Approximate 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,mgomez,acu}@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 a tool used to represent and solve decision problems under uncertainty. One of the most efficient exact methods used to evaluate Influence Diagrams is Lazy Evaluation. This paper proposes the use of trees for representing potentials involved in an Influence Diagram in order to obtain an approximate Lazy Evaluation of decision problems. This method will allow to evaluate complex decision problems that are not evaluable with exact methods due to their computational cost. The experimental work compares the efficiency and goodness of the approximate solutions obtained using different kind of trees.

Keywords: Influence Diagram, Approximate computation, Lazy Evaluation, Deterministic algorithms, Context-specific independencies.

1 Introduction

An Influence Diagram (ID) [1] is a Probabilistic Graphical Model used for representing and evaluating decision problems under uncertainty. IDs can encode the independence relations between variables in a way that avoids an exponential growth of the representation. Several approaches have been proposed to evaluate IDs such as *Variable Elimination* [2,3] and *Arc Reversal* [4]. However, if the problem is too complex the application of these methods may become infeasible due to the high requirement of resources (time and memory). A technique that improves the efficiency of the evaluation is Lazy Evaluation (LE) [5]. The basic idea is to maintain a decomposition of the potentials and to postpone computation for as long as possible. Some other deterministic methods use alternative representations for the potentials, such as *trees* [6,7]. This representation supports the exploitation of *context-specific independencies* [8]. That is, identical values of the potential can be grouped. Moreover, if potentials are too large,

C. Bielza et al. (Eds.): CAEPIA 2013, LNAI 8109, pp. 321–331, 2013.

they can be pruned and converted into smaller trees, thus leading to approximate and more efficient algorithms. The potential representation as a numerical tree was already used for LE in Bayesian networks [9]. This paper proposes the use of trees for LE of IDs in order to obtain an approximate LE of Bayesian decision problems. The experimental work compares the computing time, memory requirements and goodness of approximations using tables, numerical and binary trees.

The paper is organized as follows: Section 2 introduces some basic concepts about IDs and Lazy Evaluation; Section 3 describes key issues about numerical and binary trees and and how they are used during the lazy evaluation 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 direct acyclic graph used for representing and evaluating decision problems under uncertainty. An ID contains three types of nodes: *chance nodes* (representing random variables), *decision nodes* (mutually exclusive actions which the decision maker can control) and *utility nodes* (representing decision maker preferences). 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 according to when they are observed: $\mathcal{I}_0$ is the set of chance nodes observed before to the first decision, 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 temporal ordering: $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$.

The *universe* of the ID is $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D = \{X_1, \ldots, X_m\}$. Let us suppose that each variable X_i takes values on a finite set $\Omega_{X_i} = \{x_1, \ldots, x_{|\Omega_{X_i}|}\}$. If I is a set of indexes, we shall write $\mathbf{X}_I$ for the set of variables $\{X_i | i \in I\}$, defined on $\Omega_{\mathbf{X}_I} = \times_{i \in I} \Omega_{X_i}$. The elements of Ω_{X_I} are called configurations of $\mathbf{X}_I$ and will be represented as $\mathbf{x}_I$. 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). Let $\mathbf{X}_I$ be the set of all variables involved in a potential, then a *probability potential* denoted by ϕ is a mapping $\phi : \Omega_{\mathbf{X}_I} \to [0, 1]$. A *utility potential* denoted by ψ is a mapping $\psi : \Omega_{\mathbf{X}_I} \to \mathbb{R}$. The set of probability potentials is denoted by Φ while the set of utility potentials is denoted by Ψ.

When evaluating an ID, we must compute the best choice or *optimal policy* δ_i for each decision D_i, that is, a mapping $\delta_i : \Omega_{pa(D_i)} \to \Omega_{D_i}$ where $pa(D_i)$ are the informational predecessors or parents of D_i. To be well defined, informational predecessors of each decision in a ID must include previous decisions and informational predecessors of the previous decisions (*no-forgetting assumption*). The optimal policy maximizes the *expected utility* for each decision.

