OVERVIEW OF FINDING THE MOST PROBABLE EXPLANATION IN BAYESIAN NETWORKS

MARCEL KAPFER

Finding a most probable explanation in a Bayesian network is an important, but also complex task. Its goal is it to find a most probable configuration for all variables in the Bayesian network that are not determined by an given evidence. To solve this there are numerous different (partly dependent, partly independent) algorithms. In this paper we try to collect some of the wide-spread algorithms. These contain stochastic local search and stochastic greedy search with their initialization algorithms, bucket elimination, simulated annealing, iterative local search, and some genetic algorithms. Additionally we give an overview over some connected and relevant topics like the Generalized Bayes Factor and restart points.

1 INTRODUCTION

Bayesian networks—as defined by Pearl in 1988 [7]—are nowadays used in a wide area. Especially in medicine [8], but also in intelligent data analysis [4] like weather forecasting, economics, sociology to name just a few. The problem is, that some computations in or with Bayesian networks (BNs) are NP-complete or even harder, finding the most possible explanation being one of them[4]. Another strength of BNs is the possibility to display them graphically, which enables an intuitive access to the stored information [8].

Finding a helpful (in the sense of most probable but also most relevant) explanation in an BN is therefore an important computational task for the earlier named areas. The difficulty in this task lies in the complexity and efficiency.

This paper focuses on different algorithms for finding explanations in Bayesian networks. This includes most probably explanation as well as most relevant explanation, as defined by Yuan, Lim, and Lu[8], and will also look at maximum a posterior (MAP). The algorithms mentioned in this paper can be separated into initialization algorithms as used by stochastic local search and others. The first area takes—next to stochastic local search (SLS)—a closer look at stochastic greedy search, where definitions from different papers are reviewed. The second area consists of generalized Bayes factor, bucket and mini-bucket elimination, stochastic simulation, iterative local search, genetic algorithms, and probabilistic crowding. Especially genetic algorithms (GA) will be viewed more in-depth.

In section 2 we will define and explain some fundamentals concepts and their notation concerning Bayesian networks and explanations. In Section 3 we will cover some initialization and explanation-finding algorithms. Some further relevant topics—concerning Bayesian networks in general, as well as specific concepts used in some algorithms—are presented in section 4.

2 PRELIMINARIES

This section contains the framework for the rest of the paper. Here we will introduce the relevant notations and concepts and also take a look at differences between the definitions of those subjects in other papers. This introductory section contains information about Bayesian networks in general, explanations in BNs, stochastic local search, and its connection with BNs.

2.1 *Bayesian Networks*

Bayesian networks are in fact directed acyclic graphs (DAG) [8]. This means they consist of nodes, which represent variables, and connections between them representing conditional dependencies [8] [2]. This makes them extremely usable because of our knowledge in other areas of graph theory and probability [3] (e.g. Bayes' theorem, marginalization, and factorization property [2]) and also easy to visualize and to understand [8]. Concerning their structure it is also possible to categorize BNs in single and multiply connected graphs. The first group can be further separated into trees and polytrees [3]. Mengshoel also states that multiply connected graphs have the highest expressiveness, followed by polytrees and then trees. In this case expressiveness means the amount of information a Bayesian network can provide.

From a more mathematically point of view a Bayesian network can be defined as follows.

Definition 1 (Bayesian Network)**.** *Let (X, E) be a DAG with $n = |X|$ nodes and $m = |E|$ edges. Further let $P = \{\Pr(X_1|\Pi_{X_1}), \dots, \Pr(X_n|\Pi_{X_n})\}$ be a set of probability function for (X, E), where $\Pr(X_i|\Pi_{X_i})$ is the conditional probability distribution for X_i (also called conditional probability table (CPT)). The* *combination of P and (X, E) as (X, E, P) is now called an Bayesian network. Let now π_{X_i} be the instantiation of the parents of X_i, given by Π_{X_i}. This follows the joint probability function*

$$\Pr(x) = \Pr(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^{n} \Pr(x_i | \pi_{X_i}). \qquad (1)$$

But why are Bayesian networks interesting? It allows us to find e.g. a most probable explanation for a given evidence. In other words: the algorithms discussed later allow you to use a Bayesian network in combination with some known values (the evidence) to find the values of the other variables in the network (the explanation).

