A New Algorithm for Generating Situation-Specific Bayesian Networks Using Bayes-Ball Method

Laécio L. Santos[1], Rommel N. Carvalho[1,2], Marcelo Ladeira[1], and Li Weigang[1]

[1] Department of Computer Science, University of Brasília (UnB)
Brasília, DF, Brazil
laecio@gmail.com,{rommelnc,mladeira,weigang}@unb.br
[2] Department of Research and Strategic Information, Brazilian Office of the Comptroller General (CGU), Brasília, DF, Brazil
rommel.carvalho@cgu.gov.br

Abstract. Multi-Entity Bayesian Network (MEBN) is an expressive first-order probabilistic logic that represents the domain using parameterized fragments of Bayesian networks. Probabilistic-OWL (PR-OWL) uses MEBN to add uncertainty support to OWL, the main language of the Semantic Web. The reasoning in MEBN is made by the construction of a Situation-Specific Bayesian Network (SSBN), a minimal Bayesian network sufficient to compute the response to queries. A Bottom-Up algorithm has been proposed for generating SSBNs in MEBN. However, this approach presents scalability problems since the algorithm starts from all the query and evidence nodes, which can be a very large set in real domains. To address this problem, we present a new scalable algorithm for generating SSBNs based on the Bayes-Ball method, a well-known and efficient algorithm for discovering d-separated nodes of target sets in Bayesian networks. The novel SSBN algorithm used together with Resource Description Framework (RDF) databases and PR-OWL 2 RL, an amenable version of PR-OWL, allows reasoning with probabilistic ontologies containing large assertive bases, offering a scalable approach for the treatment of uncertainty in the Semantic Web.

1 Introduction

Uncertainty can occur in a variety of forms in several types of domains, which encouraged the proposition of various new formalisms. Bayesian networks are one of the probabilistic methodologies most widely studied and used when working with uncertainty due to their power of representation, and the well-known algorithms that make inference to them.

Multi-Entity Bayesian Network (MEBN) [9] is an expressive first-order probabilistic logic that represents the domain using parameterized fragments of Bayesian networks (BNs). MEBN is the base for the Probabilistic-OWL (PR-OWL) language [4]. PR-OWL adds uncertainty support to Web Ontology Language (OWL), the main language of the Semantic Web, allowing the construction of probabilistic ontologies. PR-OWL 2 [2] extends the original language allowing a better integration with the OWL semantics, where the modeler can define uncertainty starting with an existent OWL ontology. PR-OWL 2 RL [13] maps PR-OWL 2 language to be represented in OWL 2

RL profile, which allows reasoning in polynomial time for the main reasoning tasks. PR-OWL 2 RL uses RDF databases for both storage and reasoning with very large ontologies represented as RDF triples [5].

Inference in MEBN consists of the generation of a Situation-Specific Bayesian Network (SSBN), a minimal Bayesian network sufficient to solve a set of target nodes for which it is necessary to calculate the probability. Laskey [9] presents a Bottom-Up algorithm to generate SSBNs. This algorithm was implemented in the UnBBayes framework [1], which was the first tool to implement MEBN.

The Bottom-Up algorithm, however, does not work well in domains with large assertive bases. In the Procurement Fraud ontology presented in [3], for example, there are millions of statements about the entities related to the analyzed case (e.g., enterprises, enterprise owners, and committee members). The Bottom-Up algorithm starts the generation of a Bayesian network by instantiating the query and evidence nodes. In this kind of domain, all the information related to every entity in the dataset will become nodes instantiated and evaluated to see whether they can influence the query nodes. Since the query nodes are usually related to just a small subset of these entities, usually, most of the instantiated nodes will be removed from the final BN. Thus, a new algorithm is necessary to work with domains such as this, and with PR-OWL 2 RL language, which is designed to work with assertive bases with millions of RDF triples.

This paper proposes a novel algorithm for generating SSBNs based on Bayes-Ball algorithm. Bayes-Ball [14] is a well-known and efficient method that was developed for searching relevant information in Bayesian networks. The algorithm can be used to see which nodes are d-connected with a set of target nodes in polynomial time on the size of the graph. The idea is to extend the Bayes-Ball algorithm to generate the SSBN on demand, generating only the nodes d-connected with the query nodes.

