A unifying framework for exact and approximate Bayesian inference

Kamel Mekhnacha, Linda Smail, Juan-Manuel Ahuactzin, Pierre Bessière, Emmanuel Mazer

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A unifying framework for exact and approximate Bayesian inference

Kamel Mekhnacha — Juan-Manuel Ahuactzin — Pierre Bessière — Emmanuel Mazer — Linda Smail

N° 5797

Janvier 2006

Thèmes COG et NUM

ISSN 0249-6399 ISRN INRIA/RR--5797--FR+ENG

A unifying framework for exact and approximate Bayesian inference

Kamel Mekhnacha * , Juan-Manuel Ahuactzin * , Pierre Bessière * ,
Emmanuel Mazer * , Linda Smail†

Thèmes COG et NUM — Systèmes cognitifs et Systèmes numériques
Projet E-MOTION

Rapport de recherche n° 5797 — Janvier 2006 — 44 pages

Abstract: We present a unifying framework for exact and approximate inference in Bayesian networks. This framework has been used to design a general purpose Bayesian inference engine, called "ProBT", for probabilistic reasoning and incremental model construction.

This paper is not intended to present ProBT but to describe its underlying algorithms for both exact and approximate inference problems.

The main idea of the ProBT inference engine is to use "probability expressions" as basic bricks to build more complex probabilistic models incrementally. The numerical evaluation of these expressions is accomplished just-in-time. Indeed, a probability expression is a symbolic representation of an inferred distribution. Probability expressions are manipulated in the same way as numerical distributions, such as probability tables and standard parametric distributions. A probability expression is said to be an "exact" or an "approximate" depending on the inference method (exact or approximate) used to evaluate it.

For exact inference, we describe the "Successive Restrictions Algorithm" (SRA). Given a target distribution, the goal of the SRA is to construct a symbolic evaluation tree by finding a corresponding sum/product ordering that takes into account the computational constraints of the application (computation time and/or memory size). The optimality considerations of the SRA are also discussed.

For the approximate inference part, several approximation schemes and the corresponding algorithms are presented. An original algorithm called "MCSEM" (for Monte Carlo Simultaneous Estimation and Maximization) is proposed. This algorithm aims at solving the problem of maximizing a posteriori high-dimensional distributions containing (in the general case) high-dimensional integrals (or sums).

Key-words: bayesian networks, exact inference, approximate inference, optimisation, genetic algorithms, Monte Carlo

* Email: prenom.nom@inrialpes.fr
† Email: linda.smail@univ-mlv.fr

Un cadre unificateur pour l'inférence Bayesienne exacte et approchée

Résumé : Nous présentons un cadre unificateur pour l'inférence Bayesienne exacte et approchée dans les réseaux Bayesiens. Ce cadre a été utilisé pour la conception d'un moteur d'inférence Bayesienne généraliste nommé "ProBT". Ce moteur a comme objectif d'automatiser le raisonnement probabiliste et de faciliter la construction incrémentale des modèles.

Cet article n'a pas pour objectif de présenter ProBT, mais de décrire ses algorithmes sous-jacents aussi bien pour le problème de l'inférence exacte que pour celui de l'inférence approchée.

L'idée principale de ProBT est d'utiliser la notion d'"expressions de probabilités" comme briques de base pour la construction incrémentale de modèles probabilistes plus complexes. Une expression est une représentation symbolique d'une distribution inférée (conditionnelle ou non conditionnelle) qui n'est évaluée qu'à la demande. En effet, une expression de probabilités est manipulée de la même manière que les distributions numériques telles que les tables de probabilités et les distributions paramétrique standards. Elle est dite "exacte" ou bien "approchée" selon la méthode d'inférence (exacte ou bien approchée) utilisée pour l'évaluer.

Pour l'inférence exacte, nous décrivons l'algorithme appelé "SRA" (Successive Restrictions Algorithm). Etant donné une distribution cible, le but de l'algorithme SRA est de construire un arbre d'évaluation symbolique représentant l'ordre des sommes et des produits. Cet ordre doit prendre en compte les contraintes calculatoires de l'application (en termes de temps de calcul et/ou de mémoire utilisée). Nous discutons également l'optimalité de cet algorithme.

Pour l'inférence approchée, plusieurs niveaux d'approximations sont présentés. Nous décrivons en particulier un algorithme original appelé "MCSEM" (Monte Carlo Simultaneous Estimation and Maximization). Le but de cet algorithme est de résoudre le problème de maximisation des distributions a posteriori de grandes dimensions, nécessitant (dans le cas général) le calcul d'intégrales (ou somme) dans des espaces de grandes dimensions.

Mots-clés : réseaux bayesiens, inférence exacte, inférence approchée, optimisation, algorithmes génétiques, Monté-Carlo

1 Introduction

ProBT is a C++ library for developing efficient Bayesian software. This library has two main components: (i) a friendly Application Program Interface (API) for building Bayesian models and (ii) a high-performance Bayesian inference and learning engine allowing execution of the probability calculus in exact or approximate ways. This paper is not intended to describe ProBT. It will only focus on its underlying inference algorithms.

The aim of ProBT is to provide a programming tool that facilitates the creation of Bayesian models and their reusability. Its main idea is to use "probability expressions" as basic bricks to build more complex probabilistic models. The numerical evaluation of these expressions is accomplished just-in-time: computation is done when numerical representations of the corresponding target distributions are required. This property allows designing advanced features such as submodel reuse and distributed inference. Therefore, constructing symbolic representations of expressions is a central issue in ProBT.

Let $\mathbf{X}$ be a finite set of random variables. We denote by $P(\mathbf{X})$ the joint distribution on the conjunction of the variables in $\mathbf{X}$.

Consider a Bayesian network (BN) relative to a set $\mathbf{X}_U$ of random variables, a subset of target variables $\mathbf{X}_L \subset \mathbf{X}_U$, and another subset (possibly empty) of evidence variables $\mathbf{X}_R \subset \mathbf{X}_U$ (with $\mathbf{X}_L \cap \mathbf{X}_R = \emptyset$). A probability expression, corresponding to a target (inferred) distribution $P(\mathbf{X}_L \mid \mathbf{X}_R)$, is basically a symbolic evaluation tree allowing us to numerically compute this distribution for all possible values $\mathbf{x}_L$ and $\mathbf{x}_R$ of $\mathbf{X}_L$ and $\mathbf{X}_R$. This inferred expression can be used in numerous ways:

- It can be evaluated numerically for all (or some) values of $\mathbf{X}_L$ and $\mathbf{X}_R$ to obtain a numerical representation of $P(\mathbf{X}_L \mid \mathbf{X}_R)$.
- It can be used as an objective function to be maximized, to find the Maximum A Posteriori (MAP) solution of the inferred distribution:

  $$\mathbf{x}_L{}^* = \arg\max_{\mathbf{X}_L} P(\mathbf{X}_L \mid \mathbf{X}_R = \mathbf{x}_R),$$

  for a given evidence value $\mathbf{x}_R$, possibly using optimization heuristics. This function may also be used to sample $P(\mathbf{X}_L \mid \mathbf{X}_R = \mathbf{x}_R)$ using indirect sampling methods such as MCMC techniques.
- It can be used as an elementary distribution (possibly after renaming variables) when constructing another Bayesian network. This property is very useful for incremental probabilistic modeling and models' reusability (see Figure 1).
- It can also be used to replace an a priori distribution in the same BN ($P(\mathbf{X}_L)$ for example) with the a posteriori distribution $P(\mathbf{X}_L \mid \mathbf{X}_R = \mathbf{x}_R)$, after incorporating a given observation $\mathbf{X}_R = \mathbf{x}_R$. This property is especially useful for dynamic systems descriptions (Bayesian filters for example) (see Figure 2).

Constructing a probability expression $E(\mathbf{X}_L, \mathbf{X}_R) \equiv P(\mathbf{X}_L \mid \mathbf{X}_R)$ requires symbolic simplifications. The aim of these simplifications is to find an appropriate organization of the computation. By "appropriate", we mean that the computational complexity required to numerically evaluate this expression must satisfy the time and memory constraints of the application. In selecting the computational scheme to be used, we must be able to answer the following question:

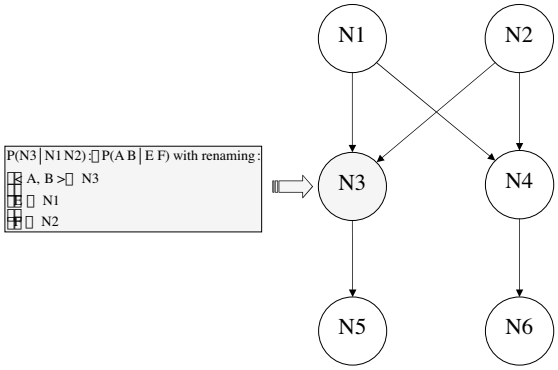
Is it possible to find a computation sequence that will build an exhaustive and exact probability table corresponding to $E(\mathbf{X}_L) \equiv P(\mathbf{X}_L \mid \mathbf{X}_R = \mathbf{x}_R)$ *and satisfy the computational constraints of the problem?*

These constraints may affect the computation time and/or the memory size required to build this table. The problem of constructing an exact and exhaustive probability table of the target distributions is known as the "exact Bayesian inference" problem. In this case, finding efficient algorithms that reduce the computational cost of building such tables is a central issue in making the exact inference problem more tractable. The problem of exact inference is NP-hard (Cooper, 1990). However, various algorithms have been proposed to make this problem more tractable.

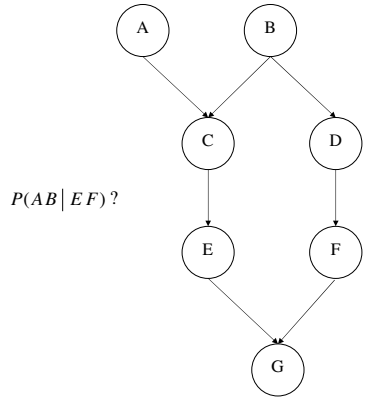
Unfortunately, for complicated real-world applications, satisfying the time/memory constraints is seldom possible when using exact calculation. This is especially the case for probabilistic problems involving a large number of variables and/or dependencies, and/or defined on variables taking values in a huge (or infinite for continuous variables) set of states. In this case, exact inference becomes intractable and approximation methods must be used. This case is known as the "approximate Bayesian inference" problem.

ProBT provides original and efficient algorithms for both exact and approximate classes of problems. It offers a unifying approach and a user-friendly Application Program Interface (API) for constructing exact, approximate, and mixed (exact/approximate) probabilistic models. To give a better idea of the computational issues of exact and approximate Bayesian inference and to introduce the viewpoint we shall take in this paper, we present the following example.

Example Consider the Bayesian network in Figure 1b. The joint distribution corresponding to this BN is:



(a)



(b)

Figure 1: A Bayesian network using an inferred distribution. The distribution $P(N3 \mid N1\ N2)$ in (a) is obtained from the Bayesian network in (b). This distribution corresponds to the inferred distribution $P(A\ B \mid E\ F)$ after renaming the variables $\langle A, B\rangle$ to $N3$ and E, F to $N1$, $N2$, respectively.

$$P(A\ B\ C\ D\ E\ F\ G) = \\ P(A)\ P(B)\ P(C \mid A\ B)\ P(D \mid B)\ P(E \mid C)\ P(F \mid D)\ P(G \mid E\ F). \qquad (1)$$

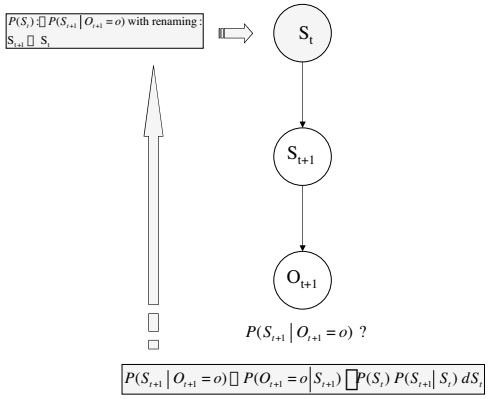


Figure 2: Replacing the a priori distribution $P(S_t)$ by the a posteriori distribution $P(S_{t+1} \mid O_{t+1} = o)$ in a Bayesian filter.

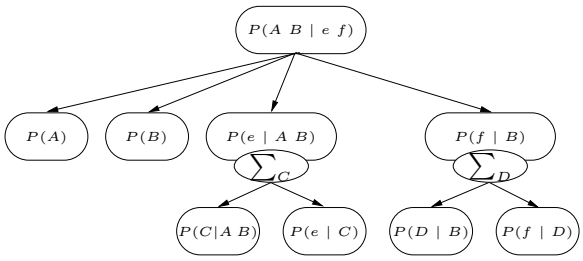


Figure 3: The evaluation tree corresponding to Equation 3.

Suppose we are given the task of computing the target distribution $P(A\ B \mid E\ F)$ for a given evidence value $(E = e)$ and $(F = f)$. This distribution will be denoted $P(A\ B \mid e\ f)$ (i.e., we will use upper case letters for variables and lower case letters for values in variables' domains).

The exact inference problem

Suppose first that A, B, C, D, E, F, and G are variables with the same state space $\mathcal{S}$ with n elements. We have the task of computing the probability table corresponding to $P(A\ B \mid e\ f)$.

By marginalizing out the variables C, D, and G from the joint distribution (Equation 1), this target distribution can be written as:

$$P(A\ B \mid e\ f) \propto$$
$$\sum_{C,D,G} P(A\ B\ C\ D\ e\ f\ G) =$$
$$\sum_{C,D,G} P(A)\ P(B)\ P(C \mid A\ B)\ P(D \mid B)\ P(e \mid C)\ P(f \mid D)\ P(G \mid e\ f). \qquad (2)$$

Suppose in this example that we have a constraint on the computation time required to build this table. We must answer the following questions:

- How many arithmetic operations (additions and multiplications) are required for an exact computation of this table?
- Does this computation cost satisfy our application time constraints?
- If the answer for the previous question is no, what can we do to satisfy these constraints?

