ProbModelXML.
A format for encoding probabilistic graphical models

Manuel Arias
Dept. Artificial Intelligence. UNED. Madrid, Spain

Francisco Javier Díez
Dept. Artificial Intelligence. UNED. Madrid, Spain

Miguel Palacios-Alonso
Computer Science Dept. INAOE, Tonantzintla, Mexico

Íñigo Bermejo
Dept. Artificial Intelligence. UNED. Madrid, Spain

Abstract

ProbModelXML is an XML format for encoding probabilistic graphical models. The main advantages of this format are that it can represent several kinds of models, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs, decision analysis networks, as well as temporal models: dynamic Bayesian networks, MDPs, POMDPs, Markov processes with atemporal decisions (MPADs), DLIMIDs, etc., and the possibility of encoding new types of networks and user-specific properties without the need to modify the format definition.

1 Introduction

A probabilistic graphical model (PGM) consists of a probability distribution and a graph, such that each node in the graph represents one of the variables on which the probability is defined, and the structure of the graph imposes some properties of independence on the probability distribution. Some PGMs are purely probabilistic, such as Bayesian networks (Pearl, 1988), while others, such as influence diagrams (Howard and Matheson, 1984), include decisions and utilities. Dynamic PGMs (Dean and Kanazawa, 1989; Murphy, 2002) are temporal models that discretize time in intervals of a fixed duration (cycle length) and create an instance of each variable for each time period.

Several formats have been developed for encoding PGMs, but almost all of them are designed for a single software tool. One exception is Fabio Cozman's XMLBIF (see Sec. 4.1), that has been implemented by several software tools, but it is restricted to Bayesian networks containing only discrete variables, with a very limited set of features. Another exception was DSC, proposed by Microsoft as a standard format for Bayesian networks and influence diagrams, that would receive contributions from the UAI (uncertainty in artificial intelligence) community; however, some time later Microsoft removed the web pages of DSC and developed a new XML format, MSBNx, limited to Bayesian networks.

For this reason, we decided to develop a new format for encoding PGMs, that presents two main advantages with respect to previous proposals. First, it can encode several types of PGMs: its current version includes Bayesian networks, Markov networks, influence diagrams, LIMIDs, decision analysis networks, dynamic Bayesian networks, MDPs, Markov processes with atemporal decisions (MPADs), POMDPs, Dec-POMDPs, and DLIMIDs (see (Arias et al., 2011) for definitions and references), and it also permits to encode new models by combining the existing *constrains* or by defining new ones (see Sec. 3.1.1). The second advantage is that it can encode user-specific features by using the `AdditionalProperties` tag (see Sec. 3.1.2). ProbModelXML was designed as the default format for OpenMarkov,[1] an open-source tool for probabilistic graphical models, but our purpose is not only to fulfill the needs of a single tool and a single research group, but to offer an extensible well-documented format that can be used by a large community of people. This is the main reason for submitting this paper to the PGM workshop: to

[1] `www.openmarkov.org`.

make the format known to a wide audience and to receive suggestions from our colleagues. In fact, we will not release version 1.0.0 of the format until we have received the feedback from the communities of PGMs and POMDPs.

The rest of the paper is structured as follows: in Section 2.1 defines the basic properties of PGMs, including dynamic models. Section 3 describes the most relevant aspects of the specification of the format. Section 4 reviews other formats for PGMs and Section 5 contains the conclusions. Given that it is impossible to present in this paper all the details of the format, we will limit ourselves to describing its main features. The complete specification can be found in (Arias et al., 2011).

2 Background: Probabilistic graphical models

2.1 Basic properties

A probabilistic graphical model (PGM) consists of a set of variables $\mathbf{V}$, a graph $\mathcal{G}$ such that each node represents a variable in $\mathbf{V}$, and a probability distribution P that satisfies certain properties of independence dictated by the structure (the links) of the graph (Pearl, 1988). In some PGMs all the nodes represent chance variables; in this case, the probability distribution is defined over $\mathbf{V}$: $P(\mathbf{v})$. Other models contains three types of nodes: chance, decision, and utility, denoted by $\mathbf{C}$, $\mathbf{D}$, and $\mathbf{U}$, respectively, such that $\mathbf{V} = \mathbf{C} \cup \mathbf{D} \cup \mathbf{U}$; in this case, the probability distribution is $P(\mathbf{c}|\mathbf{d})$.

