have an ID with treewidth w, then the complexity of VE, following an optimal elimination order, is $\mathcal{O}(n \cdot exp(w))$.

3.2. Lazy Evaluation

Lazy Evaluation (LE) was already used for making inference in BNs [24], so it can be adapted for evaluating IDs [7]. The basic idea of this method is to maintain the decomposition of the potentials 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 moral graph using a strong elimination order [20]. Nodes in the strong junction trees correspond to *cliques* (maximal complete sub-graphs) 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. Fig. 3 shows the strong junction tree for the ID shown in Fig. 1.

Propagation is performed by message-passing. Initially, each potential is associated to the clique closest 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 *CollectMessage* algorithm in the root (Algorithm 2).

Algorithm 2: CollectMessage.

```
/* 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 3);

A clique can send the message to its parent (*Absorption*) when 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 with each separator $S' \in ch(C_j)$ and then associating the obtained potentials with S_j. The original proposal [7] uses VE for removing the variables. Thus we will refer to this method as VE–Lazy Evaluation (VE–LE).

Algorithm 3: Absorption.

```
/* Let C_j be a clique, S_j be the parent separator and S' ∈ ch(C_j) be each of the child separators. If
Absorption is invoked on C_j, then: */
```

1 $\mathcal{R}_{S_j} \leftarrow \Phi_{C_j} \cup \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} (\Phi^*_{S'} \cup \Psi^*_{S'})$; // Relevant potentials for computing $\mathcal{R}_{S_j}$
2 $\mathbf{X} \leftarrow \{X | X \in C_j, X \notin S_j\}$; // Variables to be removed
3 Choose a strong 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.

4. Motivation

In order to explain the motivation of this paper, let us consider the ID shown in Fig. 1. The optimal policy for D_1 can be calculated directly from Eq. (1):

$$\delta_{D_1}(A) = \arg\max_{D_1} \sum_{G,F,E,C,B} P(G)P(F)P(E)P(C)P(B|C,E,F,G)P(A|B)\left(U_1(G,F,D_1) + U_2(E,C,D_1)\right) \tag{4}$$

The table representing the joint probability of all chance variables might be too large. For that reason, some evaluation algorithms such as VE for IDs iteratively removes variables following a strong elimination order (see Section 3.1). The advantage of VE is that the removal of a variable X_i only involves computations with those potentials with X_i in their domain. Assuming that all the variables are binary, the optimal order for removing variables in $\mathcal{I}_1$ is G, F, E, C, B and the computations done are:

1. Removal of G (96 multiplications, 48 additions, 32 divisions):

$$\sum_G P(G) \cdot P(B|C,E,F,G) = \sum_G \phi(G,B|C,E,F) = \phi(B|C,E,F)$$
$$\frac{\sum_G \phi(G,B|C,E,F) \cdot U_1(G,F,D_1)}{\phi(B|C,E,F)} = \psi(D_1,F,B,C,E)$$

2. Removal of F (48 multiplications, 24 additions, 16 divisions):

$$\sum_F P(F) \cdot \phi(B|C,E,F) = \sum_G \phi(F,B|C,E) = \phi(B|C,E)$$
$$\frac{\sum_F \phi(F,B|C,E) \cdot \psi(D_1,F,B,C,E)}{\phi(F,B|C,E)} = \psi(D_1,B,C,E)$$

3. Removal of E (24 multiplications, 28 additions, 8 divisions):

$$\sum_E P(E)\cdot\phi(B|C,E)=\sum_E\phi(E,B|C)=\phi(B|C)$$
$$\frac{\sum_E\phi(E,B|C)\cdot(\psi(D_1,B,C,E)+U_2(E,C,D_1))}{\phi(B|C)}=\psi(D_1,B,C)$$

4. Removal of C (12 multiplications, 15 additions, 4 divisions):

$$\sum_C P(C)\cdot\phi(B|C)=\sum_C\phi(B,C)=\phi(B)$$
$$\frac{\sum_C\phi(B,C)\cdot\psi(D_1,B,C)}{\phi(C)}=\psi(D_1,B)$$

5. Removal of B (12 multiplications, 15 additions, 4 divisions):

$$\sum_B\phi(B)\cdot P(A|B)=\sum_B\phi(A,B)=\phi(A)$$
$$\frac{\sum_B\phi(A,B)\psi(D_1,B)}{\phi(A)}=\psi(D_1,A)$$

We can see that the removal of $\mathcal{I}_1$ using VE with the optimal order requires 366 arithmetic operations (192 multiplications, 110 additions and 64 divisions). Independently of the elimination order used to solve this ID, VE will always have to combine the marginal potentials with a large potential. However, with a re-order of the operations this situation can be avoided:

1. Removal of F, G (112 multiplications, 40 additions, 16 divisions):

$$\sum_{F,G}P(B|C,E,F,G)\cdot(P(F)\cdot P(G))=\sum_{F,G}\phi(B,F,G|C,E)=\phi(B|C,E)$$
$$\frac{\sum_{F,G}\phi(B,F,G|C,E)\cdot U_1(G,F,D_1)}{\phi(B|C,E)}=\psi(B,C,E,D_1)$$

2. Removal of B, C, E (68 multiplications, 42 additions, 4 divisions):

$$\sum_{B,C,E}(P(A|B)\cdot(P(E)\cdot P(C)))\cdot\phi(B|C,E)=\sum_{B,C,E}\phi(A,E,C,B)=\phi(A)$$
$$\frac{\sum_{B,C,E}\phi(A,E,C,B)(\psi(B,C,E,D_1)+U_2(E,C,D_1))}{\phi(A)}=\psi(D_1,A)$$

Now the removal of $\mathcal{I}_1$ requires 282 arithmetic operations (180 multiplications and 82 additions and 20 divisions). From this example we can deduce that sometimes it could be better to combine small potentials even if they do not share any variable (e.g., $P(E)$ and $P(C)$). This combination will never be performed using VE since it is guided by the elimination order. Thus, the efficiency of the evaluation can be improved if an optimization of the order of both operations, marginalization and combination, is performed [11].

4.1. Definition of the problem

Evaluating an ID can be seen as an optimization problem in which we try to find an order that minimizes the cost of the operations involved in the evaluation. Moreover, due to temporal restrictions, the problem can be divided into two sub-problems. The first one consists of finding the optimal order of the operations involved in the removal of a set of chance variables from a set of probability and utility potentials.

Assume that we shall remove the set of chance variables $\mathbf{X}$ from the sets of probability and utility potentials $\Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$ (potentials with any variable of $\mathbf{X}$ in the domain). That is we shall calculate Eq. (5).

$$\sum_{\mathbf{X}}\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}}\phi_{\mathbf{X}}\left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}}\psi_{\mathbf{X}}\right)=\sum_{\mathbf{X}}\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}}\phi_{\mathbf{X}}\frac{\sum_{\mathbf{X}}\left(\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}}\phi_{\mathbf{X}}\left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}}\psi_{\mathbf{X}}\right)\right)}{\sum_{\mathbf{X}}\left(\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}}\phi_{\mathbf{X}}\right)}\tag{5}$$

Previous expression is a factorization of potentials, thus we should calculate the set of potentials $\Phi'_{\mathbf{X}}$ and $\Psi'_{\mathbf{X}}$ such that:

$$\prod_{\phi'\in\Phi'_{\mathbf{X}}}\phi'=\sum_{\mathbf{X}}\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}}\phi_{\mathbf{X}}\qquad\sum_{\psi'\in\Psi'_{\mathbf{X}}}\psi'=\frac{\sum_{\mathbf{X}}\left(\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}}\phi_{\mathbf{X}}\left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}}\psi_{\mathbf{X}}\right)\right)}{\sum_{\mathbf{X}}\left(\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}}\phi_{\mathbf{X}}\right)}\tag{6}$$

To compute previous expression we should find an optimal order for the operations of sum-marginalization, multiplication, addition and division. Let $Y\in\mathbf{X}$ be a chance variable, let Φ_Y and Ψ_Y be the set of probability and utility potentials containing Y in the domain, $\Phi^*=\Phi_{\mathbf{X}}\backslash\Phi_Y$ and $\Psi^*=\Psi_{\mathbf{X}}\backslash\Psi_Y$. Applying the distributive law, the removal of Y can be performed using Eq. (7).

$$\sum_{\mathbf{X}\setminus Y}\left(\prod_{\phi^*\in\Phi^*}\phi^*\left(\sum_{Y}\prod_{\phi_Y\in\Phi_Y}\phi_Y\right)\left(\sum_{\psi^*\in\Psi^*}\psi^*+\frac{\sum_Y\left(\prod_{\phi_Y\in\Phi_Y}\phi_Y\left(\sum_{\psi_Y\in\Psi_Y}\psi_Y\right)\right)}{\sum_Y\left(\prod_{\phi_Y\in\Phi_Y}\phi_Y\right)}\right)\right) \tag{7}$$

From Eq. (7) we get that a variable Y can only be removed if the product of all the potentials in Φ_Y has been calculated. Moreover, the removal must be performed at the same time from the probability and utility potentials. Similarly, the second sub-problem consists on finding the optimal order of all the operations involved in the removal of a decision variable D from the set of utility potentials Ψ_D (potentials with D in the domain). Then we should calculate Eq. (8). In this case, the decision variable is removed using max-marginalization.

$$\psi'_D=\max_D\sum_{\psi_D\in\Psi_D}\psi_D \tag{8}$$

In both sub-problems, we try to find the optimal order for all the operations involved. This optimization will reduce the size of intermediate potentials and therefore the evaluation should be more efficient.

5. Symbolic Probabilistic Inference for IDs

5.1. Overview

The *Symbolic Probabilistic Inference* algorithm (SPI) was already used for making inference in BNs [10,11]. This is also a greedy algorithm that considers the removal of a set of variables as a combinatorial factorization problem. That is, SPI tries to find the optimal order for the combinations and marginalizations by choosing at each step the best operation. Herein we describe how the SPI algorithm can be used for directly evaluating IDs. The correctness and complexity analysis is shown in Appendix A. For evaluating IDs, as VE does, SPI removes all variables in the decision problem in reverse order of the partial ordering imposed by the information constraints (called a strong elimination order [20]). That is, it first sum-marginalizes $\mathcal{I}_n$, then max-marginalizes D_n, sum-marginalizes $\mathcal{I}_{k-1}$, etc. The general scheme of SPI algorithm as presented in this paper is shown in Algorithm 4.

Algorithm 4: SPI-algorithm.

input : Φ,Ψ (sets of potentials in the ID),
$\{\mathcal{I}_0, D_1, \mathcal{I}_1, \ldots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID)

1 **for** $k \leftarrow n$ **to** 0 **do**
2 $(\Phi_{\mathbf{X}},\Psi_{\mathbf{X}}) \leftarrow (\{\phi\in\Phi | dom(\phi)\cap\mathcal{I}_k\neq\emptyset\},\ \{\psi\in\Psi | dom(\psi)\cap\mathcal{I}_k\neq\emptyset\})$;
3 $(\Phi,\Psi) \leftarrow (\Phi\setminus\Phi_{\mathbf{X}},\ \Psi\setminus\Psi_{\mathbf{X}})$;
4 $(\Phi'_{\mathbf{X}},\ \Psi'_{\mathbf{X}}) \leftarrow RemoveChanceSet(\mathcal{I}_k,\Phi_{\mathbf{X}},\Psi_{\mathbf{X}})$; // Algorithm 8
5 $(\Phi,\Psi) \leftarrow (\Phi\cup\Phi'_{\mathbf{X}},\ \Psi\cup\Psi'_{\mathbf{X}})$;
6 **if** $k > 0$ **then**
7 $(\Phi_D,\Psi_D) \leftarrow (\{\phi\in\Phi | D_k\in dom(\phi)\},\ \{\psi\in\Psi | D_k\in dom(\psi)\})$;
8 $(\Phi,\Psi) \leftarrow (\Phi\setminus\Phi_D,\ \Psi\setminus\Psi_D)$;
9 $(\Phi'_D,\ \psi'_D) \leftarrow RemoveDecision(D_k,\Phi_D,\Psi_D)$; // Algorithm 10
10 $(\Phi,\Psi) \leftarrow (\Phi\cup\Phi'_D,\ \Psi\cup\{\psi'_D\})$;

SPI and VE algorithms differ in the way they solve these problems: VE chooses at each step a variable to remove while SPI chooses a pair of potentials to combine and eliminate variables when possible. In this sense SPI is more fine-grained than VE. The latter only considers the next variable to eliminate, and not the order in which potentials are combined.

