Proc. Fourth European Workshop on Probabilistic Graphical Models (PGM-08)
Hirtshals (Denmark), 2008, pp. 25-32.

Carmen: An open source project for probabilistic graphical models

Manuel Arias and Francisco J. Díez
Dept. Inteligencia Artificial. UNED
Juan del Rosal, 16. 28040 Madrid

Abstract

Carmen is an open-source software package for probabilistic graphical models (PGMs), which aims at being useful for different research groups and for building real-world applications. After reviewing similar projects launched in the last years, we analyze the general properties of Carmen and how it adheres to the principles of software engineering, in particular by including exhaustive documentation and systematic tests. We then describe the current state of development, i.e., which algorithms and modules have been implemented so far, and discuss the results of a preliminary analysis of its performance.

1 Introduction

Graphical probabilistic models (PGMs), such as Bayesian networks and influence diagrams, are a powerful tool for uncertain reasoning in real-world problems (Pearl, 1988). Several computer packages for building and evaluating PGMs have been developed in the last years—see K. Murphy's list at `www.cs.ubc.ca/~murphyk/Software/bnsoft.html`. In particular, many researchers, practioners, and teachers of PGMs felt the advisability of having an open source package supported by a large community of programmers. One of the main reasons is that every people or group who wished to use a PGM had to do a lot of programming from scratch. Another reason was that an open source tool would allow to compare the performance of different algorithms.

Several projects started with the objective of building such a tool. However, in our opinion, it still makes sense to try to build an open source tool for PGMs, as we discuss at length in Section 2. The tool that we present in this paper is Carmen.

The rest of the paper is structured as follows. In Section 2 we review some of the open source packages developed in the last years. Then we describe Carmen's general properties (Sec. 3), the methods and modules implemented so far (Sec. 4) and a rough premilinary comparison of Carmen's performance with three well-known tools (Sec. 5). In Section 6 we state the delivery schedule and conclude in Section 7.

In this paper we assume that the reader is familiar with PGMs. Due to the lack of space, we do not give references to standard algorithms, such as variable elimination, Hugin, lazy propagation, etc., nor for well-known computer applications, such as CVS, subversion, listserv, etc.

2 Previous work

In Table 1 we list some open source programs for PGMs.[1] In the following, we analyze four of the most successful packages: BNT, PNL, Weka, and Elvira, in order to show that, despite all these open-source projects started in the last years, it still makes sense to propose a new one that aspires to involve developers from different research groups.

[1] Some members of the *free software community* insist on the difference between "free software" and "open source software", arguing that the latter is confusing. However, for most of the people "open source" means not only that the source code is visible, but also that it can be modified and distributed; i.e., both terms are synonymous. Furthermore, we prefer "open source" because it avoids the ambiguity of "free": for instance, in K. Murphy's list, the latter does not mean "free software" but "free of charge".

	BNT	PNL	OBP	JavaBayes	BNJ	Riso	BayesLine	Weka	Elvira	Carmen
Language	Matlab	C++	Python	Java	Java	Java	Java	Java	Java	Java
License	GPL	IOSL	GPL	GPL	GPL	GPL	LGPL	GPL		LGPL
User manuals	yes	no	yes	yes	yes	yes	no	yes	yes	yes
Users list/forum	yes	no	yes	no	yes	yes	yes	yes	yes	yes
Developer manuals	no	no	no	no	no	no	no	yes	yes	yes
Developers list	yes	no	yes	no	yes	yes	yes	yes	yes	yes
Source HTML docs	no	no	no	yes	no	yes	yes	yes	yes	yes
Version control	no	no	yes	no	yes	yes	yes	yes	yes	yes
Bug tracker	no	no	yes	yes	yes	yes	yes	yes	no	yes
Start	1999	2003	2006	1996	2004	2000	2003	1993	1997	2004
Stopped	2007	2005	2007	2001	2004	2004	2003	–	–	–

Table 1: Open-source packages for PGMs. The URLs for these packages can be found in K. Murphy's list, at `www.cs.ubc.ca/~murphyk/Software/bnsoft.html`. (OBP = OpenBayes for Python. IOSL = Intel Open Source License.)