There are three types of variables, depending on their domain. A variable is *finite-states* if it takes values on finite set of values. It is *numeric* if it represents the result of a measurement, including the count of the number of elements in a set. A *discretized* variable is a numeric variable that has been assigned a finite set of thresholds, which induce a finite set of intervals; each interval is considered as a state of the variable.

Links can be directed or undirected. Paths can be open or closed. If a closed path can be traversed completely crossing all its directed links forwards (i.e., from the tail of the link to its head), then that path is a *cycle*; otherwise it is a *loop*—see the examples in (Arias et al., 2011). A *self-loop* is a link whose nodes are the same; for example, A — A or $A \rightarrow A$. It can also be defined as a closed path consisting of only one link. Most PGMs do not accept self-loops, but there are exceptions, such as the representation of dynamic models in Netica and GeNIE.[2]

[2]See `www.norsys.com` and `genie.sis.pitt.edu`.

2.2 Dynamic models

Dynamic PGMs are a generalization of some Markovian models proposed several decades earlier. Thus, dynamic Bayesian networks (Dean and Kanazawa, 1989; Murphy, 2002) extend Markov chains and hidden Markov models, by allowing that the state of the system be represented by a set of variables rather than by a single variable. In the same way, Markov Decision Process (MDPs) (Bellman, 1957) are extended by factored MDPs (Boutilier et al., 1995; Boutilier et al., 2000)) and Partially Observable Markov Decision Process (POMDPs) are extended by factored POMDPs (Boutilier and Poole, 1996)). DLIMIDs (Díez and van Gerven, 2011) are very similar to POMDPs, but they allow several decision nodes per time slice.

3 The ProbModelXML format

Each ProbModelXML file may contain one network and, optionally, some *inference options*, such as the default inference algorithm or an elimination ordering. Alternatively, a file may contain some *evidence*, consisting of one or several evidence cases, each one composed of a set of *findings*. A third possibility is that the file contains a set of *policies*; it may be a strategy returned by an algorithm or a policy imposed by the user to perform what-if reasoning.

Therefore, the skeleton of a file in ProbModelXML format is:

```
<?xml version="1.0" encoding="UTF-8"?>
<ProbModelXML formatVersion="1.0">
  <ProbNet />0..1
  <Policies />0..1
  <InferenceOptions />0..1
  <Evidence />0..1
</ProbModelXML>
```

The default encoding for ProbModelXML is UTF-8, the same as for XML. Therefore, it is redundant to write encoding="UTF-8", but we prefer to say it explicitly, because it is possible to use other encodings.

3.1 Specification of probabilistic networks

The skeleton for a probabilistic network is as follows:

```
<ProbNet type=enumNetworkType >
  <AdditionalConstraints />0..1
  <Comment />0..1
  <Language />0..1
  <AdditionalProperties />0..1
  <Variables />
```

```
  <Links />
  <Potentials />
</ProbNet>
```

In the current version of the format, the type can be one of the following: *BayesianNetwork* (Pearl, 1988), *MarkovNetwork* (Pearl, 1988), *InfluenceDiagram* (Howard and Matheson, 1984), *LIMID* (Lauritzen and Nilsson, 2001), *DynamicBayesianNetwork* (Dean and Kanazawa, 1989; Murphy, 2002), *MarkovDecisionProcess* (which includes factored MDPs (Boutilier et al., 2000)), *POMDP* (which includes factored POMDPs (Boutilier and Poole, 1996) and MOMDPs (Ong et al., 2009)), and *DLIMID* (Díez and van Gerven, 2011).

3.1.1 Constraints

The main purpose of constraints is to prevent the user from doing illegal operations at the graphical user interface (GUI), such as giving an empty name to a variable or creating a cycle in a Bayesian network. Another use of constraints may be, for example, to prevent a learning algorithm from adding more than n parents to a node.