5.2. Combination candidate set

The SPI algorithm uses a data structure for storing the candidate pairs of potentials to combine in the next iteration. Herein we introduce this data structure and detail the related algorithms used in posterior sections.

Given a set of potentials $\Phi_{\mathbf{X}}$, B is a *combination candidate set* defined as $\{\{\phi_i,\phi_j\} | \phi_i,\phi_j\in\Phi_{\mathbf{X}}, \phi_i\neq\phi_j\}$. That is, B contains all pairwise combinations of elements of $\Phi_{\mathbf{X}}$. Algorithm 5 takes two input arguments: a set of potentials $\Phi_{\mathbf{X}}$ and an existing combination candidate set B. If B is empty, this algorithm returns a set B with all pairwise combinations of elements of $\Phi_{\mathbf{X}}$. Otherwise, the algorithm adds the new pairs without removing those pairs already in B. Notice that, if potentials are represented as tables, $\{\phi_i,\phi_j\}$ is equivalent to $\{\phi_j,\phi_i\}$. Thus the pair $\{\phi_j,\phi_i\}$ is not added to B if $\{\phi_i,\phi_j\}$ is already present (line 4). For example, given a set of potentials $\Phi_{\mathbf{X}}=\{\phi_1,\phi_2,\phi_3\}$, the combination candidate is $\{\{\phi_1,\phi_2\},\{\phi_1,\phi_3\},\{\phi_2,\phi_3\}\}$.

A pair is a set of two potentials, thus any operation with sets can be used with pairs. For example, given the pairs $p=\{\phi_1,\phi_2\}$, $p'=\{\phi_1,\phi_3\}$ and $p''=\{\phi_3,\phi_4\}$, the intersection $p\cap p'$ is $\{\phi_2\}$ while the intersection $p\cap p''$ is $\emptyset$. Thus, given a combination candidate set B and a pair $p\in B$, the removal of any pair $p'\in B$ containing at least one potential in common with p is denoted as $\{p'\in B | p'\cap p=\emptyset\}$. Similarly, the removal of both potentials in p from a potential set $\Phi_{\mathbf{X}}$ is denoted as $\Phi_{\mathbf{X}}\setminus p$.

Algorithm 5: addPairwiseCombinations.

input : $\Phi_{\mathbf{X}} = \{\phi_1, \phi_2, \ldots \phi_m\}$ (set of m potentials), B (existing combination candidate set)
output: B (updated combination candidate set)

```
1 for i ← 1 to m − 1 do
2     for j ← i + 1 to m do
3         p ← {φ_i, φ_j};
4         if p ∉ B then
5             B ← B ∪ {p};
6 return B;
```

Once the pairwise combination candidate set is built, a pair of potentials should be selected to combine. Algorithm 6 shows the procedure for selecting a pair from B minimizing any score or heuristic. Some examples of these heuristics are later explained in Section 5.5.

Algorithm 6: selectBest.

input : $B = \{p_1, \ldots, p_l\}$ (combination candidate set with l pairs)
output: *bestPair* (the best pair in B minimizing any score)

```
1 minScore ← +∞;
2 for i ← 1 to l do
3     s ← score(p_i);
4     if s < minScore then
5         bestPair ← p_i;
6         minScore ← s;
7 return bestPair;
```

In the original proposal of SPI algorithm for BNs, the combination candidate set B does not contain singletons. That is B is composed only of pairs of potentials. In our approach, the set B can also contain singleton potentials containing any variable that can be directly removed. That is a variable which is only present in one potential. With this improvement, the algorithm is not forced to combine at least two potentials in order to remove the first variable. On the other hand, these variables are not directly removed because it cannot be assured that the cost of this removal is lower that the cost of selecting a pair of potentials.

For example, let $\mathbf{X} = \{A, B, C\}$, be a set of chance variables that we want to remove from the set of probability potentials $\Phi_{\mathbf{X}} = \{\phi(A|B), \phi(B|C, E), \phi(C)\}$, then the candidate combination set generated is:

$$B = \Big\{ \{\phi(A|B), \phi(B|C, E)\},\ \{\phi(A|B), \phi(C)\},\ \{\phi(B|C, E), \phi(C)\},\ \{\phi(A|B)\} \Big\}$$

Previous set B contains the singleton $\{\phi(A|B)\}$ because variable A only belongs to the domain of this potential and thereby it can be sum-marginalized without need of performing first a combination. Although variable E only belongs to $\phi(B|C, E)$, this potential is not added as a singleton because E is not contained in the set $\mathbf{X}$ of variables we aim to remove. The procedure for adding the singletons to an existing combination candidate set B, given a set of variables $\mathbf{X}$ to remove from $\Phi_{\mathbf{X}}$ is shown in Algorithm 7.

Algorithm 7: addSingletons.

input : $\Phi_{\mathbf{X}} = \{\phi_1, \phi_2, \ldots \phi_{|\Phi|}\}$ (set of potentials), $\mathbf{X}$ (set of variables to remove), B (existing combination candidate set)
output: B (updated combination candidate set)

```
1 for i ← 1 to |Φ_X| do
2     if ∃Y ∈ dom(φ_i) ∩ X | ∀φ ∈ Φ_X \ {φ_i} : Y ∉ dom(φ) then
3         B ← B ∪ {{φ_i}} ;                          // {φ_i} is a singleton
4 return B;
```

5.3. 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 $\Phi_{\mathbf{X}}$ as shown in Algorithm 8. For that purpose, all possible pairwise combinations between the probability potentials are stored in the set combination candidate set B (line 3). Besides, B also contains those probability potentials or singletons (line 4) that contain any variable of $\mathbf{X}$ which is not present in any other potential of $\Phi_{\mathbf{X}}$.

Algorithm 8: RemoveChanceSet.

input : $\mathbf{X}$ (subset of chance variables), $\Phi_{\mathbf{X}}$ (set of probability potentials relevant for removing $\mathbf{X}$), $\Psi_{\mathbf{X}}$ (set of utility potentials relevant for removing $\mathbf{X}$)
output: Sets of potentials $\Phi'_{\mathbf{X}}$ and $\Psi'_{\mathbf{X}}$ resulting from removing $\mathbf{X}$ (Eq. (6))

1 $B \leftarrow \emptyset$; // Empty combination candidate set
2 **repeat**
3 $B \leftarrow addPairwiseCombinations(\Phi_{\mathbf{X}}, B)$;
4 $B \leftarrow addSingletons(\Phi_{\mathbf{X}}, \mathbf{X}, B)$;
5 $p \leftarrow selectBest(B)$;
6 **if** p *is a pair* **then**
7 $\phi_{ij} \leftarrow \phi_i \cdot \phi_j$; // p is a pair $\{\phi_i, \phi_j\}$
8 **else**
9 $\phi_{ij} \leftarrow \phi_i$; // p is a singleton $\{\phi_i\}$
10 $\mathbf{W} \leftarrow \{W \in dom(\phi_{ij}) \cap \mathbf{X} | \forall \phi \in \Phi_{\mathbf{X}} \backslash p : W \notin dom(\phi)\}$;
11 $\Psi_{\mathbf{W}} \leftarrow \{\psi \in \Psi_{\mathbf{X}} | \mathbf{W} \cap dom(\psi) \neq \emptyset\}$;
12 **if** $\mathbf{W} \neq \emptyset$ **then**
13 $(\phi'_{ij}, \Psi'_{\mathbf{W}}) \leftarrow Sum\text{-}marginalize(\mathbf{W}, \phi_{ij}, \Psi_{\mathbf{W}})$;
14 **else**
15 $(\phi'_{ij}, \Psi'_{\mathbf{W}}) \leftarrow (\phi_{ij}, \Psi_{\mathbf{W}})$
16 $B \leftarrow \{p' \in B | p' \cap p = \emptyset\}$; // Update
17 $\mathbf{X} \leftarrow \mathbf{X} \backslash \mathbf{W}$;
18 $\Phi_{\mathbf{X}} \leftarrow (\Phi_{\mathbf{X}} \backslash p) \cup \{\phi'_{ij}\}$
19 $\Psi_{\mathbf{X}} \leftarrow (\Psi_{\mathbf{X}} \backslash \Psi_{\mathbf{W}}) \cup \Psi'_{\mathbf{W}}$
20 **until** $\mathbf{X} = \emptyset$;
21 $(\Phi'_{\mathbf{X}}, \Psi'_{\mathbf{X}}) \leftarrow (\Phi_{\mathbf{X}}, \Psi_{\mathbf{X}})$;
22 **return** $(\Phi_{\mathbf{X}}, \Psi_{\mathbf{X}})$;

At each iteration, an element of B is selected (line 5). If it is a pair (line 7), both potentials are combined. The procedure stops when all variables in $\mathbf{X}$ have been removed. A variable can be removed at the moment it only appears in a single probability potential. Thus, after each combination it is computed the set of variables $\mathbf{W} \subseteq \mathbf{X}$ satisfying such condition (line 10) and the set of utility potentials $\Psi_{\mathbf{W}}$ containing any variable in $\mathbf{W}$ (line 11). If there is any variable that can be removed, a similar procedure is performed for combining utilities and sum-marginalizing the variables in $\mathbf{W}$ (line 13). At the end of each iteration sets $B, \mathbf{X}, \Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$ are updated. Notice that this algorithm produces a factorization of potentials as stated in Eq. (6).

To illustrate Algorithm 8, let us suppose we aim to remove the set of variables $\mathbf{X} = \{A, C\}$ from $\Phi_{\mathbf{X}} = \{\phi(A), \phi(C), \phi(B|AC)\}$ and $\Psi_{\mathbf{X}} = \{\psi(A), \psi(A, B, C)\}$. Initially, the combination candidate set B is $\{\{\phi(A), \phi(C)\}, \{\phi(A), \phi(B|AC)\}, \{\phi(C), \phi(B|AC)\}\}$. Suppose that our heuristic chooses the pair $\{\phi(A), \phi(C)\}$ in line 5, then we combine both potentials obtaining $\phi(A, C)$ as a result in line 7. Notice that these two potentials are never combined by the VE algorithm since they do not share any variable. The set of removable variables $\mathbf{W}$ is equal to $\emptyset$ and the *sum-marginalize* algorithm is not invoked in line 13. At the end of this iteration, set B is now empty and $\Phi_{\mathbf{X}}$ is $\{\phi(A, C), \phi(B|A, C)\}$. The sets $\mathbf{X}$ and $\Psi_{\mathbf{X}}$ have not changed. In the second iteration the single pair in $B = \{\{\phi(A, C), \phi(B|A, C)\}$ are combined, obtaining $\phi(A, B, C)$ as a result. Now $\mathbf{W}$ is equal to $\{A, C\}$ and the *sum-marginalization* algorithm is invoked in line 13.