2.2 Lazy Evaluation

The principles of Lazy Evaluation (LE) were already used for making inference in BNs [10], so it can be adapted for evaluating IDs [5]. 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 and independence relations introduced by evidence. 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. 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. An example of a strong junction tree is shown in Fig. 1. The original IDs and details for building this strong junction tree are given in [2].

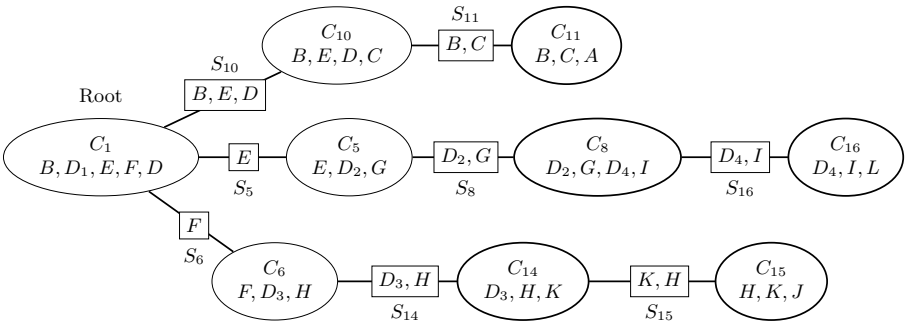


Fig. 1. Example of a Strong Junction Tree obtained from an ID

Propagation is performed by message-passing. Initially, each potential is associated to one clique containing all its variables. These potentials are not combined, so during propagation each clique and separator keeps two lists of potentials (one for probabilities and another for utilities). The message propagation starts by invoking the *Collect Message* algorithm in the root (Definition 1).

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

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

A clique can send the message to its parent (*Absorption*) if it has received all the messages from their children. Consider a clique C_j and its parent separator S. Absorption in C_j amounts to eliminating the variables of $C_j \backslash S$ from the list of probability and utility potentials Φ and Ψ associated with C_j and the separators

of $ch(C_j)$ and then associating the obtained potentials with S. Algorithms for marginalizing a variable from a list of potentials are given in [5]. For example, when absorption is invoked in clique C_{14}, the variable K is removed from the list of potentials in C_{14} and in the child separator S_{15}. The resulting potentials are stored in the parent separator S_{14}.

Definition 2 (Absorption). *Let C_j be a clique, S be the parent separator and be $ch(C_j)$ the set of child separators. If Absorb Evidence is invoked on C_j, then:*

1. *Let $\mathcal{R}_S = \Phi_{C_j} \cup \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} (\Phi^*_{S'} \cup \Psi^*_{S'})$.*
2. *Let $\mathbf{X} = \{X | X \in C_j, X \notin S\}$ 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 $\mathcal{R}_S$. Let Φ^*_S and Ψ^*_S the set of probability and utility potentials obtained. During this step, potential containing each variable are combined.*
5. *Associate Φ^*_S and Ψ^*_S to the parent separator S.*

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 (EU)* for the decision D_i. The values of the decision that maximizes the expected utility is the policy for D_i.

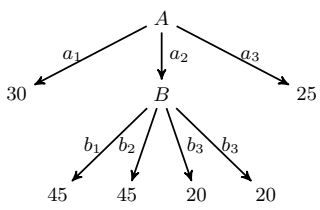
3 Lazy Evaluation with Trees

3.1 Numerical and Binary Trees

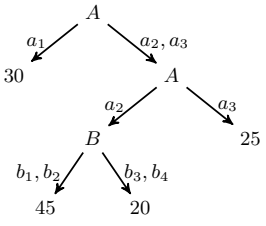
Traditionally, potentials have been represented using tables. However, several alternative representations have been proposed in order to reduce the storage size of the potentials and improve the efficiency of the evaluation algorithms. An example are trees (numerical [6] and binary [7]) that will be denoted by NT and BT respectively. Figure 2 shows three different representations for the same utility potential: (a) table, (b) a NT and (c) a BT.

