aGrUM: a Graphical Universal Model framework

Christophe Gonzales, Lionel Torti, Pierre-Henri Wuillemin

aGrUM: a Graphical Universal Model framework

Christophe Gonzales, Lionel Torti, and Pierre-Henri Wuillemin

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, Paris, France, email: firstname.lastname@lip6.fr

Abstract. This paper presents the **aGrUM** framework, a C++ library providing state-of-the-art implementations of graphical models for decision making, including Bayesian Networks, Influence Diagrams, Credal Networks, Probabilistic Relational Models. This is the result of an ongoing effort to build an efficient and well maintained open source cross-platform software, running on Linux, MacOS X and Windows, for dealing with graphical models. The framework also contains a wrapper, pyAgrum, for exploiting **aGrUM** within Python.

1 Introduction

The **aGrUM** project started eight years ago at the artificial intelligence and decision department of University Pierre and Marie Curie (`http://www.lip6.fr`). Developed by several contributors, in particular the authors of the present paper, the project grew into an extensive open source graphical model framework. This one includes the **aGrUM** C++ library, a Python wrapper and some applications, all running on Linux, MacOS and Windows (supported compilers include g++, clang, mvsc, mingw). The framework is freely available at `http://agrum.lip6.fr`[1]. There also exists a dedicated website (`http://pyagrum.lip6.fr`) for the python wrapper: **pyAgrum**.

The goal of **aGrUM** is the development of an efficient, easy-to-use and well maintained framework for dealing with graphical models for decision making (e.g., Bayesian Networks, Influence Diagrams, *etc.*). The emphasis is set on high standards for performance, code quality and usability. The **aGrUM** framework is now used by academics and industrials around the world, both end-users and algorithm designers. European projects DREAM, MIDAS and SCISSOR as well as French ANR projects SKOOB, INCALIN, LARDONS and DESCRIBE also exploit **aGrUM**. It is a placeholder for its authors' research and more than fifty papers published in international conferences and journals use **aGrUM** for implementation and benchmarking. The framework's name, aGrUM, stands for "A GRaphical Universal Model" but let us be clear that aGrUM does not provide a *universal* model but offers serveral puns in the French language.

2 aGrUM features

The **aGrUM** C++ library is divided into seven modules, the majority of which relate to different graphical models:

[1] The website also contains installation instructions, the library's documentation and support.

– BN: Bayesian Networks.
– Learning: Bayesian Network learning algorithms [5], [2].
– CN: Credal Networks [3].
– FMDP: Factorized Markov Decision Processes [4].
– ID: Influence Diagrams.
– PRM: Probabilistic Relational Models [6].
– Core: common data structures and utilities.

The BN module provides flexible and efficient implementations of Bayesian Networks. Those can be read from (and written to) files of different formats (BIF, DSL, net, cnf, BIFXML, UAI). They can also be generated (randomly) from several "generators" or learnt from data using the Learning module. The **aGrUM** library allows users to define BNs using traditional Conditional Probability Tables (CPT), but also using Noisy OR or Noisy AND gates, Logit models, aggregators (and, or, max, min, exists, forall, *etc.*). In addition, for a high level of efficiency, CPTs can be encoded using different representations (arrays, sparse matrices, algebraic decision diagrams, *etc.*). Those are exploited in various inference algorithms like Lazy Propagation, Shafer-Shenoy, Variable Elimination, Gibbs sampling, *etc.*, including relevant reasoning methods.

A specific module is provided for learning the structure and/or parameters of BNs from datasets. Currently, those can be either CSV files or SQL databases. Here again, the library has been designed in order to be as flexible as possible and follows a component-based approach: structure learning algorithms are a combination of a handler for reading the database, a score among (BD, BDeu, K2, AIC, BIC/MDL) with, possibly, some additional *a priori* (smoothing or Dirichlet), a component for scheduling local structure changes and a set of constraints that the user wishes to be satisfied. The latter includes structural constraints like requiring/forbidding arcs, limiting the indegrees and imposing a partial ordering on the nodes. The learning algorithms currently implemented using this framework are greedy hill climbing, local search with tabu list and K2. BN parameters can also be learnt either by maximum likelihood or maximum a posteriori. All the learning algorithms are highly parallelized thanks to the OpenMP library.

Beside BNs, other graphical models have been implemented: Credal Networks (module CN), Factorized Markov Decision Processes (FMDP), Influence Diagrams (ID) and Probabilistic Relational Models (PRM). These modules follow the same philosophy as the BN module: high flexibility, inference efficiency, extended file format support. For instance, all these models are shipped with tailored inference algorithms, e.g., loopy propagation and Monte Carlo for CN, SPUDD for FMDPs, Shafer-Shenoy for IDs.