In Algorithm 8 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. Thus, in order to avoid additional computations, the utilities are only 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 9) involves finding a good order for summing the utility potentials. The procedure for that is quite similar to the one for combining probabilities, the main difference is that at the moment a variable can be removed, the probability and utility potentials resulting from the marginalization are computed (line 14). Notice that this procedure is invoked on $\Psi_{\mathbf{W}} \subseteq \Psi_{\mathbf{X}}$.

For sake of example, suppose Algorithm 9 is invoked to remove $\mathbf{W} = \{A, C\}$ from $\phi = \phi(A, B, C)$ and $\Psi_{\mathbf{W}} = \{\psi(A), \psi(A, B, C)\}$. Initially, B' is equal to $\{\{\psi(A), \psi(A, B, C)\}, \{\psi(A, B, C)\}\}$. Notice that now the combination candidate set contains the singleton $\psi(A, B, C)$ because variable C is in only one potential. Suppose that in the first iteration we select this singleton, then $\mathbf{V}$ is equal to $\{C\}$. Thus in line 14 variable C is sum-marginalize out obtaining as a result the potentials $\phi(A, B)$ and $\psi(A, B)$. Now the sets are updated as $\mathbf{W} = \{A\}$ and $\Psi_{\mathbf{W}} = \{\psi(A), \psi(A, B)\}$. In the second iteration, B' is $\{\psi(A), \psi(A, B)\}$ so the utilities potentials in the single pair are added and the sum-marginalization of A is performed. Finally the output of the algorithm is the probability potential $\phi(B)$ and set of utilities $\{\psi(B)\}$.

5.4. Removal of a decision

The removal of a decision variable D from a set probability potentials Φ_D and from a set of utility potentials Ψ_D is shown in Algorithm 10. This procedure does not imply the combination of any probability potential since any decision is d-separated from its predecessors [18] and any successor has already been removed (the removal order of the disjoint subsets

Algorithm 9: Sum-marginalize.

input : $\mathbf{W}$ (subset of chance variables), ϕ (probability potential relevant for removing $\mathbf{W}$), $\Psi_{\mathbf{W}}$ (set of utility potentials relevant for removing $\mathbf{W}$)
output: Potential ϕ' and set of potential $\Psi'_{\mathbf{W}}$ resulting from removing $\mathbf{W}$

1 $B' \leftarrow \emptyset$; // Empty combination candidate set
2 **if** $\Psi_{\mathbf{W}} = \emptyset$ **then**
3 **return** $(\sum_{\mathbf{W}} \phi, \emptyset)$;
4 **repeat**
5 $B' \leftarrow addPairwiseCombinations(\Psi_{\mathbf{W}}, B')$;
6 $B' \leftarrow addSingletons(\Psi_{\mathbf{W}}, \mathbf{W}, B')$;
7 $q \leftarrow selectBest(B')$;
8 **if** *q is a pair* **then**
9 $\psi_{ij} \leftarrow \psi_i + \psi_j$; // q is a pair $\{\psi_i, \psi_j\}$
10 **else**
11 $\psi_{ij} \leftarrow \psi_i$; // q is a singleton $\{\psi_i\}$
12 $\mathbf{V} \leftarrow \{V \in dom(\psi_{ij}) \cap \mathbf{W} | \forall \psi \in \Psi_{\mathbf{W}} \backslash q : V \notin dom(\psi)\}$;
13 **if** $\mathbf{V} \neq \emptyset$ **then**
14 $(\phi'_{\mathbf{V}}, \psi'_{\mathbf{V}}) \leftarrow (\sum_{\mathbf{V}} \phi, \ \frac{\sum_{\mathbf{V}}(\phi \otimes \psi_{ij})}{\phi'_{\mathbf{V}}})$;
15 **else**
16 $(\phi'_{\mathbf{V}}, \psi'_{\mathbf{V}}) \leftarrow (\phi, \psi_{ij})$;
17 $B' \leftarrow \{q' \in B' | q' \cap q = \emptyset\}$; // Update
18 $\mathbf{W} \leftarrow \mathbf{W} \backslash \mathbf{V}$;
19 $\phi = \phi'_{\mathbf{V}}$;
20 $\Psi_{\mathbf{W}} \leftarrow (\Psi_{\mathbf{W}} \backslash q) \cup \{\psi'_{\mathbf{V}}\}$
21 **until** $\mathbf{W} = \emptyset$;
22 $(\phi', \Psi'_{\mathbf{W}}) \leftarrow (\phi, \Psi_{\mathbf{W}})$;
23 **return** $(\phi, \Psi_{\mathbf{W}})$;

of variables must respect the temporal constraints). Thus, any probability potential $\phi(D_k, \mathbf{X})$ must be directly transformed 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 $i < k$. In practice, this means that each probability potential $\phi \in \Phi_D$ is restricted to any of the values $d \in \Omega_D$ (step 3). This restriction is denoted $\phi^{R(D=d)}$.

Algorithm 10: RemoveDecision.

input : D (decision variable), Φ_D (set of probability potentials relevant for removing D),
Ψ_D (set of utility potentials relevant for removing D)
output: Φ'_D (set of probability potentials resulting from removing D), ψ'_D (utility potential resulting from removing D)

1 $\Phi'_D \leftarrow \emptyset$;
2 **foreach** $\phi \in \Phi_D$ **do**
3 $\Phi'_D \leftarrow \Phi'_D \cup \{\phi^{R(D=d)}\}$;
4 $\psi'_D \leftarrow max\text{-}marginalize(D, \Psi_D)$;
5 **return** (Φ'_D, ψ'_D);

Algorithm 11 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 11: max-marginalize.

input : D (decision variable), Ψ_D (set of utility potentials relevant for removing D)
output: ψ'_D (utility potential resulting from removing D)

1 $B' \leftarrow \emptyset$;
2 **while** $|\Psi_D| > 1$ **do**
3 $B \leftarrow addPairwiseCombinations(\Psi_D, B')$;
4 $q \leftarrow selectBest(B')$;
5 $\psi_{ij} \leftarrow \psi_i + \psi_j$;
6 $B' \leftarrow \{q' \in B' | q' \cap q = \emptyset\}$; // Update
7 $\Psi_D \leftarrow (\Psi_D \backslash q) \cup \{\psi_{ij}\}$;
8 Let ψ^D be the single potential in Ψ_D;
9 $\psi'_D \leftarrow \max_D \psi^D$;
10 $\delta_D \leftarrow \arg\max_D \psi^D$
11 **return** ψ'_D;

In order to illustrate Algorithms 10 and 11, let us consider that we aim to remove decision D with $\Omega_D = \{d_0, d_1, d_2\}$ from $\Phi_D = \{\phi(A|D)\}$ and $\Psi_D = \{\psi(A, D), \psi(D, B, A), \psi(D)\}$. In lines 2 and 3 of Algorithm 10, decision is removed

from. That is, $\Phi'_D = \{\phi(A|D)^{R(D=d_0)}\} = \{\phi(A)\}$. Then, *max-marginalize*(D, Ψ_D) is invoked. The combination candidate set B' is $\{\{\psi(A,D), \psi(D,B,A)\}, \{\psi(A,D), \psi(D)\}, \{\psi(D,B,A), \psi(D)\}\}$. Suppose that the pair $\{\psi(A,D), \psi(D)\}$ is selected then both utility potentials are added giving as a result a new utility potential $\psi(A,D)$. In the second iteration B' is $\{\{\psi(A,D), \psi(D,B,A)\}\}$ so the addition $\psi(D,B,A) = \psi(A,D) + \psi(D,B,A)$ is done. Finally, in lines 9 and 10 D is max-marginalized out and its optimal policy is recorded.

5.5. Combination heuristics

During the removal of a set of chance variables, at each iteration a pair of probability potentials is selected to be combined (Algorithm 8, line 5). Since computing the cost of future combinations and marginalizations could be extremely expensive, the decision must be taken using a heuristic. Some heuristics used with VE for selecting the next variable to remove can be adapted for choosing a pair instead in the SPI algorithm. Let $p = \{\phi_i, \phi_j\}$ be a candidate pair to be combined, let $\phi_{ij} = \phi_i \cdot \phi_j$ be the resulting potential of the combination. Then, the heuristics *minimum size* [25], and *minimum weight* [20] are defined as:

$$min\_size(p) = |dom(\phi_i) \cup dom(\phi_j)| = |dom(\phi_{ij})| \tag{9}$$

$$min\_weight(p) = \prod_{X \in dom(\phi_{ij})} |\Omega_X| \tag{10}$$

The previous heuristics choose the next pair to combine using only information from the probability potentials involved. However, they do not consider if the pair chosen will imply a costly combination with the utilities. As explained in Section 5.3, utilities are only combined at the moment a variable can be removed. Let $\mathbf{W}$ be the set of variables that can be removed after combining potentials in the pair p, let $\Psi_{\mathbf{W}}$ be the set of utility potentials containing any variable in $\mathbf{W}$ and let $\psi = \sum_{\psi_k \in \Psi_{\mathbf{W}}} \psi_k$. Then the heuristic *minimum utility* can be defined as follows:

$$min\_utility(p) = \prod_{X \in dom(\phi_{ij})} |\Omega_X| \cdot \prod_{Y \in dom(\psi) \setminus dom(\phi_{ij})} |\Omega_Y| \tag{11}$$

Any of the heuristics previously mentioned can also be used for selecting a pair of utility potentials at steps 7 and 4 of Algorithms 9 and 11 respectively. These heuristics will be considered in the experimental analysis.

5.6. Probabilistic barren

A variable is *probabilistic barren* if it is barren when only the set of probability potentials of the ID is considered. In other words, such variable only belongs to one probability potential and, at least to one utility potential. Let Y be probabilistic barren and let $\phi(Y|\mathbf{X}_I)$ be a probability potential, then:

$$\sum_Y \phi(Y|\mathbf{X}_I) = 1_{\mathbf{X}_I}$$

where $1_{\mathbf{X}_I}$ is a unity potential, that is a potential defined on $\mathbf{X}_I$ assigning the value 1 to each configuration of $\Omega_{\mathbf{X}_I}$. When a probabilistic barren variable is removed, the sum-marginalization and division can be avoided (Algorithm 9, line 14). By contrast, the sum-marginalization from the utility potentials cannot be avoided. Unlike barren nodes (see Section 2.3), probabilistic barren variables cannot be removed during the minimalization phase as they have impact on the decisions. In conclusion, the efficiency of the computation can be improved if singletons are allowed and a detection a priori of probabilistic barren variables is performed. This point is empirically demonstrated in the experimental section.