2.1 BNT and its successors

BNT (BayesNetToolbox) was built by Kevin Murphy (2001) on Matlab, a numerical matrix-oriented environment, which includes a specific programming language. The reasons for its success are the many features implemented in BNT (Bayesian networks, influence diagrams, dynamic models, learning and inference with both discrete and continuous variables...), its robustness, and the clarity of the code and its documentation.

The main limitations of BNT stem from its dependence on Matlab, which requires an expensive licence (although there is a relatively cheap student licence), is much slower than other programming languages, and has limited support for object-oriented programming.

This has motivated several attempts to build a new program on an efficient language, but none of them has been able to replace BNT. The **OpenBayes** project, initiated by R. Dybowski and K. Murphy in 2001, was abandonned some months later without even arriving at a decision about which programming language to use. In 2005, Intel Labs in Russia, with the collaboration of K. Murphy, released **PNL** (Probabilistic Networks Library), an open-source C++ library for PGMs. The project was abandonned that same year. Similarly, the project **OpenBayes for Python** was stopped before the release of version 0.2, which was announced in February 2007.

2.2 Weka

This project started in 1993 at the University of Waikato, in New Zealand, with the purpose of including several data mining methods in a single tool. The Bayesian network (BN) classifiers were implemented by Remco R. Bouckaert (2004).

The main advantages of Weka is that it has good documentation, which facilitates the development of new learning algorithms, and the possibility of empirically comparing many methods, not only those that build BNs, but also "traditional algorithms", like C4.5, nearest neighbors, etc.

2.3 gR

R is "a language and environment for statistical computing and graphics". It is an evolution of a previous language, S. gR was a subproject of R aimed at providing facilities for PGMs. The development of gR stoppped in 2003.

2.4 Elvira

Elvira (Elvira Consortium, 2002) started in 1997 as a join project of several Spanish universities. The main objectives were to foster the collaboration of the different groups working on PGMs in our country and to build a package

that be (1) a workbench for new PGM algorithms, (2) a tool for tuition, (3) a tool for building real-world applications, and (4) an open-source program. It was very successful in almost all of them. The meetings of the project were an interesting forum for the exchange of ideas. The Elvira program, whose development still continues, has served to try new methods that led to around 15 or 20 PhD theses and many papers; as a consequence, Elvira is probably the tool having more algorithms for inference and learning PGMs. As a tool for tuition, Elvira has been used by hundreds of computer science studends and postgraduate medical students, in at least 8 countries. Several applications were built or are currently under development using Elvira, most of them for medical problems, but also for other domains.

However, Elvira has not been so successful as an open-source tool: as far as we can tell, aside from the Spanish groups involved in the project, only a few Mexican researchers use it. Some European colleagues who started using it gave up because of the difficulties they encountered. In our opinion, it was a consequence of the "publish or perish" pressure: some members of Elvira argued that writing extensive documentation, reorganizing the source code, optimizing the basic data structures, improving the interface, giving support to external developers, etc., are time-consuming tasks that return little benefit.

On the other hand, an important drawback of Elvira is the lack of efficiency (see the experiments in 5), due to the fact that some of the basic data structures and algorithms do not scale up properly. Unfortunately, any attempt to optimize one class or method might cause an unpredictable number of conflicts with other classes: currently Elvira contains over 600 Java files; the biggest one occupies 390 KB and has almost 10,000 lines; the second and the third occupy around 200 KB. Finally, the program is still buggy, mainly its GUI, but debugging Elvira is not easy. Similarly, adding new functionalities is more and more a daunting task.

These are the main reasons why we decided to build a new tool, that takes profit from the many good ideas implemented in Elvira, and also from the lessons that we learnt during its development, thus trying to avoid some of the mistakes that we made.