2.2 *Explanations*

An explanation in an Bayesian network is a set of variables that can be computed if the Bayesian network is initialized with another set of known variables (the so-called evidence). While there are many different algorithms for calculating the explanation, the goal of this section is, to provide an overview of the "different" kind of explanation that can be calculated as well as other relevant basic information for dealing with the algorithms. There are also methods using singleton explanations and simplifying, but those are not able to give an explanation in more complex cases [8]. This section as well as the algorithms viewed later focus on multivariate explanations.

For us are now three different kinds of explanation relevant: maximum a posteriori (MAP), most probable explanation (MPE), and the k most probable explanations (K-MPE) [8]. The first, maximum a posteriori (MAP), takes evidence for a set of variables and returns an explanation for a given set of target variables [8]. The concept of a most probable explanation (MPE) is quite similar: the only difference is, that a explanation will be calculated for all variables and not just for a chosen set [8]. K-MPE is the same as MPE but instead of the most relevant explanation, the k most relevant explanations are calculated.

But what is a good explanation? Yuan, Lim and Li are using the words *precise* and *concise* for answering that question [8]. This means that an explanation should only contain distinct and relevant results in an as-short-as-possible form. In their paper they introduce the "most relevant explanation" (MRE) to fulfill these requirements. MRE calculates an explanation for the target variables that has the greatest generalized Bayes factor (GBF) (see 4.4) [8]. As such it is partly an extension to MPE where some irrelevant variables are purged and another relevance measure is used. To further fight against non-concise explanations two combinable solutions are proposed by Yuan, Lim and Lu: pre-pruning and post-pruning. Pre-pruning consists of removing unneeded variables before calculating an explanation and post-pruning prunes non-important variables from the calculated explanation. They further write, that these pruning methods wont work with methods using probability as a relevance measure. To use “likelihood of the evidence” as a method is also not recommended, due to the fact, that there focus is laid on preciseness but not on conciseness [8]. Yuan, Lim and Lu then use their generalized form of the Bayes factor (GBF). The Bayes factor compare two hypothesis while the generalized one is used to compare a posterior with a prior hypothesis [8].

Additionally it is worth mentioning, that finding an MPE or MRE is NP-hard [8] [1].

3 ALGORITHMS

In this section we will look at explanation-finding algorithms. This includes, next to stochastic local search and stochastic greedy search (SGS), bucket elimination, stochastic simulation, simulated annealing, genetic algorithms, and probabilistic crowding. Those algorithms are responsible for actually finding the most probable/relevant explanation (see 2.2).

3.1 *Stochastic Local Search*

Stochastic local search (SLS) is an algorithm that especially uses the knowledge in graph theory and probability research. It depends on three things: noise, an initialization algorithm (4.2) and restarts at certain points (4.1) [6]. While SLS may deliver good solution under idealistic conditions [1], its run time is highly variable, which also depends on the used initial explanation [6].

It is also possible to achieve quite good, but not optimal, solutions using heuristic. This brings the advantage of lower space and time requirements [1]. Using SLS without heuristics may bring a poor performance.

In more advanced use cases different algorithms and heuristics are combined in an so called portfolio. This allows more efficient solving of computational hard problems [5]. It is further stated that SLS is also competitive in comparison with other MPE and MAP hypothesis calculation.

While SLS can be seen as an own algorithms there are also various algorithms based on the concepts of SLS which promises a more optimal MPE computation. SLS can therefore also rather be viewed as a group of algorithms [1] which consists for example of stochastic greedy search 3.2 and stochastic simulation 3.3.

3.2 *Stochastic Greedy Search*

As already mentioned, stochastic greedy search (SGS) is a stochastic local search algorithm for finding MPEs in BNs [6]. Its strength lies in the initialization step, which makes it competitive to other approaches. A few algorithms for this initialization are listed in 4.2, but GraphBDP/GraphFDP seems to be preferred [6].

Concerning its structure, SGS has some similarities with the WalkSAT family which again shows the benefits of BNs through their similarities with some other well researched and known areas [6].