5.7. Example

Let us consider the ID in Fig. 1 in order to illustrate the behaviour of the SPI algorithm as described in this paper. For sake of simplicity, $\phi(X_1, \ldots, X_n)$ will be denoted $\phi_{X_1,\ldots,X_n}$. First, SPI proceeds to remove variables in the chance set $\mathcal{I}_1 = \{B, C, E, F, G\}$ using Algorithm 8. The initial combination candidate set is:

$$\{\phi_C, \phi_E\}, \{\phi_C, \phi_F\}, \{\phi_C, \phi_G\}, \{\phi_C, \phi_{BCEFG}\}, \{\phi_C, \phi_{AB}\}, \{\phi_E, \phi_F\}, \{\phi_E, \phi_G\}, \{\phi_E, \phi_{BCEFG}\},$$
$$\{\phi_E, \phi_{AB}\}, \{\phi_F, \phi_G\}, \{\phi_F, \phi_{BCEFG}\}, \{\phi_F, \phi_{AB}\}, \{\phi_G, \phi_{BCEFG}\}, \{\phi_G, \phi_{AB}\}, \{\phi_{BCEFG}, \phi_{AB}\}$$

If the *minimum size* heuristic is used for selecting the next pair of potentials, there are 6 pairs minimizing this score. Let us suppose that the pair $\{\phi_C, \phi_E\}$ is chosen, then the resulting potential is ϕ_{CE}. At this point there is not any variable that can be removed, since variables C and E are contained in another potential (e.g., ϕ_{BCEFG}). Then, the set B is updated by

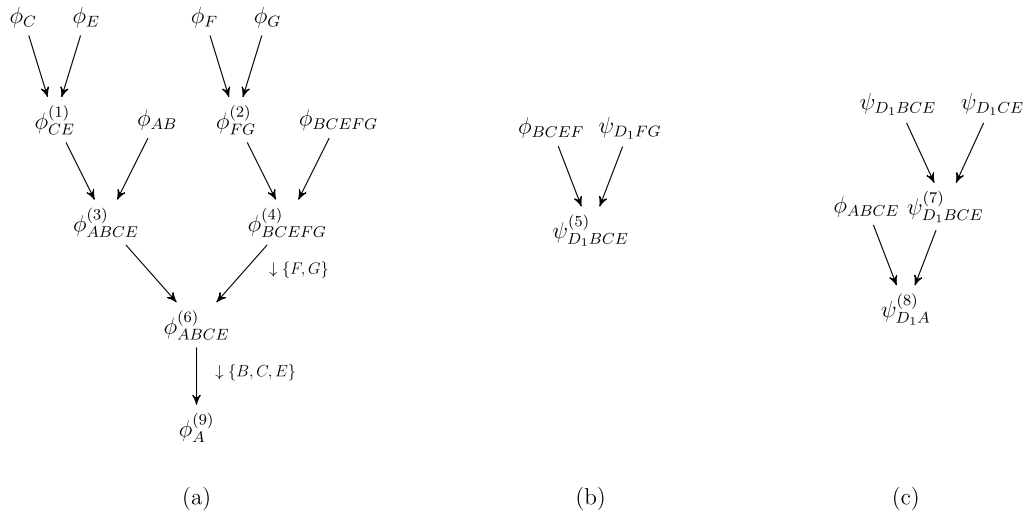

Fig. 4. Combination order of the probability potentials obtained using SPI for removing the chance set $\mathcal{I}_1 = \{B, C, E, F, G\}$ during the evaluation of the ID shown in Fig. 1.

removing pairs containing ϕ_C or ϕ_E and by adding new pairwise combinations with ϕ_{CE}:

$$\{\phi_{CE}, \phi_F\}, \{\phi_{CE}, \phi_G\}, \{\phi_{CE}, \phi_{BCEFG}\}, \{\phi_{CE}, \phi_{AB}\}, \{\phi_F, \phi_G\},$$
$$\{\phi_F, \phi_{BCEFG}\}, \{\phi_F, \phi_{AB}\}, \{\phi_G, \phi_{BCEFG}\}, \{\phi_G, \phi_{AB}\}, \{\phi_{BCEFG}, \phi_{AB}\}$$

The process will keep on choosing pairs to combine until all variables have been removed. The whole process is shown in Fig. 4 using three factor graphs [9]. Nodes without any parent correspond to initial potentials while child nodes to the resulting potentials of a combination. The numbers above each potentials indicate the combination order and arcs labels indicate the variables that are sum-marginalized.

In the 4th iteration, after generating the potential ϕ_{BCEFG}, variables F and G can be removed. Then, the Algorithm 9 is executed in order to combine utility potentials and max-marginalize these variables: the combination candidate set of utility potentials is $B = \{\{\psi_{D_1FG}\}\}$ and the resulting potentials are ϕ_{BCE} and ψ_{D_1BCE}. Similarly, in the 5th iteration, variables B, C and E can be removed. Now, the combination candidate set contains a pair and a singleton, that is $B = \{\{\psi_{D_1CE}, \psi_{D_1BCE}\}, \{\psi_{D_1BCE}\}\}$. The element selected from B is the pair $\{\psi_{D_1CE}, \psi_{D_1BCE}\}$. The variables B, C and E can be removed after adding both utility potentials in the pair, thus it is not necessary to perform any additional iteration. The resulting potentials are ϕ_A and ψ_{D_1A} which are also, in this example, the resulting potentials in Algorithm 8. SPI will now proceed to remove decision D_1 using Algorithm 10 and chance variable A using Algorithm 8.

6. SPI Lazy Evaluation

LE is based on message passing between cliques in a strong junction tree (see Section 3.2). Basically, the computation of these messages consists on removing variables not present in the parent separator from the sets of potentials in a clique. The original approach [7] uses VE for removing the variables. Thus, we will refer to this method as VE–Lazy Evaluation (VE–LE).

Here we propose SPI Lazy Evaluation (SPI–LE), which is a variant of Lazy Evaluation that uses SPI instead of VE in order to compute the messages. This idea was already considered in a previous paper [26] where the SPI algorithm was used in Lazy Propagation in BNs. 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 12).

The main difference is that the set of variables to remove is partitioned into disjoint subsets of chance variables or single sets of decisions because variables in an ID should be removed according to an order that respects 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 $\mathbf{X}_k$. Another difference is the way variables are removed: in SPI–LE, procedures explained in Sections 5.3 and 5.4 are used instead of VE.

7. Optimization of Variable Elimination

When evaluating an ID using the VE algorithm, variables are removed in reverse order of information precedence (see Section 3.1). The removal of a variable X_i implies combining all probability and utility potentials with X_i in the domain

Algorithm 12: Absorption-SPI.

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

1 $(\Phi^*_{S_j},\ \Psi^*_{S_j}) \leftarrow (\Phi_{C_j} \cup \bigcup_{S' \in ch(C_j)} \Phi^*_{S'},\ \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} \Psi^*_{S'})$; // Relevant potential sets
2 $\mathbf{X} \leftarrow \{X | X \in C_j, X \notin S_j\}$; // 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** $k \leftarrow n$ **to** 1 **do**
5 $(\Phi_{\mathbf{X}_k},\ \Psi_{\mathbf{X}_k}) \leftarrow (\{\phi \in \Phi^*_S | \mathbf{X}_k \cap dom(\phi) \neq \emptyset\},\ \{\psi \in \Psi^*_S | \mathbf{X}_k \cap dom(\psi) \neq \emptyset\})$;
6 $(\Phi^*_{S_j},\ \Psi^*_{S_j}) \leftarrow (\Phi^*_{S_j} \backslash \Phi_{\mathbf{X}_k},\ \Psi^*_{S_j} \backslash \Psi_{\mathbf{X}_k})$;
7 **if** $\mathbf{X}_k \subseteq \mathcal{U}_C$ **then**
8 $(\Phi'_{\mathbf{X}_k},\ \Psi'_{\mathbf{X}_k}) \leftarrow RemoveChanceSet(\mathbf{X}_k, \Phi_{\mathbf{X}_k}, \Psi_{\mathbf{X}_k})$; // Algorithm 8
9 $(\Phi^*_{S_j},\ \Psi^*_{S_j}) \leftarrow (\Phi^*_{S_j} \cup \Phi'_{\mathbf{X}_k},\ \Psi^*_{S_j} \cup \Psi'_{\mathbf{X}_k})$;
10 **else**
11 Let D_k the single variable in $\mathbf{X}_k$;
12 $(\Phi'_{\mathbf{X}_k},\ \psi'_{\mathbf{X}_k}) \leftarrow RemoveDecision(D_k, \Phi_{\mathbf{X}_k}, \Psi_{\mathbf{X}_k})$; // Algorithm 10
13 $(\Phi^*_{S_j},\ \Psi^*_{S_j}) \leftarrow (\Phi^*_{S_j} \cup \Phi'_{\mathbf{X}_k},\ \Psi^*_{S_j} \cup \{\psi'_{\mathbf{X}_k}\})$;
14 Associate $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$ to the parent separator S_j.

Algorithm 13: combineProbabilities.

input : Φ_{X_i} (set of probability potentials relevant for removing X_i)
output: ϕ_{X_i} (probability potential resulting from combining all the potentials in Φ_{X_i})
1 $B \leftarrow \emptyset$;
2 **while** $|\Phi_{X_i}| > 1$ **do**
3 $B \leftarrow addPairwiseCombinations(\Phi_{X_i}, B)$;
4 $p \leftarrow selectBest(B)$; // p is a pair $\{\phi_i, \phi_j\}$
5 $\phi_{ij} \leftarrow \phi_i \cdot \phi_j$;
6 $B \leftarrow \{p \in B' | p' \cap p = \emptyset\}$; // Update
7 $\Phi_{X_i} \leftarrow (\Phi_{X_i} \backslash p) \cup \{\phi_{ij}\}$;
8 Let ϕ_{X_i} be the single potential in Φ_{X_i};
9 **return** ϕ_{X_i};

(Algorithm 1 line 2). Here we propose using a greedy algorithm for optimizing the combination of the potentials involved in the removal of a variable. The procedure for combining a set of probability potentials is shown in Algorithm 13.

This algorithm is quite similar to the procedure used by Shenoy [13] for building the binary join trees. However, in our approach no tree-like structure is created, potentials are directly combined. A similar approach must be considered for adding all utility potentials (see Algorithm 14).

Algorithm 14: addUtilities.

input : Ψ_{X_i} (set of utility potentials relevant for removing X_i)
output: ψ_{X_i} (probability potential resulting from combining all the potentials in Ψ_{X_i})
1 $B' \leftarrow \emptyset$;
2 **while** $|\Psi_{X_i}| > 1$ **do**
3 $B' \leftarrow addPairwiseCombinations(\Psi_{X_i}, B')$;
4 $q \leftarrow selectBest(B')$; // q is a pair $\{\psi_i, \psi_j\}$
5 $\psi_{ij} \leftarrow \psi_i + \psi_j$;
6 $B' \leftarrow \{q' \in B' | q' \cap q = \emptyset\}$; // Update
7 $\Psi_{X_i} \leftarrow (\Psi_{X_i} \backslash q) \cup \{\psi_{ij}\}$;
8 Let ψ_{X_i} be the single potential in Ψ_{X_i};
9 **return** ψ_{X_i};

8. Experimental work

8.1. Procedure and objectives

In general, the aim of the experimental work is to analyze the behaviour of all the algorithms considered in the paper. We compare VE and SPI for directly evaluating an ID and for computing clique-to-clique messages in LE as well. The objectives of this experimentation are:

(a) Analyze if the SPI algorithm offers better results if probabilistic barren nodes are exploited and singletons are allowed. The combination heuristics explained in Section 5.5 are also compared.

Table 1
Features of the IDs used in the experimentation.

ID	Chance nodes	Decisions	Utilities	$\max \lvert\mathcal{I}_i\rvert$	Average potential size
Motivation ID	6	1	2	5	27.5
Oil	2	2	2	1	7.25
Oil Split Costs	2	2	3	1	6.2
NHL	17	3	1	11	468.111
Jaundice	21	2	1	10	41.5
Maze	14	2	1	6	3100.2
Car Buyer	3	3	1	1	64.5
Mildew 1	6	1	2	6	28.375
Mildew 4	7	2	2	4	32.222
Poker	7	1	1	7	94
ChestClinic	8	2	2	5	5.2
Appendicitis	4	1	1	2	3.6
Jensen et al. 1	4	2	2	3	5
Jensen et al. 2	12	4	4	8	4.875
Thinkbox	5	2	4	2	20.444
Threat of Entry	3	9	1	3	46
Wildlife	9	1	1	9	5
Competitive Asymm.	10	9	1	10	35.182

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

	Cliques	$\lvert C\rvert$		$w(C)$	
		min	max	min	max
Motivation ID	2	3	6	30	432
Oil	1	4	4	36	36
Oil Split Costs	1	4	4	36	36
NHL	8	5	12	32	$5.530 \cdot 10^5$
Jaundice	9	4	12	16	$1.555 \cdot 10^5$
Maze	3	5	12	$1.984 \cdot 10^4$	$4.032 \cdot 10^5$
Car Buyer	1	6	6	384	384
Mildew 1	2	3	3	64	112
Mildew 4	4	4	6	256	9408
Poker	5	3	3	32	324
ChestClinic	5	3	6	8	64
Appendicitis	1	4	4	16	16
Jensen et al. 1	3	3	4	8	16
Jensen et al. 2	9	3	5	8	32
Thinkbox	2	4	6	8	384
Threat of Entry	3	4	9	48	1728
Wildlife	7	3	4	8	16
Competitive Asymm.	3	4	6	96	1152

(b) Compare the improved version of VE (Section 7) with the original one.
