Evaluating Asymmetric Decision Problems with Binary Constraint Trees

Rafael Cabañas, Manuel Gómez-Olmedo, and Andrés Cano

Dept. Computer Science and Artificial Intelligence
University of Granada, CITIC-UGR, Spain
{rcabanas,mgomez,acu}@decsai.ugr.es

Abstract. This paper proposes the use of *binary trees* in order to represent and evaluate asymmetric decision problems with Influence Diagrams (IDs). Constraint rules are used to represent the asymmetries between the variables of the ID. These rules and the potentials involved in IDs will be represented using binary trees. The application of these rules can reduce the size of the potentials of the ID. As a consequence the efficiency of the inference algorithms will be improved.

Keywords: Influence diagrams, asymmetric decision problems, binary trees, probability trees.

1 Introduction

Influence Diagrams (IDs) [11] are a tool to represent and solve decision problems under uncertainty. Their main advantage is that they can encode the independence relations between variables allowing a compact representation. However, they have weaknesses: decision problems are usually asymmetric in the sense the set of legitimate states of variables may vary depending on different states of other variables [1]. To be represented as an ID, an asymmetric decision problem must be symmetrized and a considerable amount of unnecessary computation may be involved. Several approaches have been made to solve this drawback. Call and Miller [4], Fung and Shachter [18], Smith et al. [21], Qi et al. [17], Covaliu and Oliver [7], Shenoy [20], Nielsen and Jensen [15], Demirer and Shenoy [8], Díez and Luque [9] have proposed modifications to the IDs framework in order to deal with asymmetries.

In this paper we propose representing the qualitative information about the problem (constraints, due to asymmetries) using *binary trees* (BTs). Constraints can be easily applied to potentials reducing the number of scenarios to consider. Moreover, if BTs are too large, they can be pruned and converted into smaller trees, thus leading to approximate algorithms. We compare BTs with a previous approach for representing constraints, *numerical trees* (NTs), and show that more efficient algorithms are obtained.

The paper is organized as follows: Section 2 introduces some basic concepts about IDs and trees; Section 3 describes key issues about asymmetries and how they can be represented using BTs; Section 4 describes the evaluation algorithm

L.C. van der Gaag (Ed.): ECSQARU 2013, LNAI 7958, pp. 85–96, 2013.

adapted for working with constraints; Section 5 includes the experimental work and results; finally Section 6 details our conclusions and lines for future work.

2 Preliminaries

2.1 Influence Diagrams

An ID [11] is a generalization of a Bayesian network (BN) [16] used for representing and solving 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). Fig. 1 shows an example of an ID that represents the Car Buyer problem [17].

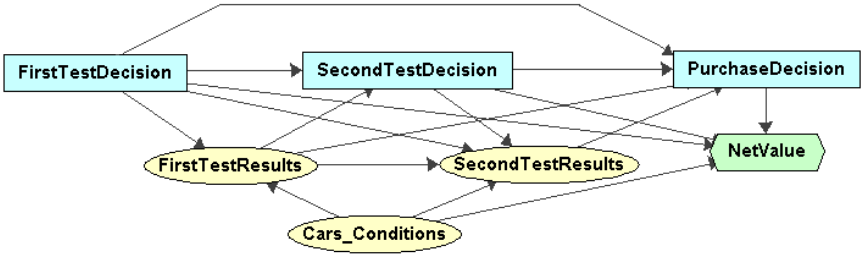

Fig. 1. Example of an ID representing the Car Buyer problem

The set of chance nodes is denoted by $\mathcal{U}_C$, the set of decision nodes is denoted by $\mathcal{U}_D$, and the set of utility nodes is denoted by $\mathcal{U}_V$. 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 prior to any decision, and $\mathcal{I}_i$ is the set of chance nodes observed after D_i is taken and before deciding about D_{i+1}. Finally, $\mathcal{I}_n$ is the set of chance nodes never observed or observed after the last decision. That is, there is a partial temporal ordering: $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$. For example, the temporal ordering of the ID in Fig. 1 is $FirstTestDecision \prec FirstTestResults \prec SecondTestDecision \prec SecondTestResults \prec PurchaseDecision \prec Car\_Conditions$.