$U(A,B)$	b_1	b_2	b_3	b_4
a_1	30	30	30	30
a_2	45	45	20	20
a_3	25	25	25	25

(a)



(b)



(c)

Fig. 2. Potential represented as a table (a), as a NT (b) and as BT (c)

The main advantage of trees is that they allow the specification of *context-specific independencies* [8]. For example, the table in Fig. 2 requires 12 values for representing the potential. When $A = a_1$, the potential will always take the value 30, regardless of the value of B. Similarly, when $A = a_3$, the utility value is also independent of B. Therefore, a NT representing the same utility potential requires 8 nodes. Moreover, the representation of the potential as a BT requires only 7 nodes since leaves labelled with 45 and 20 can be collapsed in two leaves. That is, BTs allow representing context-specific independencies which are finer-grained than those represented using NTs. These finer-grained independencies are also called *contextual-weak independencies* [11]. If trees are too large, they can be pruned and converted into smaller trees, thus leading to approximate algorithms. When a tree is pruned, leaves with a similar value are represented with a single leaf labelled with their mean. The prune is controlled with a threshold $\varepsilon \geq 0$. A low ε value will produce large trees with a low error, while a high ε value will produce small trees with a big error. When $\varepsilon = 0$, the exact prune is performed. That is, only identical values are grouped. When building a tree, variables are sorted in a way that the most informative variables must be situated in the highest nodes of the tree. This operation will reduce the error obtained when pruning a tree. More details for the building and pruning processes are given in [6,7].

3.2 Evaluation with Trees

LE algorithm for IDs can be easily adapted for working with trees (NTs or BTs). Evaluation algorithms for IDs require five operations with potentials: *restriction, combination*, *sum-marginalization*, *max-marginalization* and *division*. These operations must be implemented for operating with trees instead of tables. The general scheme of LE adapted for working with trees is shown below.

1. Initialization phase:
   (a) Build initial trees:
      – for each $\phi \in \Phi$ obtain $\mathcal{T}_\phi$.
      – for each $\psi \in \Psi$ obtain $\mathcal{T}_\psi$.
   (b) Prune all trees
   (c) Build the Strong Junction Tree from the ID.
2. Propagation phase:
   (a) Associate each potential (trees) in $\Phi \cup \Psi$ to only one clique containing all the variables in its domain.
   (b) Call the method *CollectMessage* in the root node. After removing each variable during the propagation, trees can be pruned again.

The main difference is that now it requires an initialization phase where initial trees are built from tables (1.a) and pruned in order to obtain smaller trees (1.b). After that, the Strong Junction Tree is built. Besides pruning initial potentials, an additional pruning can be performed after removing each variable during propagation. That is, after step 4 in Definition 2.

4 Experimentation

For testing purposes, two different families of IDs are used. First, a real world ID used for the treatment of gastric NHL disease [12] with 3 decisions, 1 utility node and 17 chance nodes. Second, a set of 20 randomly generated IDs with 2 decisions, 1 utility node, and a random number (between 6 and 17) of chance nodes. All IDs are evaluated using different variations of the LE algorithm: using tables (LET); using NTs and BTs pruning only initial potentials present in the ID (LETNT and LETBT); and using NTs and BTs using the pruning operation after removing each variable (LETNTPR and LETBTPR). The threshold used for pruning is ranged in the interval $[0, 1.0]$. All the algorithms are implemented in Java with the Elvira Software[1].