(c) Compare the algorithms VE and SPI.

A set of 18 IDs from the literature are used: NHL and Jaundice are two real world IDs used for medical purposes [27,28]; the oil wildcatter's problem with one and two utilities [29,30]; an ID representing the Car Buyer problem [31]; an ID used to evaluate the population viability of wildlife species [32]; the Chest Clinic ID [33] obtained from the Asia BN; an ID representing the decision problem in the poker game [18]; two different IDs used at agriculture for treating mildew [18]; an ID to model a simplified version of the dice game called *Think-box*[1]; and ID for solving the maze problem [34]; finally, three synthetic IDs are used: the motivation example shown in Fig. 1 and two IDs proposed by Jensen et al. [8]. The details of these IDs are shown in Table 1, which contains the number of nodes of each kind, the size of the largest partition $\mathcal{I}_i$ of chance nodes and the total table size (number of entries in a table containing all the variables).

To compare VE and SPI for computing the clique-to-clique messages, a strong junction tree is built from each ID using the *minimum size* heuristic [25] for triangulating the graph. Table 2 shows, for each tree, the number of cliques, the minimum and maximum clique sizes $|C|$ and clique weights $w(C)$.

The SPI algorithm and the improved version of VE may have non-trivial additional computational costs for selecting the next potentials to combine. In order to check that these new algorithms do not have a large overhead, all the comparisons

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

Table 3
Cumulative time (ms) for evaluating all the IDs using the basic version of the SPI algorithm with different combination heuristics and considering the improvements of singletons and probabilistic barren (SPI_B, SPI_S and SPI_{BS}) and without them (SPI).

	min_size	*min_weight*	*min_utility*
SPI	21 540	35 100	345 100
SPI_B	21 350	34 980	343 800
SPI_S	35 400	73 960	17 650
SPI_{BS}	31 730	69 250	**16 420**

are made in terms of computation time.[2] Moreover, the portion of time corresponding to this overhead is shown in all the graphics. To avoid the influence of outliers, each ID is evaluated 100 times with each evaluation scheme.

8.2. Singletons and probabilistic barren

Here, the objective (a) is considered, that is we analyze if the efficiency of the computation can be improved if singletons are allowed and a detection a priori of probabilistic barren is performed. For that purpose each ID is evaluated using different schemes and algorithms. For the basic version of the SPI algorithm, four evaluation schemes are considered: SPI, SPI_B, SPI_S and SPI_{BS} where the subscript B means that probabilistic barren nodes are detected and the subscript S means that singletons are allowed. Similarly, each ID is also evaluated using the SPI–LE algorithm considering the schemes *SPI–LE*, $SPI\text{–}LE_B$, $SPI\text{–}LE_S$ and $SPI\text{–}LE_{BS}$. Fig. 5 shows the average computation time needed for evaluating each ID using the schemes of the basic version of the SPI algorithm. The combination heuristics considered are those explained in Section 5.5. For the majority of the IDs, the algorithm without any improvement (SPI) offers the worst performance. By contrast, the best results are obtained if both of the improvements proposed are applied (SPI_{BS}). This scheme is the fastest for evaluating 11, 14, and 10 networks when using the heuristics *min_size*, *min_weight* and *min_utility* respectively.

If we analyze the effect of adding singletons to the combination candidate set, we can observe that for some large IDs such as NHL the computation time can increase if these improvements are considered (SPI_S and SPI_{BS} with *min_weight* heuristic). The reason for that is that the search space is too large and the algorithm will give preference to selecting singletons even if the operations with the utilities are costly. This problem disappears if the heuristic considers the cost of operations with the utility (*min_utility*). The detection of probabilistic barren nodes (SPI_B) does not have any drawback: it is a simple procedure that will never increase the number of operations and in many cases will reduce it.

Fig. 5 also includes the overhead introduced by the SPI algorithm (bars in black). That is, the time required for selecting the next pair to combine (operations with the combination candidate set) and for updating the sets of potentials and variables. It can be observed that for most of the IDs, this overhead is small or insignificant: most of the time corresponds with the time required for computing with potentials. However, when evaluating the Wildlife ID the overhead is high. In this ID, all the chance nodes are in a single and large partition $\mathcal{I}$. As the overhead increases exponentially in the number of potentials, the overhead will be high. Even though there are other IDs with partitions of a similar size, they contain larger potentials. As a consequence, the overhead is smaller compared to the time required for computing with potentials.

Table 3 shows the total time for evaluating all the IDs. It can be observed that the lowest cumulative time is obtained with SPI_{BS} using the *minimum utility* heuristic. Considering also that, in most of the IDs, the scheme SPI_{BS} offers the best performance and that the heuristic *min_weight* avoids the problems produced by considering the singletons, we state that the best results are obtained with SPI_{BS} using the *minimum utility* heuristic.

Similarly, Fig. 6 shows the average time required for evaluating each ID using the four schemes of LE with SPI for computing the clique-to-clique messages (*SPI–LE*, $SPI\text{–}LE_B$, $SPI\text{–}LE_S$ and $SPI\text{–}LE_{BS}$). This evaluation time includes the time required for building the strong junction tree and for propagating the messages. It can be observed that, for most of the IDs considered, $SPI\text{–}LE_{BS}$ is the most efficient scheme. Moreover, it can also be observed that the problem with large IDs such as NHL in the growth of the evaluation time disappears: the search space for selecting a pair is now smaller (a large part of this search is now made during the building of the strong junction tree). In fact, there are less differences between schemes and heuristics: the combinatorial problem is divided into several sub-problems. As a consequence there is less room for improvement but the overhead is smaller. In fact, the large overhead introduced by the algorithm for evaluating the Wildlife ID is now insignificant. $SPI\text{–}LE_{BS}$ with *minimum weight* will be considered as the best scheme since it offers the best results for all the IDs used in the experimentation. In addition, the lowest cumulative time for evaluating all the IDs is obtained with this heuristic (see Table 4).

8.3. Optimization of Variable Elimination

In Section 7 a variation of the algorithm VE is proposed. This version of the algorithm optimizes the combination of the potentials involved in the removal of a variable using a greedy algorithm. Thus, the performance of VE should be

[2] The raw data obtained in the experimentation can be found in http://leo.ugr.es/rcabanas/spi/.

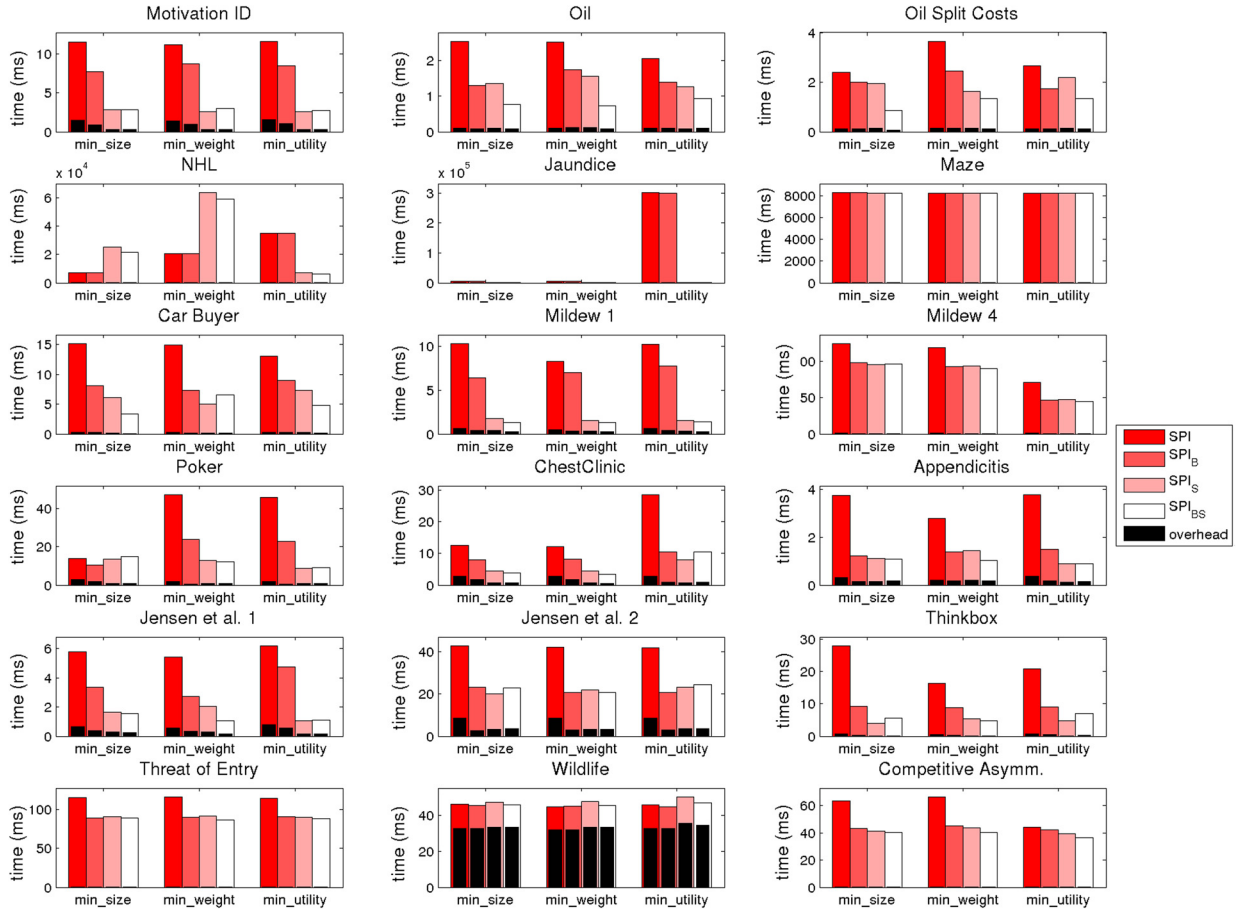

Fig. 5. Comparison of the computation time using the basic version of the SPI algorithm with different combination heuristics and considering the improvements of singletons and probabilistic barren (SPI_B, SPI_S and SPI_{BS}) and without them (SPI).

Table 4
Cumulative time for evaluating all the IDs using the basic version of the SPI–LE algorithm with different combination heuristics and considering the improvements of singletons and probabilistic barren ($SPI\text{–}LE_B$, $SPI\text{–}LE_S$ and $SPI\text{–}LE_{BS}$) and without them ($SPI\text{–}LE$).

	min_size	*min_weight*	*min_utility*
$SPI\text{–}LE$	12 800	12 050	12 610
$SPI\text{–}LE_B$	11 790	10 880	11 410
$SPI\text{–}LE_S$	12 980	12 160	12 690
$SPI\text{–}LE_{BS}$	11 210	**10 590**	11 100

improved (objective (b)). In order to simplify the experimentation, the equivalent heuristic used for selecting the variable to remove is used for selecting the pair of potentials to combine. In particular, heuristics *minimum size* and *minimum weight* are considered.