Let us suppose that each variable X_i of the ID 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 $\mathbf{x}_I$. Parents or direct predecessors of a variable X_i are denoted $pa(X_i)$.

In an ID, each chance node X_i has a conditional probability distribution $P(X_i|pa(X_i))$ attached, where $pa(X_i)$ are the parents of X_i. In the same way, each utility node V_i has a utility function $U(pa(V_i))$ attached. In general, we will talk about potentials (probability distributions are normalized potentials). Let $\mathbf{X}_I$ be the set of all variables involved in a potential, then a *probability potential*

denoted by ϕ is a mapping $\phi : \Omega_{\mathbf{X}_I} \rightarrow [0, 1]$. A *utility potential* denoted by ψ is a mapping $\psi : \Omega_{\mathbf{X}_I} \rightarrow \mathbb{R}$.

When evaluating an ID, it must be computed the best choice or *optimal policy* δ_i for each decision D_i, that is a mapping $\delta_i : \Omega_{pa(D_i)} \rightarrow \Omega_{D_i}$. The optimal policy maximizes the *expected utility* for the decision. A strategy is an ordered set of policies $\Delta = \{\delta_1, \ldots, \delta_n\}$ including a policy for each decision. An optimal strategy $\widehat{\Delta}$ returns the optimal choice the decision maker should take for each decision.

2.2 Numerical and Binary Trees

Traditionally, potentials involved in an ID have been represented using tables. An alternative representation are trees (numerical and binary)[6, 3] that will be denoted NT and BT respectively. Each internal node of the tree is labelled with a variable (random variable or decision). We use L_t to denote the *label of a node* t. Each leaf node is labelled with a number (a probability or a utility value). In a NT, each internal node has an outgoing arc for each state of the variable associated with that node. The difference between NTs and BTs is that internal nodes in BTs always have two children. As a consequence, outgoing arcs in a BT can be labelled with more than one state. We denote by $L_{lb(t)}$ and $L_{rb(t)}$ the left and right labels of a node t respectively.

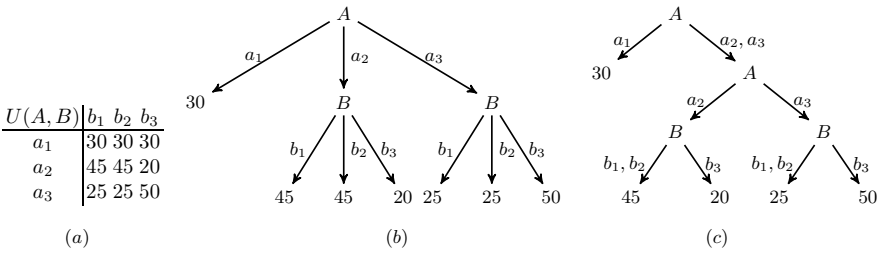

Fig. 2. Utility potential represented as (a) a table, (b) a NT and (c) a BT

Fig. 2 shows three different representations for the same utility potential: (a) a table , (b) a NT, and (c) a BT. The use of trees offer the possibility of taking advantage of *context-specific independencies* [2]. For example, when $A = a_1$, the potential will always take the value 30, regardless of the value of B. Therefore, less space is needed for representing it as a tree. Besides, BT can capture finer-grained independencies: when $A = a_2$ and $B \in \{b_1, b_2\}$, the potential will always be 45.

In a previous work [3], a comparison of the evaluation of IDs using NTs and BTs was performed using the *Variable Elimination* algorithm. The experiments showed that BTs offer better approximate solutions than NTs. The same error level will be achieved using a BT of smaller size than the corresponding NT.

2.3 Building and Approximating Trees

When a BT is built, variables are sorted in such a way that the most informative variables must be situated at the highest nodes in the tree. BTs are built from tables using a top-down approach, choosing at each step a variable and two partitions of its states that maximizes the information gain.