If we proceed in the naive way (using Equation 2), we find that for each of the n^2 values of $\mathcal{S} \times \mathcal{S}$ (set $\{A, B\}$), we must evaluate n^3 terms in the sum (set $\{C, D, G\}$). Each term requires six multiplications and one addition, so the total number of operations will be $n^2 \times n^3 \times 7 = 7n^5$.

It is clear that proceeding in the above way is not efficient. This computational cost may be reduced using:

1. a factorization phase to organize the sums,
2. a caching phase to compute a probability table for each factor.

Using the distributive law, Equation 2 may be rewritten as (factorization phase):

$$P(A\ B \mid e\ f) \propto$$
$$\sum_{C,D,G} P(A)\ P(B)\ P(C \mid A\ B)\ P(D \mid B)\ P(e \mid C)\ P(f \mid D)\ P(G \mid e\ f) =$$
$$P(A)\ P(B) \left(\sum_{C} P(C \mid A\ B)\ P(e \mid C)\right) \left(\sum_{D} P(D \mid B) P(f \mid D)\right) \left(\sum_{G} P(G \mid e\ f)\right) =$$
$$P(A)\ P(B) \left(\sum_{C} P(C \mid A\ B)\ P(e \mid C)\right) \left(\sum_{D} P(D \mid B) P(f \mid D)\right). \qquad (3)$$

Figure 3 is the evaluation tree corresponding to Equation 3.

Using Equation 3, we can simplify the computation of the table of $P(A\ B \mid e\ f)$ as follows (caching phase).

1. First we compute a table of the function:

$$\alpha(A,B) = \sum_C P(C \mid A\ B)\ P(e \mid C).$$

   Note that $P(C \mid A\ B)\ P(e \mid C) = P(C\ e|\ A\ B)$ then $sum_C P(C \mid A\ B)\ P(e \mid C) = P(e \mid A\ B)$. The computation of $\alpha(A,B)$ requires n^3 terms, each requiring one addition and one multiplication, so the total number of operations to create a table of $\alpha(A,B)$ is $2n^3$.

2. Then we compute a table of the function:

$$\beta(B) = \sum_D P(D \mid B)P(f \mid D) = P(f \mid B),$$

   which requires n^2 terms, each term requiring one addition and one multiplication, so the total number of operations to create the table of $\beta(B)$ is $2n^2$.

3. Finally, we compute for each of the n^2 values of $\mathcal{S} \times \mathcal{S}$ the product $P(A\ B \mid e\ f) = P(A)P(B)\alpha(A,B)\beta(B)$, which requires $3n^2$ multiplications.

The total number of arithmetic operations to create the table of $P(A\ B \mid e\ f)$ using Equation 3 is only $2n^3 + 2n^2 + 3n^2 = 2n^3 + 5n^2$, as compared to $7n^5$ for the direct naive method using Equation 2. For instance, with $n = 10$ we would require $2n^3 + 5n^2 = 2500$ operations versus $7n^5 = 700,000$.

The simplification in this example was easy to accomplish and the computational gain is relatively modest compared with more complicated instances. This is especially the case when trying to compute various target distributions for which simplifications can be harder to find, although the gain can be dramatic.

The algorithm used to build this expression (evaluation tree) is a main component of ProBT for single-target exact inference problems. This algorithm will be described in Section 3.

The approximate inference problem

Many real-world systems are too complex to allow exact computation. This complexity has two main causes:

- *The number of arithmetic operations is too large.* For example in Equation 3, when the cardinality n of $\mathcal{S}$ is large enough to violate the application's constraints.
- *The inferred expression involves real-valued functions that are not analytically integrable.* For example, consider Equation 3 and suppose now that A, B, C, D, E, F,

and G are continuous variables. By marginalizing out the variables C, D, and G from the joint distribution, we obtain:

$$P(A\ B \mid e\ f) \propto \int P(A\ B\ C\ D\ e\ f\ G)\ dC\ dD\ dG = P(A)P(B) \int P(C \mid A\ B)P(D \mid B)P(e \mid C)P(f \mid D)\ dC\ dD \quad (4)$$

The previous expression can result in an intractable numerical integration problem.

The alternative to the exact computation is to compute an approximate solution of the original problem.

Obviously, different approximations will lead to different solutions. Therefore, the expertise of the programmer is crucial in such a task in providing the approximation scheme to be used.

For a given abstract expression, ProBT provides four schemes (levels) of approximation:

- *Integrals (sums) estimation*: Evaluating the abstract expression for a given point of the target space by computing a more or less accurate numerical estimation of the integrals (sums) involved by the expression. In Equation 4, this concerns the estimation of the integral $I(A, B) = \int P(C \mid A\ B)\ P(D \mid B)\ P(e \mid C)\ P(f \mid D)\ dC\ dD$ for given values for A and B.

- *Numerical representation of a posteriori distributions*: Constructing a numerical representation of the distribution by selectively visiting high-probability regions of the target space. For our previous examples (Equation 3 and Equation 4), the task is to build a numerical representation $T(A, B) \equiv P(A\ B \mid e\ f)$ using a finite set of points of the target space. This numerical representation must have the capacity of generalization so that the distribution can be evaluated at each point in the target space.

- *A posteriori distributions sampling*: Sampling the target distribution $P(A\ B \mid e\ f)$ using MCMC methods to generate a sample set of N points $\{(a^{(i)}, b^{(i)})\}_{i=1}^{N}$.

- *A posteriori distributions maximization*: Finding the MAP solution of the problem (i.e., $\langle a^*, b^* \rangle = \arg\max_{A,B} P(A\ B \mid e\ f)$), using optimization heuristics instead of computing $P(A\ B \mid e\ f)$ for all possible states of the target space.

A set of approximation algorithms is proposed for each level, so time/memory constraints in an application can be taken into account in a simple way. These four approximation levels and the underlying algorithms constitute the approximate inference part of ProBT and will be described in Section 5.

Organization of paper This paper is organized as follows. We first report related work for exact inference in Section 2. Then, in Section 3, we present the exact inference calculation in ProBT, the Successive Restrictions Algorithm (SRA), and discuss its optimality considerations. Section 4 presents a brief review of the related techniques and approaches for the approximate inference problem. The approximate inference algorithms of ProBT are described in Section 5. Four schemes for approximating high-dimensional problems and the corresponding algorithms are presented. We will especially detail an original algorithm called "MCSEM" (for Monte Carlo Simultaneous Estimation and Maximization). This algorithm aims at solving the problem of maximizing a posteriori high-dimensional distributions containing (in the general case) high-dimensional integrals (or sums). Finally, conclusions and perspectives are presented in Section 6.

2 General purpose algorithms for exact Bayesian inference

Even though the problem of exact inference is NP-hard (Cooper, 1990), various algorithms have been proposed to deal with its exponential complexity. The aim of this section is to give a brief review of general purpose algorithms for the exact Bayesian inference problem.

Depending on the target distribution or distributions to be updated, we can classify belief update algorithms into two main classes. The first class, called "belief propagation algorithms", aims to construct the marginal distributions on all variables (nodes) efficiently, given values of another set of variables (called "evidence variables").

The objective of the second class, called "variable elimination algorithms", is to compute an arbitrary joint distribution on a subset of variables efficiently, given a set of evidence variables.

2.1 Belief propagation algorithms

Belief propagation algorithms are mainly generalizations of the message propagation algorithm originally proposed by Pearl in (Pearl, 1982). This algorithm has polynomial complexity but can only be applied to poly-tree shaped networks (graphs with no cycles).

Two main approaches have been proposed to generalize this message- passing algorithm for arbitrary (multiply connected) networks. These two approaches are described below.

2.1.1 The "loop cutset conditioning" approach

"Loop cutset conditioning" was proposed by Pearl (Pearl, 1986, 1988). Its main idea is to convert a network to a singly connected one by instantiating a selected subset of nodes referred to as a "loop cutset". The resulting singly connected network is solved using the poly-tree message propagation algorithm and the results of each instantiation are then combined by weighting them using their prior probabilities in the message-passing process. The

complexity grows exponentially with the size of the loop cutset. It is thus important to minimize the size of the loop cutset, which is an NP-hard problem.

2.1.2 The "junction tree" approach

The second generalization of the "message propagation algorithm" was proposed by Lauritzen and Spiegelhalter in (Lauritzen & Spiegelhalter, 1988) and improved in (Jensen, 1996). This generalization requires a more complicated modification of the original network. This network is converted to a tree of cliques called a "junction tree" by clustering the nodes. The first step in the clustering process is to construct the "moralized graph", where the parents of each node are connected directly and all the edges are undirected, then triangulate the moral graph to obtain a junction tree. In the junction tree, each node is associated with a potential that represents the marginal joint distribution over the variables of the clique. In the message propagation phase of the algorithm, potentials are passed between neighboring cliques. At the end of this phase, every clique potential contains the correct marginal distribution over the clique variables. Obtaining the marginal distribution of a given variable requires a simple marginalization over all the other variables that are in the same clique. Many other node clustering algorithms have been proposed to improve the original junction tree algorithm. We can cite for example Shenoy–Shafer (Shenoy & Shafer, 1990), JLO (Jensen, 1996), and lazy propagation (Madsen & Jensen, 1998).

The junction tree algorithm in its original version constructs a tree of cluster nodes independently of the distribution or distributions of interest. It aims to compute efficiently the marginal distributions of all cluster nodes (i.e., $P(\mathbf{X}_{C_i} \mid \mathbf{X}_E)$ where $(C_i)_{i=1}^{n_c}$ is the set of the n_c clusters of the junction tree and $\mathbf{X}_E$ are evidence variables). If we are interested in building a given target distribution $P(\mathbf{X}_L \mid \mathbf{X}_E)$, this is done efficiently as long as the variables in $\mathbf{X}_L$ are in the same clique, by computing a simple marginalization over the other variables of the clique. To address the problem of computing target distributions on variables that are not in the same clique, some adaptations of this algorithm have been proposed. Two examples of these extensions are the "variable firing" and the "variable propagation" algorithms (Jensen, 2001). However, for highly connected networks these algorithms are not practical.

The complexity of all these clustering-based algorithms is exponential in the size of the largest clique of the junction tree (Lauritzen & Spiegelhalter, 1988). Unfortunately, finding the optimal triangulation in junction tree construction is an NP-hard problem.

2.2 Variable elimination algorithms

This approach, which first appeared in (Zhang & Poole, 1994) is called "variable elimination". It aims to compute an arbitrary joint distribution on a subset of variables, given a set of evidence variables. The main idea of this goal-oriented approach is to sum over a set of variables from a list of factors one by one. This approach has been generalized in the "bucket elimination" algorithm (Dechter, 1996, 1999) for various inference problems: belief updating, Most Probable Explanation (MPE), MAP, and Maximum Expected Utility (MEU). An

ordering of these variables is required as an input and is called an elimination ordering. The computation depends on the order of elimination: different elimination orderings produce different factors. The complexity of this algorithm is exponential in the maximum arity of the generated factors. Finding the optimal ordering is equivalent to the problem of finding the minimal tree width of the network (Dechter, 1996), which has been demonstrated to be NP-hard (Arnborg & Proskurowski, 1989).

3 Exact inference in ProBT: the Successive Restrictions Algorithm (SRA)

Given a target joint distribution query, exact inference in ProBT consists of:

- constructing an "exact expression" (evaluation tree) that organizes the sum/product operations sequence (see Figure 3),
- using this evaluation tree to compute or update the corresponding probability table.

The algorithm we developed to construct such an evaluation tree is called the "Successive Restrictions Algorithm". It is a goal-oriented algorithm that tries to find a marginalization (elimination) ordering for an arbitrary target joint distribution. We describe the principles of this algorithm in this section. More mathematical justifications of the SRA can be found in L. Smail's Ph.D. thesis (Smail, 2004).

Given a Bayesian network relative to a set of random variables $\mathbf{X}_U = \{X_1, X_2, ..., X_n\}$ taking values in finite sets $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n\}$, we are interested in computing the joint probability distribution (called the target) of a subset of random variables $\mathbf{X}_L \subset \mathbf{X}_U$, conditionally to another subset (possibly empty) of random variables $\mathbf{X}_R \subset \mathbf{X}_U$, where $(\mathbf{X}_L \cap \mathbf{X}_R) = \emptyset$. This target distribution is denoted $P(\mathbf{X}_L \mid \mathbf{X}_R)$.

According to Bayes's theorem, we have:

$$P(\mathbf{X}_L \mid \mathbf{X}_R) = \frac{P(\mathbf{X}_L \cup \mathbf{X}_R)}{P(\mathbf{X}_R)} = \frac{P(\mathbf{X}_L \cup \mathbf{X}_R)}{\sum_{\mathbf{X}_L} P(\mathbf{X}_L \cup \mathbf{X}_R)}. \tag{5}$$

Therefore, to compute this conditional probability we must calculate the probability distribution of $(\mathbf{X}_L \cup \mathbf{X}_R)$, which requires marginalizing out a set of variables $\mathbf{X}_{\overline{U}} = (\mathbf{X}_U - (\mathbf{X}_L \cup \mathbf{X}_R))$, from the joint distribution $P(\mathbf{X}_U)$ corresponding to the BN. The main idea in our algorithm is to find a way to manage the succession of summations on all random variables $X_i \in \mathbf{X}_{\overline{U}}$.

3.1 Motivations

The objective of finding a marginalization (elimination) ordering for an arbitrary target joint distribution is shared by other variable elimination algorithms, such as (Dechter, 1999) and (Zhang & Poole, 1994). However, the SRA algorithm has two additional objectives:

1. The construction of a symbolic probability expression (evaluation tree) representing the elimination ordering regardless of the numerical values to be used in the effective numerical evaluation.

2. All intermediate computations produce probability distributions instead of simple potentials. In other words, each node of the evaluation tree represents a probability distribution on a subset of variables. This property is very important in ProBT because each node of this tree (expression) may be replaced at runtime by another distribution (Figure 1).