2.5 Discussion

The end of the development of BNT and the "death" of its successors shows that it is still desirable to build an open-source program that does not depend on Matlab. Another drawback of BNT is that it has ceased to add new features—see `www.cs.ubc.ca/~murphyk/Software/BNT/whyNotSource`

Weka might be a good candidate, but in our opinion, a general-purpose package for PGMs should be developed as an independent project, rather than as an appendix of a data mining tool: even though the two fields overlap in the construction of BN classifiers, they have very different goals and needs. For instance, when building stand-alone applications, such as medical expert systems, it would be desirable to have a library that has only some inference algorithms, without the need to include all the learning methods implemented in Weka. Additionally, it is significant, in our opinion, that despite the excellent work done by R. Bouckaert, nobody in the PGM community has tried to use Weka for any purpose other than learning BNs.

Elvira has many of the features of an open source program that might be used by a large community of programmers, but unfortunately this was not one of the priorities of the project: its developers concentrated their efforts in creating new algorithms rather than in analyzing design issues (which would have made the code more efficient and easier to mantain) and writing extensive documentation (which would facilitate the task of external developers).

We have discussed in detail these packages to explain why, in spite of the excellent contributions that they have made, we think that it still makes sense to develop a new open source package. In the rest of the paper, we will try to show how the new package that we are developing might meet the needs and expectations of the PGM community.

3 Carmen's general properties

3.1 Programming language

The development language for Carmen is Java, mainly in order to allow it to run on different platforms. Since the begining we used version 1.5, which introduced new syntactical features, such as typed collections (for instance, `ArrayList<CertainClass>`) and enhanced loops, which significantly facilitate the iteration on lists.

3.2 Documentation

There are two kinds of documentation for the Carmen project: on-line comments and PDF documents. On-line documentation consists of comments included in the Java files, and works in combination with Sun's Javadoc utility,[2] which generates a set of HTML pages with many cross-references. We devoted a significant effort to carefully choosing the names of fields, methods, and variables, and to writing thorough and clear explanations.

Sun has proposed a set of tags for documenting the Java code, such as `@author`, `@param`, `@return`, etc. We have extended this collection with new tags especially intended for guaranteeing the correctness of the code. For instance, `@precondition` indicates a condition that must be fulfilled before invoking a certain method; `@postcondition` is a logical condition fulfilled after the execution of a certain method; `@paramCondition` indicates a property that the parameters must satisfy; `@invariant` refers to a logical property always satisfies by the objects of a certain class; `@frozen` means that an attribute is set by the constructor and will not be modified afterwards; and `@sideEffect` refers to secondary effects of the the execution of a method. These tags may be useful for the verification of the source code by human programmers and in the future might be adapted to be used by automatic verification tools.

Additionally, we are generating PDF documents, which offer a general overview of Carmen and contain several UML diagrams, mainly of types class, object, sequence, and components.

[2] `http://java.sun.com/j2se/javadoc`.

3.3 Testing

We have built a test suite for each class in Carmen with JUnit.[3] After introducing a modification in Carmen, we run the battery of tests in order to detect possible bugs in the program. It would be desirable that each new package contributed by an external developer come with its JUnit test, at least for those packages thay might be used by real-world applications.

3.4 Version control and support for developers

As a version control tool, we chose subversion, which offers several advantages over CVS. Following subversion's standard, Carmen's repository is organized in three directories: trunk, branch, and tag, which facilitates the collaboration of different programmers. The utility WebSVN allows Carmen's developers to receive customized notifications of changes in subversion, via RSS. In the near future we will install a distribution list (majordomo or listserv) and a bug tracking utility (probably Trac,[4] because it was designed to integrate with subversion). Later on, we will set up a web utility for the automatic registration of users and developers.

In our opinion, these facilities are a requisite for the success of any open source project.

4 Carmen's implementation

4.1 Basic data structures

The package graph implements graph operations, such as adding nodes or links. In principle it can be used for any kind of graph. In the case of a probabilistic network, each node represents a chance variable, a decision, or a utility; in a cluster tree, each node represents a set of chance variables; in the case of a Markov transition diagram, each node might represent a state of a variable.

The package networks is specific of probabilistic graphical models. Each network has an associated graph and a set of restrictions (see Figure 1). This allows to maximun flexibility for defining new types of PGMs. For example, the

[3] `http://www.junit.org`.

[4] `http://trac.edgewall.org`.