Definition 1 (Information Gain). *Let ϕ be the potential to be represented as a tree $\mathcal{BT}_j$ and $\mathcal{BT}_j(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i})$ the tree resulting of expanding the leaf node t with the candidate variable X_i and a partition of its available states into sets $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$. Let $D(\phi, \mathcal{BT}_j)$ be the distance between a potential and a tree. The information gain can be defined as:*

$$I(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i}) = D(\phi, \mathcal{BT}_j) - D(\phi, \mathcal{BT}_j(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i})) \quad (1)$$

Kullback Leibler divergence and Euclidean distance are the distance measure used for computing the information gain with probability and utility trees respectively. If the size of a BT needs to be reduced, it can be pruned in order to get a new BT which approximates the potential. Pruning a BT consists in replacing a *terminal tree* (a node whose children are all leaves) by the average value of its leaves. If the information gain between a terminal tree and the resulting pruned tree is lower than a threshold ε, the tree is pruned. The decision of pruning a terminal tree is independent of the decision of pruning other terminal trees. Variables in trees generated during evaluation can be sorted again using the same procedure than for building: the most informative variables may not be the same than in initial trees. This process allows obtaining better approximations, but it can be a very time consuming task since it implies building again the tree. Further details about building and pruning BTs are given in [5, 3].

3 Asymmetries and Constraints

The drawback of using IDs to model asymmetric decision problem is well known. An asymmetric decision problem must be symmetrized to be represented as an ID. Therefore, a considerable amount of unnecessary computation may be involved during the evaluation. It is sometimes possible to identify the source of asymmetry and represent it with relations between variables. In our solution we try to keep qualitative (constraints due to asymmetries) and quantitative (potentials) knowledge separate, merely because qualitative knowledge may affect several distributions, with some of them not being present in the model (i.e. distributions managed during the evaluation process and derived from the initial ones). On the other hand, we attempt to store both kinds of knowledge in similar structures, making their joint application easier. In order to represent the qualitative knowledge about a decision problem, we therefore propose the use of *constraint rules*.

A *constraint rule* is an expression *antecedent* $\Rightarrow$ *consequent*. An *atomic sentence* is a pair (variable, set of values): $X_i \in \{x_i, \ldots x_j\}$. Atomic sentences can

be connected with logical operators to form *logical sentences*. Valid logical operators are $\wedge(and)$, $\vee(or)$ and $\neg(not)$. For constraint rules, both antecedents and consequents are expressed using logical sentences. For example, let us suppose that X, Y and Z take values receptively on the sets $\Omega_X = \{x_1, x_2, x_3\}$, $\Omega_Y = \{y_1, y_2\}$, $\Omega_Z = \{z_1, z_2\}$, then the constraint rule:

$$X \in \{x_1, x_3\} \wedge Y \in \{y_2\} \Rightarrow Z \in \{z_2\} \tag{2}$$

states that if X is equal to x_1 or x_3 and Y is equal to y_2 then the variable Z will always take the value z_2. Considering this constraint rule and the conditional probability $P(Z|X, Y)$, we can state that:

$$P(Z = z_1|X = x_1, Y = y_2) = P(Z = z_1|X = x_3, Y = y_2) = 0$$

The configurations $\{x_1, y_2, z_1\}$ and $\{x_3, y_2, z_1\}$ are impossible scenarios that must not be considered for computations. An atomic sentence could have an empty set of values for the consequent. For example, the constraint rule

$$X \in \{x_1, x_3\} \wedge Y \in \{y_2\} \Rightarrow Z \in \{\} \tag{3}$$

means that $\{x_1, y_2, z_1\}$, $\{x_1, y_2, z_2\}$, $\{x_3, y_2, z_1\}$ and $\{x_3, y_2, z_2\}$ are impossible scenarios.

To decide if a constraint rule for the variables $\mathbf{X}_J$ is applicable to a potential ϕ (probability or utility) for the variables $\mathbf{X}_I$, we have to check the *applicability* of the constraint rule. The applicability of the complete constraint rule depends on the logical operator involved with the atomic sentences of the rule. We use the following definitions to decide if the constraint rule is applicable. We say that an atomic sentence in a constraint rule for $\mathbf{X}_J$ is applicable to a potential $\mathbf{X}_J$ for $\mathbf{X}_I$ if the variable X_i of the atomic sentence is in $\mathbf{X}_J \cap \mathbf{X}_I$. The negation of a sentence is applicable if and only if the sentence itself is applicable. A conjunction is applicable if and only if the two conjuncts are applicable. A disjunction is applicable if and only if at least one of the disjuncts is applicable. With these definitions, the constraint rule is applicable if and only if both the antecedent and the consequent are applicable.