The graphics included in Fig. 3 show the storage requirement for storing all the potentials during the NHL ID evaluation, that is, potentials associated to the cliques and intermediate potentials used to compute the messages. For space restrictions only results using the thresholds values 0 and 0.05 are shown. The vertical axis indicate the storage size using a logarithmic scale: number of values in the tables and number of nodes in the trees. The horizontal axis indicate the evaluation stage when the storage size is measured. These measurements are performed after combining potentials containing a variable to be removed, and after pruning the resulting potentials of marginalization. It can be observed that less space is needed for representing the potentials as BTs than using NTs or tables. When the exact prune is performed (threshold = 0), differences are not very significant, which become even less significant during the latest stages. This is due to the combination of potentials during evaluation that makes the effect of initial prune disappear. Moreover, at some stages of the evaluation tables require less space for storing the potentials than trees, which need an additional space for storing the internal nodes. However, with a higher threshold value, the reduction of the size is more noticeable since the additional storage requirements are compensated.

The reduction of the potential sizes should lead to more efficient algorithms: operations with smaller potentials should be faster. In Figure 4 it is shown the computation time for evaluating the NHL ID with different threshold values. It can be observed that pruning after removing each variable (LENTPR and LEBTPR) is not efficient when the exact prune is performed: the overhead introduced by pruning and sorting trees eats all the gain obtained with the smaller potentials. The evaluation time with BTs (LEBT) is faster than the evaluation with NTs (LENT). However, with higher threshold values, all variants of the evaluation algorithm with trees obtain similar results. By contrast, worst results are obtained using tables (LET).

Figure 5 includes six different graphics representing *root mean squared error* calculated over all the configurations in the potential (horizontal axis) against tree size (vertical axis) for the expected utilities corresponding to decisions 0, 1, and 2 for the NHL ID. Each point in the graphics corresponds to a different

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

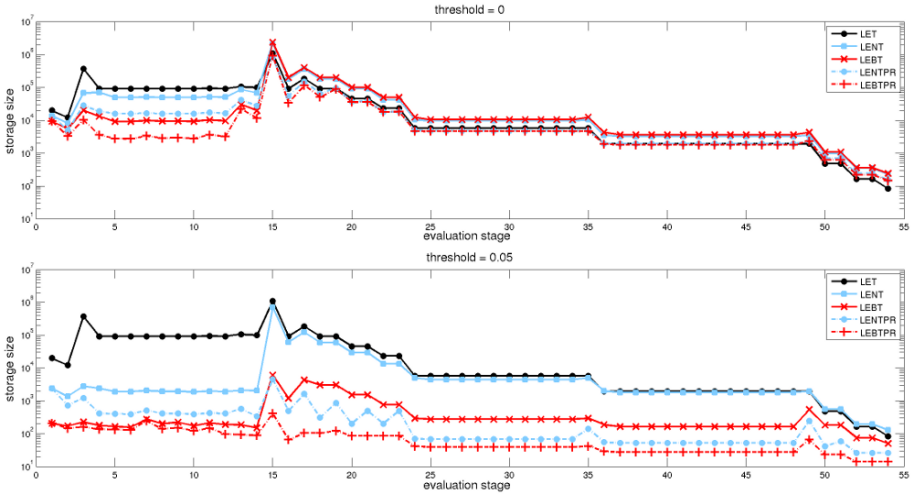


Fig. 3. Size of all potentials stored in memory during the NHL ID evaluation comparing tables, NTs and BTs with different threshold values