An important constant in SGS is MAX-FLIPS, which sets the amount of flipping a variable before the computation is restarted. If the calculation is restarted a new start explanation is generated using the favored algorithm [6].

For "storing" the algorithms in use SGS has two portfolios. The first for greedy and noise search algorithms and the other for initialization algorithms [6].

The duration of the calculation is given using two different techniques. First the run length distribution, which can be calculated from reviewing the algorithm and second the run time distribution, which is the needed time. While the first is obviously only dependent on the algorithm, the second one depends heavily on the used hardware and software [6].

There are also two different kinds of SGS algorithms, which are named SimpleSGS and OperatorSGS. OperatorSGS is like described above: it uses portfolios and some way for choosing in those which can be based on round-robin or on probability. SimpleSGS on the other hand does not have portfolios and works only with previously set algorithms [3] [5].

In another source ([3]) it is also stated that SGS is an generalization of GSAT and share similarities to stochastic simulation and iterative local search. Mengshoel writes also the controversy opinion, that SGS is not an SLS algorithm but independently and concurrently created with SLS, while sharing some concepts and ideas.

3.3 *Stochastic Simulation*

Next to stochastic local search and stochastic greedy search, stochastic simulation (also known as Gibbs Sampling) is another stochastic algorithm . It is a quite simple concept based on random initial assignments, where each sample is computed by changing a variable from the previous one. This variable may be picked on three different ways: at random, by a schedule, or—the most complex variant—based on a probability distribution and the neighbors current values. An stochastic simulation returns either an sample with the highest probability or the most frequent one as a result [1].

3.4 *Simulated Annealing*

Following the listing of stochastic algorithms for MPE finding, another one is named simulated annealing. Simulated annealing combines hill-climbing and random selection for changing variables [1].

Simulated annealing starts with a random assignment of the variables (A), a so-called temperature (t) and a cost function (h). The iteration process is repeated until t is 0 and consists of the following steps. First a neighbor for A is selected, called A'. If the cost function of A' is higher than the one of A (that means, if $h(A') > h(A)$) then A will be set to A', otherwise this will only be done with the probability $e^{\frac{h(A)-h(A'))}{t}}$. Finally t is reduced and the process starts over [1].

3.5 *Bucket Elimination*

Next to the reviewed stochastic algorithms there are also some, which are not stochastic. One of them is bucket elimination [1].

A concept for this algorithm is called min-with variable ordering for ordering the graph from last to first. In the next step the remaining graph with non-empty variables is viewed and the one with the fewest neighbors chosen. For each variable a so-called bucket is created and at the initialization phase specific probability matrices are placed into the variable buckets [1].

Following that initialization phase, every bucket is processed in a first phase on its own from last to first creating a new function of the existing probability matrices and eliminating the bucket. The second phase then consists of creating the solution. For this a value is assigned to the value following the existing order [1].

There is also an process called mini-bucket elimination which can approximate the result of the algorithm described above. Using the mini-bucket elimination algorithm subsets all current bucket functions are put into mini-buckets which are then processed the same [1].

3.6 *Iterative Local Search*

Another approach for finding the most probable explanation in a Bayesian network is iterative local search which uses probabilistic hill-climbing (also known as neighborhood search). This form of hill-climbing combines next-ascend hill-climbing with random-mutation hill-climbing. Iterative local search has the limitation to stop at local maxima. The wished, global maxima can be found after some iterative steps [3].

3.7 *Genetic Algorithms*

Genetic algorithms (GAs) are another family which are a bit different than the others. Their concept is inspired by nature and relies on stochastic and population-based search. GAs are especially used for adaption, search, optimization, and complex learning. Generally the comparison happens through evaluating a population representing candidate for its fitness (survival of the fittest replacement). New "populations" are generated using operators like crossover and mutation and are based on the previous one. Their robustness, simplicity and generality make them quite interesting. As a representative for this family we will focus on the simple genetic algorithm (SGA) [3].