Sometimes the constraint rules are not applicable to any distribution of the model. For example, we could have the following situation: a constraint links the values of two decision nodes , but there is no distribution containing both variables. However, during the evaluation process the value node will depend on both of them and this will be the moment to activate the constraint.

The use of constraint rules have several advantages. First of all, they make the elicitation process easier (reducing the number of scenarios and therefore the number of parameters to assess); secondly, they help to make both qualitative and quantitative knowledge consistent; and thirdly, they clearly state invalid scenarios, making the contingent nature of the decision problem clear.

3.1 Binary Constraint Trees

In a previous work, it was proposed the use of NTs for representing constraint rules and applying them during the ID evaluation [10]. In the present paper, BTs are proposed for representing constraint rules: *binary constraint trees* (BCTs). BT can capture finer-grained independencies than those captured using NT. Potentials and constraints need less space to be represented as a BT than as NT. As a consequence, more efficient evaluation algorithms will be obtained.

Leaf nodes in a BCT contain the values 0 or 1. If $\mathcal{T}^c$ is a BCT for a constraint rule with variables $\mathbf{X}_J$, then a value of 0 in a leaf node t_n, means that the configuration of its ancestor variables corresponds to an impossible scenario in the ID. A value equal to 1 means that, taking into account only this constraint tree, the configuration is possible (it can be impossible according to another constraint).

Constraint rules and trees are useful when evaluating the ID in order to reduce the size of the potentials (probability trees and utility trees). This reduction causes that the complexity of operations (combination and marginalization) is also reduced. For applying a constraint tree $\mathcal{T}^c$ to a tree $\mathcal{T}_\phi$ from a potential ϕ, non-common variables are removed from $\mathcal{T}^c$ using max-marginalization. Then, the resulting constraint tree is combined with $\mathcal{T}_\phi$. Fig. 3 shows an example of the application of the constraint tree $\mathcal{T}^c(X, Y, X)$ obtained from the rule in Equation 3 to a utility tree $\psi(X, Y, Z)$. In the constraint tree variable Z is not present since it has previously been pruned.

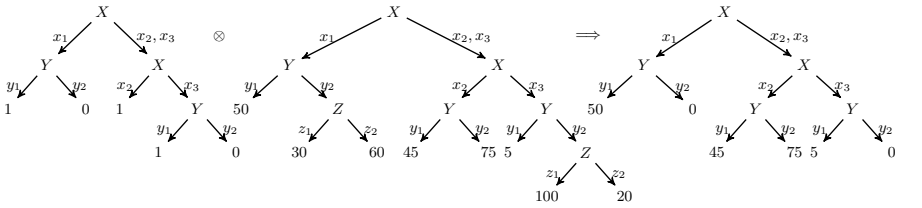

Fig. 3. Application of the constraint tree obtained from the rule in Equation 3 to a utility tree

4 Evaluating Influence Diagrams

4.1 Variable Elimination with Constraint Trees

This section shows how BCTs can be applied to evaluate IDs. In particular we have decided to work with the *Variable Elimination* algorithm (VE) and can be used for solving BNs [22] and IDs [12]. This method uses the temporal order between the decision nodes to partition the whole set of nodes according to when they are observed. Once this order has been established, the algorithm eliminates all the variables one by one, with two operations: sum-marginalization and max-marginalization. The method adapted for working with constraints is shown below.

1. Initialization phase
   (a) Build initial trees: for each potential in $\Phi \cup \Psi$ obtain a tree $\mathcal{T}_\phi$.
   (b) Apply constraints: $\forall \mathcal{T}^c$, if $\mathcal{T}^c$ is applicable to $\mathcal{T}_\phi$ then: $\mathcal{T}'_\phi = \mathcal{T}^c \otimes \mathcal{T}_\phi$, else $\mathcal{T}'_\phi = \mathcal{T}_\phi$.
   (c) Sort and prune all trees
2. While there are variables to remove
   (a) Decide next variable to remove (X) and combine all potentials (trees) containing X: Φ_X, Ψ_X. Remove X using sum-marginalization (chance nodes) or max-marginalization (decision nodes). New trees are obtained as result: $\mathcal{T}_\phi$ and $\mathcal{T}_\psi$
   (b) Apply constraints to resulting trees $\mathcal{T}_\phi$ and $\mathcal{T}_\psi$, as in 1.b.
   (c) Sort and prune the resulting trees (optional).