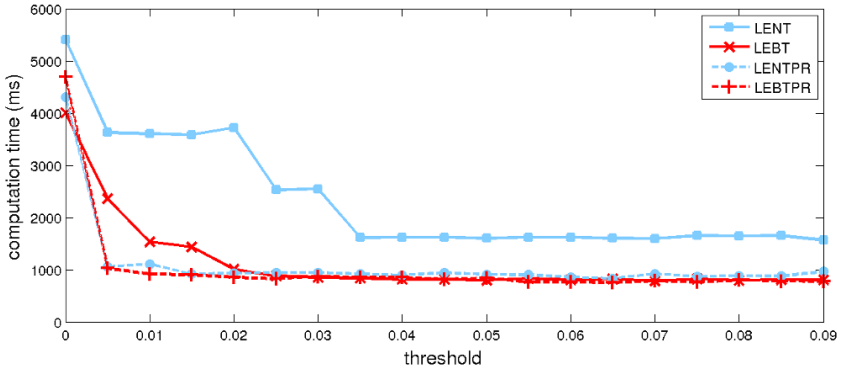


Fig. 4. Evaluation time for the NHL ID with different threshold values. The evaluation time using tables (LET) is approximately 15000 ms. and it is independent from the threshold value.

evaluation with a certain threshold value. Analyzing graphics corresponding to evaluations with a single initial prune (left column), it can be seen that in most of the evaluations the same error level is achieved using BTs of smaller size than the corresponding NTs. However, if the exact prune is performed (RMSE=0), the size is smaller using NTs. By contrast, if trees are also pruned after removing each variable (right column), NTs offer better approximate solutions than BTs. These conclusions can also be obtained with Table 1, which shows the hyper-volume indicators obtained from the evaluation of the NHL ID. The hyper-volume [13] is an unary indicator that measures the area of the dominated portion of the space. It is defined in the interval $[0, 1]$, being 1 the optimal solution and 0 the worst. The hyper-volume indicator can be used to evaluate a

set of different approximations of a potential if preferences about size and error are unknown. Each row corresponds to one of the expected utilities, whereas each column corresponds to the evaluation scheme. In case of a single prune, the hyper-volume values (H_{LEBT}) obtained using BTs are always larger (better) than the corresponding hyper-volume values (H_{LENT}) using NTs. In case of pruning after removing each variable (H_{LENTPR} and H_{LEBTPR}), the hyper-volume values obtained using BTs are lower for Decisions 2 and 1.

Table 1. Hyper-volume values of the approximate expected utilities

	H_{LENT}	H_{LEBT}	H_{LENTPR}	H_{LEBTPR}
Decision 2	0.7259	**0.7846**	**0.9756**	0.9673
Decision 1	0.6981	**0.7419**	**0.9758**	0.9645
Decision 0	0.2930	**0.5584**	0.9592	**0.9626**

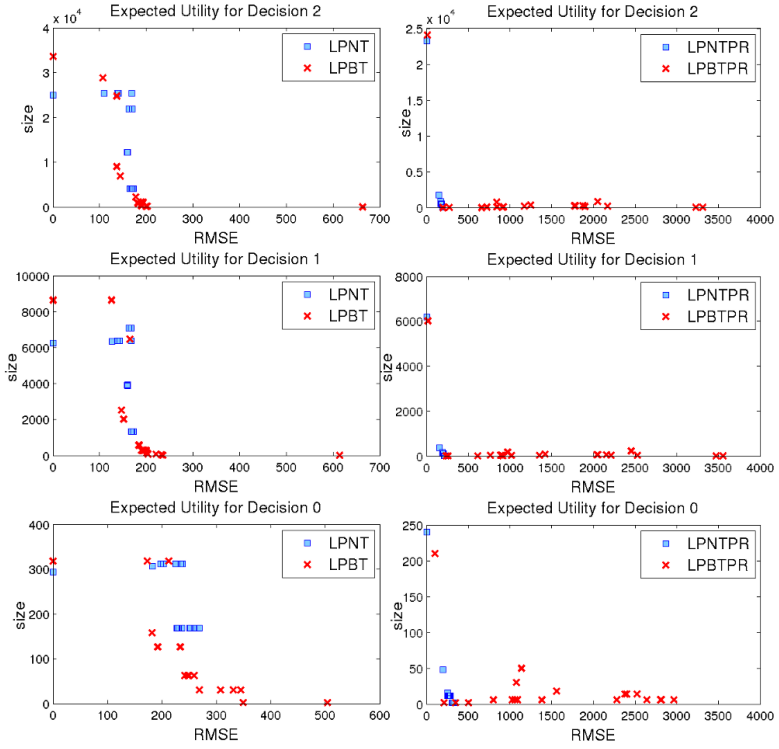


Fig. 5. Results of the expected utilities approximation for the NHL ID performing a single prune at the beginning (left) and pruning after removing each variable (right)