All the aforementioned modules rely on the core module for their data structures and common algorithms. These include classical data structures like lists, hashtables, AVL search trees, sets, heaps, *etc.*, that have been implemented in the library in such a way that they are both safe and particularly efficient. More complex data structures and algorithms are provided, like graph definitions and algorithms (including, e.g., a whole hierarchy of triangulations, notably incremental ones) and the different flavors of multidimensional tables described in the preceding page. The core of the **aGrUM** library also provides some tools used to make sure that **aGrUM**'s code satisfies the highest quality standards and is memory leak free.

3 Extensions

Beside the **aGrUM** library, the **aGrUM** framework provides a wrapper for Python: **pyAgrum** . It also implements the specific probabilistic graphical models (PGM) language **O3PRM** (`http://O3PRM.lip6.fr`).

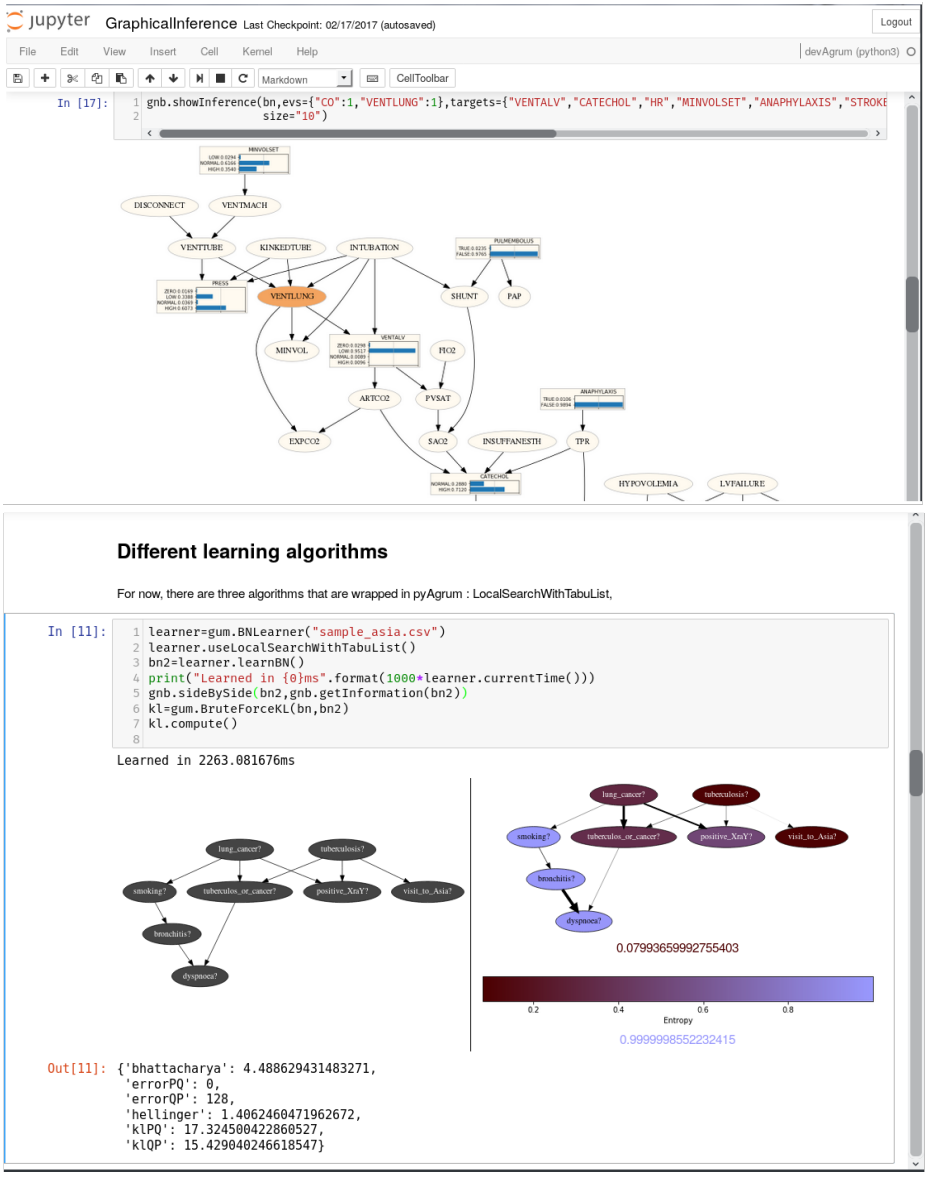

Fig. 1: Some Python notebooks using **pyAgrum**.