The rest of paper is organized as follows. Section 2 presents some necessary concepts for this work: MEBN, PR-OWL, and Bayes-Ball algorithm. Section 3 presents the Bottom-Up algorithm for generating SSBNs with some considerations about the implemented version in the UnBBayes framework. Section 4 further discusses the proposed algorithm for extending Bayes-Ball algorithm to generate SSBNs. Section 5 presents a comparison between Bottom-Up and Bayes-Ball algorithms using plug-ins of PR-OWL 2 and PR-OWL 2 RL implemented in UnBBayes framework. Finally, Section 6 describes some concluding remarks and future works.

2 Fundamentals

In this section we will present the main concepts related to the new algorithm proposed for generating SSBNs: MEBN, PR-OWL, and the Bayes-Ball algorithm.

2.1 Multi-Entity Bayesian Networks

Multi-Entity Bayesian Network (MEBN) is a first-order language that specifies probabilistic knowledge bases as parameterized fragments of Bayesian networks [9]. By the incorporation of first-order logic expressiveness, MEBN solves the main limitation of

Bayesian networks: the impossibility of representing situations where the number of random variables is unknown.

The domain is modeled as a MEBN Theory (MTheory). An MTheory consists of a set of MEBN Fragments (MFrags) that satisfy consistency conditions ensuring the existence of a unique joint probability distribution over its random variables [9]. Each MFrag has a collection of parametrized related random variables representing the properties and relationships of the entities. MEBN works as a template: an MFrag can be instantiated to create as many instances of the hypotheses as needed [4].

MFrags are composed of resident, input, and context nodes. To illustrate this, Figure 1 shows the MFrag *Front of Enterprise*, from the domain Procurement Fraud, presented in [3]. This domain models the possibility of a fraud in public procurements using information such as the existence of relationships between members of the procurement committee and the winning enterprise. Resident nodes (node 9) and input nodes (nodes 6-8) are random variables representing the properties and relationships of the entities. They are parametrized with arguments (ordinary variables) that will be filled with the entities during the instantiation of the model. Resident node `isFrontFor`, for example, has the arguments `person` and `enterprise`. Each resident node has a Local Probability Distribution (LPD) defined in its home MFrag. Input nodes point to resident nodes in other MFrags. Context nodes (nodes 1-5) are first-order logic expressions that define constraints that must be verified and valid in order to allow the instantiating of the corresponding MFrag with its defined LPD per resident node. If the context nodes are not satisfied, the MFrag will be instantiated using a default distribution.

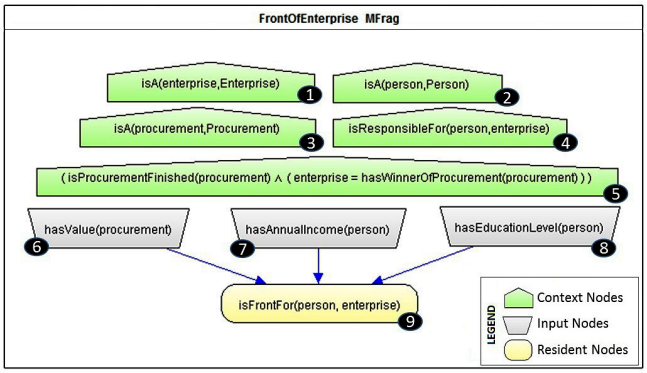

Fig. 1. MFrag "Front of Enterprise"

Inference in MEBN consists of answering queries posed to assess the belief in a set of target random variables given a set of evidence random variables (findings) [4]. Therefore, the MTheory will be instantiated using the entities and findings of the knowledge base, which generates a Situation-Specific Bayesian Network (SSBN), a minimal Bayesian network sufficient to compute the correct probability of the query nodes [11].

Over the SSBN generated, the inference can be made using any algorithm, exact and approximate, existent for Bayesian networks.