In the OpenMarkov tool, each network type is defined by a set of basic constraints. For example, one of the constraints that define a Bayesian network is that it can not contain cycles; another constraint is that it can not contain decision nor utility nodes. The basic constraints available in ProbModelXML and the set of constraints that define each network type are listed in (Arias et al., 2011). Decoupling the constraints from the rest of the code has facilitated the treatment of new network types in the future.

Additionally, the user can assign to a network other constraints that are not imposed by the network type; for example, the constraint that the network contains only finite-state variables. This way, an algorithm that can only solve Bayesian networks with finite-state variables may check that the network has that constraint assigned (an immediate check) or that all the variables are finite-states (which would require examining all the variables). If the network does not satisfy this constraint, the algorithm will throw an exception, explaining why it can not evaluate that network. Obviously, other software tools can treat constraints in different ways.

3.1.2 Additional properties

This tag permits to extend the ProbModelXML format by representing user-defined properties. This tag can appear in the context of *ProbNet* (see Section 3.1), *Variable*, *State*, *Potential*, *EvidenceCase*, and *Policy* (see below). In all cases, its skeleton is:

```
<AdditionalProperties>
  <Property name=string value=string/>1..n
</AdditionalProperties>
```

The main use of this tag is to store properties that are not part of the ProbModelXML format. For example, in Elvira each variable has a name, which is a string with some restrictions, and a title. In ProbModelXML we do not need this distinction, because the name can be any string. If the software package Elvira had to encode a variable in ProbModelXML, the name can be stored as the attribute name, while the title can be stored in the list of AdditionalProperties: <Property name="elvira.title" value="Test result"/>. Similarly, in GeNIE each node has a fill-in color, that can be stored as follows: <Property name="genie.interior.color" value="e5f6f7"/>. This way, any tool can save its models in ProbModelXML without missing information. Furthermore, a second tool might read that network, storing apart the properties that it does not understand, and edit it; when saving the network, it can write down also the properties stored apart. Then, the first tool can open again the modified network, thus recovering the properties that were not understood by the second.

Another use of the AdditionalProperties tag is to encode a multi-lingual version of the network. For example, if the network has a property <Language>en</Language> (cf. Sec. 3.1), which means that its default language is English, we may have the following variable:

```
<Variable name="Fever">
  ...
  <AdditionalProperties>
    <Property name="name.es" value="Fiebre"/>
    <Property name="name.fr" value="Fièvre"/>
    <Property name="name.de" value="Fieber"/>
  </AdditionalProperties>
<Variable>
```

When the network is displayed in English, the name of this variable will appear as "Fever", when in Spanish, as "Fiebre", etc.

3.1.3 Variables

The skeleton for encoding a variable is:

```
<Variable name=string type=enumDomainType
    role=enumNodeRole>
  <Comment />0..1
  <Coordinates />0..1
```

```
  <AdditionalProperties />0..1
  specification_of_domain
</Variable>
```

The type is an enumerate that can take on three values: *FiniteStates*, *Numeric*, and *Discretized*. The role can be *Chance*, *Decision*, or *Utility*. The domain of a finite-states variable is specified by a list of <States>:

```
<States>
  <State name=string>
    <AdditionalProperties />0..1
  </State>2..n
</States>
```

The domain of a numeric variable is specified by two thresholds, that define an interval, plus a number that denotes the precision with which its value is measured.

```
<Unit>string</Unit>0..1
<Precision>decimalNumber</Precision>
<Interval>
  <Threshold value=number
    belongsTo=enumSide/>2
</Interval>
```

Given that a discretized variable can be viewed as being finite-states and numeric at the same time, its skeleton has the tags of both:

```
<Unit />0..1
<Precision />
<Thresholds>
  <Threshold />3..n
</Thresholds>
<States />2..n
```

3.1.4 Links

The skeleton of a link is:

```
<Links>
  <Link var1=string var2=string directed=boolean>
    <Comment />0..1
    <Label>string</Label>
    <AdditionalProperties />0..1
  </Link>0..n
</Links>
```

The AdditionalProperties tag can be used to declare new types of links. For example, when learning a Bayesian network from a database we can use a *model network* that determines which links must be necessarily present in the learned network, which links are forbidden, etc. This can be accomplished by assigning AdditionalProperties to the links in the model network.