3.1 pyAgrum

pyAgrum is a Python wrapper for the C++ **aGrUM** library. It provides a very user friendly high-level interface for manipulating **aGrUM**'s graphical models while keeping the high performance level of the C++ library. Within Python Notebooks, **pyAgrum** can be easily used as a PGM graphical editor. Figure 1 shows such notebooks, illustrating, e.g., how BN structure learning and inferences can be performed. Note that many computations' outputs are provided graphically in order to facilitate their analysis by the users. Other learning libraries, such as **Pandas** (`http://pandas.pydata.org`), can also be used in conjunction with **pyAgrum**'s models. The latter include Bayesian Networks, Credal Networks and Influence Diagrams. All these features make **pyAgrum** a very versatile and efficient PGM package. Tutorials, demos and downloading/installation instructions can be found at `http://pyagrum.lip6.fr`.

Figure 2 is taken from one of many examples provided with the pyAgrum notebooks (notebooks are available on pyAgrum website `http://pyagrum.lip6.fr`). In this example, we use pyAgrum to iterate over 100 probabilistic inferences to produce these results. Without entering into details, the idea is to visualize the impact of evidence over one variable on another. Here the x axis represents an increasing belief that the $MINVOLSET$ variable of the classical benchmark Bayesian network *Alarm* equals $NORMAL$. The y axis indicates the posterior probability of the $VENTALV$ variable given the evidence over $MINVOLSET$. Each curve indicates the probability of a particular value of $VENTALV$ given the evidence on $MINVOLSET$.

3.2 O3PRM

The **aGrUM** library contains a specific module named PRM for Probabilistic Relational Models. They are a fully object-oriented extension of Bayesian Networks, as specified in [7]: they implement the notions of classes, interfaces, instances, attributes, reference slots, slot chains, systems, *etc*. Their object-oriented nature greatly reduces the maintenance and creation costs of complex systems with many repeated subcomponents. Highly efficient inference engines like structured variable elimination (SVE) or SVE with relevant reasoning are provided in the module. A bridge with the BN module exists that enables grounding PRMs into BNs, thereby allowing the exploitation of all the available BN-related algorithms of **aGrUM**. Finally, a domain specific language **O3PRM** has been developed to enable users to easily create PRMs.

4 Towards aGrUM 1.0

aGrUM is under active development and, even if many of its features are robust and well designed, aGrUM is still missing some fundamental algorithms and useful features that we strive to implement.

Regarding approximate probabilistic inference, we wish to add various Belief Propagation algorithms. For exact inference, we still have to parallelize and further optimize our inference engines. With these additions, aGrUM will offer a wide variety of optimized probabilistic inference algorithms, making it a complete framework for probabilistic inference.