At the beginning the algorithm starts with a random population of the "first generation" and iterates over each individual in the old population (the current one is at the position j) until the amount of generations reaches a given maximum. Each individual holds an binary alphabet called "chromosomes". During each (inner) iteration two additional individuals will be selected based on their fitness. The next step is called "crossover" where two given chromosomes are crossed over by a certain probability ($\Pr(\text{Crossover})$). The results are added to the new population at j and $j+1$. If the crossover happens, only one position will be affected. After the crossover-step the chromosomes will be mutated by another probability ($\Pr(\text{Mutation})$). The mutation function are also applied on j and $j+1$. Finally the fitness of the new individuals j and $j+1$ is calculated [3].

3.8 *Probabilistic Crowding*

Probabilistic Crowding follows a slightly different approach for finding an MPE. Its goal is not to find the one, but multiple one. It thereby follows two concepts. The first one is converging to multiple, highly possible, but different solutions and the second one is to find one solution (if required) through slow down the convergence [3].

Those concepts are—for example—realized with the deterministic crowding algorithm, which has one problem: the convergence is not analyzed and it cannot be viewed what is computed.

Probabilistic crowding fixes that problem by using a probabilistic instead of deterministic acceptance function. For this algorithms there are several variants: one using mutation, one using crossover and a combination of those two.

4 FURTHER RELEVANT TOPICS

Next to the above listed algorithms there are some interesting and relevant topics which are important to look at. Some of them (like restart points and the generalized Bayes factor) are used among various algorithms while other (like complexity) are general topics.

4.1 *Restart Points*

Restart points quite important for stochastic local search and stochastic greedy search as a termination clause. The correct use of the restart points can increase efficiency significantly. If the restart point also known as MAX-FLIPS is too low, then much time will be wasted through initialization. The goal is, to minimize the expected number of operations. For that, it is relevant to know an expected number of operation. Mengshoel, Wilkins and Roth start by defining a total number of operations in a try with $Z = X + Y$. X being number of initialization operations and Y the number of flips. In connection with the portfolio, those variables are generalized as $Z(a, m)$ where a is the algorithm and $m = \text{MAX}-\text{FLIPS}$ ($m = \infty$ means no restart). Further is the probability for success and failure defined as $p_s(a, m) := \sum_{i=0}^{m} \Pr(Y(a, \infty) = i)$ and $p_f(a, m) := 1 - p_s(a, m)$. The expected number of operations can now be calculated with

$$E(Z(a, m)) = \frac{mp_f(a, m) + \sum_{i=0}^{m} i\Pr(Y(a, \infty) = i) + 1}{p_s(a, m)}. \tag{2}$$

and the expectation-optimal MAX-FLIPS with

$$m'(a) = \text{argmin}(E(Z(a, m))), m \in \mathbb{N} \tag{3}$$

[6].

4.2 *Initialization Algorithms for Stochastic Algorithms*

This subsection focuses on initialization algorithms for stochastic local search (an algorithm for finding an explanation). The algorithms reviewed are unified initialization, forward simulation and—perhaps the most interesting—dynamic programming with GraphBDP and GraphFDP. The task of an initialization algorithm consists of pre-filling all non-evidence nodes [6].

"Unified initialization" (UI) is a initialization algorithm for stochastic local search. UI assigns at random and independently. For each non-evidence node every possible state has an equal probability of being chosen for initialization. This algorithm is in use in iterative local search, simulated annealing, and genetic algorithms, but also in more general SAT solvers. UI has a complexity of $O(n)$ since it needs to visit every node exactly one time to check if it is a non-evidence node and in case act on it. [6].

Another initialization algorithm is forward simulation, which starts by dividing all nodes of the network in root and non-root ones [6]. The root nodes are first initialized depending on the initial explanation or the prior distribution [3]. This initialization is done independently. Following that the non-root and non-evidence nodes are initialized at random following their conditional distribution.

A third approach for finding a starting point is called dynamic programming and includes two generalized algorithms: GraphBDP and GraphFDP. The generalization lies in splitting the graph into independent trees. [6]. These trees are then initialized using a Viterbi approach and GraphBDP/GraphFDP then collects the subexplanations. These algorithms may find quite good and efficient algorithms for some tree-like BNs.