It can be seen in the algorithm that the global structure of VE is not changed: the only difference is that it is performed a pre-processing at the beginning and a post processing after removing each variable which modify the potentials (their size is reduced). Constraints are applied to initial potentials (1.b) and to potentials obtained from the removal of variables (2.b). This application reduces the size of potentials, but a greatest reduction can be achieved if trees are sorted and pruned (see Section 2.3). The pruning process is a time consuming task. For that reason, if the features of the problem require to evaluate the ID in a short period of time, it must only be perform during the initialization phase. However, if the reduction in the storage size is more important, it can also be performed after removing each variable.

4.2 Modified Operations

The application of constraints is performed by combining the BCTs and potential trees when needed. Combine operation was described in a previous work about using BTs for BNs inference [5]. However, the combination does not reduce the size of potentials after applying a constraint. In fact, bigger potentials can be obtained. For that reason, it should be necessary to prune the trees after applying the constraints. The inconvenient of performing the pruning process is that evaluation time can be increased.

In order to avoid pruning the trees, here we propose to modify the combine operation (see Algorithm 1). When combining two trees, it is checked if one of them is a leaf node with the value 1 or 0. In case of a 0, the algorithm will return a leaf node with the value 0 (step 1). On the other hand, if it is the value 1, it will return the other tree (steps 1 and 1). This operation requires restricting a $\mathcal{BT}$ to a set of states L of a variable X_i, denoted $\mathcal{BT}^{R(X_i,L)}$.

It must be noticed that, in order get the benefits from this new version of the operator combine, the constraint tree must be the first input argument $\mathcal{BT}_1$. The combination process is illustrated in Fig. 4. It shows the differences in the process between combining two trees with the modifications (bottom) and without them (top). The same considerations can be made for the division operation, which is used after the removal of each variable.

Input : $t1$ and $t2$ (root nodes of $\mathcal{BT}_1$ and $\mathcal{BT}_2$)
Output: The root of $\mathcal{BT} = \mathcal{BT}_1 \otimes \mathcal{BT}_2$

Build a new node t
if *($t1$ is a leaf node* ***and*** $L_{t1} == 0$*)* ***or*** *($t2$ is a leaf node* ***and*** $L_{t1} == 0$*)* **then**
  Set $L_t = 0$ the label of t
else if $t1$ *is a leaf node* ***and*** $L_{t1} == 1$ **then**
  Set $t = t2$
else if $t2$ *is a leaf node* ***and*** $L_{t2} == 1$ **then**
  Set $t = t1$
else if $t1$ *is a leaf node* **then**
  if $t2$ *is a leaf node* **then**
    $L_t = L_{t1} \cdot L_{t2}$
  else
    Set $L_t = L_{t2}$ the label of t
    Set $L_{lb(t)} = L_{lb(t2)}$ and $L_{rb(t)} = L_{rb(t2)}$ labels of the two branches of t
    Set Combine($t1$,$t2_l$) the left child of t
    Set Combine($t1$,$t2_r$) the right child of t
else
  Suppose X_j is the variable labelling node t_1
  Set $L_t = L_{t1}$ the label of t
  Set $L_{lb(t)} = L_{lb(t1)}$ and $L_{rb(t)} = L_{rb(t1)}$ labels of the two branches of t
  Set Combine($t1_l$,$\mathcal{BT}_2^{R(X_j, L_{lb(t1)})}$) the left child of t
  Set Combine($t1_r$,$\mathcal{BT}_2^{R(X_j, L_{rb(t1)})}$) the right child of t
return t