Other methods of constructing symbolic probability expressions have been proposed. We can cite for example the SPI algorithm (D'Ambrosio, Shachter, & DelFavero, 1990) and the Query DAG concept (Darwiche & Provan, 1997). The SRA algorithm differs from these methods in the nature of the expressions constructed and in the objective of manipulating them. Indeed, the SPI algorithm uses the notion of a probability expression to state that belief network inference can be viewed as an expression factoring problem for which results from optimization theory can be used. Query DAGs are intended to represent expressions containing only low-level arithmetic operations (additions and multiplications). These DAGs are first generated using a given inference algorithm. Then, they are evaluated in online applications on systems with restrictions on the available software and hardware resources. On the other hand, the objective of the probability expressions constructed by the SRA algorithm is to provide an abstraction tool for building probabilistic models incrementally.

3.2 Algorithm principles

Before detailing the principles of the SRA algorithm, we introduce the concept of close descendants, which is useful for the sequel of this section.

A Bayesian network $\mathcal{B}$ relative to a set of variables $\mathbf{X}_U$ is a quadruplet $(\mathbf{X}_U, \mathcal{N}, \mathcal{E}, \mathcal{P})$, where:

- $\mathcal{N}$ is a partition of $\mathbf{X}_U$. An element $\mathbf{X} \in \mathcal{N}$ is called a "node".
- $\mathcal{E}$ is a set of ordered pairs of distinct elements of $\mathcal{N}$. If $(\mathbf{X}_i, \mathbf{X}_j) \in \mathcal{E}$, we say there is an edge from $\mathbf{X}_i$ to $\mathbf{X}_j$ and that $\mathbf{X}_i$ is the parent of $\mathbf{X}_j$. Conversely, we say that $\mathbf{X}_j$ is a child of $\mathbf{X}_i$.
- $\mathcal{P}$ is the collection $\{P_{\mathbf{X}_i|pa(\mathbf{X}_i)} \mid (\mathbf{X}_i, pa(\mathbf{X}_i)) \in \mathcal{E}\}$ of conditional distributions, with $pa(\mathbf{X}_i)$ denoting the set of parents of $\mathbf{X}_i$.

By definition, the $\mathcal{B}$ codes the joint distribution $P(\mathbf{X}_U)$.

Definition 1 *Let $\mathcal{B} = (\mathbf{X}_U, \mathcal{N}, \mathcal{E}, \mathcal{P})$ be a Bayesian network. We define the* close descendants *(denoted $cd(\mathbf{X}_i)$) of $\mathbf{X}_i \in \mathcal{N}$ as the set (possibly empty) of nodes containing the children of $\mathbf{X}_i$ and all the nodes located in a path between $\mathbf{X}_i$ and one of its children.*

Suppose first that the problem is simply to compute the joint distribution $P(\mathbf{X}_L \cup \mathbf{X}_R)$. We address the problem of taking into account the normalization constant $P(\mathbf{X}_R) = \sum_{\mathbf{X}_L} P(\mathbf{X}_L \cup \mathbf{X}_R)$ at the end of this section.

Given a Bayesian network $\mathcal{B} = (\mathbf{X}_U, \mathcal{N}, \mathcal{E}, \mathcal{P})$ and a target joint distribution $P(\mathbf{X}_L \cup \mathbf{X}_R)$ (with $\mathbf{X}_L, \mathbf{X}_R \subset \mathbf{X}_U$ and $\mathbf{X}_L \cap \mathbf{X}_R = \emptyset$), the main idea of the Successive Restrictions Algorithm is to build a sequence $\langle \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_\ell \rangle$ of Bayesian networks where $\ell \leq (Card(\mathbf{X}_{\overline{U}}) + 1)$ with $\mathbf{X}_{\overline{U}} = \mathbf{X}_U - (\mathbf{X}_L \cup \mathbf{X}_R)$. For each iteration i of the algorithm, we obtain a new Bayesian network $\mathcal{B}_{i+1} = (\mathbf{X}_{U_{i+1}}, \mathcal{N}_{i+1}\mathcal{E}_{i+1}, \mathcal{P}_{i+1})$ such that on completion of the algorithm, the Bayesian network $\mathcal{B}_\ell$ corresponds to the target distribution $P(\mathbf{X}_L \cup \mathbf{X}_R)$.

The sequence $\langle \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_\ell \rangle$ of Bayesian networks is constructed such that:

1. $\mathcal{B}_1$ is the initial Bayesian network.
2. $\mathcal{B}_{i+1}$ is computed from $\mathcal{B}_i$.
3. If a leaf node $\mathbf{X}$ is a subset of $\mathbf{X}_{\overline{U}}$ then the node is removed. The corresponding variables in the node are the so-called "barren variables" (i.e., summing-to-one variables).
4. The number of nodes in the Bayesian network $\mathcal{B}_{i+1}$ is less than or equal to the number of nodes in $\mathcal{B}_i$. That is $Card(\mathcal{N}_{i+1}) \leq Card(\mathcal{N}_i)$. Indeed, $\mathcal{N}_{i+1}$ contains at least one fewer node than $\mathcal{N}_i$. This node is said to be the *marginalization node*. The set of variables in the marginalized node belonging to $\mathbf{X}_{\overline{U}}$ are said to be marginalized out (or summed over). In addition, $\mathcal{N}_{i+1}$ contains a new node, not included in $\mathcal{N}_i$, resulting from the marginalization.
5. Once $\mathcal{B}_\ell$ is obtained, we obtain $\mathbf{X}_{U_\ell} = (\mathbf{X}_L \cup \mathbf{X}_R)$, and the probability distribution of $P(\mathbf{X}_L \cup \mathbf{X}_R)$ can be computed as the product of the conditional probabilities in $\mathcal{P}_\ell$.

Iteration i of the algorithm consists of selecting a marginalization node $\mathbf{X}_M$ such that $(\mathbf{X}_M \cap \mathbf{X}_{\overline{U}}) \neq \emptyset$. Once $\mathbf{X}_M$ is selected, the set of summed variables is $(\mathbf{X}_M \cap \mathbf{X}_{\overline{U}})$.

Indeed, the aim of the algorithm is to sum over a single variable at each step (i.e., $(\mathbf{X}_M \cap \mathbf{X}_{\overline{U}})$ contains one variable). However, in some cases it is impossible to take a single variable. This occurs when $\mathcal{B}_1$ contains nodes with more than one variable. For example, consider the BN corresponding to the joint distribution $P(X_0X_1X_2X_3) = P(X_0X_1)P(X_2 \mid X_0)P(X_3 \mid X_1)$, in which we are interested in computing $P(X_2X_3)$. We obtain:

$$P(X_2X_3) = \sum_{X_0X_1} P(X_0X_1)P(X_2 \mid X_0)P(X_3 \mid X_1),$$

in which $(\mathbf{X}_M \cap \mathbf{X}_{\overline{U}}) = \{X_0, X_1\}$.

The selected $\mathbf{X}_M$ must satisfy the constraint that it has no descendants in $\mathbf{X}_{\overline{U}}$. This constraint is a sufficient condition to ensure the coherence of the Bayesian network constructed at each iteration. However, several marginalization nodes could be available (i.e., several nodes respect this constraint), so additional criteria in relation to the computational cost of the inference task may be applied (see Subsection 3.3 below).

Once the marginalization node $\mathbf{X}_M$ is selected, the output Bayesian network $\mathcal{B}_{i+1}$ has as its nodes

$$\mathcal{N}_{i+1} = \{\mathbf{X}_M^{new}\} \cup \mathcal{N}_i',$$

where $\mathbf{X}_M^{new}$ is a new node that contains the set of variables on the close descendants of $\mathbf{X}_M$ and $\mathcal{N}_i'$ is the set of nodes on $\mathcal{N}_i$ other than $\mathbf{X}_M$ and those in $cd(\mathbf{X}_M)$.

The set of edges $\mathcal{E}_{i+1}$ results from $\mathcal{E}_i$ in the following way:

1. We delete the edges containing elements in the erased nodes $(\{\mathbf{X}_M\} \cup cd(\mathbf{X}_M))$.
2. We keep all other edges in $\mathcal{E}_i$.
3. We introduce to $\mathbf{X}_M^{new}$ a set of parents, $pa(\mathbf{X}_M^{new})$, which includes
   (a) all parents of $\mathbf{X}_M$,
   (b) all parents of the close descendants of $\mathbf{X}_M$ (other than $\mathbf{X}_M$ and those in $cd(\mathbf{X}_M)$).
4. We introduce as children of $\mathbf{X}_M^{new}$ all children of the nodes in $cd(\mathbf{X}_M)$, other than those in $cd(\mathbf{X}_M)$ itself.

The probabilistic data associated with $\mathcal{P}_{i+1} = \{P_{\mathbf{X}|pa(\mathbf{X})} | \mathbf{X} \in \mathcal{N}_{i+1}\}$ can be computed from those associated with $\mathcal{P}_i = \{P_{\mathbf{X}|pa(\mathbf{X})} | \mathbf{X} \in \mathcal{N}_i\}$ in the following way:

1. For each edge $(\mathbf{X}, pa(\mathbf{X})) \in (\mathcal{E}_i \cap \mathcal{E}_{i+1})$, we conserve the probability distribution $P_{\mathbf{X}|pa(\mathbf{X})}$.
2. For each child $\mathbf{X}_c$ of the close descendants of $\mathbf{X}_M$, its probability (conditionally to its parents) is preserved by substitution of $\mathbf{X}_M^{new}$ in the set of the parents of $\mathbf{X}_c$ belonging to $cd(\mathbf{X}_M)$ (and we conserve the information that only these variables are involved in $p_{\mathbf{X}_c|\mathbf{X}_M^{new}}$). Indeed, conditionally to its parents in $\mathcal{N}_i$, $\mathbf{X}_c$ is independent from those variables in $\mathbf{X}_M^{new}$ that are not in the parents of $\mathbf{X}_c$.
3. We create the probability of $\mathbf{X}_M^{new}$ conditionally to $pa(\mathbf{X}_M^{new})$, which can be computed using the following formula:

$$P(\mathbf{X}_M^{new} \mid pa(\mathbf{X}_M^{new})) = \sum_{\mathbf{X}_m} \left[\left(\prod_{\mathbf{X} \in cd(\mathbf{X}_M)} P(\mathbf{X} \mid pa(\mathbf{X})) \right) P(X_{\mathbf{X}_M} \mid pa(\mathbf{X}_M)) \right], \tag{6}$$

   where $\mathbf{X}_m = (\mathbf{X}_M \cap \mathbf{X}_{\overline{U}})$.

Applying the previous steps for each iteration leads to an incremental construction of a sum/product evaluation tree. On completion of the algorithm (i.e., $\mathcal{B}_\ell$), all the variables in $\mathbf{X}_{\overline{U}}$ are summed over. The probability distribution of $P(\mathbf{X}_L \cup \mathbf{X}_R)$ can be computed as the product of the probability distributions in the Bayesian network $\mathcal{B}_\ell$.

To take the normalization constant $P(\mathbf{X}_R) = \sum_{\mathbf{X}_L} P(\mathbf{X}_L \cup \mathbf{X}_R)$ into account, we can reason as follows.

The inferred expression $P(\mathbf{X}_L \mid \mathbf{X}_R)$ can be used in two possible ways:

1. In the first way, we fix, at each use, the value of $\mathbf{X}_R = \mathbf{x}_R$ as evidence and obtain the unconditional distribution $P(\mathbf{X}_L \mid \mathbf{X}_R = \mathbf{x}_R)$. In this case, $P(\mathbf{X}_R = \mathbf{x}_R)$, in Equation 5, is simply the normalization constant obtained implicitly after computing $P(\mathbf{X}_L\ \mathbf{X}_R = \mathbf{x}_R)$ on the variables of $\mathbf{X}_L$.

2. In the second approach, we use $P(\mathbf{X}_L \mid \mathbf{X}_R)$ as a conditional distribution (without fixing $\mathbf{X}_R$). $P(\mathbf{X}_L \mid \mathbf{X}_R)$ is used, for example, as a node distribution of another Bayesian network (see Figure 1). In this case, we must find an evaluation tree that allows us to compute $P(\mathbf{X}_R)$.

A simple way to construct the evaluation tree corresponding to $P(\mathbf{X}_R)$ is to initialize another SRA algorithm with the Bayesian network obtained for $P(\mathbf{X}_L \cup \mathbf{X}_R)$ (i.e., $\mathcal{B}_\ell$) and to continue marginalizing over $\mathbf{X}_L$ to derive another Bayesian network $\mathcal{B}_{norm}$ corresponding to $P(\mathbf{X}_R)$. However, some terms in $P(\mathbf{X}_R)$ may also be found in $P(\mathbf{X}_L \cup \mathbf{X}_R)$. This will be expressed by the existence of some common distributions in the corresponding networks $\mathcal{B}_{norm}$ and $\mathcal{B}_\ell$.

Because these terms can be simplified in the fraction $\frac{P(\mathbf{X}_L \cup \mathbf{X}_R)}{P(\mathbf{X}_R)}$, additional simplifications are possible by removing the corresponding distributions from $\mathcal{B}_\ell$ (and consequently from $\mathcal{B}_{norm}$, because $\mathcal{B}_\ell$ is used as an initialization to build $\mathcal{B}_{norm}$). The nodes to be removed from $\mathcal{B}_\ell$ are simply those defining probability distributions exclusively on variables belonging to $\mathbf{X}_R$. Removing these terms simplifies the computation of $P(\mathbf{X}_L \mid \mathbf{X}_R)$, whether the normalization is computed implicitly or explicitly. In the extreme case, the nodes removed from $\mathcal{B}_\ell$ correspond to the whole set $\mathbf{X}_R$ of variables and the normalization $P(\mathbf{X}_R)$ is simply canceled (i.e., $\mathcal{B}_\ell$ is already normalized).

We now show an example of the procedures described above.