Fig. 7 shows the computation time required by the basic version of VE and the optimized one (VE_{opt}) for evaluating each ID. For most of the IDs, the optimized version requires less time and, in those IDs where VE_{opt} offers the worst performance, the results are quite similar. These graphics also include the overhead introduced by the optimization but also the one corresponding to the time required for choosing the next variable to remove. It can be observed that the optimization does not introduce a large overhead.

VE_{opt} using *minimum size* or *minimum weight* heuristics are considered as the best configuration schemes since the total time for evaluating all the IDs are the lowest (see Table 5).

Similarly, Fig. 8 shows the comparison of LE using both methods for computing the clique-to-clique messages ($VE\text{–}LE$ and $VE_{opt}\text{–}LE$). Again, the optimized version is faster for evaluating most of the IDs. $VE_{opt}\text{–}LE$ with any of the heuristics are considered as the best configuration schemes for computing the clique-to-clique messages since the cumulative time is the lowest (see Table 6).

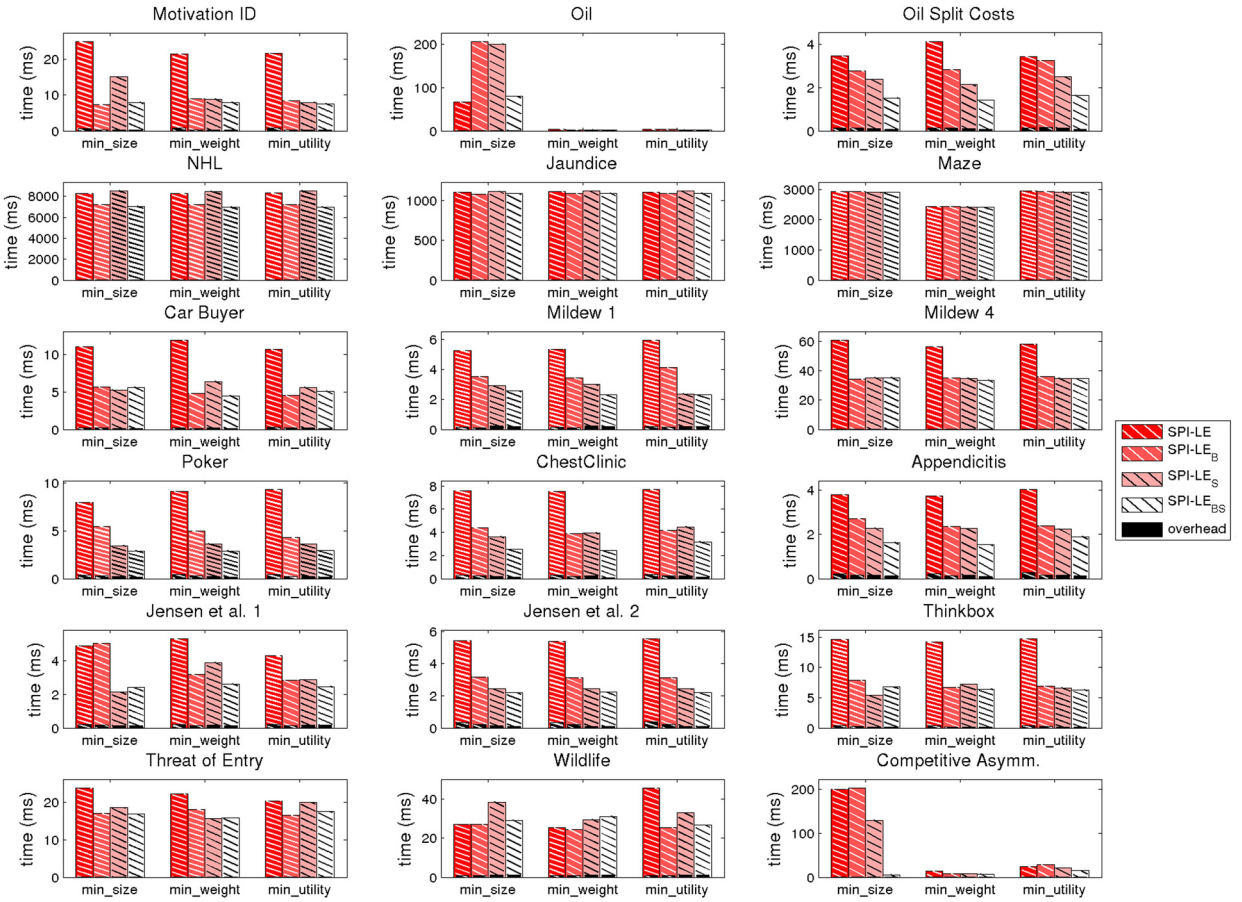

Fig. 6. Comparison of the average computation time using the basic version of the SPI–LE algorithm with different combination heuristics and considering the improvements of singletons and probabilistic barren ($SPI\text{–}LE_B$, $SPI\text{–}LE_S$ and $SPI\text{–}LE_{BS}$) and without them ($SPI\text{–}LE$).

Table 5
Cumulative time for evaluating all the IDs using *VE* and the optimized version with different heuristics.

	min_size	*min_weight*
VE	1749	1751
VE_{opt}	**1532**	1534

Table 6
Cumulative time for evaluating all the IDs using *VE–LE* and the optimized version with different heuristics.

	min_size	*min_weight*
VE–LE	9045	9048
$VE\text{–}LE_{opt}$	**8621**	8662

8.4. Comparison of SPI and VE

In previous subsections, it has been studied which is the best configuration for the algorithms SPI and VE. Fig. 9 shows the average computation time comparing the best scheme of *VE* against *SPI* and the best scheme of *VE–LE* against *SPI–LE* for evaluating each ID (objective (c)). First, it is compared VE_{opt} using *minimum size* heuristic against SPI_{BS} with *minimum utility* heuristic. The SPI algorithm offers the best results in 10 out 18 IDs. If we analyze those IDs were VE offers better performance than SPI, in 3 of them there are not great differences (ChestClinic, Jensen et al. 2 and Threat of Entry). Secondly both algorithms are also compared for computing clique-to-clique messages. That is $VE\text{–}LE_{opt}$ using *minimum size* heuristic against SPI_{BS} with *minimum weight* heuristic. Now the SPI algorithm for computing the messages offers the best results for evaluating 16 out of 18 IDs.

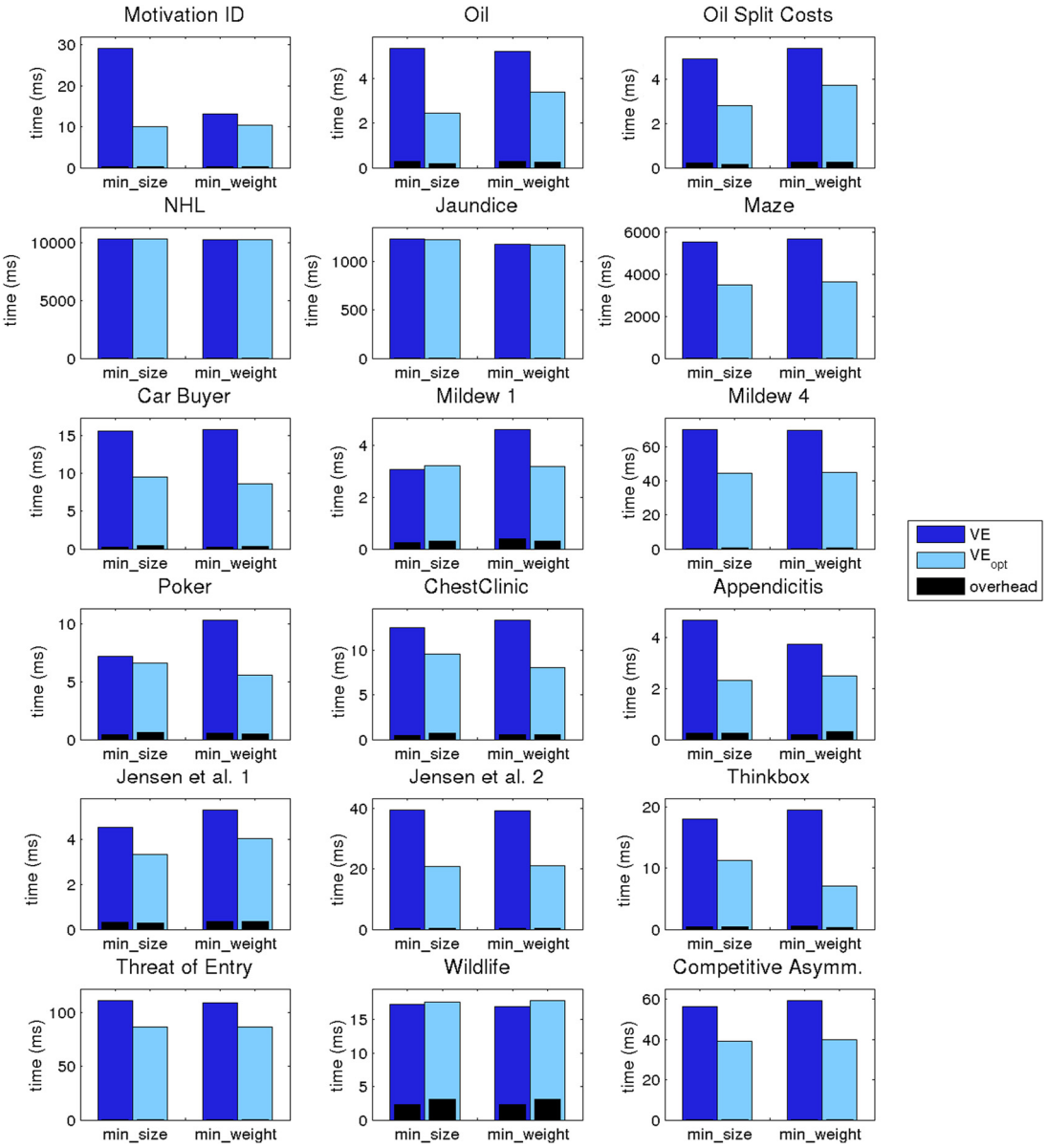

Fig. 7. Comparison of the average computation time required by *VE* and the optimized version with different heuristics.

8.5. Pre-analysis algorithm

In the results previously given, we have seen that *SPI* can outperform *VE* in many IDs. However, it cannot be assured that this will always be the case. The efficiency of these algorithms depend on several heuristics for determining the order of the operations involved in the evaluation. Therefore, we cannot assure which method (or heuristic) will offer the best results with a given ID until we evaluate it. However, we know that there is a correlation between the number of arithmetic operations and the efficiency of the methods. Table 7 shows the number of arithmetic operations and evaluation time for evaluating each ID with VE_{opt} and SPI_{BS} using the heuristics *min_size* and *min_utility* respectively. It can be observed that in 16 out of 18 a reduction in the number of operations means a reduction in the evaluation time. In the largest IDs (NHL, Jaundice and Maze), where it is more important to determine which is the best method, there is a high correlation. By contrast, in the Car Buyer and Mildew 1 IDs the reduction in the number of operations does not means a reduction in the evaluation time. These two IDs are quite small and the time required by the arithmetic operations has a lower weight in the total evaluation time.