Algorithm 1. Modified combine operation

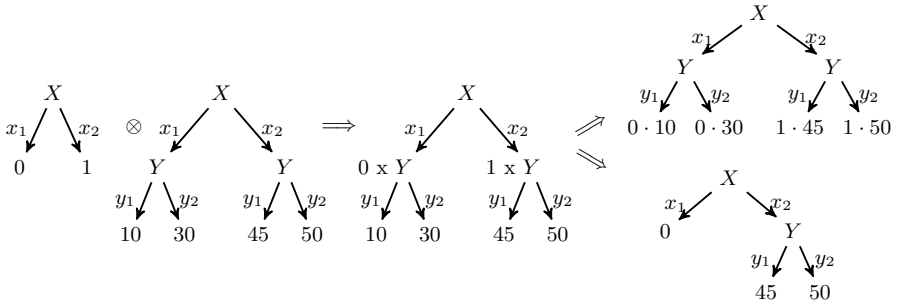

Fig. 4. Combination process: It shows the differences in the process between combining two trees with the modifications (bottom) and without them (top).

5 Experimentation

For testing purposes, two different IDs were used. First, a real world ID used for the treatment of gastric NHL disease [13] with 3 decisions, 1 utility node and 17 chance nodes. This ID contains two constraint rules between its decisions:

$$HelicobacterTreatment = \{NO\} \Longrightarrow Surgery = \{NONE\}$$
$$HelicobacterTreatment = \{NO\} \Longrightarrow CT\_RT\_Schedule = \{NONE\}$$

The second ID used represents the Car Buyer problem [17], which is shown in Fig. 1. This ID has 3 decisions, 1 utility, 3 chance nodes and the following constraint rules:

$$FirstTestDecision = \{NO\} \Longleftrightarrow FirstTestResult = \{NONE\}$$
$$FirstTestResult = \{defects2\} \Longrightarrow FirstTestDecision = \{FuelElectrical\}$$
$$SecondTestDecision = \{NO\} \Longleftrightarrow SecondTestResult = \{NONE\}$$