2.2 PR-OWL

Probabilistic OWL (PR-OWL) [4] adds uncertainty support to OWL using MEBN. PR-OWL is an upper-ontology that consists of a set of classes, subclasses, and properties that collectively form a framework for building probabilistic ontologies [4].

Figure 2, extracted from [4], shows how PR-OWL models the main concepts involved in defining a MEBN Theory. The probabilistic part of the ontology is modeled using the MTheory class, composed of a set of MFrags. MFrags are composed of nodes that represent random variables with their respective probability distribution and an exhaustive set of possible values, individuals of the Entity class.

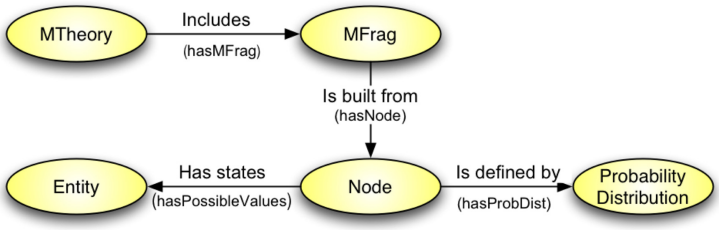

Fig. 2. PR-OWL Concepts

PR-OWL 2 [2] extends the original language solving its two major limitations: the lack of a formal mapping between random variables of PR-OWL and relationships defined in OWL, and the lack of compatibility between PR-OWL types and the types that already exist in OWL language. To solve the first limitation, PR-OWL 2 utilizes the relationship `definesUncertaintyOf`, which creates a link between a PR-OWL random variable and an OWL property. With this construction, and the additional properties `isSubjectIn` and `isObjectIn`, it is possible to model ontologies containing interrelated probabilistic and deterministic declarations. To solve the second limitation, PR-OWL 2 maps PR-OWL types to types already defined in OWL. These modifications approximate PR-OWL semantics to OWL semantics.

PR-OWL 2 RL [13] adapts PR-OWL 2 to the OWL 2 RL profile, allowing the use of triplestores - a type of database that stores the knowledge using RDF triples. The language is designed to work with probabilistic ontologies that have large assertive bases. Along with restricting the ontologies to the OWL 2 RL profile to allow inference in polynomial time, PR-OWL 2 RL also restricts the set of logical expressions allowed to evaluate them using queries submitted to triplestores (see [13]).

Both versions of PR-OWL were implemented as plug-ins for UnBBayes. UnBBayes [3] is an open-source framework developed to work with probabilistic reasoning. The PR-

[3] http://sourceforge.net/projects/unbbayes

OWL plug-in was the first worldwide implementation of MEBN [1]. It includes a Graphical User Interface (GUI) for visual modeling of an MTheory, a knowledge representation and reasoning system, and one implementation of the Bottom-Up algorithm proposed in [9] for generating the SSBN. After generating the SSBN, UnBBayes uses the Junction Tree algorithm [8] for compiling and propagating evidence in the Bayesian network generated. Junction Tree is an exact algorithm that transforms the network in a group of cliques, in which probabilities can be propagated in polynomial time.

2.3 Bayes-Ball Algorithm

The Bayes-Ball algorithm [14] allows the verification of which nodes of a Bayesian network are d-separated from a set of target nodes, given a set of evidence nodes. Thus, it allows the identification of the probabilistically irrelevant (independent) nodes.

Given a Bayesian network $B = (N, A)$, where N is the set of nodes and A the set of arcs, Bayes-Ball algorithm allows the calculation of the set $N_i(J|K)$, which contains the irrelevant nodes to a set of target nodes X_J given the set of evidence X_K, as shown in Equation 1.

$$N_i(J|K) = \{i \in N : X_i \perp X_J | X_K\} \ . \tag{1}$$

The d-separation, stated in Theorem 1, is a graphical criterion that identifies conditional independences in Bayesian networks.

Theorem 1. (d-separation) *Two variables, A and B, in a directed acyclic graph are d-separated if for every trail between A and B there is an intermediate variable W, such as: a) connection is serial or divergent, and the state of W is known, or, b) connection is convergence and neither W nor its descendants have any evidence [7].*

Figure 3 illustrates the criterion: the probability goes through serial and divergent connections, but in convergent connections it advances only if exists evidence for the intermediate variable W or for one of its descendants.