Taking into account this correlation, a pre-analysis algorithm for predicting which method is the most efficient one can be developed. This algorithm computes the number of arithmetic operations by evaluating an ID with each method in a

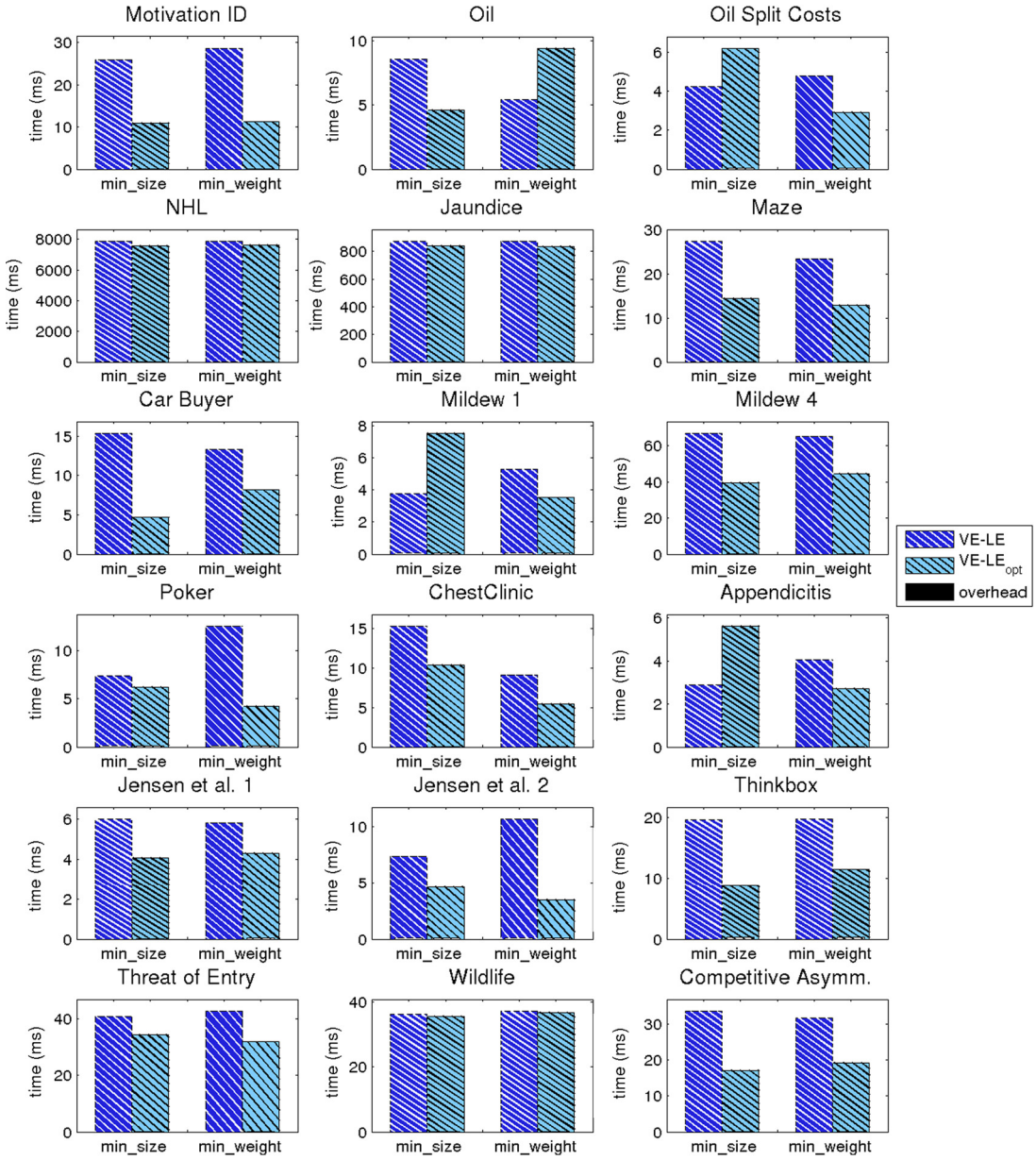

Fig. 8. Comparison of the average computation time required by the basic *VE–LE* and the optimized version with different heuristics.

qualitative way. This kind of evaluation is similar to the numerical (usual) evaluation but it does not perform the operations with potentials (domain of the resulting potentials are only computed). As the pre-analysis algorithm knows the domains of potentials involved in each operation with potentials, the number of arithmetic operations can be easily computed. Finally, the pre-analysis algorithm will base its decision on the number arithmetic operations.

In the rightmost column of Table 7, the time for performing this pre-analysis. It can be observed that, for large IDs, the pre-analysis time is much more smaller than the time required for evaluating the ID. By contrast, in smaller IDs, this pre-analysis could take more time than the evaluation.

9. Conclusions and future work

In this paper two algorithms for optimizing the operation order in the evaluation of IDs are described. First, the details of the adaptation of the SPI algorithm for evaluating IDs are given. This method was already used for making inference on BNs and it is more fine-grained than other methods in the literature such as VE. For this adaptation, differences between IDs and BNs have been taking into account: two kinds of potentials, temporal order between variables, etc. Secondly, an

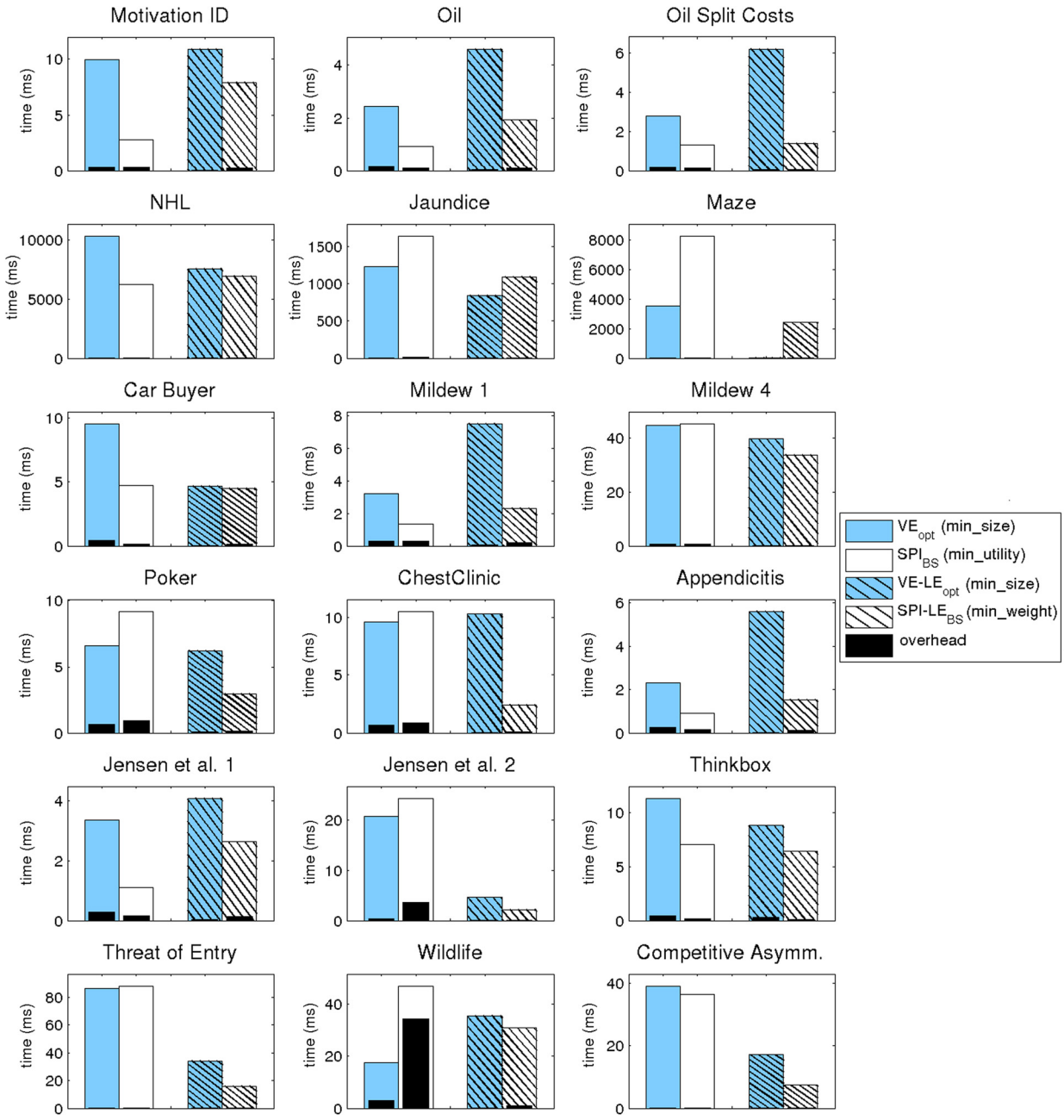

Fig. 9. Average computation time comparing the best scheme of *VE* against *SPI* and the best scheme of *VE–LE* against *SPI–LE* for evaluating each ID.

optimization of VE have also been proposed. This improved version consists of using a greedy algorithm for minimizing the cost of the combination of all the potentials involved in the removal of a variable. Both algorithms have been described for the direct evaluation of IDs and for the computation of clique-to-clique messages as well.

In the experimental work, these algorithms have been tested using a set of IDs from the literature. It has been demonstrated that, for many of the IDs considered, the SPI algorithm is more efficient if singletons are allowed and a priori detection of probabilistic barren is performed. For some large IDs the improvement of allowing singletons can produce an important growth in the evaluation time. However, this problem is solved if a combination heuristic that also considers the size of the utility potentials is used. For the computation of clique-to-clique messages, where the search space is smaller, this growth disappears. By contrast, the detection of probabilistic barren will never increase the evaluation time (it will remain the same or lower). Secondly, it has also been demonstrated that the optimized version of VE offers better results than the basic version. Finally, the best configuration schemes of both algorithms have been compared. For the direct evaluation of IDs and for the clique-to-clique message computation, SPI can outperform VE in many IDs.

Table 7
Number of arithmetic operations and evaluation time for evaluating each ID with VE_{opt} and SPI_{BS} using the heuristics *min_size* and *min_utility* respectively. The time for evaluating each ID with both methods in a qualitative way is also given (pre-analysis time).

ID	*# operations*		*time* (ms.)		Time pre-analysis
	VE_{opt} *min_size*	SPI_{BS} *min_utility*	VE_{opt} *min_size*	SPI_{BS} *min_utility*	
Motivation ID	**5939**	**1982**	**9.927**	**2.776**	2.4
Oil	**133**	**121**	**2.456**	**0.932**	3.319
Oil Split Costs	**145**	**133**	**2.794**	**1.354**	1.753
NHL	$\mathbf{3.570 \cdot 10^6}$	$\mathbf{2.384 \cdot 10^6}$	$\mathbf{1.031 \cdot 10^4}$	**6240.162**	75.438
Jaundice	$\mathbf{4.791 \cdot 10^5}$	$\mathbf{8.704 \cdot 10^5}$	**1227.203**	**1639.544**	220.016
Maze	$\mathbf{1.802 \cdot 10^6}$	$\mathbf{4.846 \cdot 10^6}$	**3515.281**	**8256.285**	48.482
Car Buyer	1375	1487	9.536	4.748	3.334
Mildew 1	395	511	3.211	1.361	1.884
Mildew 4	$\mathbf{3.272 \cdot 10^4}$	$\mathbf{4.432 \cdot 10^4}$	**44.636**	**45.154**	6.446
Poker	**1586**	**7034**	**6.573**	**9.163**	2.886
ChestClinic	**593**	**2465**	**9.601**	**10.52**	3.572
Appendicitis	**65**	**59**	**2.309**	**0.896**	1.721
Jensen et al. 1	**123**	**119**	**3.351**	**1.105**	3.95
Jensen et al. 2	**449**	**779**	**20.725**	**24.358**	6.358
Thinkbox	**1891**	**1793**	**11.31**	**7.079**	3.86
Threat of Entry	**3755**	**3971**	**86.247**	**87.95**	4.774
Wildlife	**155**	**163**	**17.657**	**46.948**	5.092
Competitive Asymm.	**3431**	**3135**	**39.094**	**36.536**	2.747

The SPI algorithm used only considers next pair of potentials to combine. Thus, a line of future research could be studying the behaviour of the algorithm using a higher neighbourhood degree. It could also be interesting making a study that let us to characterize which features of an ID make it for a possible candidate for choosing SPI over VE. Concerning to the pre-analysis, alternatives indicator to the number of operations could be studied.