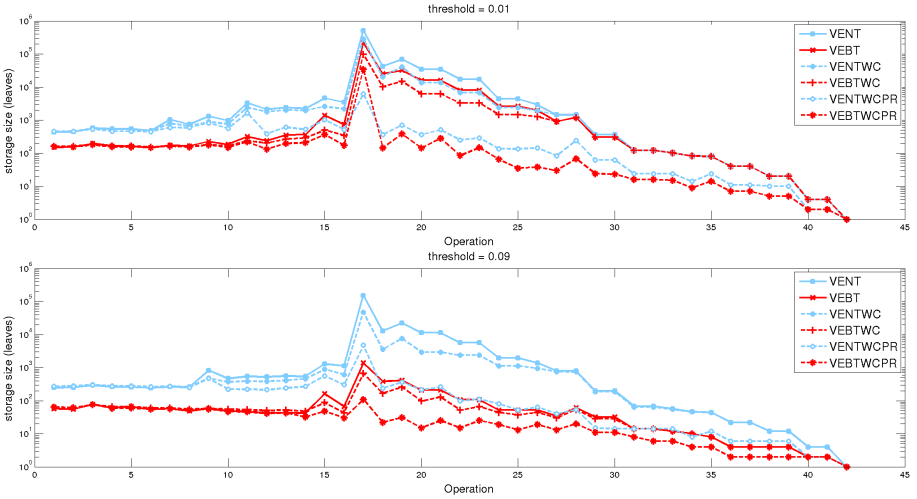

Fig. 5. Size of all potentials stored in memory during the NHL ID evaluation

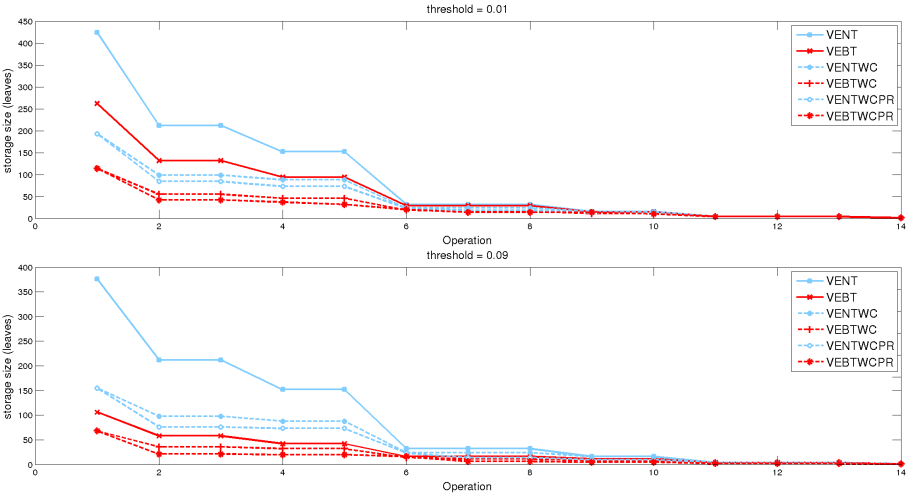

Fig. 6. Size of all potentials stored in memory during the Car Buyer ID evaluation

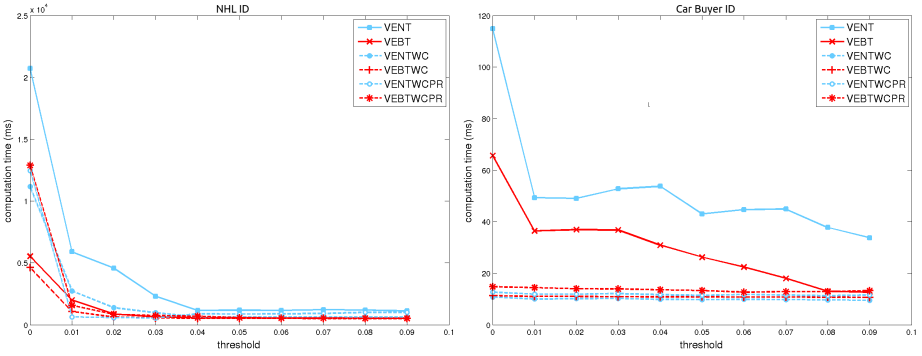

Fig. 7. Computation time for the evaluation of the diagrams NHL and Car Buyer

Both IDs were evaluated using different variations of the VE algorithm: using NTs and BTs without constraints (VENT and VEBT); using NTs and BTs with constraints (VENTWC and VEBTWC); and using NTs and BTs with constraints and using the sort and pruning operation after removing each variable (VENTWCPR and VEBTWCPR); The modifications in combination operation shown in Section 4.2 were only employed for the evaluation with constraints. The ε threshold used for pruning was ranged in the interval $[0, 0.1]$. All the algorithms were implemented in Java with the Elvira Software[1]. The tests were run on a Intel Xeon Processor E3510 (4 cores, 1.6GHz).

Graphics included in Fig. 5 and Fig. 6 show the storage requirement for both IDs using different thresholds. The vertical axis indicates the number of leaves necessary for storing all the potentials and constraints. The horizontal axis shows de number of operation. The measurement was performed after combining potentials containing a variable to be removed, and after pruning the resulting potentials of marginalization. That is, after steps 2.a and 2.c in the schema shown in Section 4.1. If evaluations with NTs (VENT, VENTWC and VENTWCPR) are compared with their equivalents using BTs (VEBT, VEBTWC and VEBTWCPR), it can be observed that, in general, less space is needed when using BTs. If constraints are applied, the size of potentials is reduced even more (VENTWC and VEBTWC), and if potentials are pruned after each operation (VENTWCPR and VEBTWCPR), the reduction is more significant. Even though constraint rules are applicable to initial potentials, an important reduction is obtained if they are applied to intermediate potentials: impossible configurations may appear in intermediate potentials during evaluation.

The reduction of the potential sizes should lead to more efficient algorithms: operations with smaller potentials should be faster. In Fig. 7 it is shown the computation time for evaluating both IDs. It can be observed that pruning after applying constraints (VENTWCPR and VEBTWCPR) is not always efficient for lower threshold values: it requires an additional computing time that is not compensated by the smaller potentials. Moreover, with higher threshold values,

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

all variants of the evaluation algorithm obtain similar results. The fastest evaluation is obtained using BTs with constraints and without pruning after each operation (VEBTWC). By contrast, worst results are obtained using NTs without constraints (VENT).