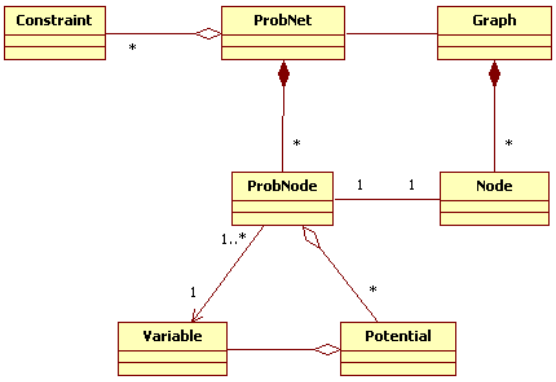

Figure 1: Main data structures for probabilistic networks.

only default restriction for a Markov network is OnlyUndirectedLinks. Both a Bayesian network and an influence diagram have the restrictions OnlyDirectedLinks and NoCycles; the former also has an OnlyChanceNodes restriction, while the latter has additional restrictions for preventing certain types of links; for instance, the children of utility nodes can only be utility nodes.

The distinction between ProbNode and Variable (cf. Fig. 1) is introduced to save memory space when a variable shared by several structures; for instance, a Bayesian network, its potentials, its moral graph (in fact, a Markov network), and a clique tree.

The package editSupport has two purposes: to allow undo/redo operations on ProbNets and to inform the listeners, i.e., the objects interested in the changes performed on a probNet. Each modification, such as adding or removing a node or a link or modifying a potential, is performed by first creating an instance of PNEdit and then passing this object to an instance of PNESupport (where PNE stands for "probabilistic network edit"), which informs the listeners, executes the "edit", and tracks it in a pushdown list to be able to undo it if necessary.

4.1.1 Edits

In general, each modification of a probabilistic network is performed by building a PNEdit. This design address three goals. First, to implement the undo/redo operations, because class PNEdit implements the Java Swing interface UndoableEdit. Second, to supervise the fulfillment of restrictions. For instance, the NoCycles restriction may register as a listener at a ProbNet to be able to veto certain additions of links. And third, to inform other classes about the changes performed to a ProbNet. For instance, an elimination heuristic might store the number of neighbors of each node, or a learning algorithms might store some scores in a cache; such information should be updated after the removal of a variable or the addition of a link. Similarly, the GUI might be interested in displaying the changes introduced by an inference algorithm (like variable elimination or arc reversal) or by a learning algorithm. This way, the possibility of adding listeners to the network offers a high degree of flexibility for fulfill these purposes and many others that future developers might imagine.

4.2 Inference

4.2.1 Purely probabilistic networks

The package inference contains methods for computing the posterior probabilities of Bayesian networks, Markov networks, and in general any kind of network that satisfies the OnlyChanceNodes restriction. The key feature of these models is that the joint probability is given by the product of a list of probabilistic potentials.

Each inference algorithm is implemented as a class that implements the Java interface EvidencePropagation, whose main methods are individualProbabilities and joinProbability. The former receives a list of variablesOfInterest and an evidenceCase, and returns a list of potentials, each one giving the posterior probability of a variable. An evidence case consists of several findings. Each finding is formed by a variable and the value that it takes. The method joinProbability receives a list of variables of interest, called query, and an evidenceCase, and returns a single potential.

Currently, we have implemented three inference algorithms for purely probabilistic networks: variable elimination, Hugin propagation, and lazy propagation. The two latter are

implemented as subclasses of ClusterPropagation, because they operate on the same structure, HuginForest. Currently Carmen offers three elimination heuristics: SimpleElimination, which chooses the node having fewer neighbors, CanoMoralElimination (Cano and Moral, 1995), and FileElimination.[5]

In the future, we will propose our students to add other heuristics and other exact and approximate algorithms, and to carry out experimental comparisons among them.

4.3 Influence diagrams

We have implemented the standard variable elimination method for influence diagrams (Jensen and Nielsen, 2007) and will later code a variable elimination method for diagrams with super-value nodes (Luque and Díez, 2004).