3.1.5 Potentials

A potential ψ defined on a set of variables $\mathbf{V}$ is a function that assigns a real number to each configuration $\mathbf{v}$ of $\mathbf{V}$. The skeleton of a potential in ProbModelXML is:

```
<Potential type=enumPotentialType
    role=enumPotentialRole label=string>
  <Comment />0..1
  <AdditionalProperties />0..1
  specification_of_the_potential
</Potential>
```

The role of a potential can be *JointProbability*, *ConditionalProbability*, *Utility*, or *Policy*.

In the current version of the format we have several types of potentials. A *Table* can be used when all the variables are finite-states or discretized; essentially, it consists of a list of numeric values (parameters), one for each configuration of the variables; it is also possible to assign second-order probability distributions to the parameters: *Beta*, *Dirichlet*, *Normal*, *LogNormal*, etc., that can be used for sensitivity analysis.

A *Tree/ADD* is a compact way of storing tables in which some of the numeric values repeat themselves; each branch outgoing from a finite-states variable corresponds to one or several of its states; each branch outgoing from a numeric variable corresponds to an interval. The difference between a tree and an ADD (algebraic decision diagram (Bahar et al., 1993)) is that the structure of the latter is an acyclic directed graph. In ProbModelXML there is only one potential type for both trees and ADDs.

ICIModel (where ICI stands for "independence of causal influence") is the type of potential used to represent canonical models (Díez and Druzdzel, 2006), such as the noisy-OR, noisy-MAX, etc. It contains a potential for each link in the family and, optionally, another subpotential for the leak probability.

```
<Potential type="ICIModel"
    role="ConditionalProbability" >
  <Model>or</Model>
  <Subpotentials>
    ...
  </Subpotentials>
</Potential>
```

The *LinearCombination* potential is used when a **numeric** variable Y depends **deterministically** on a mixed set of variables $\mathbf{X} = \mathbf{D} \cup \mathbf{C}$—which in general will be the parents of node Y in the graph—where $\mathbf{C}$ is the set of numeric variables and $\mathbf{D}$ is

the set of finite-states variables:

$$y = \alpha(\mathbf{d}) + \sum_{i=1}^{m} \beta_i(\mathbf{d}) \cdot c_i \ .$$

This function allows the parents of a utility node to be any combination of chance, decision, and utility nodes, a feature that we missed when building an influence diagram for a medical problem, because standard influence diagrams do not admit this possibility.

In a *ConditionalGaussian* potential (Lauritzen and Wermuth, 1989), the conditional probability density for Y is a univariate normal distribution:

$$f(y|\mathbf{d},\mathbf{c}) \sim \mathcal{N}(\mu(\mathbf{d},\mathbf{c}), \sigma^2(\mathbf{d})) \ ,$$

where

$$\mu(\mathbf{d},\mathbf{c}) = \alpha(\mathbf{d}) + \sum_{i=1}^{m} \beta_i(\mathbf{d}) \cdot c_i \ .$$

The potentials *Exponential* and *MixtureOfExponentials*, used in combination with *Tree/ADD*, can represent mixtures of truncated exponentials (Moral et al., 2001). Other potentials available in ProbModelXML are *LogisticRegression* and *Delta* .

3.2 Specification of evidence

A **finding** is the assignment of a value to a variable. A set of findings is an **evidence case**. Therefore, the skeleton for evidence is:

```
<Evidence>
  <EvidenceCase>
    <Finding variable=string state=string
      stateIndex=integer
      numericValue=number/>0..n
  </EvidenceCase>0..n
</Evidence>
```

The attribute variable is compulsory. If the variable is of type finite-states, then either the state or stateIndex must be present. If the variable is numeric, then numericValue must be present. If the variable is discretized, only one of the attributes may be present.

3.3 Specification of policies

In an influence diagram, a policy for a decision D is the assignment of a probability distribution to each configuration of the informational predecessors of D, i.e., the variables whose values are known when making the decision. A deterministic policy is the assignment of a degenerate distribution to each configuration. Therefore, the specification of a policy is essentially the same as that of a conditional probability potential (cf. Sec. 3.1.5).