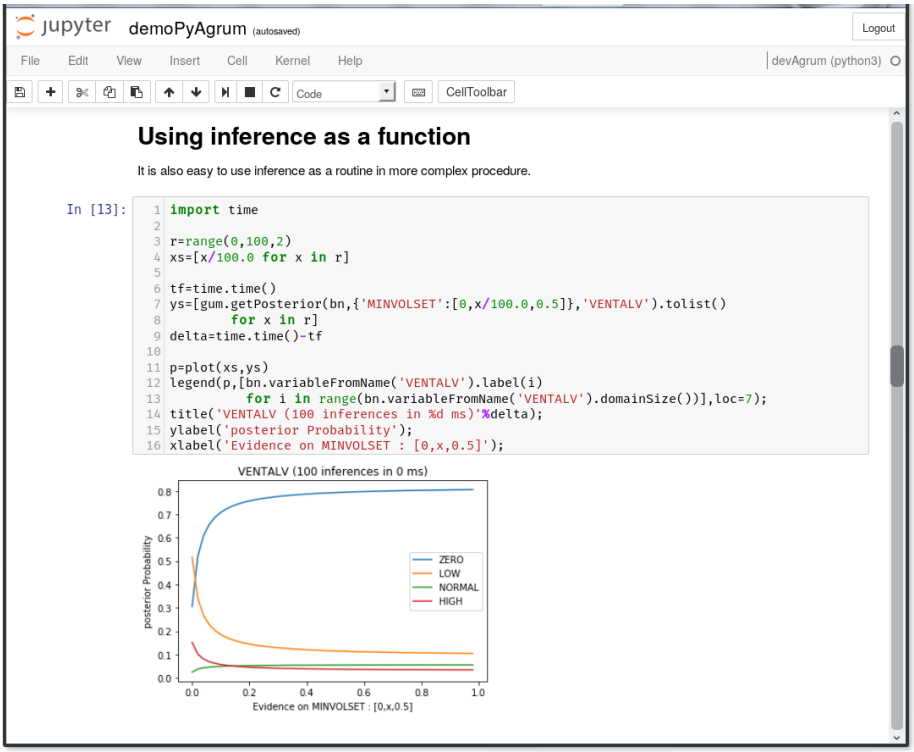

Fig. 2: pyAgrum in action: sensibility analysis

We plan to add the Expectation-Maximization (EM) algorithm and its structural counterpart SEM into aGrUM's learning module. The EM algorithm is widely used in machine learning for finding maximum likelihood or maximum a posteriori estimates of parameters. In conjunction with the learning algorithms already implemented in aGrUM, the framework will offer a broad range of methods for learning Bayesian Networks and other graphical models.

We also plan to add into aGrUM mixed discrete/continuous extensions of Bayesian networks, including, e.g., that proposed in [1], and to provide efficient learning and inference algorithms for these models.

Algorithms are not the only way we wish to improve aGrUM for a first *stable* version. Indeed, documentation and tutorials are as important as algorithms for spreading aGrUM's use. Even if we try to provide the most complete and up-to-date documentation, we still feel that its readability and examples can be improved.

As for all open source projects, aGrUM's community is very important to us and we hope to convince more people from various scientific communities to adopt aGrUM and pyAgrum as their main tool for modeling graphical models. To achieve this goal we are putting a lot of efforts in making aGrUM and pyAgrum easier to use: distributing PyPi and conda packages, porting aGrUM to Windows, talking about aGrUM in various

conferences. Another important change for aGrUM is its open source license. Currently, aGrUM is distributed under GPL2.0, which can forbid its use due to the contaminant nature of GPL2.0. We plan to switch to LGPL or another integration friendly open source license.

We hope to release version 1.0 of aGrUM in 2017. Afterwards, we plan to improve aGrUM's performance with integration of GPU support and memory optimization. We also plan to test aGrUM against other open source framework with the goal to provide the most performing graphical model framework in the open source community.

5 Conclusion

This paper has presented **aGrUM**, a powerful framework for manipulating graphical models for decision making. It is designed to be flexible, well maintained and highly efficient. The core of the framework is the C++ **aGrUM** library but wrappers like **pyAgrum** enable users to exploit **aGrUM** within high level and easy-to-use programming languages like Python.

The development of the **aGrUM** framework has not only been stimulated by academic research, it is also the result of different industrial collaborations. For instance, **aGrUM**'s **O3PRM**s are exploited in ongoing projects with EDF (the French national electricity provider) on risk management in nuclear power plants and with IBM on the exploitation of probabilities in rule-based expert systems. The BN learning module is exploited in projects with IRSN, the French Institute for Nuclear Safety, for nuclear incident scenario reconstruction. Other projects with Airbus Research and the Open Turns project use **aGrUM** for structural learning in copules with continuous variables.

References

1. Cortijo, S., Gonzales, C.: Bayesian networks with conditional truncated densities. In: Florida Artificial Intelligence Research Society Conference (FLAIRS'16). pp. 656–661 (2016)
2. Gonzales, C., Dubuisson, S., Manfredotti, C.E.: A new algorithm for learning non-stationary dynamic Bayesian networks with application to event detection. In: Florida Artificial Intelligence Research Society Conference (FLAIRS'15). pp. 564–569 (2015)
3. Hourbracq, M., Baudrit, C., Wuillemin, P.H., Destercke, S.: Dynamic Credal Networks: introduction and use in robustness analysis. In: International Symposium on Imprecise Probability: Theories and Applications (ISIPTA'13). pp. 159–169 (2013)
4. Magnan, J.: Représentations graphiques de fonctions et processus décisionnels Markoviens factorisés. Ph.D. thesis, University Pierre and Marie Curie, Paris, France (2016)
5. Magnan, J.C., Wuillemin, P.H.: Efficient Incremental Planning and Learning with Multi-Valued Decision Diagrams. Journal of Applied Logic (2016)
6. Torti, L.: Structured probabilistic inference in object-oriented probabilistic graphical models. Ph.D. thesis, University Pierre and Marie Curie, Paris, France (2012)
7. Torti, L., Wuillemin, P.H., Gonzales, C.: Reinforcing the object-oriented aspect of probabilistic relational models. In: Workshop on Probabilistic Graphical Models (PGM'10). pp. 273–280 (2010)