Example Consider the Bayesian network in Figure 4a. The corresponding joint distribution is given by the equation:

$$\begin{aligned} P(\mathbf{X}_{U_1}) \;=\; & P(X_1)\ P(X_2)\ P(X_3)\ P(X_4 \mid X_1)\ P(X_5 \mid X_2\ X_4)\ P(X_6 \mid X_5) \\ & P(X_7 \mid X_3\ X_5)\ P(X_8 \mid X_1\ X_6)\ P(X_9 \mid X_2\ X_6)\ P(X_{10} \mid X_2\ X_7), \end{aligned}$$

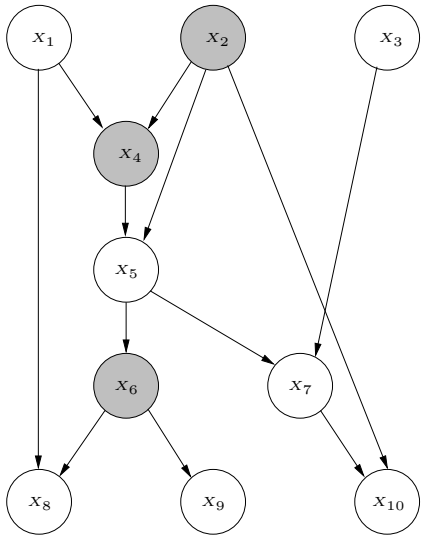
with $\mathbf{X}_{U_1} = \{X_1, X_2, ..., X_{10}\}$.

Suppose we are interested in computing the marginal distribution:

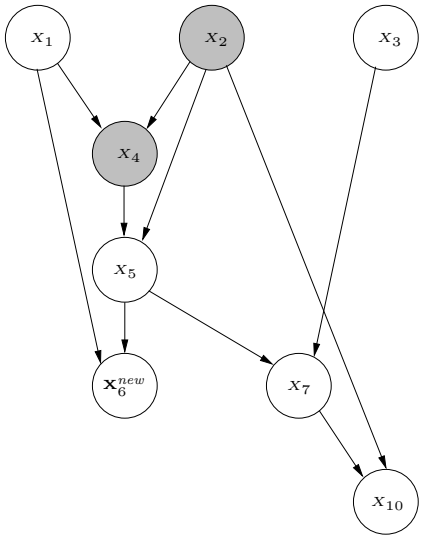
$$P(\mathbf{X}_L \cup \mathbf{X}_R) = \sum_{\mathbf{X}_{\overline{U}}} P(\mathbf{X}_{U_1}),$$

where $\mathbf{X}_L \cup \mathbf{X}_R = \{X_1, X_3, X_5, X_7, X_8, X_9, X_{10}\}$.

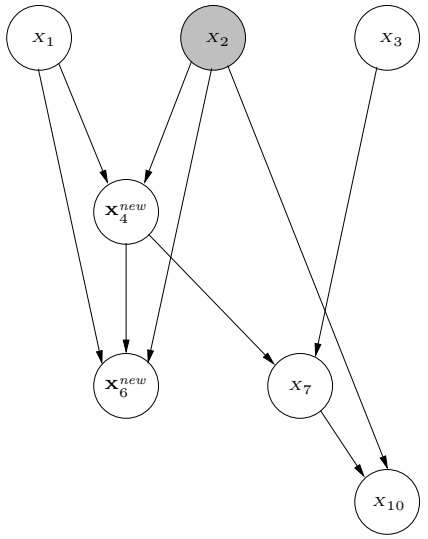
This computation requires marginalizing out the variables in $\mathbf{X}_{\overline{U}} = \{X_2, X_4, X_6\}$ (i.e., X_2, X_4, and X_6). For each step of the algorithm, we must choose a variable in $\mathbf{X}_{\overline{U}}$ to marginalize out.



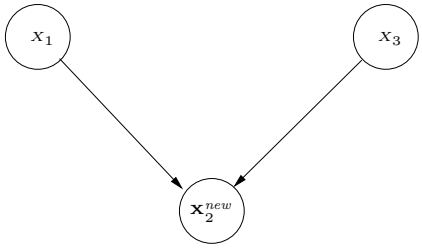
(a)



(b) $\mathbf{X}_6^{new} = \{X_8, X_9\}$



(c) $\mathbf{X}_4^{new} = \{X_5\}$



(d) $\mathbf{X}_2^{new} = \{X_5, X_7, X_8, X_9, X_{10}\}$

Figure 4: The initial Bayesian network $\mathcal{B}_1$ (Nodes in $\mathbf{X}_{\overline{U}} = \{X_2, X_4, X_6\}$ are shadowed) (*a*), and the Bayesian networks $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$ obtained after marginalizing X_6 (*b*), X_4 (*c*), and X_2 (*d*).

Step 1
For the first step, note that only X_6 has no descendants in $\mathbf{X}_{\overline{U}}$. By marginalizing out X_6, we obtain:

$$\sum_{X_6} P(X_8 \mid X_1\ X_6)\ P(X_9 \mid X_2\ X_6)\ P(X_6 \mid X_5) = P(X_8\ X_9 \mid X_1\ X_2\ X_5).$$

The resulting graph is given in Figure 4b. The resulting Bayesian network is constructed as follows:

1. Node X_6 is erased.
2. A new node $\mathbf{X}_6^{new} = \{X_8, X_9\}$ is created. Its parents are X_1, X_2, and X_5 (i.e., the initial parents of X_6 and the parents of its close descendants X_8 and X_9).

The joint distribution corresponding to $\mathcal{B}_2$ can be written as:

$$\begin{aligned} P(\mathbf{X}_{U_2}) \quad = \quad & P(X_1)\ P(X_2)\ P(X_3)\ P(X_4 \mid X_1)\ P(X_5 \mid X_2\ X_4)\ P(\mathbf{X}_6^{new} \mid X_1\ X_2\ X_5) \\ & P(X_7 \mid X_3\ X_5)\ P(X_{10} \mid X_2\ X_7), \end{aligned}$$

where $\mathbf{X}_{U_2} = \mathbf{X}_{U_1} - \{X_6\}$ and $\mathbf{X}_6^{new} = \{X_8, X_9\}$.

Step 2
For the second step of the algorithm, note that both X_4 and X_2 have no descendants in $\mathbf{X}_{\overline{U}}$. They are therefore both candidates to be marginalized out. To decide which variable should be selected, additional criteria related to the computational cost induced by each choice can be introduced (see Subsection 3.3).

Suppose we choose X_4. By marginalizing X_4 out we obtain:

$$\sum_{X_4} P(X_5 \mid X_2\ X_4)\ P(X_4 \mid X_1) = P(X_5 \mid X_1\ X_2).$$

The resulting graph is given in Figure 4c and is constructed as follows:

1. Node 4 is erased.
2. A new node $\mathbf{X}_4^{new} = \{X_5\}$ is created. Its parents are X_1 and X_2 (i.e., the initial parents of X_4 and the parents of its close descendants X_5 and $\mathbf{X}_6^{new}$).

The joint distribution corresponding to this new BN can be written as:

$$\begin{aligned} P(\mathbf{X}_{U_3}) \quad = \quad & P(X_1)\ P(X_2)\ P(X_3)\ P(\mathbf{X}_4^{new} \mid X_1\ X_2)\ P(\mathbf{X}_6^{new} \mid X_1\ X_2\ X_5) \\ & P(X_7 \mid X_3\ X_5)\ P(X_{10} \mid X_2\ X_7), \end{aligned}$$

with $\mathbf{X}_{U_3} = \mathbf{X}_{U_1} - \{X_6, X_4\}$ and $\mathbf{X}_4^{new} = \{X_5\}$.

Step 3
Finally, we must marginalize out X_2 and we obtain:

$$P(X_7 \mid X_3\ X_5)\ \textstyle\sum_{X_2}\ P(\mathbf{X}_6^{new} \mid X_1\ X_2\ \ \mathbf{X}_4^{new})\ P(\mathbf{X}_4^{new} \mid X_1\ X_2)\ P(X_{10} \mid X_2\ X_7)\ P(X_2) = P(\mathbf{X}_6^{new}\ \mathbf{X}_4^{new}\ X_7\ X_{10} \mid X_1\ X_3).$$

The resulting graph is given in Figure 4d and is constructed as follows:

1. Node X_2 is suppressed.
2. A new node $\mathbf{X}_2^{new} = \mathbf{X}_6^{new} \cup \mathbf{X}_4^{new} \cup \{7, 10\} = \{X_8, X_9, X_5, X_7, X_{10}\}$ is created. Its parents are X_1 and X_3.

The joint distribution corresponding to this new BN can be written as:

$$P(\mathbf{X}_{U_4}) = P(X_1)\ P(X_3)\ P(\mathbf{X}_2^{new} \mid X_1\ X_3),$$

with $\mathbf{X}_{U_4} = \mathbf{X}_{U_1} - \{X_6, X_4, X_2\} = (\mathbf{X}_L \cup \mathbf{X}_R)$ and $\mathbf{X}_2^{new} = \{X_5, X_7, X_8, X_9, X_{10}\}$. This joint distribution corresponds to our target distribution $P(\mathbf{X}_L \cup \mathbf{X}_R)$, so

$$P(\mathbf{X}_L \cup \mathbf{X}_R)\ =\ P(X_1)\ P(X_3)\ P(\mathbf{X}_2^{new} \mid X_1\ X_3).$$

3.3 Optimality considerations

When selecting a node to marginalize out in the general step of the SRA, there may be several choices (several nodes having no descendants in $\mathbf{X}_{\overline{U}}$ may be available). For optimality considerations, additional criteria concerning the computational cost of the inferred expression can be taken into account when selecting the variable (variables) to be marginalized out in the current step.

The aim of applying these additional criteria is to build an evaluation tree that minimizes the total computational cost of compiling (building the corresponding probability table) of the target distribution $P(\mathbf{X}_L|\ \mathbf{X}_R)$ for a given value of $\mathbf{X}_R$. The problem of finding the optimal (marginalization) elimination ordering (and, therefore, the problem of finding the optimal triangulation in junction tree construction) is NP-hard (Dechter, 1996; Arnborg & Proskurowski, 1989). However, algorithms exist that find optimal solutions (Shokhet & Geiger, 1997) or constant-factor approximations (Eyal, 2001; Reed, 1992; Robertson & Seymour, 1995; Becker & Geiger, 2001) for the optimal triangulation problem. These algorithms are not practical for graphs with large tree widths. Heuristic methods have been also proposed (Rose, 1970; Kjaerulff, 1990; Cowell, Dawid, Lauritzen, & Spiegelhalter, 1999). The problem of finding the optimal elimination ordering for given query (target) distributions has been also addressed as a combinatorial optimization problem in (Li & D'Ambrosio, 1994). An optimal polynomial time algorithm has been proposed for tree-structured (singly connected) networks and an heuristic algorithm called the "set factoring algorithm" has been proposed for the arbitrary (multiply connected) networks case.

The SRA algorithm is intended to be used in two kinds of situations: (i) statically for off-line construction of expressions in an initialization phase and (ii) dynamically for online (runtime) construction. Therefore, the heuristic used must be efficient enough to satisfy the online construction constraint. The heuristic we propose is to apply an optimality criterion locally. It aims to minimize the additional computational cost for each step of the algorithm. This local minimization is accomplished by selecting the variable that minimizes the computational cost induced by the step when building the target table. Therefore, the worst possible complexity of this heuristic is $n \ + \ (n-1) \ +, \cdots, + \ 3 \ + \ 2 = \frac{(n-1)\times(n+2)}{2}$, where n is the number of nodes to be marginalized out.

It is clear that applying the criterion locally (for each step) does not ensure the optimality of the whole constructed evaluation tree. However, in practice, this heuristic leads to very interesting results.

ProBT allows choice among the following three optimality criteria when constructing the inferred expression (the corresponding evaluation tree):

1. The size of the required memory.
2. The computation time (i.e., the number of arithmetic operations) to build the table corresponding to $P(\mathbf{X}_L|\ \mathbf{X}_R)$ for the first time.
3. The computation time required to update an already built table $P(\mathbf{X}_L|\ \mathbf{X}_R = \mathbf{x}_{R0})$ for a new value $\mathbf{x}_{R1}$ of $\mathbf{X}_R$ (i.e., $P(\mathbf{X}_L|\ \mathbf{X}_R = \mathbf{x}_{R1})$).

The first criterion considers the total memory size required to build the target probability table. This total memory size is basically the sum of the sizes of all local probability tables corresponding to all nodes of the evaluation tree.

The second and third criteria concern computation time. More precisely, they concern the total number of arithmetic operations required to build the corresponding probability tables. The total number of arithmetic operations is the sum of the numbers required to compute the local probability table of each node.

The difference between the second and the third criteria is that updating a target probability table corresponding to a given evaluation tree only requires reevaluating the nonconstant subtrees of the whole evaluation tree. Nonconstant subtrees are those depending on the values of the evidence variable E, while the other subtrees are constant (i.e., do not require reevaluation when changing the value of the evidence). Optimizing the update time therefore amounts to keeping the largest possible part of the tree constant. To do so, the variables (nodes) having evidence variables (nodes) as parents are placed as close as possible to the root of the evaluation tree. This is done, for each step, by trying first to select a variable having no evidence variables as parents. If taking such a variable is impossible, then the selection is done on the remaining variables.

Before detailing the functions to minimize for each of the three available optimality criteria, let us define $SIZE(\mathbf{X})$ as the number of possible values taken by the conjunction of

the random variables in $\mathbf{X}$. That is

$$SIZE(\mathbf{X}) = \prod_{j=1}^{Card(\mathbf{X})} |\mathcal{S}_j|,$$

where $\mathcal{S}_j$ is the state space of variable $X_j \in \mathbf{X}$.

According to Equation 6, the table corresponding to $P(\mathbf{X}_M^{new} \mid pa(\mathbf{X}_M^{new}))$ contains $TableSize(\mathbf{X}_M)$ elements, where:

$$TableSize(\mathbf{X}_M) = SIZE\Big((\mathbf{X}_M^{new} \cup pa(\mathbf{X}_M^{new})) - \mathbf{X}_R\Big). \tag{7}$$

When trying to minimize the amount of memory used, the function (7) must be minimized (i.e., choose the node $\mathbf{X}_M* = \arg\min_{\mathbf{X}_M} TableSize(\mathbf{X}_M)$).

When computation time is to be minimized, we can reason as follows.

Equation 6 shows that constructing a table corresponding to $P(\mathbf{X}_M^{new} \mid pa(\mathbf{X}_M^{new}))$ requires computing $TableSize(\mathbf{X}_M)$ (size of the table) table elements. Each element involves $SIZE(\mathbf{X}_M)$ sum terms. Because each sum term requires one addition and $(Card(\mathbf{X}_M^{new}) - 1)$ multiplications, the total number of operations required by the current step is then:

$$N(\mathbf{X}_M) = TableSize(\mathbf{X}_M) \times SIZE(\mathbf{X}_M^{new}) \times Card(\mathbf{X}_M^{new}). \tag{8}$$

Thus, when trying to minimize computation time we must minimize the function (8) (i.e., find $\mathbf{X}_M^* = \arg\min_{\mathbf{X}_M} N(\mathbf{X}_M)$).