3.4 Specification of dynamic models

The representation of dynamic models in their compact form (see Sec. 2.2) is similar to that of other types of models. There are, however, specific network tags, such as `TimeUnit`, `CycleLength`, `Horizon`, and `CoordinatesShift`, which is used to determine the relative positions of temporal variables in different time slices.

Temporal variables are recognized because they have an attribute indicating the time slice that contains the variable:

There are also specific potentials for dynamic models, such as *CycleLengthShift* and *WeibullDistribution*.

4 Related work: other formats

Several formats have been developed for probabilistic graphical models (PGMs) and Markov decision processes (MDPs). In this section we review briefly those that are more related to ProbModelXML.

4.1 Formats for Bayesian networks and influence diagrams

DNET (Netica) DNET was developed by Norsys Software Corp. as the default format for their software package, Netica.[3] This format can represent Bayesian networks, influence diagrams, MDPs and POMDPs, with both discrete (finite-states) or continuous variables. Its specification is available at `www.norsys.com/downloads/DNET_File_Format.txt`.

Elvira Elvira (Elvira Consortium, 2002) started in 1997 as a joint project of several Spanish universities.[4] The Elvira format, which uses a C-like syntax, can represent Bayesian networks and influence diagrams with continuous and finite-state variables, canonical models (Díez and Druzdzel, 2006), uncertain parameters (defined by intervals), etc. Its specification can be found at `leo.ugr.es/elvira/devel/Formato/formato.html`. Some of the features of the Elvira format were inspired on DNET, and in turn many of ProbModelXML's features are borrowed from Elvira.

XMLBIF It was proposed by Fabio Cozman, with suggestions from Marek Druzdzel and others—see `www.poli.usp.br/p/fabio.cozman/Research/InterchangeFormat`. It is restricted to the representation of Bayesian networks with finite-state

[3] See `www.norsys.com/netica.html`.

[4] `leo.ugr.es/elvira` and `www.ia.uned.es/~elvira`.

variables. It is the default format for Cozman's JavaBayes tool.[5] Weka[6] and many of the tools for PGMs can read and write networks in this format.

MSBNx This XML format was proposed by Microsoft as the default format for their Microsoft Bayesian Network (MSBNx) tool, as a replacement for the old non-XML format DSC. It can only represent Bayesian networks. See `research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx` and `research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx/msbnx/File_Formats.htm`.

XDSL It is the default format for SMILE and GeNIE, two programs developed by the Decision Systems Laboratory at the University of Pittsburgh.[7] It can represent Bayesian networks, influence diagrams, and some dynamic models, with both continuous and finite-state variables. It can also represent canonical models. The XML schemas (XSDs) that define it are available at `genie.sis.pitt.edu/SMILEHelp/Appendices/XDSL_File_Format_-_XML_Schema_Definitions.htm`.

4.2 Formats for POMDPs

Cassandra's format Anthony Cassandra has used a format for flat (i.e., non-factored) POMDPs, that is available at `www.cassandra.org/pomdp/code/pomdp-file-grammar.shtml` and `www.cassandra.org/pomdp/code/pomdp-file-spec.shtml`—see also `www.cassandra.org/pomdp/examples`. The software package Perseus (Spaan and Vlassis, 2005) can also read files in Cassandra's format.

SPUDD It is the format used by the software package SPUDD, which stands for "Stochastic Planning using Decision Diagrams" (Hoey et al., 1999). It can encode factored MDPs and POMDPs. Potentials are represented by algebraic decision diagrams (ADDs)—see Section 3.1.5. Some example POMDPs (files having the extension `.txt`) are included in a `tar.gz` file, together with SPUDD's C++ source code, which is available at `www.computing.dundee.ac.uk/staff/jessehoey/spudd/index.html`.

The tool Symbolic Perseus,[8] by Pascal Poupart (Poupart, 2005), uses a subset of the SPUDD format, whose grammar is specified in this file: `www.cs.uwaterloo.ca/~ppoupart/software/symbolicPerseus/problems/SYNTAX.txt`— see also the examples at `www.cs.uwaterloo.ca/~ppoupart/software/symbolicPerseus/problems`.