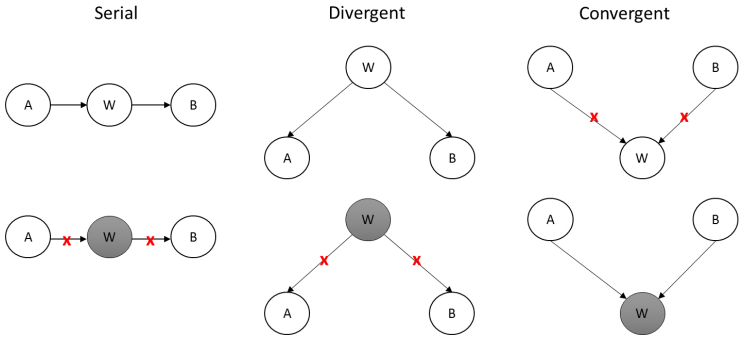

Fig. 3. d-separation Criterion

The idea of the Bayes-Ball algorithm is to pass an imaginary ball from the target nodes to the nodes that have an active trail to them. The ball is initially received from each target node from a virtual child node and is passed from one node to another following rules based on d-separation. Depending whether the ball was received from a parent or from a child, and whether the node that received it is evidence or not, the ball is passed to its parents, to its children, or blocked. Figure 4 illustrates the following rules:

R1 If the ball is received from a parent and the node does not have evidence, the ball is passed to all children of this node.
R2 If the ball is received from a child and the node does not have evidence, the ball is bounced back to all the children of the node and passed to all parents of the node.
R3 If the ball is received from a parent and the node has evidence, the ball is bounced back to all parents of the node.
R4 If the ball is received from a child and the node has evidence, the ball is blocked (it is not passed to any node).

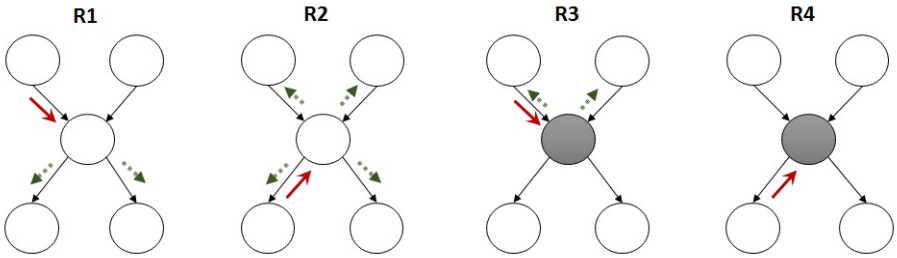

Fig. 4. Bayes-Ball Rules

The irrelevant nodes, $N_i(J|K)$, are the nodes not marked as visited. The set of irrelevant nodes calculated by Bayes-Ball algorithm correspond to the set of nodes d-separated from target nodes given K [14].

The complexity of the algorithm is O($|N|+|Av|$) [14], where Av are the arcs visited by the algorithm. It is necessary to pass to every node to initialize the flags. In the worst case scenario, the algorithm is linear on the size of the graph [14] because every node and arc will be visited.

3 Bottom-Up Algorithm for Generating SSBNs

This section presents the version of the SSBN Bottom-Up construction algorithm implemented in UnBBayes framework.

The steps are described below. The input is the generative MTheory, the knowledge base containing the assertive information, and the queries to be solved.

1 **Initialization:** Create the initial set (S_1) with the query nodes (set Q) and the evidence nodes (set E). The evidence nodes are recovered from the knowledge base. The nodes of S_1 are marked as `Unfinished`. Set $i = 1$ (number of the iteration) and initialize the constant $LIMIT\_ITR$ with a limit for the number of iterations.

2 **Structure Building:** If $i = LIMIT\_ITR$, or if S_i is empty, go to the next step. Else, create an empty set S_{i+1}, and, for each node x of the set S_i do steps 2.1 to 2.4 below. After this, set $i = i + 1$.