Equation 8 is also used when trying to minimize the update time. The only difference is that the algorithm tries first to select a variable having no evidence variables as parents.

Example Consider the Bayesian network in Figure 5, for which we are interested in constructing the target distribution $P(B \mid A)$. The d parameter represents the "depth" of the network.

The purpose of this example is to show the different evaluation trees constructed for the target distribution $P(B \mid A)$ depending on the chosen optimization criterion. The corresponding computational costs are quantified for each case.

First, we assume that the d parameter (depth of the BN) is fixed to 1. We will also assume that all variables of the network take values in a finite set $\mathcal{S}$ with n elements.

Using the "first compilation time minimization" criterion Let us suppose that the time required to compile the target distribution $P(B \mid A)$ for the first evidence value of A is the main issue. In this case, the "first compilation time minimization" criterion is used and the corresponding evaluation tree given in Figure 6 is constructed.

Using this evaluation tree, the number N_C of arithmetic operations (additions and multiplications) required to compile $P(B \mid A = a)$ and the one number N_U required to update this table for a new evidence value of A are respectively:

$$N_C = 10n^5 + 6n^4 + 2n^3 + 2n^2,$$

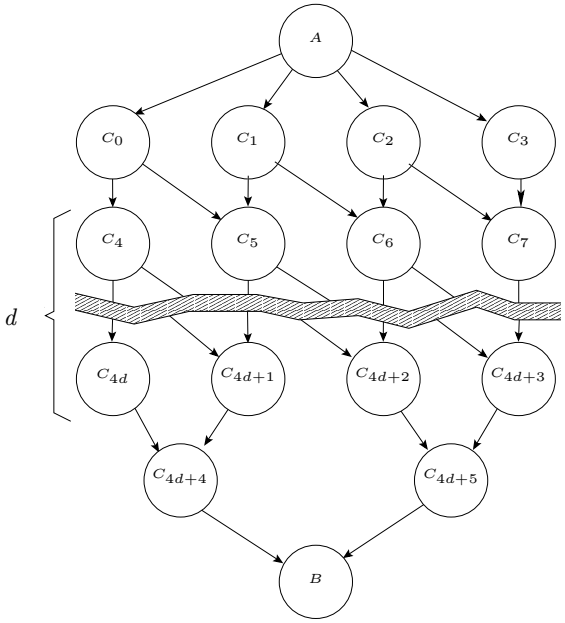


Figure 5: A Bayesian network example.

and

$$N_U = 4n^5 + 6n^4 + 2n^3 + 2n^2.$$

Notice that the constant subtree (requiring no update when the evidence value of A changes) corresponds to the node distribution $P(B \mid C_0\ C_1\ C_9)$.

Using the "update time minimization" criterion Suppose now that we are interested in using the target distribution $P(B \mid A)$ in a program loop that compiles it, for each iteration, for a different observed evidence value of A. In this case, minimizing the time required to update the corresponding probability table is the main issue and the "update time minimization" criterion is used. Figure 7 shows the evaluation tree constructed for this case.

Using this evaluation tree, the number N_C of arithmetic operations (additions and multiplications) required to compile $P(B \mid A = a)$ and the number N_U of operations required to update this table for a new evidence value of A are respectively:

$$N_C = 6n^6 + 8n^5 + 2n^4 + 2n^3 + 2n^2,$$

and

$$N_U = 2n^5 + 2n^4 + 2n^3 + 2n^2.$$

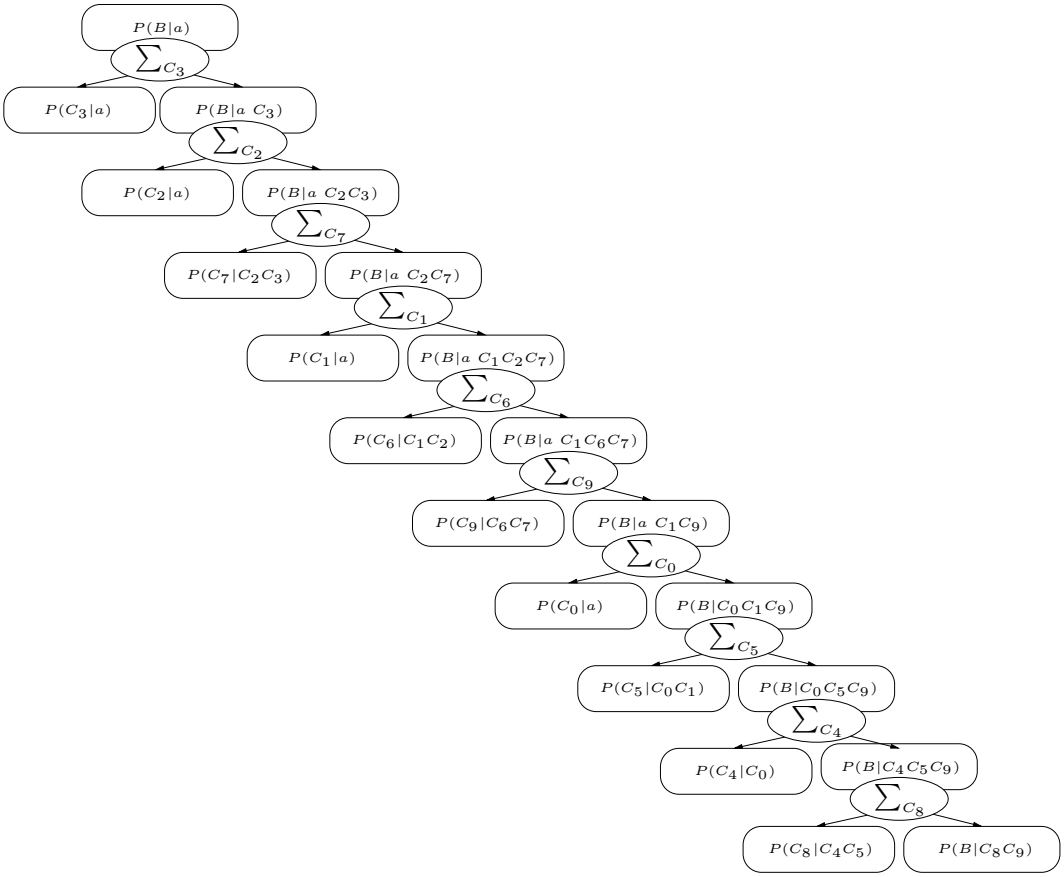


Figure 6: The evaluation tree constructed using the "first compilation time minimization" criterion.

Notice that the constant subtree (requiring no update when the evidence value of A changes) corresponds to the node distribution $P(B \mid C_0\ C_1\ C_2\ C_3)$.

Let us now assume that n is fixed to 2. Table 1 gives the number of arithmetic operations (addition and multiplication) required to build the probability table of $P(B \mid a)$ (N_C) and to update it (N_U) as a function of the parameter d for the "first compilation time minimization" and "update time minimization" criteria.

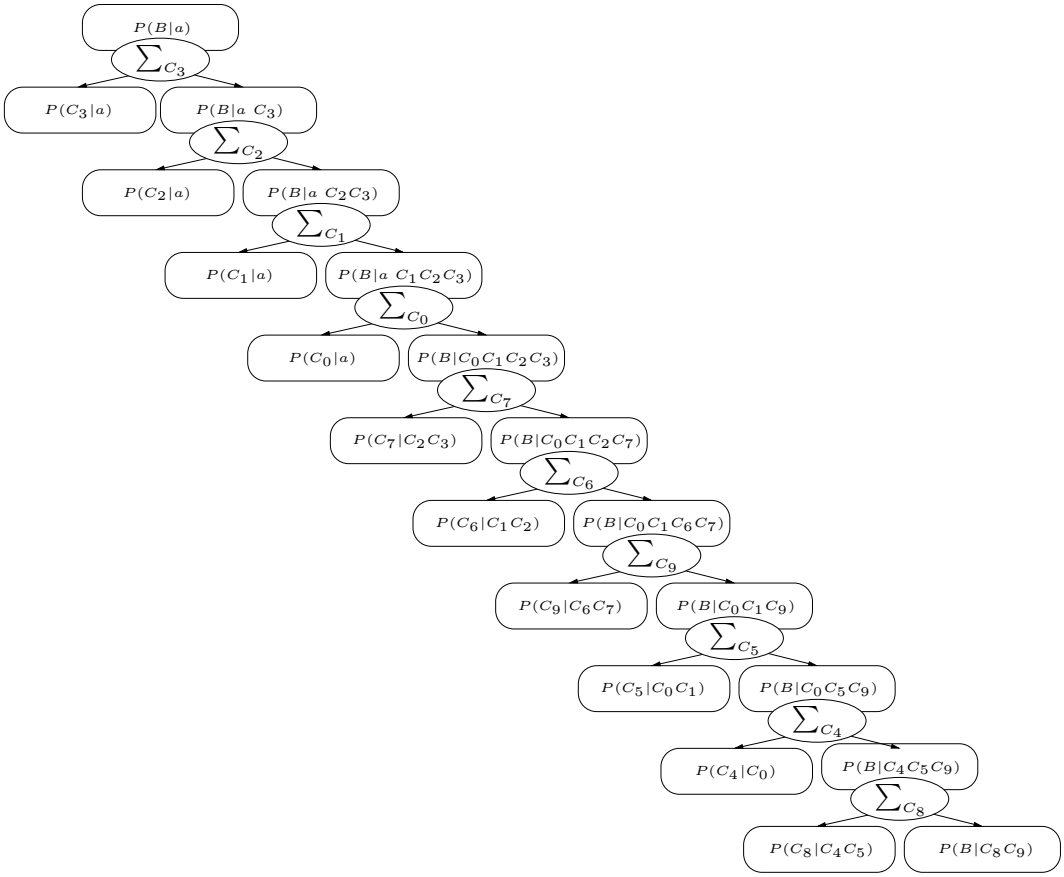


Figure 7: The evaluation tree constructed using the "update time minimization" criterion.

4 General purpose algorithms for approximate Bayesian inference

A major problem in probabilistic modeling with many variables is the computational complexity involved in typical calculations for inference. For sparsely connected probabilistic networks, this problem has been solved by the introduction of efficient algorithms for exact inference. However, in large or densely connected models, exact inference is often intractable. This means that the computation time increases exponentially with the problem size. In

	Optimization criterion			
	Compilation		Update	
d	N_C	N_U	N_C	N_U
1	**440**	248	696	**120**
2	**1112**	536	1592	**120**
3	**2072**	600	2488	**120**
6	**5016**	216	5176	**120**
10	**7064**	216	7224	**120**
100	**53144**	216	53304	**120**
300	**155544**	216	155704	**120**

Table 1: Computational cost for different values of d when using the "first compilation time minimization" (left) and "update time minimization" (right) criteria. The results of the chosen optimization criterion are shown in bold.

such cases, approximate inference techniques are alternative solutions that make the problem more tractable.

This section will briefly review the main approximate probabilistic inference techniques. These techniques are presented in two classes of approaches. The first class groups the sampling-based techniques, while the second concerns the variational methods.

4.1 Sampling-based approaches

Sampling-based (or Monte Carlo) approaches for approximate Bayesian inference group together several stochastic simulation techniques that can be applied to solve optimization and numerical integration problems in large-dimensional spaces. Since their introduction in the physics literature in the 1950s, Monte Carlo methods have been at the center of the recent Bayesian revolution in applied statistics and related fields (Geweke, 1996). Their application in other fields such as image synthesis (Keller, 1996), CAD modeling (Mekhnacha, Mazer, & Bessière, 2001; Mekhnacha, Bessière, & Mazer, 2000a), and mobile robotics (Dellaert, Fox, Burgard, & Thrun, 1999; Fox, Burgard, Dellaert, & Thrun, 1999) is more recent.

The aim of this section is to present some of the most popular sampling-based techniques and their use in the problem of approximate Bayesian inference.

4.1.1 Sampling high-dimensional distributions

Sampling distributions is a central issue in approximate inference. Sampling is required when:

1. using Monte Carlo methods for numerical estimation of integrals (see Subsection 4.1.2),
2. sampling a posteriori distributions.

The problem of drawing samples from a given distribution is still a challenging one, especially in high-dimensional spaces.

If we have an acceptable uniform random generator at our disposal, it is possible in some simple cases to use a "transformation function" to sample a given nonuniform parametric distribution (Rubinstein, 1981). One of the more important and well-known transformation functions is the "Box–Muller transformation" (Box & Muller, 1958). It permits generation of a set of random numbers drawn from a one-dimensional normal distribution using another set of random numbers drawn from a uniform random generator.

Therefore, direct sampling techniques are available for some standard simple distributions. However, for more complicated cases, indirect sampling methods such as "forward sampling", "importance sampling," "rejection sampling," and "Markov Chain Monte Carlo" (MCMC) are alternative solutions.

In this section we present some of the most popular variants of these algorithms: forward sampling, importance sampling, rejection sampling, Gibbs sampling, and Metropolis sampling. Two excellent starting points on Monte Carlo methods are the tutorials by (Neal, 1993) and (MacKay, 1996).

Forward sampling Using the "forward sampling" algorithm to sample a joint distribution represented as a Bayesian network consists of drawing values of the variables one by one, starting with the root nodes and continuing in the implicit order defined by the Bayesian network graph. In other words, first each root variable X_i is drawn from $P(X_i)$. Then each nonroot variable X_j is drawn, in the ancestral ordering, from $P(X_j \mid pa(X_j))$, where $pa(X_j)$ are the parents of X_j, for which values have been already drawn.

Suppose for example that we are interested in drawing a point from the distribution $P(X_1X_2) = P(X_1)\ P(X_2 \mid X_1)$ where $P(X_1)$ and $P(X_2 \mid X_1)$ are simple distributions for which direct sampling methods are available. Drawing a point $x^{(i)} = (x_1^{(i)}, x_2^{(i)})$ from $P(X_1X_2)$ using forward sampling consists of:

1. drawing $x_1^{(i)}$ from $P(X_1)$,
2. then drawing $x_2^{(i)}$ from $P(X_2 \mid X_1 = x_1^{(i)})$.