4.4 Learning

Carmen can learn Bayesian networks from databases using basic *search and score* techniques. The only search method implemented so far is hill climbing. The metrics implemented currently are Bayesian (which includes K2 and BDe as particular cases), cross-entropy, AIC, and MDL—see (Bouckaert, 2004; Neapolitan, 1990) for references. The file formats that Carmen can read are `dbc` (used by Elvira), `arff` (used by Weka), and Microsoft Excel.

In the future we will add other search methods, learning algorithms based on the detection of conditional independences, and learning algorithms for databases with missing values.

4.5 Markov models

One of our postgraduate students is implementing dynamic Bayesian networks, DBNs (Dean and Kanazawa, 1989), and factored Markov decision processes, MDPs (Boutilier et al., 2000), and another one will implement dynamic limited-memory influence diagrams, DLIMIDs (van Gerven et al., 2007). Later we would like to add partially observable Markov decision processes, POMDPs (Åström, 1965), which are much more difficult to solve than MDPs.

4.6 Graphical user interface (GUI)

Two undergraduate students are implementing a GUI for Carmen, whose look will be very similar to Elvira's.

5 Performance of Carmen

As a first approach to assessing the performance of Carmen, we have compared it with some well-known software tools, such as Elvira (version 0.16), GeNIE (v. 2.0), Hugin (v. 5.6), and Netica (v. 3.14).[6] We have built some test networks with a double requirement: small number of nodes and states (to make the network tractable by the demo versions of some programs) and big clusters in the clique tree, to make the measurement of time more reliable.

A solution has been the definition of networks containing $m \times n$ nodes $X_{i,j}$, m nodes R_i, and n nodes C_j. Each node R_i is a child of all the $X_{i,j}$'s (the i-th row) and each C_j is a child of all the $X_{i,j}$'s (the j-th column), with $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$. The size of the largest clique is roughly $m \times n$, which means that the complexity of a network grows extremely fast with the number of nodes.

We have made some experiments with a 6×6 network by introducing evidence on all the R and C nodes. Both GeNIE and Netica needed around 3-4 seconds to compile the network and 1.5 seconds to propagate evidence, i.e., to compute the posterior probabilities of all the X nodes. Hugin needed around 10 seconds to compile the network and the same amount of time to propagate evidence; these times were the same for the four triangulation algorithms offered by that version of Hugin. Carmen, in turn, needed 0.42 seconds to compile the network and 16.0 to propagate evidence. Given that GeNIE and Netica are all implemented in C or C++, it is not surprising that they are around 10 times

[5] FileElimination is not properly a heuristic method, because it reads the list of variables from a file. It is used for forcing Carmen to use a certain elimination order, for sake of comparison with other software tools (see Sec. 5).

[6] See `www2.sis.pitt.edu/~genie`, `www.hugin.com`, and `www.norsys.com`.

faster than Carmen, which is implemented in Java.

Elvira ran out of memory for the 6×6 network; when both tools were compared on a 5×5 network, Carmen was over 100 times faster than Elvira.

The fact that in our experiments Hugin was slower than GeNIE and Netica might due to the fact that we used an old version of that program, which seems to be based on non-efficient triangulations. In fact, when analyzing Hugin's log files, we saw that the biggest clique for that network contained 24 variables, while Carmen, which used the heuristic method by Cano and Moral, built a tree whose biggest clique contains only 22 variables. When we forced Carmen to use the same elimination ordering as Hugin and, consequently, to propagate evidence on a tree containing the same cliques, Carmen needed 50.4 seconds, i.e., it was five times slower than Hugin with the same triangulation and it was three times slower than Carmen itself with the tree built with the Cano-Moral heuristic.

However, the fact that Carmen is slower when propagating evidence can be compensated by the fact that it is able to compile the network around 7 to 10 times faster than GeNIE and Netica. This means that, instead of always using the same clique tree, we can speed up the propagation of evidence by pruning the *barren nodes* (i.e., nodes that are neither variables of interest nor parents of evidence nodes) before compiling the network, which in general speeds up significantly the propagation of evidence: the time spent in compiling the pruned network is negligible compared to the time saved in the phase of inference.

In any case, we insist, these are only very preliminary results. It is necessary to perform further experiments with diffent kinds of networks, such as as those in the *Bayesian Network Repository*,[7], comparing Carmen with other software tools.