- 2.1 Recover the MFrag M_x, in which x is a resident node;
- 2.2 Instantiate the ordinary variables of M_x using the argument values of node x. Evaluate the context nodes of M_x. Context nodes that have ordinary variables not filled from the x arguments are queried against the knowledge base for values that validate the formula. If some context nodes are evaluated `false`, x and M_x are marked to use default distribution and x is marked as `Finished`. If all context nodes of M_x are evaluated to `true`, the evaluation of the MFrag continues. If some context nodes c_i have ordinary variables unfilled, it utilizes an uncertainty reference strategy, where the context node and the possible parent nodes (one for each possible value of the OV unfilled) become parents of node x. In this case, the node c_i will work like a multiplexer.
- 2.3 Instantiate parents of node x inside M_x. The parent nodes should be input or resident nodes. The values of the ordinary variables of the new nodes are defined from the values of the ordinary variables of the child node x or from the values recovered by the context nodes for ordinary variables not present in x. The new nodes are marked as `Unfinished` and added to the set S_{i+1}.
- 2.4 Mark node as `Finished`;

3 **Structure Prune:** From the SSBN, remove the nodes that do not influence the probabilistic distributions of the query nodes given the evidence nodes.

4 **Probabilistic Distribution Generation:** Build the Conditional Probabilistic Tables (CPTs) for the SSBN nodes. The CPT of each node is generated from the LPD depending on the parent and states of the node.

5 **Compilation and Initialization of the generated SSBN:** The SSBN is compiled and initialized with the information about the evidence available.

The Bayesian network built during the second step is known as Grand BN and it will possibly contain disconnected networks. The prune phase is important for removing this disconnected networks and for eliminating nodes that are irrelevant for determining the probabilistic distribution of the query node. Laskey [9] proposes pruning the following nodes:

- **d-separated nodes**: nodes d-separated from query nodes given the evidence.
- **Barren nodes** : nodes that not contain descendants that are query or evidence nodes. Barren nodes should depend on evidence, but they cannot change the probability of a query node and are computationally irrelevant [10].
- **Nuisance nodes**: nodes that are computationally related to target nodes given the evidence but that are not part of any evidential trail from any node in the evidence set to any node in the query set [10]. They are relevant for the queries only by their marginal distributions: if those are pre-computed, these nodes can be removed from the SSBN.

4 New Algorithm for Generating SSBNs

The Bottom-up algorithm is not adequate to work with domains that contain large assertive bases and, because of this, it is not adequate to work with PR-OWL 2 RL language. It starts in each query and evidence node, which can originate the generation of disconnected networks and irrelevant nodes in domains with a great quantity of evidence, consuming time and space in the Grand BN generation and evaluating some unnecessary MFrags. For each MFrag, we have to evaluate the context nodes, making searches and inferences on the knowledge base. This can be very complex using an expressive logic such as first-order logic, or OWL 2 DL logic. Since irrelevant nodes will be removed in the pruning phase, one solution that avoids its generation in the building phase can improve the algorithm scalability.

We propose an algorithm for generating SSBNs based on the Bayes-Ball algorithm. The SSBN nodes are generated when the Bayes ball visits that node. Since the ball is passed only for nodes relevant to answer the queries, the algorithm warranty that only d-connected nodes to the query nodes will be generated. Consequently, only MFrags of relevant nodes are evaluated, saving time in the evaluation of context nodes.

The new algorithm makes the following modifications to the Bottom-Up algorithm: in the initialization phase, only query nodes are included in the initial set S_1; in the structure building phase, the nodes are created based on Bayes-Ball algorithm rules; and, in the structure Prune phase, it is no longer necessary to prune d-separated nodes because the algorithm generated only nodes that have an active path to the query nodes.

To control the evaluation of the algorithm, some marks are necessary on each SSBN node during the reception and transmission of the virtual Bayes ball:

`receivedBallFromChild`: true if the ball was received from a child;
`receivedBallFromParent`: true if the ball was received from a parent;
`evaluatedTop`: true if the node was evaluated above;
`evaluatedBottom`: true if the node was evaluated below;
`isFinding`: true if the node has evidence;
`visited`: true if the node was visited by the Bayesian ball.