This sampling scheme may be used when no evidence values are available or when evidence concerns only the conditioning (right side) variables.

When evidence on the conditioned (left side) variables is available, forward sampling may also be used by introducing rejection of samples that are not consistent with the evidence. In this case, this algorithm may be very inefficient (have a high rejection rate) for evidence values with small probabilities of occurrence. Moreover, applying this algorithm is impossible when evidence concerns continuous variables.

Importance sampling Suppose we are interested in sampling a distribution $P(X)$ for which no direct sampling method is available and that we are able to evaluate this distribution for each point x of the state space.

Suppose also that we have a simpler distribution $Q(X)$ (called the "proposal distribution") that we can evaluate for each point x and for which a direct sampling method is available.

Using "importance sampling" to sample $P(X)$ consists of generating n pairs $\{(w_i, x_q^{(i)})\}_{i=1}^n$ where $\{x_q^{(i)}\}_{i=1}^n$ are drawn from $Q(X)$ and

$$w_i = \frac{P(x_q^{(i)})}{Q(x_q^{(i)})}. \tag{9}$$

Rejection sampling Suppose we are interested in sampling a distribution $P(X)$ for which no direct sampling method is available and that we can evaluate this distribution for each point x of the state space.

Suppose also that we have a simpler distribution $Q(X)$ that we can evaluate for each point x_i and for which a direct sampling method is available, respecting the constraint:

$$\exists c,\ \forall x,\ c \times Q(x) > P(x).$$

Using rejection sampling to draw a point of $P(X)$ consists of drawing a point x_q from $Q(X)$ and accepting it with a probability of $\frac{c \times Q(x_q)}{P(x_q)}$:

1. draw a candidate point x_q from $Q(X)$,
2. evaluate $c \times Q(x_q)$,
3. generate a uniform random value u in $[0, c \times Q(x_q)]$,
4. if $P(x) > u$ then the point x_q is accepted. Otherwise, the point is rejected.

It is clear that this rejection sampling is efficient if the distribution $Q(X)$ is a good approximation of $P(X)$. Otherwise, the rejection rate will be very important.

Gibbs sampling Gibbs sampling is an example of "Markov Chain Monte Carlo" (MCMC) sampling techniques. MCMC methods use a Markovian process in which a sequence of states $\{x^{(t)}\}$ is generated. Each new state $x^{(t)}$ depends on the previous one $x^{(t-1)}$. These algorithms are based on the theory of Markov chains (Feller, 1968; Fill, 1991; Neal, 1993).

The Gibbs sampling method came into prominence only recently with the work of (Geman & Geman, 1984) and (Smith & Roberts, 1993). It is a method for sampling from distributions over at least two dimensions. It is assumed that while $P(X)$ is too complex to draw samples from directly, its conditional distributions $P(X_i|\{X_j\}_{i\neq j})$ are tractable to work with. In the general case of a system of N variables, a single iteration involves sampling one parameter at a time:

$$\begin{aligned}
x_1^{(t+1)} &\sim P(X_1 \mid x_2^{(t)} x_3^{(t)} \cdots x_N^{(t)}) \\
x_2^{(t+1)} &\sim P(X_2 \mid x_1^{(t)} x_3^{(t)} \cdots x_N^{(t)}) \\
&\vdots \\
x_N^{(t+1)} &\sim P(X_N \mid x_1^{(t)} x_2^{(t)} x_3^{(t)} \cdots x_{N-1}^{(t)}).
\end{aligned}$$

Metropolis algorithm The Metropolis algorithm (Metropolis, Rosenblusth, Rosenblusth, Teller, & Teller, 1953) is another example of MCMC methods. It is one of the more widely used techniques for sampling high-dimensional probability distributions and densities. This algorithm only requires a way to evaluate the sampled expression for each point of the space.

The main idea of the Metropolis algorithm is to use a proposal distribution $Q(x_c, x^{(t)})$, which depends on the current state $x^{(t)}$. This proposal distribution can be a simple distribution (a normal distribution having $x^{(t)}$ as mean value for example).

Suppose the current state is $x^{(t)}$. A candidate x_c is generated from $Q(X_c, x^{(t)})$. To accept or reject this candidate we must compute

$$a = \frac{P(x_c)Q(x_c, x^{(t)})}{P(x^{(t)})Q(x^{(t)}, x_c)}.$$

If $a > 1$ then x_c is accepted, otherwise it is accepted with probability a. If x_c is accepted, we set $x^{(t+1)} = x_c$. If x_c is rejected, then we set $x^{(t+1)} = x^{(t)}$.

If the proposal distribution $Q(X_c, x^{(t)})$ is symmetrical (a normal distribution having $x^{(t)}$ as mean value for example), then $\frac{Q(x_c, x^{(t)})}{Q(x^{(t)}, x_c)} = 1$ and we obtain:

$$a = \frac{P(x_c)}{P(x^{(t)})}.$$

One drawback of MCMC methods is that we must in general wait for the chain to reach equilibrium. This can take a long time and it is sometimes difficult to tell when it happens.

4.1.2 Numerical estimation of high-dimensional integrals

Integral (sums) calculus is a central issue in Bayesian inference. Unfortunately, analytic methods for integral evaluation seem very limited in real-world applications, where integrands may have complex shapes and integration spaces may have very high dimensionality. Furthermore, these techniques are not useful for general purpose inference, where the distributions may be simple probability tables.

In this section, we will generally assume that X is a k-dimensional vector with real or discrete components, or a mixture of real and discrete components. We will also use the symbol $\int$ as a generalized integration operator for both real integrals (over real components) and sums (over discrete components).

The aim of Monte Carlo methods for numerical integration is to approximate efficiently the k-dimensional (where k can be very large) integral

$$I = \int P(X)g(X)\ d^k X. \tag{10}$$

Assuming that we cannot visit every location x in the state (integration) space, the simplest solution we can imagine to estimate the integral (10) is to uniformly sample the

integration space and then estimate I by $\hat{I}$:

$$\hat{I} = \frac{1}{N} \sum_i P(x^{(i)}) g(x^{(i)}).$$

where $\{x^{(i)}\}_{i=1}^N$ are randomly drawn in the integration space.

Because high-dimensional probability distributions are often concentrated on a small region T of the state (integration) space $\mathcal{S}$, known as its "typical set" (MacKay, 1996), the number N of points drawn uniformly for the state (integration) space $\mathcal{S}$ must be sufficiently large to cover the region T containing most of the probability mass of $P(X)$.

Instead of exploring the integration space uniformly, Monte Carlo methods try to use the information provided by the distribution $P(X)$ to explore this space more efficiently. The main idea of these techniques is to approximate the integral (10) by estimating the expectation of the function $g(X)$ under the distribution $P(X)$

$$I = \int P(X) g(X) \; d^k X = \langle g(X) \rangle.$$

Clearly, if we are able to generate a set of points (vectors) $\{x^{(i)}\}_{i=1}^N$ from $P(X)$, the expectation of $\hat{I}$ is I. As the number of samples N increases, the variance of the estimator $\hat{I}$ will decrease as $\frac{\sigma^2}{N}$ where σ^2 is the variance of g:

$$\sigma^2 = \int P(X)(g(X) - \langle g(X) \rangle)^2 \; d^k X.$$

Suppose we are able to obtain a set of samples $\{x^{(i)}\}_{i=1}^N$ (k-vectors) from the distribution $P(X)$. We can use these samples to find the estimator

$$\hat{I} = \frac{1}{N} \sum_i g(x^{(i)}). \tag{11}$$

This Monte Carlo method assumes the capacity to sample the distribution $P(X)$ efficiently. It is called "perfect Monte Carlo integration".

This is possible when $P(X)$ is a simple standard distribution with a direct sampling method, or a product of standard distributions on which direct sampling is possible using the forward sampling algorithm (see Subsection 4.1.1). For example, if

$$P(X) = P(X_1) P(X_2 | X_1),$$

where $P(X_1)$ is a uniform distribution on $[a..b[$ and $P(X_2|X_1)$ is a normal distribution having the value of X_1 as mean and a fixed value as variance, then drawing a point $x^{(t)} = (x_1^{(t)}, x_2^{(t)})$ consists of:

1. drawing $x_1^{(t)}$ from $P(X_1)$ (i.e., uniformly between a and b).

2. drawing $x_2^{(t)}$ from the normal distribution $P(X_2|X_1 = x_1^{(t)})$.

Suppose now that we are unable to generate sample points directly from the distribution $P(X)$ but only from a simpler distribution $Q(X)$ called the "sampling distribution".

Using importance sampling (see Section 4.1.1), a set of N points is generated from $Q(X)$. If these sample points were generated from $P(X)$, we could estimate I using Equation 11. However, because these points have been generated from $Q(X)$ and not from $P(X)$, the values x for which $Q(x)$ is greater than $P(x)$ will be over-represented and the values for which $Q(x)$ is less than $P(x)$ will be under-represented. To take this problem into account, we introduce the notion of "weight":

$$w_i = \frac{P(x_q^{(i)})}{Q(x_q^{(i)})}. \tag{12}$$

These weight values are used to add the "importance" of each point in the estimator:

$$\hat{I} = \frac{1}{\sum_i w_i} \sum_i w_i g(x_q^{(i)}). \tag{13}$$

(Neal, 1993) is a good survey of Monte Carlo sampling techniques.

4.2 Variational methods

Variational methods in graphical models refer to a set of optimization techniques used to develop approximate inference algorithms. These approaches have been used successfully in a considerable number of specific models where exact inference becomes intractable, that is, when the graph is highly connected. Variational methods allow an approximation of the desired distribution by providing lower and upper bounds of the desired distribution. The basic idea is to transform the original inference problem $P(X)$ into a tractable optimization problem $F(X, \lambda)$ where the vector λ contains the variational parameters of the initial distribution. A concave function $P(X)$ can be reformulated as follows:

$$P(X) = \inf_{\lambda} F(X, \lambda).$$

A deterministic iterative algorithm (e.g., coordinate descent, gradient-based, fixed-point) then solves the minimization problem. For convex functions a maximization over $F(X, \lambda)$ is executed. This reformulation requires $P(X)$ to be concave or convex. If $P(X)$ is neither concave nor convex then it is necessary to find an invertible transformation function $f(P(X))$ that allows us to obtain a concave or convex function, for instance a logarithmic function. In some cases, a second transformation function $g(X)$ for the argument x must also be executed.

The principles of convex duality and conjugate functions are at the core of variational transformations. In effect, $F(X, \lambda)$ is defined as follows:

$$F(X, \lambda) = \lambda g(X) - P^*(\lambda),$$

where $g(X)$ is a function such that $P(X)$ is concave with respect to $g(X)$. The function P^* is said to be the *dual function* of $P(X)$ and is defined as:

$$P^*(\lambda) = \inf_X F^*(X, \lambda),$$

with

$$F^*(x, \lambda) = \lambda g(X) - P(\lambda).$$

If an invertible transformation function f is used, we replace P by f and P^* by f^* in the previous expressions. Again, for a convex function we replace the minimization by a maximization. The transformation of $P(X)$ by the principle of convex duality allows us to eliminate some of the nodes or interconnections in the original graph. In fact, the transformation will allow us to take out some expressions in the product of probabilities, reducing them to a simple constant. A simpler graph is obtained where some of the nodes have new parameter values. The new graph then allows us to obtain an expression where the summation over the missing variables can be evaluated by an exact method, for instance, the junction tree inference algorithm.

Jordan (Jordan, Ghahramani, Jaakkola, & Saul, 1999; Jordan & Weiss, 2002) presented some common graphical problems where the variational methods were shown to be useful and for which it is evident that running an exact inference algorithm becomes infeasible. However, despite the successful application of variational methods in common graphical models, a methodology for including this approach into a general probabilistic inference engine seems difficult to propose. Indeed, variational methods require finding a convenient transformation, a task that is not always systematic for nonspecific classes of graphical models.

General introductions to variational methods may be found in (Jordan et al., 1999), (Jordan & Weiss, 2002), and (Jaakkola & Jordan, 1999).

5 Approximate inference in ProBT

Exact inference is often intractable for large and/or densely connected models. This is especially the case when the variables involved take values in huge sets of states. Moreover, exact inference is impossible for arbitrary models involving continuous variables[1].

ProBT uses a set of sampling-based techniques to propose four levels of approximation. These techniques are used for variables taking values in finite sets as well as for continuous ones.

5.1 Approximation in computing marginalization

The first level of approximation proposed in ProBT concerns the problem of numerically estimating integrals. When evaluating an expression $E(X)$ for a given point of the target

[1]Exact inference using continuous variables is only possible for models allowing analytical calculation.

space $\mathcal{S}$, ProBT allows the computation of a more or less accurate numerical estimation of the integrals (sums) involved in this expression using perfect Monte Carlo methods (see Subsection 4.1.2). When the evaluated expression involves a marginalization over a conjunction of numerous variables that possibly take values in a huge (or infinite for continuous variables) set of states, its exact evaluation may have a very high computational cost. When using this approximation scheme, the expression is said to be a "Monte Carlo expression".

Let us recall our example, in particular Equation 4. This approximation level concerns the evaluation of $P(A\ B \mid E\ F)$ for some given values $(A = a)$, $(B = b)$, $(E = e)$, and $(F = f)$. More precisely, the problem is to find an efficient way to approximate $P(a\ b \mid e\ f)$ using a Monte Carlo estimation of the integral

$$I = \int P(C \mid a\ b)\ P(D \mid b)\ P(e \mid C)\ P(f \mid D)\ dC\ dD.$$

We must answer the following question. Is it more efficient to use:

1. $$\hat{I}_1 = \frac{1}{N} \sum_i P(e \mid c^{(i)})\ P(f \mid d^{(i)}), \tag{14}$$

   where $\{c^{(i)}\}_{i=1}^{N}$ and $\{d^{(i)}\}_{i=1}^{N}$ are generated from $P(C \mid a\ b)$ and $P(D \mid b)$, respectively,

2. or:
   $$\hat{I}_2 = \Big(\frac{1}{N_C} \sum_j P(e \mid c^{(j)})\Big)\ \Big(\frac{1}{N_D} \sum_k P(f \mid d^{(k)})\Big), \tag{15}$$

   where $\{c^{(j)}\}_{j=1}^{N_C}$ and $\{d^{(k)}\}_{k=1}^{N_D}$ are generated from $P(C \mid a\ b)$ and $P(D \mid b)$, respectively?