PomdpX It is an XML format for POMDPs, developed at the University of Singapore: `bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.PomdpXDocumentation`. It admits flat POMDPs, as well as MOMDPs. In a MOMDP the state space is factored into two variables, X (observable) and Y (unobservable), and there is a third variable in each time slice, O, which provides indirect information about Y (Ong et al., 2009). There is a companion XML format for representing the policy obtained when evaluating a POMDP: `bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.PolicyXDocumentation`.

PPDDL and RDDL Two additional formats were proposed to encode the problems proposed at the Probabilistic Planning Track of the International Planning Competition (IPC). The Probabilistic Planning Domain Definition Language (PPDDL),[9] used at the 4th and 5th IPC in 2004 and 2006 respectively, was able to encode factored MDPs with finite-state variables. The Relational Dynamic Influence Diagram Language (RDDL), used at the 7th IPC in 2011, was able to represent relational (PO)MDPs with both finite-state and continuous variables.[10]. These formats were not intended to encode non-temporal Bayesian networks and influence diagrams, but they has enough expressive power to represent them as well.

4.3 Discussion

We have seen that most formats proposed so far have important limitations: each format can represent either Bayesian networks (sometimes in conjunction with influence diagrams) or POMDPs, with the exception of Netica's DNET, which claims to admit both types of models, but it uses a non-standard representation of POMDPs, which makes it incompatible with existing packages for POMDPs.

A drawback of many of these formats is that they use a syntax inspired on C++ or Lisp, which complicates the creation of parsers for them. In contrast, there are efficient tools for writing parsers

[5] `www.pmr.poli.usp.br/ltd/Software/javabayes`.

[6] `www.cs.waikato.ac.nz/ml/weka`.

[7] See `genie.sis.pitt.edu`.

[8] See `www.cs.uwaterloo.ca/~ppoupart/software.html`. The file `ParseSPUDD.java` contains the parser for the SPUDD format.

[9] `www.tempastic.org/papers/CMU-CS-04-167.pdf`.

[10] `users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf`.

that can read XML files into C++ or Java programs. An inconvenience of XML is its verbosity, which leads to significantly larger files than when using a C-like syntax. This would be relevant if a user had to encode the information—a PGM, in our case—into an XML format manually. Fortunately, there are many computer programs that facilitate that task; in our case, it is possible to use OpenMarkov's GUI to build PGMs without caring about the syntax of the format, and we expect that other software tools may adopt this format in the future. Given that the simplicity of parsing XML files compensates for the size of the files, XML is used more and more for defining new formats in almost every field of computing. However, two of the three XML formats proposed previously for PGMs can only encode Bayesian networks with finite-state variables, the third can only encode Bayesian networks and influence diagrams, and the only XML format for POMDPs cannot encode factored models.

Finally, an important limitation of all these formats is that they are not extensible, which implies that they cannot encode any property that has not been explicitly declared in its specification.

5 Conclusion

In this paper we have presented an overview of a new XML format for encoding probabilistic graphical models (PGMs). One of its advantages is the possibility of representing several types of models, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs, as well as dynamic models: dynamic Bayesian networks, MDPs, POMDP, and DLIMIDs, and it is easy to add new types of models by combining the existing constraints or by defining new ones.

Another advantage with respect to existing formats is the possibility of encoding user-defined properties without modifying the specification of the format, by placing them under the `AdditionalProperties` tag. The third advantage is its XML syntax, which permits to use the utilities available for generating parsers and writers in different languages: Java, C++, etc.

For these reasons, we believe that ProbModelXML may be very useful as an interchange format for PGMs. Clearly, ProbModelXML is a very rich format, which makes it difficult to implement all its features. However, each software package can implement only the subset of features required for its purpose: it suffices to throw an error message when the parser encounters an unknown feature. Even our software tool OpenMarkov is currently unable to cope with several features of the format, but we have decided to include them in the format to satisfy the needs of other research groups. For this reason, we believe that the format presented in this paper can be very useful for interchanging several types of PGMs between different software tools and research groups.