Algorithm 1 presents how the new algorithm builds the SSBN. The algorithm uses the following procedures:

evaluateNode: verify whether the node is an evidence (using the knowledge base). If it is, set `isFinding` = `true`; instantiate MFrag M_x in which the node is resident and evaluate its context nodes.

evaluateTop: evaluate the nodes above by instantiating the parents of x, setting `receivedBallFromChild` to `true`. If y, a parent node of x, is an input node, the MFrag M_y where y is resident will be instantiated using the values for that ordinary variables in M_x to fill the arguments of y. If y is a resident node, only instantiate it, using the arguments of M_x. Add the new generated nodes to `nodeList` (list of nodes to be evaluated in the next iteration).

evaluateBottom: evaluate the nodes below, generating its descendants. This includes resident nodes that are children of x in M_x and resident nodes that are children of y in M_y, where y is an input node located in MFrag M_y that makes reference to x. In

Algorithm 1 Bayes-Ball SSBN Generator Algorithm

Input: SSBN object created with the list of queries and
Input: Knowledge base contain the evidences
Output: nodeList containing the generated nodes

```
1: for each node in SSBN.queryList do
2:     node.receivedBallFromChild ← true
3:     unvaluatedNodeList.add(node)
4:     nodeList.add(node)
5: while unvaluatedNodeList.size() > 0 do
6:     for each node in unvaluatedNodeList do
7:         evaluateNode(node)
8:         if node.receivedBallFromChild then
9:             if node.isFinding == false then
10:                if node.evaluatedTop == false then
11:                    evaluateTop(nodelist, node);
12:                    node.evaluatedTop ← true;
13:                if node.isEvaluatedBottom == false then
14:                    evaluateBottom(nodeList, node);
15:                    node.evaluatedBottom ← true;
16:        if node.isReceivedBallFromParent then
17:            if node.isFinding == false then
18:                if node.isEvaluatedBottom == false then
19:                    evaluateBottom(nodelist, node);
20:                    node.evaluatedBottom ← true;
21:            else
22:                if node.isEvaluatedTop == false then
23:                    evaluateTop(nodeList, node);
24:                    node.evaluatedTop ← true;
25:        node.visited ← true
26:    unvaluatedList.clear()
27:    for each node in nodeList do
28:        if node.visited == false then
29:            unvaluatedList.add(node)
```

the last case, the MFrag M_y will also be instantiated, starting with the ordinary variables of x. The other ordinary variables will be recovered using the context node evaluation. New nodes created are added to the `nodeList` and `receivedBallFromParent` is set to `true`.

Before creating a new node, the algorithm verifies whether the node already exists in the SSBN. If it exists, the algorithm only updates the information of the flags `receivedBallFromChild` and `receivedBallFromParent`, of the existing node, and sets `visited` to `false` if any of these flags were changed.

5 Evaluation of the new Algorithm

We developed a plug-in [4] for UnBBayes that implements PR-OWL 2 RL and the Bayes-Ball algorithm. This section shows the results of comparative tests made between the new plug-in and the PR-OWL 2 plug-in, the most recent one that uses the Bottom-Up algorithm.

PR-OWL 2 plug-in utilizes the reasoner HermiT [6] to work with the knowledge base, represented in OWL 2 DL. PR-OWL 2 RL plug-in utilizes a triplestore, GraphDB Free[5], that implements the OWL 2 RL profile over RDF graphs, using materialization for inference. The materialization consists of expanding the rules in loading time by calculating all new expressions that emerge from the set of added expressions.

In order to carry out the tests, we developed a benchmark able to generate assertive bases for the probabilistic ontology Procurement Fraud. Table 1 shows the bases used. Base 15, for example, has 15 persons, 1 procurement, and 2 enterprises. Using these entities, the benchmark creates the relationships (`isProcurementFinished`, `hasProcurementOwner`, `hasWinnerOfProcurement`, `isParticipantIn`, `isMemberOfCommittee`, etc.) based on the probabilities available on the ontology. The final base has a total of 97 assertions.