More generally, is it more efficient to use the sum/product evaluation tree built using the SRA algorithm (see Section 3) to estimate the integrals (sums) using Monte Carlo approximation?

To answer this question, we must consider error propagation in the estimation of intermediate terms and the convergence of this estimation.

In ProBT, we use Equation 14 to estimate integrals (sums). In other words no elimination ordering is required. This choice is motivated as follows:

- It is more efficient to use the estimator in Equation 14 to avoid error propagation.
- Monte Carlo methods for integral estimation perform better in high-dimensional spaces (Neal, 1993).

ProBT allows two ways to control the cost/accuracy of the estimate.

The first way is to specify the number of sample points to be used for estimating the integral. This allows the user to express constraints on the computational cost and on the required accuracy of the estimate. This parameter (i.e., the number of sample points) is also

used internally in the MCSEM algorithm (see Subsection 5.4) for a posteriori distributions optimization.

The second way to control the cost/accuracy of the estimate is to provide a convergence threshold. The convergence of the estimate is supposed to be reached when adding sampling points does not sensibly modify the value of the estimate. This is accomplished by checking the convergence criterion at each incremental estimation of the integral as follows. Starting with an initial number n_0 of sampling points, an estimate $\hat{E}_{n_0}(x)$ is computed. For each step s, the number of sampling points is increased according to a given scheduling scheme and the estimate $\hat{E}_{n_{s-1}}(x)$ is updated to give the estimate $\hat{E}_{n_s}(x)$. Then, given a threshold value ϵ_{estim}, the convergence criterion is checked. This convergence criterion is defined as:

$$\frac{|\hat{E}_{n_s}(x) - \hat{E}_{n_{s-1}}(x)|}{\hat{E}_{n_{s-1}}(x)} < \epsilon_{estim}.$$

5.2 Approximation in building numerical representation of distributions: the MRBT representation

The problem of numerically representing probability distributions is central to computational efficiency (memory use and computation time) in probabilistic inference. The main problems arise when dealing with high-dimensional distributions on a conjunction of numerous variables that possibly take values in a huge (or infinite for continuous variables) set of states. In these cases, it is seldom possible to build exhaustive probability tables.

This problem is approached in ProBT by approximating any multidimensional distribution using an adaptive discretization of the space. This representation is based on a binary tree and its main idea is to partition the space into regions at different resolution levels. High-probability regions of the distribution are represented at high resolutions while low-probability regions are represented at low resolutions.

These binary trees are called "MRBT"s (for Multi-Resolution Binary Tree). They allow us to visit high-probability regions of the target space selectively instead of constructing an exhaustive table of probabilities. Besides allowing an efficient storage of the target distribution, this numerical representation has the capacity of generalization so that we can compute, for each point x of the target space $\mathcal{S}$, the probability value $P(X = x)$.

An MRBT is built incrementally by inserting N pairs $\{(x^{(i)}, P(X = x^{(i)}))\}_{i=1}^{N}$. This set of points $\{x^{(i)}\}_{i=1}^{N}$ is generated using an external process such as a Genetic algorithm or a Metropolis sampler.

Suppose that X is k-dimensional. Each node encodes a region of $\mathcal{S}$ using its min and max values for each dimension $([min_1, max_1], \cdots, [min_k, max_k])$. When inserting a given pair $(x^{(i)}, P(X = x^{(i)}))$, the binary tree is updated so that the resolution of the region (node) including the drawn point $x^{(i)}$ is increased by splitting the corresponding node on the current splitting dimension. Two child nodes are thus created and the probability value of each child node is computed as a function of its own volume and the probability value of the inserted point $P(X = x^{(i)})$. This construction process is incremental and has the

"anytime" property: each time a point is inserted, the resulting MRBT represents a valid and operational approximation of the distribution.

ProBT uses MRBTs as default numerical representations to perform nonexhaustive compilation of a given expression. Indeed, when compiling a given expression, the ProBT API allows the user to express constraints on memory use, by fixing the number of points to use when constructing the MRBT, or in terms of computational cost by fixing the maximum time allowed for this compilation. Satisfying these constraints is possible thanks to the anytime property of MRBT's construction process.

Once built, an MRBT can be used to find the probability value of a given point x (i.e., to compute $P(X = x)$), by searching (using dichotomy) the leaf node corresponding to x and returning its probability value. To draw a point from the distribution $P(X)$, we also use dichotomy to draw a leaf node. Starting from the root node, we choose to go to its first or second child according to their respective total probability values. This procedure is iterated until a leaf node is reached. Given this leaf node, drawing a value x from $P(X)$ consists of drawing for each dimension j a value between min_j and max_j.

More details on the implementation and use of MRBTs can be found in (Bellot & Bessière, 2003).

5.3 Approximation in sampling distributions

ProBT implements a drawing method for each probability distribution (or density function), whether or not it is a standard built-in distribution or an inferred expression.

ProBT implements standard direct sampling methods for simple distributions such as normal (with one or many dimensions), Poisson, gamma, ... It also implements a dichotomy-based algorithm using the repartition function to sample discrete distributions represented as probability tables.

For inferred expressions (exact or approximate), a direct sampling is possible if we construct an explicit numerical representation of the expression by compiling it. Compilation may be exhaustive, to obtain a corresponding probability table, or approximate, to obtain a corresponding MRBT. In both cases, a direct drawing method is available. Unfortunately, this compilation process is often very expensive and seldom possible for high-dimensional distributions.

An alternative to compiling an expression before sampling it is to use an indirect sampling method that does not require a global numerical construction of the target distribution. The idea is to apply an indirect sampling method on the expression to be sampled. The following two cases must be considered.

Expressions with no evidence on the conditioned variables For expressions containing no evidence or containing evidence exclusively on the conditioning variables, the simple (without rejection) forward sampling algorithm (see Section 4.1.1) is used. Suppose we have a given expression

$$E(X_1X_2) = P(X_1X_2 \mid e) = P(X_1 \mid e)\ P(X_2 \mid X_1),$$

where $P(X_1 \mid e)$ is a given probability table and $P(X_2 \mid X_1)$ is a normal distribution having the value of X_1 as mean and a fixed value as variance. Drawing a point $x^{(i)} = (x_1^{(i)}, x_2^{(i)})$ using forward sampling consists of:

1. drawing $x_1^{(i)}$ from $P(X_1 \mid e)$ using the corresponding repartition function,
2. drawing $x_2^{(i)}$ from the normal distribution $P(X_2 \mid X_1 = x_1^{(i)})$ using the Box–Muller algorithm.

The same method is used for expressions involving sums (integrals) and containing no evidence values in the left side of all components (i.e., simple marginalization expressions). This is done by drawing from the corresponding joint distribution and then simply projecting the drawn point on the target space. For example, suppose we have the expression

$$E(X) = P(X \mid e) \propto \int P(Y \mid e)\ P(X \mid Y)\ dY.$$

To draw from $E(X)$, we must draw $P(X\ Y \mid e)$ from the joint distribution using the forward sampling method, then take the X component as a sampling point of $E(X)$.

Expressions with evidence on the conditioned variables Suppose now that we have the task of sampling the expression

$$E(X_1X_2) = P(X_1X_2 \mid e) \propto P(X_1)\ P(e \mid X_1)\ P(X_2 \mid X_1).$$

This expression contains evidence on the conditioned variable E. Using forward sampling with rejection in this case consists of drawing points from $P(X_1X_2\ E)$ using the standard forward sampling above, then rejecting points that are not coherent with the evidence (i.e., points where $e^{(i)} \neq e$). This algorithm may be very inefficient (high rejection rate), especially for evidence values having small probability of occurrence (small value of $P(e)$) and/or for continuous variables (i.e., when E is continuous).

Therefore, for expressions containing evidence on the conditioned variables of at least one component, ProBT uses the Metropolis algorithm (see Section 4.1.1).

5.4 Approximation in computing MAP solutions: the MCSEM algorithm

Searching a MAP solution for a given inference problem consists of maximizing the corresponding target distribution $P(X \mid E = e)$ (i.e., finding $x^* = \arg\max_X P(X \mid E = e)$), where e is a given evidence value.

X may be a conjunction of numerous variables that possibly take values in a huge (or infinite for continuous variables) set of states. Moreover, evaluating $P(X \mid E = e)$ for a given value x of $\mathcal{S}$ space may require evaluating an integral (or sum) over numerous variables taking themselves values in possibly huge (or infinite) sets of states.

In these cases, evaluating the target distribution $P(X \mid E = e)$ for all possible states of $\mathcal{S}$ is intractable and an appropriate numeric optimization technique must be used.

For general purpose Bayesian inference problems, the optimization method to be used must satisfy a set of criteria in relation to the shape and nature of the objective function (target distribution) to optimize. The method must:

1. be global, because the distribution to optimize (objective function) is often multimodal,
2. allow multiprecision evaluation of expressions requiring integral (sums) computing; estimation with high accuracy may require long computation times,
3. allow parallel implementation to improve efficiency.

The resolution method used in ProBT to solve this double integration/optimization problem is based on an adaptive genetic algorithm. The accuracy of integral numerical estimation is controlled by the optimization process to reduce computation time.

Genetic Algorithms (GAs) are stochastic optimization techniques inspired by the biological evolution of species. Since their introduction by Holland (Holland, 1975) in the seventies, these techniques have been used for numerous global optimization problems, thanks to their ease of implementation and their relative independence of application fields. They are widely used in a large variety of domains including artificial intelligence (Grefenstette, 1988) and robotics (Mazer, Ahuactzin, & Bessière, 1998).

Biological and mathematical motivations of genetic algorithms and their principles are not discussed in this paper. We only discuss the practical problems we face when using standard genetic algorithms in Bayesian inference. We give the required improvements and the corresponding algorithms.

In the following, we use $E(X)$ to denote the probability expression corresponding to the target distribution to be optimized. This expression is used as an evaluation function in our genetic algorithm. $E(X)$ may contain integrals (sums) to be evaluated exactly if the expression is an exact one, or approximately (using Monte Carlo) if the expression is an approximate one.

5.4.1 Narrowness of the objective function—constraint relaxation

The objective function $E(X)$ may have a narrow support (the region where the value is not null) for very constrained problems. The initialization of the population with random individuals from the search space may give null values of the function $E(X)$ for most individuals. This will make the evolution of the algorithm very slow and its behavior will be similar to random exploration.

To deal with this problem, a concept inspired from classical simulated annealing algorithms (Corana, Martini, & Ridella, 1987) consists of introducing a notion of "temperature". The principle is to first widen the support of the function by changing the original function to obtain nonnull values even for configurations that are not permitted (i.e., with probability zero). To do so, we introduce an additional parameter T (for temperature) in the objective

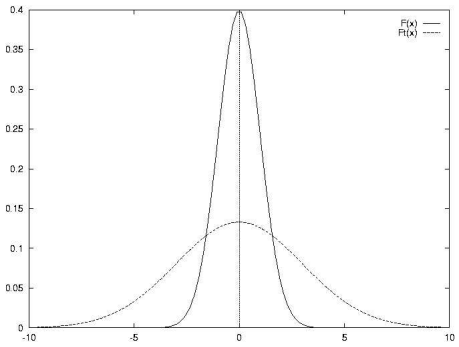


Figure 8: A normal (Gaussian) distribution at different temperature values.

function $E(X)$. Our goal is to obtain another function $E^T(X)$ that is smoother and has wider support, with

$$\lim_{T \to 0} E^T(X) = E(X).$$

To widen the support of $E(X)$, all elementary terms (distributions) of $E(X)$ are widened.

Because the main idea is to perform successive optimization cycles for successive values of the temperature T, we must respect the following additional condition:

$$\forall x_1, x_2 \in \mathcal{S}, \quad \forall t_1, t_2 \in I\!R^+, \qquad E^{t_1}(x_1) \leq E^{t_1}(x_2) \Rightarrow E^{t_2}(x_1) \leq E^{t_2}(x_2).$$

To do so, all elementary distributions must accept an additional temperature parameter and must themselves satisfy the following condition:

$$\forall x_1, x_2 \in \mathcal{S}, \quad \forall t_1, t_2 \in I\!R^+, \qquad f^{t_1}(x_1) \leq f^{t_1}(x_2) \Rightarrow f^{t_2}(x_1) \leq f^{t_2}(x_2).$$

For example, for a normal (Gaussian) distribution (see Figure 8) we have:

$$\begin{aligned} f(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}, \\ f^T(x) &= \frac{1}{\sqrt{2\pi}\sigma(1+T)} e^{-\frac{1}{2}\frac{(x-\mu)^2}{[\sigma(1+T)]^2}}. \end{aligned}$$

For nonparametric distributions (probability tables for example), this additional parameter may be simply ignored.

5.4.2 Accuracy of the estimates—multiprecision computing

When solving problems involving approximate expressions (i.e., Monte Carlo numerical estimation of integrals), the second problem we face is that only an approximation $\hat{E}(X)$ of $E(X)$ is available, of unknown accuracy.

This accuracy depends on the number of points N used for the estimation: more points mean more accurate estimations. However, using a large number of points to obtain sufficient accuracy in the whole optimization process may be very expensive in computation time and is seldom possible for complicated problems. The main idea we propose to solve this problem is to introduce N as an additional parameter to define a new function $\hat{E}_N(X)$.

The main assumption in using this additional parameter is that the final population of a GA initialized and run for some cycles with $\hat{E}_{N_1}(X)$ as its evaluation function is a good initialization for another GA having $\hat{E}_{N_2}(X)$ as evaluation function with $N_2 > N_1$.

5.4.3 General optimization algorithm

In the following, we label the evaluation function (the objective function) by the temperature T and the number N of points used for estimation. It is denoted by $E_N^T(X)$.

Our optimization algorithm may be described by the following three phases.