6 Conclusions and Future Work

In the present paper, it is proposed a new method for representing and evaluating asymmetric decision problems with IDs: potentials and asymmetries (constraint trees) are represented using BTs. Using this kind of representation allows to reduce the number of scenarios to consider and also to approximate the potentials. The paper shows how constraints can be used to improve the efficiency of the VE algorithm. In the experimental work, it was proved that evaluating IDs with BTs and BCTs is faster and requires less storage size than using NTs. However, if BTs are pruned after removing each variable and applying constraints, the evaluation with BTs is not efficient: the overhead introduced by pruning and sorting trees is larger using BTs than with NTs.

As regards future directions of research, we shall study if applying constraints reduces the error committed when approximating potentials. It could also be interesting to study the behaviour of BTs with constraints using alternatives to the VE inference algorithm, like *Arc Reversal* [19], *Lazy propagation* [14], etc

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 "Consejería de Economía, Innovación y Ciencia de la Junta de Andalucía" under projects TIC-06016 and P08-TIC-03717.

Bibliography

1. Bielza, C., Shenoy, P.P.: A comparison of graphical techniques for asymmetric decision problems. Management Science 45(11), 1552–1569 (1999)
2. 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)
3. Cabañas, R., Gómez, M., Cano, A.: Approximate inference in influence diagrams using binary trees. In: Proceedings of the Sixth European Workshop on Probabilistic Graphical Models, PGM 2012 (2012)
4. Call, H.J., Miller, W.A.: A comparison of approaches and implementations for automating decision analysis. Reliability Engineering & System Safety 30(1), 115–162 (1990)
5. Cano, A., Gómez-Olmedo, M., Moral, S.: Approximate inference in Bayesian networks using binary probability trees. International Journal of Approximate Reasoning 52(1), 49–62 (2011)
6. Cano, A., Moral, S., Salmerón, A.: Penniless propagation in join trees. International Journal of Intelligent Systems 15(11), 1027–1059 (2000)

7. Covaliu, Z., Oliver, R.M.: Representation and solution of decision problems using sequential decision diagrams. Management Science 41(12), 1860–1881 (1995)
8. Demirer, R., Shenoy, P.P.: Sequential valuation networks for asymmetric decision problems. European Journal of Operational Research 169(1), 286–309 (2006)
9. Díez, F.J., Luque, M.: Representing decision problems with decision analysis networks. Technical report, UNED, Madrid, Spain (2010)
10. Gómez, M., Cano, A.: Applying numerical trees to evaluate asymmetric decision problems. In: Nielsen, T.D., Zhang, N.L. (eds.) ECSQARU 2003. LNCS (LNAI), vol. 2711, pp. 196–207. Springer, Heidelberg (2003)
11. Howard, R.A., Matheson, J.E.: Influence diagram retrospective. Decision Analysis 2(3), 144–147 (2005)
12. Jensen, F.V., Nielsen, T.D.: Bayesian networks and decision graphs. Springer (2007)
13. Lucas, P.J.F., Taal, B.: Computer-based decision support in the management of primary gastric non-hodgkin lymphoma. UU-CS, (1998-33) (1998)
14. Madsen, A.L., Jensen, F.V.: 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)
15. Nielsen, T.D., Jensen, F.V.: Representing and solving asymmetric Bayesian decision problems. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 416–425. Morgan Kaufmann Publishers Inc. (2000)
16. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausble Inference. Morgan Kaufmann Pub. (1988)
17. Qi, R., Zhang, L., Poole, D.: Solving asymmetric decision problems with influence diagrams. In: Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence, pp. 491–497. Morgan Kaufmann Publishers Inc. (1994)
18. Shachter, R., Fung, R.: Contingent influence diagrams. Advanced Decision Systems (1990)
19. Shachter, R.D.: Evaluating influence diagrams. Operations Research, 871–882 (1986)
20. Shenoy, P.P.: Valuation network representation and solution of asymmetric decision problems. European Journal of Operational Research 121(3), 579–608 (2000)
21. Smith, J.E., Holtzman, S., Matheson, J.E.: Structuring conditional relationships in influence diagrams. Operations Research 41(2), 280–297 (1993)
22. Zhang, N.L., Poole, D.: Exploiting causal independence in Bayesian network inference. Journal of Artificial Intelligence Research 5, 301–328 (1996)