Table 1. Procurement Fraud Test Bases

	Persons	Procurements	Enterprises	Total Assertions
Base 15	15	1	2	97
Base 50	50	2	5	296
Base 100	100	2	5	574
Base 1,000	1,000	20	50	5,514
Base 10,000	10,000	200	500	55,104
Base 100,000	100,000	2,000	5,000	551,004
Base 1,000,000	1,000,000	20,000	50,000	5,510,004

The tests were performed on a computer with an i5 processor and 8 GB of RAM, 5 GB dedicated to Java Runtime Engine running UnBBayes. The test consists of solving the query `isSuspiciousProcurement` for the entity `procurement_0`, generating the SSBN, and compiling it using the Junction Tree algorithm.

Figure 5 shows a plot with the time used for both plug-ins to evaluate bases 15, 50, and 100. When generating the SSBN using PR-OWL 2 plug-in, the time to solve the query increases at a greater rate than using PR-OWL 2 RL plug-in. This occurs for two reasons: the Bottom-Up algorithm starts on each evidence existent in the base; and HermiT reasoner takes more time to solve expressions submitted to it because it makes inference in $\mathcal{SHROIQ}$(D), the description logic in which OWL 2 DL is based, that has complexity N2EXPTIME-complete [12]. PR-OWL 2 RL plug-in performs better since it uses the Bayes-Ball algorithm, which generates only nodes relevant to solve the queries. It also uses a triplestore that makes the inference in polynomial time, over

[4] Available in https://sourceforge.net/p/unbbayes/

[5] http://graphdb.ontotext.com/

a less expressive version of OWL (OWL 2 RL), using materialization when the user loads data on the base.

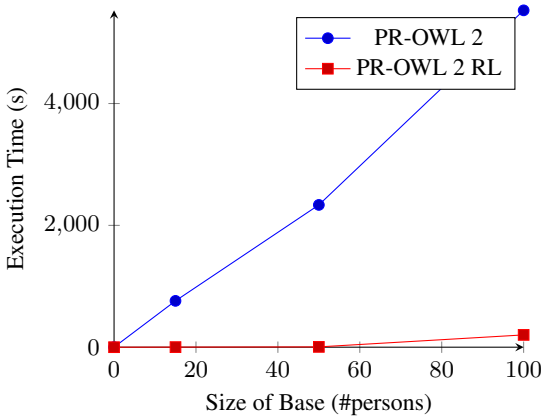

Fig. 5. Generation of SSBN in UnBBayes PR-OWL 2 and PR-OWL 2 RL plug-ins

The test with Base 1,000 using PR-OWL 2 plug-in was not completed after 10 hours. As a result, the following tests were not executed in this plug-in. Table 2 shows the time necessary to execute the query in Base 1,000 to 1,000,000 using the PR-OWL 2 RL plug-in. Step 1 is the initialization; Step 2 the construction of Grand BN; Step 3 the pruning phase; Step 4 the construction of CPT's; and Step 5 the compilation of the network.

Table 2. Time to execute queries with PR-OWL 2 RL plug-in (in seconds)

	Step 1	Step 2	Step 3	Step 4	Step 5	Total Time
Base 1,000	0.01	3.0	0.002	233.0	303.2	539.0
Base 10,000	0.04	2.6	0.003	229.2	310.3	542.2
Base 100,000	0.03	2.0	0.003	234.2	301.3	537.6
Base 1,000,000	0.03	8.4	0.001	243.4	313.1	565.0

The table shows that the tests were executed at an approximately constant time. This occurs because we fixed the number of persons by procurement and enterprises by procurement to 4. When we used more than this, we have found space problems for generating the CPTs because nodes with a large number of parents were generated. SSBN generation using Bayes-Ball algorithm uses only the information related to the query. Subsequently, while increasing the assertion base, as the amount of these relationships remained constant, the time needed to solve the query remained approximately constant. This resolved the limitation of the Bottom-Up algorithm, which starts on every information of the assertive base. In real applications, usually only a small part of the

database information will be related to a user query, which makes the new algorithm more adequate.