Acknowledgements

This research was supported by the Spanish Ministry of Economy and Competitiveness under project TIN2013-46638-C3-2-P, the European Regional Development Fund (FEDER), the FPI scholarship program (BES-2011-050604). The authors have also been partially supported by "Junta de Andalucía" under projects P10-TIC-06016.

Appendix A. Correctness and complexity of the SPI algorithm

Herein we prove the correctness of the SPI algorithm by proving that it is a correct reordering of the operations used by VE (the correctness of VE can be found in [18]). Evaluating an ID involves performing several sum-marginalizations and max-marginalizations of the variables in the ID (see Eqs. (1), (2) and (3)). The order of these two operations are not interchangeable and therefore variables must be removed according to a strong elimination order. This condition is satisfied by the SPI algorithm (as described in Algorithm 4): *RemoveChanceSet* sum-marginalizes out $\mathcal{I}_n$, *RemoveDecision* max-marginalizes out D_n, *RemoveChanceSet* sum-marginalizes out $\mathcal{I}_{n-1}$, etc.

As explained in Section 4.1, the *RemoveChanceSet* algorithm should give as a result a factorization of probability potentials such that $\sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}}$. Let us suppose that our method aims to remove a set of variables $\mathbf{V} \subseteq \mathbf{X}$ (Algorithm 9, line 14). Previously all probability potentials in $\Phi'_{\mathbf{X}} \subseteq \Phi_{\mathbf{X}}$ have been combined. In other words the input argument ϕ of Algorithm 9 is equal to $\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}}$. Notice that it is guaranteed that any probability potential containing any variable in $\mathbf{V}$ has been combined (Algorithm 8, line 12). Using the distributive law we get:

$$\sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} =$$

$$= \sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \right) =$$

$$= \sum_{\mathbf{X} \setminus \mathbf{V}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \sum_{\mathbf{V}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \right) =$$

$$= \sum_{\mathbf{X} \setminus \mathbf{V}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \sum_{\mathbf{V}} \phi \right) =$$

$$= \sum_{\mathbf{X}\setminus\mathbf{V}} \left(\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}\setminus\Phi'_{\mathbf{X}}} \phi_{\mathbf{X}}\phi'_{\mathbf{V}} \right)$$

We see that the result of eliminating $\mathbf{V}$ implies removing each potential in $\Phi'_{\mathbf{X}}$ and replacing them by ϕ'_V. This is done in line 18 of Algorithm 8. Therefore operations with probability potentials are correct. Similarly, *RemoveChanceSet* algorithm should give also as a result a factorization of utility potentials $\alpha \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}} \psi_{\mathbf{X}} \right)$ with $\alpha = \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}} \phi_{\mathbf{X}}$. Let us now consider computations with utility potentials needed for removing $\mathbf{V}$. Previously all the addition of all utility potentials in $\Psi'_{\mathbf{X}} \subseteq \Psi_{\mathbf{X}}$ has been done. Notice that it is guaranteed that any utility potential containing any variable in $\mathbf{V}$ has been combined (Algorithm 9, line 13). Using the distributive law we get:

$$\alpha \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}} \psi_{\mathbf{X}} \right) =$$

$$= \alpha \sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}\setminus\Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i\in\Phi'_{\mathbf{X}}} \phi_i \left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}\setminus\Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \sum_{\psi_i\in\Psi'_{\mathbf{X}}} \psi_i \right) \right) =$$

$$= \alpha \sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}\setminus\Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i\in\Phi'_{\mathbf{X}}} \phi_i \sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}\setminus\Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}\setminus\Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i\in\Phi'_{\mathbf{X}}} \phi_i \sum_{\psi_i\in\Psi'_{\mathbf{X}}} \psi_i \right) =$$

$$= \alpha \sum_{\mathbf{X}\setminus\mathbf{V}} \prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}\setminus\Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \sum_{\mathbf{V}} \prod_{\phi_i\in\Phi'_{\mathbf{X}}} \phi_i \left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}\setminus\Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \frac{\sum_{\mathbf{V}} \left(\prod_{\phi_i\in\Phi'_{\mathbf{X}}} \phi_i \sum_{\psi_i\in\Psi'_{\mathbf{X}}} \psi_i \right)}{\sum_{\mathbf{V}} \prod_{\phi_i\in\Phi'_{\mathbf{X}}} \phi_i} \right) =$$

$$= \frac{\sum_{\mathbf{X}\setminus\mathbf{V}} \prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}\setminus\Phi'_{\mathbf{X}}} \phi_{\mathbf{X}}\phi'_{\mathbf{V}} \left(\sum_{\psi_{\mathbf{X}}\in\Psi_{\mathbf{X}}\setminus\Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \psi'_{\mathbf{V}} \right)}{\sum_{\mathbf{X}\setminus\mathbf{V}} \prod_{\phi_{\mathbf{X}}\in\Phi_{\mathbf{X}}\setminus\Phi'_{\mathbf{X}}} \phi_{\mathbf{X}}\phi'_{\mathbf{V}}}$$

From previous expression, we see that the result of eliminating $\mathbf{V}$ also implies removing each potential in $\Psi'_{\mathbf{X}}$ and replacing them by ψ'_V. This is done in line 20 of Algorithm 9. Therefore, operations with utility potentials are also correct.

When *removeDecision* is invoked for removing D_n out of Ψ_D, the result should be $\psi'_D = \max_D \sum_{\psi_D\in\Psi_D} \psi_D$. This procedure is correct since in Algorithm 11 max-marginalization operation is not made until the addition of all utility potentials is performed.

Similarly to VE, the complexity of the SPI algorithm is also linear to the size of the largest potential generated during the evaluation. The largest potential will be obtained right before the sum-marginalization or max-marginalization of a variable. That is, the result of the combination $\phi \cdot \psi_{ij}$ in lines 14 of Algorithm 9 or the potential ψ^D in line 8 of Algorithm 11. Thus, the complexity of the SPI algorithm for evaluating and ID with n variables (chance or decision) $\mathcal{O}(nN'^{n}_{max})$ where N'_{max} is the largest potential ever created during the evaluation. However the size of a potential is exponential to the number of variables in its domain. Thus, the computational cost of the VE algorithm depends on the sizes of the intermediate potentials generated and on the order of operations with potentials.

References

[1] R. Cabañas, A.L. Madsen, A. Cano, M. Gómez-Olmedo, On SPI for evaluating Influence Diagrams, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, 2014, pp. 506–516.

[2] R. Cabañas, A. Cano, M. Gómez-Olmedo, A.L. Madsen, On SPI–lazy evaluation of influence diagrams, in: Probabilistic Graphical Models, Springer, 2014, pp. 97–112.

[3] R.A. Howard, J.E. Matheson, Influence diagram retrospective, Decis. Anal. 2 (3) (2005) 144–147.

[4] U.B. Kjærulff, A.L. Madsen, Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis, vol. 22, Springer, 2013.

[5] R.D. Shachter, D. Bhattacharjya, Solving influence diagrams: exact algorithms, in: Wiley Encyclopedia of Operations Research and Management Science, 2010.

[6] R.D. Shachter, Evaluating influence diagrams, Oper. Res. (1986) 871–882.

[7] A.L. Madsen, F.V. Jensen, Lazy evaluation of symmetric Bayesian decision problems, in: Proceedings of the 15th Conference on Uncertainty in AI, Morgan Kaufmann Publishers Inc., 1999, pp. 382–390.

[8] F. Jensen, F.V. Jensen, S.L. Dittmer, From Influence Diagrams to junction trees, in: Proceedings of the 10th International Conference on Uncertainty in AI, Morgan Kaufmann Publishers Inc., 1994, pp. 367–373.

[9] M. Bloemeke, M. Valtorta, 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, Morgan Kaufmann Publishers Inc., 1998, pp. 16–23.

[10] R.D. Shachter, B. D'Ambrosio, B. Del Favero, Symbolic probabilistic inference in belief networks, in: AAAI, vol. 90, 1990, pp. 126–131.

[11] Z. Li, B. D'Ambrosio, Efficient inference in Bayes networks as a combinatorial optimization problem, Int. J. Approx. Reason. 11 (1) (1994) 55–81.

[12] B. D'Ambrosio, S. Burgess, Some experiments with real-time decision algorithms, in: Proceedings of the 12th International Conference on Uncertainty in AI, Morgan Kaufmann Publishers Inc., 1996, pp. 194–202.

[13] P.P. Shenoy, Binary join trees for computing marginals in the Shenoy–Shafer architecture, Int. J. Approx. Reason. 17 (2) (1997) 239–263.
[14] D. Geiger, T. Verma, J. Pearl, et al., Identifying independence in Bayesian networks, Networks 20 (5) (1990) 507–534.
[15] R.D. Shachter, Bayes-ball: rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams), in: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 480–487.
[16] E. Fagiuoli, M. Zaffalon, A note about redundancy in influence diagrams, Int. J. Approx. Reason. 19 (3) (1998) 351–365.
[17] D. Nilsson, S.L. Lauritzen, Evaluating influence diagrams using LIMIDs, in: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2000, pp. 436–445.
[18] F. Jensen, T. Nielsen, Bayesian Networks and Decision Graphs, Springer-Verlag, 2007.
[19] N. Zhang, D. Poole, Exploiting causal independence in Bayesian network inference, J. Artif. Intell. Res. 5 (1996) 301–328.
[20] U. Kjærulff, Triangulation of graphs – algorithms giving small total state space, Research report R-90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1990.
[21] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[22] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, in: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, ACM, 1993, pp. 226–234.
[23] N. Robertson, P.D. Seymour, Graph minors. IV. Tree-width and well-quasi-ordering, J. Comb. Theory, Ser. B 48 (2) (1990) 227–254.
[24] A.L. Madsen, F.V. Jensen, Lazy propagation: a junction tree inference algorithm based on lazy evaluation, Artif. Intell. 113 (1–2) (1999) 203–245.
[25] D. Rose, A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, in: Graph Theory and Computing, vol. 183, 1972, p. 217.
[26] A.L. Madsen, Variations over the message computation algorithm of lazy propagation, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 36 (3) (2005) 636–648.
[27] P. Lucas, B. Taal, Computer-based decision support in the management of primary gastric non-Hodgkin lymphoma, UU-CS (1998-33).
[28] C. Bielza, M. Gómez, S. Ríos Insua, J.A. Fernández del Pozo, P. García Barreno, S. Caballero, M. Sánchez Luna, Ictneo system for jaundice management, Rev. R. Acad. Cienc. Exactas Fís. Nat. 92 (4) (1998) 307–315.
[29] H. Raiffa, Decision Analysis: Introductory Lectures on Choices under Uncertainty, Addison–Wesley, 1968.
[30] P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks, Springer, 2007.
[31] R. Qi, L. Zhang, D. Poole, Solving asymmetric decision problems with influence diagrams, in: Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1994, pp. 491–497.
[32] B. Marcot, R. Holthausen, M. Raphael, M. Rowland, M. Wisdom, Using bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement, For. Ecol. Manag. 153 (1) (2001) 29–42.
[33] C. Goutis, A graphical method for solving a decision analysis problem, IEEE Trans. Syst. Man Cybern. 25 (8) (1995) 1181–1193.
[34] C. Yuan, X. Wu, E. Hansen, Solving multistage influence diagrams using branch-and-bound search, in: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, UAI 2010, 2010, pp. 691–700.