[7] `www.cs.huji.ac.il/labs/compbio/Repository`.

5.1 Discussion

In the implementation of Carmen we have used several well-known software design patterns (Gamma et al., 2005): the undo/redo operations are based on the Command pattern, listeners an example of Observer, etc. In the same way, the GUI will be based on the arquitectural pattern MVC (Model-View-Controler). Some of these patterns have been combined and adapted to our particular needs. In a future paper we will analyze in detail how we are applying the principles and methods of software engineering in an attempt to make Carmen as efficient, robust, clearly organized, and extensible as possible.

6 Release schedule

We intend to release a beta version of Carmen before the end of 2008. As an advance, some preliminary information can be found at `http://www.cisiad.uned.es/carmen`. It is possible to browse the JavaDoc pages, linked to the source code, at `http://www.cisiad.uned.es/carmen/javadoc`. After receiving the feedback from the PGM community (hopefully), we will later release the first stable version of Carmen.

7 Conclusion

In Section 2 we showed that, even thought it may seem that there are many open source packages for PGMs, only Weka and Elvira are currently active, and we argued that it still makes sense to offer a new package that might be useful for researchers of different groups and for building real-world applications. For this reason we decided to build Carmen, a project in which we are trying to adhere to the principles of software engineering in order to make our package robust, efficient, scalable, and extensible. A particular effort has been devoted to clearly documenting the source code, by means of tools such as Javadoc and UML, not only to facilitate the work of the programmers that will use Carmen, but also as a requisite to make the software robust to future additions and changes. We have also developed an extensive battery of

tests (in JUnit) for checking the stability of Carmen under new modifications.

We have already implemented algorithms for inference in Bayesian networks and influence diagrams with discrete variables, as well as the standard "search and score" learning algorithms. Other learning methods, several types Markovian decision models, and a GUI are under development.

A preliminary evaluation of Carmen's performance seems to indicate that it is efficient enough (when compared to commercial tools) to be used in real-world applications.

We would like to present Carmen to the PGM community at the PGM-08 conference, in Hirtshals, Denmark, to attrack the interest of other researchers that might be willing to contribute to this project.

Acknoledgements

José E. Mendoza and Alberto M. Ruiz are implementing Carmen's GUI, Jorge Fernández is implementing Markov decision processes, and Jesús Oliva is implemeting some learning algorithms, all under the supervision of the authors of this paper. Manuel Luque has helped to set up a web server for Carmen.

We thank all the members of the Elvira project for all that they have taught us.

This work has been supported by the Spanish Ministry of Education and Science, under grant TIN-2006-11152.

References

[Åström1965] K. J. Åström. 1965. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205.

[Bouckaert2004] R. R. Bouckaert. 2004. Bayesian networks in Weka. Technical Report 14/2004, Computer Science Department, University of Waikato, New Zealand.

[Boutilier et al.2000] C. Boutilier, R. Dearden, and M. Goldszmidt. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107.

[Cano and Moral1995] A. Cano and S. Moral. 1995. Heuristic algorithms for the triangulation of graphs. In B. Bouchon-Meunie, R. R. Yager, and I. A. Zadeh, editors, *Advances in Intelligent Computing (IPMU-94)*, pages 98–107. Springer-Verlag, Berlin.

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

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

[Gamma et al.2005] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. 2005. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, MA.

[Jensen and Nielsen2007] F. V. Jensen and T. D. Nielsen. 2007. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, second edition.

[Luque and Díez2004] M. Luque and F. J. Díez. 2004. Variable elimination for influence diagrams with super-value nodes. In P. Lucas, editor, *Proceedings of the Second European Workshop on Probabilistic Graphical Models*, pages 145–152.

[Murphy2001] K. Murphy. 2001. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33:1–20.

[Neapolitan1990] R. E. Neapolitan. 1990. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. Wiley-Interscience, New York.

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

[van Gerven et al.2007] M. A. J. van Gerven, F. J. Díez, B. G. Taal, and P. J. F. Lucas. 2007. Selecting treatment strategies with dynamic limited-memory influence diagrams. *Artificial Intelligence in Medicine*, 40:171–186.