4.3 *Complexity*

The complexity is certainly an quite interesting and relevant topic. As already mentioned MPE-finding is NP-hard [2], but there are some efficient algorithms [1], even if those are just optimized for specific cases. Mengshoel also further states, that the expressiveness of a BN correlates with its computational complexity.

It is further said, that MPE-finding is generally not tractable because of the exponential amount of variable assignments, but still possibly, when the treewidth of the graph is low or the MPE-probability is high. Kwisthout mentions further, that $P = NP$ would be necessary, so that MPE-finding can be approximated [2].

4.4 *Generalized Bayes Factor*

For some algorithms and for comparison sometimes it is necessary to have a value for a certain hypothesis. The generalized Bayes factor (GBF) (as presented by Yuan [8]) is made to solve that task. Yuan describes the GBF as a „weight of evidence“ and uses it to compare a hypothesis to an alternative one to figure out, which is more relevant. If those hypothesis are statistical the factor is more likely called likelihood ratio. The name Bayes factor is used, when there are unknown variables. The problem with the "normal" Bayes factor is, that it can only handle two hypotheses at a time.

This is why Yuan introduces the generalized Bayes factor [8], which is not limited in that way. Another advantage is that it can be used to compare prior and posterior probabilities of a singly hypothesis.

The GBF can be calculated using the following formula [8]:

$$\mathrm{GBF}(x;e) = \frac{P(e|x)}{P(e|\bar{x})} = \frac{P(x|e)(1-P(x))}{P(x)(1-P(x|e))}. \tag{4}$$

5 FURTHER WORK AND CONCLUSION

This paper provides an overview over some algorithms for finding an most possible or relevant explanation. It therefor takes also an look at some preliminaries (like: what is an explanation?) and mentions further relevant aspects (e.g. complexity) which are quite relevant for a deeper understanding.

Due to limited time and space, not all existing algorithms for MPE finding could be reviewed. To created an bigger overview would be a possible starting point, but it would also be interesting to take a closer look at the algorithms in pseudo-code and especially their complexity. But perhaps the largest area that could be covered is a practical comparison of the algorithms. While there are some test results, made under different circumstances, it would be certainly a great opportunity to make experiments with modern software and hardware for all listed (maybe even more) MPE-finding algorithms.

This would maybe also show, that each listed algorithm has some special cases, where it is quite efficient and others where it is not. The other question in this case is also, if it is wise to concentrate on efficiency and complexity or if the quality of the results is more important.

REFERENCES

[1] Kalev Kask and Rina Dechter. “Stochastic Local Search for Bayesian Networks”. In: 1999.

[2] Johan Kwisthout. “Most probable explanations in Bayesian networks: Complexity and tractability”. In: *International Journal of Approximate Reasoning* 52 (Aug. 2011), pp. 1452–1469. DOI: 10.1016/j.ijar.2011.08.003.

[3] Ole J. Mengshoel. *Efficient Bayesian Network Inference: Genetic Algorithms, Stochastic Local Search, and Abstraction*. Urbana, Illinois, 1989.

[4] Ole J. Mengshoel. “Understanding the role of noise in stochastic local search: Analysis and experiments”. In: *Artifcial Intelligence* 172 (2008), pp. 955–990. DOI: 10.1016/j.artint.2007.09.010.

[5] Ole J. Mengshoel, Dan Roth, and C. Wilkins. “Portfolios in Stochastic Local Search: Efficiently Computing Most Probable Explanations in Bayesian Networks”. In: *J Autom Reasoning* 46 (2011), pp. 103–160. DOI: 10.1007/s10817-010-9170-5.

[6] Ole J. Mengshoel, David C. Wilkins, and Dan Roth. “Initialization and Restart in Stochastic Local Search: Computing a Most Probable Explanation in Bayesian Networks”. In: *IEEE Transactions on Knowledge and Data Engineering* 23.2 (Feb. 2011), pp. 235–247. DOI: 10.1109/TKDE.2010.98.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann Publishers, 1988. ISBN: 9781558604797.

[8] Changhe Yuan, Heejin Lim, and Tsai-Ching Lu. “Most Relevant Explanation in Bayesian Networks”. In: *Journal of Artifcial Intelligence Research* 42 (Nov. 2011), pp. 309–352.