In order to get more solid conclusions in relation to the error obtained when approximating potentials with trees, a similar procedure can be done with random IDs. The hyper-volume values for the these IDs are shown in Table 2. Each

row corresponds to a different random ID, whereas the columns indicate the evaluation scheme used and expected utility analyzed. Obtaining a 0 hyper-volume value means that the size of the expected utility is independent of threshold value used for pruning. This happen because the prune operation may reduce part of the trees which must be completely developed during posterior combinations. It can also be observed that the hyper-volume value H_{LEBT} is usually larger than the corresponding H_{LENT} value. By contrast there are less differences if H_{LEBTPR} and H_{LENTPR} are compared. Moreover, for Decision 0 H_{LENTPR} is usually larger than H_{LEBTPR}. To prove these hypothesis four Wilcoxon signed-rank are performed comparing H_{LENT} against H_{LEBT} and H_{LENTPR} againts H_{LEBTPR} for each decision. The results of these test are shown in Table 3. For each test, it shows the *p-value*, the percentage of IDs where the use of BTs give better results and whether the null hypothesis was rejected (NTs and BTs are not equal) with a significance level of 10%. For the comparison H_{LENT} and H_{LEBT} the hypothesis was rejected, then better approximate solutions are obtained using BTs than using NTs if only a single initial prune is performed. In case of pruning trees after removing each variable, the null hypothesis is not rejected for Decision 1. In case of Decision 0, the null hypothesis is rejected. Taking into account that H_{LENTPR} is usually larger than H_{LEBTPR}, it can be deduced that NTs offer better approximate solutions than BTs if trees are pruned after removing each variable.

Table 2. Hyper-volume values of utility trees comparing NTs and BTs

Decision 1		Decision 2		Decision 1		Decision 2	
H_{LENT}	H_{LEBT}	H_{LENT}	H_{LEBT}	H_{LENTPR}	H_{LEBTPR}	H_{LENTPR}	H_{LEBTPR}
0.28	**0.995**	0.296	**1**	**0.991**	0.989	**0.999**	0.999
0.745	0.671	0.177	**0.827**	**0.703**	0.665	**0.901**	0.844
0.26	**0.922**	0.261	**0.944**	0.981	**0.985**	**0.942**	0.937
0.447	0.393	0.613	**0.65**	0.364	**0.384**	**0.438**	0.277
0.529	0.341	**0.333**	0.163	**0.431**	0.341	0.548	**0.641**
0.671	0.635	0.369	**0.812**	0.606	**0.635**	**0.863**	0.809
0.0712	**0.168**	0.521	**0.747**	0.0695	**0.202**	0	**0.312**
0.8	**0.979**	0.605	**0.95**	0.95	**0.972**	**0.941**	0.937
0.891	**0.997**	0.395	**0.997**	0.99	**0.997**	**0.996**	0.995
0.443	**0.998**	0.334	**0.99**	0.994	**0.999**	0.926	**0.987**
0.178	**0.968**	0.283	**0.997**	**0.987**	0.972	**1**	0.999
0.259	0.259	0	**0**	0.247	**0.248**	**0.338**	0
0.577	0.571	0.445	**0.66**	**0.562**	0.559	**0.686**	0.631
0.456	0.378	0	**0.548**	**0.449**	0.376	**0.882**	0.856
0.319	**0.975**	0.253	**0.972**	**0.991**	0.968	**0.992**	0.96
0.533	0.524	0.34	**0.96**	0.507	**0.52**	**0.944**	0.94
0.278	**0.992**	0.295	**1**	**0.979**	0.975	0.988	**0.997**
0.413	0.388	0.411	**0.898**	0.334	**0.416**	**0.975**	0.892
0.686	0.656	0	**0.856**	**0.679**	0.648	**0.817**	0.816
0.746	**0.976**	0.45	**0.946**	**0.986**	0.979	**0.993**	0.982