6 Conclusion and Future Work

This paper presented an algorithm based on Bayes-Ball method to generate a SSBN starting only on the query nodes. The new SSBN algorithm facilitates working with domains that have a large assertive base, such as the Procurement Fraud detecting system.

This new algorithm is part of the development of PR-OWL 2 RL, a new version of PR-OWL amenable for working with RDF databases (triplestores). Together with the Bayes-Ball SSBN algorithm and a grammar of FOL formulas that allow evaluation over triplestores, the new language can be used to work with domains containing large assertive databases. Tests comparing PR-OWL 2 RL plug-in to PR-OWL 2 plug-in show that the Bayes-Ball algorithm together with the use of triplestores improved a lot the scalability of SSBN generation in this kind of domains.

The tests presented in this work are only an initial evaluations of the algorithm. We plan to carry out more complete tests in the future, including tests using real data from the Procurement Fraud domain. We also plan to make tests with others domains to validate the Bayes-Ball algorithm and the PR-OWL 2 RL language.

Some issues still need to be studied in relation to the Bayesian network generated, given that the inference with it still depends on the algorithm's sensibility to the size of the network. The exact inference Junction Tree algorithm implemented in UnBBayes presents problems when the size of the cliques is large. Future research will focus on approximate inference, which seems to be a more effective approach.

References

1. Rommel N. Carvalho, Marcelo Ladeira, Laécio L. Santos, Shou Matsumoto, and Paulo C. G. Costa. A GUI tool for plausible reasoning in the semantic web using MEBN. In *Innovative Applications in Data Mining*, pages 17–45. 2009.
2. Rommel N. Carvalho, Kathryn B. Laskey, and Paulo C. G. Costa. PR-OWL 2.0–bringing the gap to OWL semantics. In *Uncertainty Reasoning for the Semantic Web II*, pages 1–18. Springer, 2013.
3. Rommel N. Carvalho, Shou Matsumoto, Kathryn B. Laskey, Paulo C. G. da Costa, Marcelo Ladeira, and Laécio L. Santos. Probabilistic ontology and knowledge fusion for procurement fraud detection in brazil. In *Uncertainty Reasoning for the Semantic Web II*, pages 19–40, 2013.
4. Paulo C. G. Costa, Kathryn B. Laskey, and Kenneth J. Laskey. PR-OWL: a Bayesian ontology language for the semantic web. In *Uncertainty Reasoning for the Semantic Web I: ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers*, pages 88–107. Springer-Verlag, 2008.
5. Laécio L. dos Santos, Rommel N. Carvalho, Marcelo Ladeira, Weigang Li, and Gilson Libório Mendes. PR-OWL 2 RL - A language for scalable uncertainty reasoning on the semantic web information. In *Proceedings of the 11th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2015), Bethlehem, USA, October 12, 2015.*, pages 14–25, 2015.

6. Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.
7. Finn V Jensen. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.
8. Finn V Jensen, Steffen L Lauritzen, and Kristian G Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational statistics quarterly*, 1990.
9. Kathryn B. Laskey. MEBN: a language for First-Order Bayesian knowledge bases. *Artificial Intelligence*, 172(2-3):140–178, 2008.
10. Yan Lin and Marek J. Druzdzel. Computational advantages of relevance reasoning in bayesian belief networks. In *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, pages 342–350, 1997.
11. Suzanne M. Mahoney and Kathryn B. Laskey. Constructing Situation Specific Belief Networks. In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, pages 370–37, 1998.
12. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, and Carsten Lutz Achille Fokoue. OWL 2 Web Ontology Language profiles. http://www.w3.org/TR/owl2-profiles/, 2012.
13. Laécio L. Santos, Rommel N. Carvalho, Marcelo Ladeira, Li Weigang, Kathryn B. Laskey, and Paulo C. G. Costa. Scalable uncertainty treatment using triplestores and the OWL 2 RL profile. In *18th International Conference on Information Fusion, FUSION 2015, Washington, DC, USA, July 6-9, 2015*, pages 924–931, 2015.
14. Ross D. Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, pages 480–487, 1998.