The tasks we have scheduled for the near future are to improve the syntax for certain properties in the light of the feedback we have received from some colleagues, and to extend it to cover new types of potentials (such as mixtures of polynomials (Shenoy, 2011; Shenoy and West, 2010)), submodels (as in GENIE), and new types of networks, such as object-oriented Bayesian networks (Koller and Pfeffer, 1997) and probabilistic relational models (Jaeger, 1997; Koller and Pfeffer, 1996).

Acknowledgments

This work has been supported by grants TIN2006-11152 and TIN2009-09158, of the Spanish Ministry of Science and Technology, and by FONCICYT grant 85195. I.B. has received a predoctoral fellowship from the Universidad Nacional de Educación a Distancia (UNED).

The reviewers of the PGM-2012 workshop have made useful comments about this paper.

References

[Arias et al.2011] M. Arias, F. J. Díez, and M. P. Palacios. 2011. ProbModelXML. A format for encoding probabilistic graphical models. Technical Report CISIAD-11-02, UNED, Madrid, Spain.

[Bahar et al.1993] R. I. Bahar, E. A. Frohm, C. M. Gaona, et al. 1993. Algebraic decision diagrams and their applications. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'93)*, pages 188–191, Santa Clara, CA.

[Bellman1957] R. E. Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

[Boutilier and Poole1996] C. Boutilier and D. Poole. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96*, pages 1168–1175, Portland, OR. AAAI Press.

[Boutilier et al.1995] C. Boutilier, R. Dearden, and M. Goldszmidt. 1995. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI–95)*, pages 1104–1111, Montreal, Canada.

[Boutilier et al.2000] C. Boutilier, R. Dearden, and M. Goldszmidt. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121:49–107.

[Dean and Kanazawa1989] T. Dean and K. Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.

[Díez and Druzdzel2006] F. J. Díez and M. J. Druzdzel. 2006. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain.

[Díez and van Gerven2011] F. J. Díez and M. A. J. van Gerven. 2011. Dynamic LIMIDs. In L. E. Sucar, J. Hoey, and E. Morales, editors, *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, pages 164–189. IGI Global, Hershey, PA.

[Elvira Consortium2002] The Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In J. A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 1–11, Cuenca, Spain.

[Hoey et al.1999] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 279–288, Stockholm, Sweden. Morgan Kaufmann, San Francisco, CA.

[Howard and Matheson1984] R. A. Howard and J. E. Matheson. 1984. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762. Strategic Decisions Group, Menlo Park, CA.

[Jaeger1997] M. Jaeger. 1997. Relational Bayesian networks. In *Proceedings of the Thirteenth Conference in Artificial Intelligence (UAI-97)*, pages 266–273, San Francisco, CA. Morgan Kaufmann.

[Koller and Pfeffer1996] D. Koller and A. Pfeffer. 1996. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98*, pages 580–587, Madison, WI.

[Koller and Pfeffer1997] D. Koller and A. Pfeffer. 1997. Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Conference in Artificial Intelligence (UAI-97)*, pages 302–313, San Francisco, CA. Morgan Kaufmann.

[Lauritzen and Nilsson2001] S. L. Lauritzen and D. Nilsson. 2001. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251.

[Lauritzen and Wermuth1989] S. L. Lauritzen and N. Wermuth. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17:31–57.

[Moral et al.2001] S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. *Lecture Notes in Artificial Intelligence*, 2143:156–167.

[Murphy2002] K. Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, Computer Science Division, University of California, Berkeley.

[Ong et al.2009] S. C.W. Ong, S. W. Png, D. Hsu, and W. S. Lee. 2009. POMDPs for robotic tasks with mixed observability. In *Proceedings of Robotics: Science and Systems V*, Seattle, WA.

[Pearl1988] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

[Poupart2005] P. Poupart. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. Ph.D. thesis, Dept. of Computer Science, University of Toronto, Canada.

[Shenoy and West2010] P. P. Shenoy and J. C. West. 2010. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52:641–657.

[Shenoy2011] P. P. Shenoy. 2011. A re-definition of mixtures of polynomials for inference in hybrid Bayesian networks. In W. Liu, editor, *Proceedings of the 11th European conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'11)*, pages 98–109. Springer, Heidelberg.

[Spaan and Vlassis2005] M. T. J. Spaan and N. Vlassis. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220.