Table 3. Results for the Willcoxon sign-rank test with the random ID

	H_{LENT} vs H_{LEBT}			H_{LENTPR} vs H_{LEBTPR}		
	p-value	% BT wins	rejected	p-value	% BT wins	rejected
Decision 1	0.0793	50	yes	0.94	50	no
Decision 0	$1.82 \cdot 10^{-4}$	95	yes	0.0859	20	yes

5 Conclusions and Future Work

This paper proposes to combine the use of trees for representing the potentials and Lazy Evaluation to evaluate IDs which allows to perform an approximate Lazy Evaluation of decision problems. It is explained how trees are used during the evaluation of IDs and compared the computing time, memory requirements and goodness of approximations using tables, NTs and BTs. The experiments shows that less space is used for storing potentials as a BT than as NT or as a table. Moreover, if potentials are approximated the reduction is more noticeable. In general, the evaluation with trees is faster than using tables. Concerning to the error level achieved, the experiments showed that BTs offer better approximate solutions than NTs if only initial potentials are pruned. If trees are pruned after removing each variable during evaluation NTs offer better results.

As regards future directions of research, we would like to study alternative heuristics or scores for triangulating the graph. The heuristics used until now consider that potentials are represented as tables, not as trees. Finally, another direction of research could be the integration of restrictions with binary trees. This would allow the treatment of asymmetric decision problems [14], where the set of legitimate states of variables may vary depending on different states of other variables.

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) and the FPI scholarship programme (BES-2011-050604). The authors have been also partially supported by " Junta de Andalucía" under projects TIC-06016 and P08-TIC-03717.

References

1. Howard, R.A., Matheson, J.E.: 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 Tenth International Conference on Uncertainty in Artificial Intelligence, pp. 367–373. Morgan Kaufmann Publishers Inc. (1994)
3. Jensen, F., Nielsen, T.: Bayesian networks and decision graphs. Springer (2007)
4. Shachter, R.: Evaluating influence diagrams. Operations Research, 871–882 (1986)
5. 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)

6. Gómez, M., Cano, A.: Applying numerical trees to evaluate asymmetric decision problems. In: ESCQARU, pp. 196–207 (2003)
7. Cabañas, R., Gómez, M., Cano, A.: Approximate inference in influence diagrams using binary trees. In: Proceedings of the 6th European Workshop on Probabilistic Graphical Models, PGM 2012, pp. 43–50 (2011)
8. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Proceedings of the 12th International Conference on Uncertainty in AI, pp. 115–123. Morgan Kaufmann Publishers Inc. (1996)
9. Cano, A., Moral, S., Salmerón, A.: Penniless propagation in join trees. International Journal of Intelligent Systems 15(11), 1027–1059 (2000)
10. Madsen, A., Jensen, F.: Lazy propagation: a junction tree inference algorithm based on lazy evaluation. Artificial Intelligence 113(1-2), 203–245 (1999)
11. Wong, S., Butz, C.: Contextual weak independence in Bayesian networks. In: Proceedings of the 15th Conference on Uncertainty in AI, pp. 670–679. Morgan Kaufmann Publishers Inc. (1999)
12. Lucas, P., Taal, B.: Computer-based decision support in the management of primary gastric non-hodgkin lymphoma. UU-CS (1998-33) (1998)
13. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)
14. Bielza, C., Shenoy, P.: A comparison of graphical techniques for asymmetric decision problems. Management Science 45(11), 1552–1569 (1999)