1. Initialization and initial temperature determination.
2. Reduction of temperature to recreate the original objective function.
3. Augmentation of the number of points to increase the accuracy of the estimates.

Initialization: The population of the GA is initialized at random from the search space. To minimize computing time in this initialization phase, we use a small number N_0 of points to estimate integrals. We propose the following algorithm as an automatic initialization procedure for the initial temperature T_0, able to adapt to the complexity of the problem.

```
INITIALIZATION()
BEGIN
    FOR each population[i] DO
        REPEAT
            population[i] = random(S)
            value[i] = E^T_{N_0}(population[i])
            if (value[i] == 0.0)
              T = T + ΔT
        UNTIL (value[i]> 0.0)
    END FOR
    Reevaluate_population()
END
```

where ΔT is a small increment value.

This phase of the algorithm is schematized in Figure 9.

Temperature reduction: To obtain the original objective function ($T = 0.0$), a possible scheduling procedure consists of multiplying the temperature, after running the GA for a given number of cycles nc_1, by a factor α ($0 < \alpha < 1$). A small value for α may cause the divergence of the algorithm, while a value too close to 1.0 may considerably increase the computation time. In ProBT, the value of α has been experimentally fixed to 0.8 as default; however, it can be fixed by the user. We summarize the algorithm as follows:

```
TEMPERATURE_REDUCTION()
BEGIN
        WHILE (T > T_ε) DO
                FOR i=1 TO nc_1 DO
                        Run_GA()
                END FOR
                T = T * α
                Reevaluate_population()
        END WHILE
        T = 0.0
        Reevaluate_population()
END
```

where T_ϵ is a small threshold value.

This phase of the algorithm is schematized in Figure 10.

Increasing the number of points N: At the end of the temperature reduction phase, the population may contain several possible solutions for the problem. To decide between these solutions, we must increase the accuracy of the estimates. One approach is to increase N, after running the GA for a given number of cycles nc_2, by a factor β ($\beta > 1$) so that the variance of the estimate is divided by β:

$$Var(E^0_{\beta * N}(X)) = \frac{1}{\beta} Var(E^0_N(X)).$$

We can describe this phase by the following algorithm.

```
NUMBER_OF_POINTS_INCREASING()
BEGIN
        WHILE (N < N_max) DO
                FOR i=1 TO nc_2 DO
                        Run_GA()
                END FOR
                N = N * β
                Reevaluate_population()
        END WHILE
END
```

where N_{max} is the number of points that allows convergence of the estimates $\hat{E}^0_N(X)$ for all individuals of the population.

This phase of the algorithm is schematized in Figure 11.

ProBT also provides an anytime version of the MCSEM algorithm. In this version, the user is allowed to fix the maximum number of evaluations of the objective function or the maximum time to be used to maximize it.

A preliminary implementation of the MCSEM algorithm and its use in high-dimensional inference problems has been presented in (Mekhnacha et al., 2001; Mekhnacha, Mazer, & Bessière, 2000b) in which this algorithm is used as a resolution module in a probabilistic CAD system.

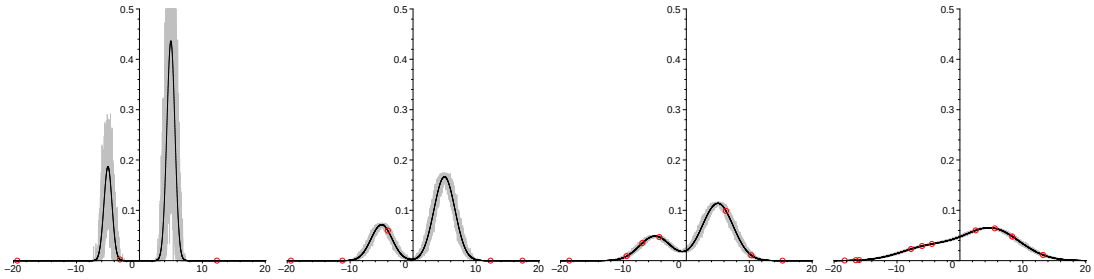


Figure 9: The initialization phase of the MCSEM algorithm. In black, the theoretical distribution to maximize and in gray, the estimated one using Monte Carlo numerical integration. From left to right, the T (temperature) parameter is increased starting from zero (i.e., the initial distribution).

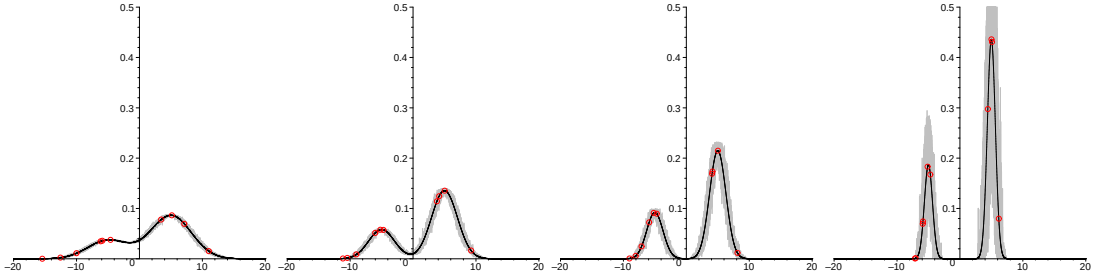


Figure 10: The "temperature reduction" phase of the MCSEM algorithm. In black, the theoretical distribution to maximize and in gray, the estimated one using Monte Carlo numerical integration. From left to right, the T (temperature) parameter is decreased to obtain the original distribution (i.e., $T = 0.0$).

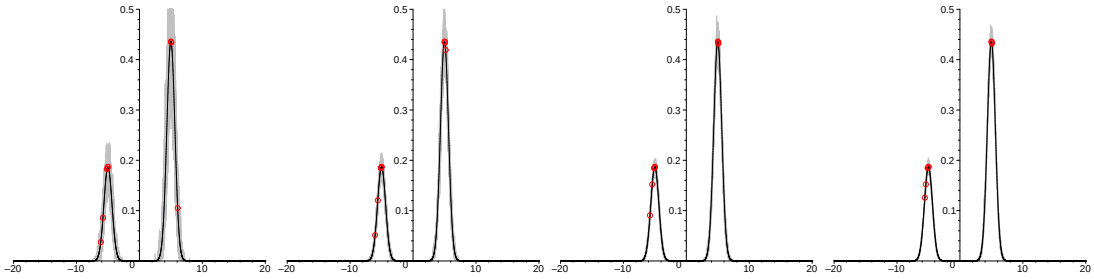


Figure 11: The "increasing number of points" phase of the MCSEM algorithm. In black, the theoretical distribution to maximize and in gray, the estimated one using Monte Carlo numerical integration. From left to right, the parameter N (number of sampling points used to estimate the integral) is increased to obtain increasingly accurate estimations of the distribution.

6 Conclusion and future research

We have proposed in this paper a unifying framework for exact and approximate Bayesian inference. Symbolic probability expressions have been presented as an abstraction tool for exact, approximate, and mixed (exact/approximate) probabilistic models' incremental construction and reuse.

For exact inference, the Successive Restrictions Algorithm (SRA) has been presented. This algorithm aims to construct a symbolic representation of the target distribution by finding a marginalization ordering that takes into account the computational constraints of the application. Preliminary implementations of the SRA algorithm on parallel architectures (SMP and Cluster) have also been developed. However, future studies will search for more appropriate optimality criteria for the cluster architecture. These criteria must especially take into account network communication latencies and data transfer between cluster nodes.

For the approximate inference part, several approximations levels were proposed and the corresponding algorithms presented. The algorithm called "MCSEM" (for Monte Carlo Simultaneous Estimation and Maximization) has been especially proposed to solve the problem of maximizing a posteriori high-dimensional distributions that involve (in the general case) high-dimensional integrals (or sums). Experimental results have demonstrated the effectiveness and the robustness of this algorithm. However, additional studies are required to improve them for both the integration and the optimization problems. For the integration problem, numerical integration can be avoided when the integrand is a product of generalized normals (Dirac delta functions and Gaussians) and when the model is linear or can be linearized (variances are small enough). The optimization algorithm may also be improved by using a local derivative-based method after the convergence of the MCSEM algorithm.

References

Arnborg, S., & Proskurowski, A. (1989). Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete and Applied Mathematics*, *23*(1), 11–24.

Becker, A., & Geiger, D. (2001). A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence*, *125*(1-2), 3–17.

Bellot, D., & Bessière, P. (2003). Approximate discrete probability distribution representation using a multi-resolution binary tree. In *Proc. of the International Conference on Tools for Artificial Intelligence - ICTAI 2003*, Sacramento, California.

Box, G. E. P., & Muller, M. E. (1958). *Annals Math. Stat.*, Vol. 29, chap. A note on the generation of random normal deviates, pp. 610–611.

Cooper, G. (1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, *42*, 393–405.

Corana, A., Martini, C., & Ridella, S. (1987). Minimizing multimodal functions of continuous variables with the 'simulated annealing' algorithm. *ACM Transactions on Mathematical Software*, *13*, 262–280.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*.

D'Ambrosio, B., Shachter, R. D., & DelFavero, B. A. (1990). Symbolic probabilistic inference in belief networks. In *Proc. of AAAI'90*, pp. 126–131.

Darwiche, A., & Provan, G. (1997). Query DAGs: A Practical Paradigm for Implementing Belief Network Inference. *J. Artif. Intellig. Res. (JAIR)*, *6*, 147–176.

Dechter, R. (1996). Bucket elimination: A unifying framework for probabilistic inference. In Horvits, E., & Jensen, F. (Eds.), *Proc. of the Twelveth Conf. on Uncertainty in Artificial Intelligence*, pp. 211–219, Portland, Oregon.

Dechter, R. (1999). Bucket elimination: A unifying framework for probabilistic inference. *Artificial Intelligence*, *113*(1-2), 41–85.

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI.

Eyal, A. (2001). Efficient approximation for triangulation of minimum treewidth. In *Proc. of Uncertainty in Artificial Intelligence (UAI'17)*, pp. 7–15.

Feller, W. (1968). *An introduction to Probability Theory and its applications. Third Edition*. John Wiley, New York.

Fill, J. A. (1991). *Annals of Applied Probability*, Vol. 1, chap. Eigenvalue bounds on convergence to stationarity for non reversible Marcov chains, with an application to the exclusion process, pp. 62–87.

Fox, D., Burgard, W., Dellaert, F., & Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–741.

Geweke, J. (1996). Monte Carlo simulation and numerical integration. In Amman, H., Kendrick, D., & Rust, J. (Eds.), *Handbook of Computational Economics*, Vol. 13, pp. 731–800. Elsevier North-Holland, Amsterdam.

Grefenstette, J. J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, *3*, 225–245.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Jaakkola, T. S., & Jordan, M. I. (1999). Variational probabilistic inference and the QMR-DT network. *J. Artif. Intellig. Res. (JAIR)*, *10*, 291–322.

Jensen, F. (1996). *An Introduction to Bayesian Networks*. UCL Press.

Jensen, F. (2001). *Bayesian Networks and Decision Graphs*. Statistics for Engineering and Information Science. Springer.

Jordan, M., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*.

Jordan, M., & Weiss, Y. (2002). Graphical models: Probabilistic inference. In Arbib, M. (Ed.), *The Handbook of Brain Theory and Neural Networks*. MIT Press, Amsterdam.

Keller, A. (1996). The fast calculation of form factors using low discrepancy point sequence. In *Proc. of the 12th Spring Conf. on Computer Graphics*, pp. 195–204, Bratislava.

Kjaerulff, U. (1990). Triangulation of graphs - algorithms giving small total state space. Technical report R 90-09, Dept. of Mathematics and Computer Science, Aalborg University.

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their applications to expert systems. In *Proc. of the Royal Statistical Society*, No. 50 in B, pp. 154–227.

Li, Z., & D'Ambrosio, B. (1994). Efficient Inference in Bayes Networks As A Combinatorial Optimization Problem. *Int. J. of Approximate Reasoning*, *11*, 55–81.

MacKay, D. G. C. (1996). Introduction to monte carlo methods. In *Proc. of an Erice summer school, ed. M.Jordan*.

Madsen, A. L., & Jensen, F. V. (1998). Lazy propagation in junction trees. In *Proc. of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 362–369. Morgan Kaufmann Publishers.

Mazer, E., Ahuactzin, J. M., & Bessière, P. (1998). The Ariadne's Clew algorithm. *J. Artif. Intellig. Res. (JAIR)*, *9*, 295–316.

Mekhnacha, K., Bessière, P., & Mazer, E. (2000a). A Bayesian framework for geometric uncertainties handling. In *Proc. of the Twent. Int. Workshop on Inference and Maximum Entropy Methods in Science and Engineering (MaxEnt 2000)*, Gif-sur-Yvette, France.

Mekhnacha, K., Mazer, E., & Bessière, P. (2000b). A robotic CAD system using a Bayesian framework. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2000)*, Vol. 3, pp. 1597–1604, Takamatsu, Japan.

Mekhnacha, K., Mazer, E., & Bessière, P. (2001). The design and implementation of a Bayesian CAD modeler for robotic applications. *Advanced Robotics, the Int. J. of the Robotics Society of Japan*, *15*(1), 45–70.

Metropolis, N., Rosenblusth, A. W., Rosenblusth, M. N., Teller, A., & Teller, E. (1953). Equation of state by fast computing machines. *Journal of Chemical Physics*, *21*, 1087–1092.

Neal, R. M. (1993). Probabilistic inference using Markov Chain Monte Carlo methods. Research report CRG-TR-93-1, Dept. of Computer Science, University of Toronto.

Pearl, J. (1982). Reverand bayes on inference engines: A distributed hierarchical approach. In *Proc. of the AAAI National Conference on AI*, pp. 133–136, Pittsburgh.

Pearl, J. (1986). *Uncertainty in Artificial Intelligence*, chap. A constraint-propagation approach to probabilistic reasoning, pp. 371–382.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Mateo, California.

Reed, B. A. (1992). Finding approximate separators and computing tree width quicly. In *Proc. of the 24th Annual ACM Symposium on the Theory of Computing*, pp. 221–229.

Robertson, N., & Seymour, P. D. (1995). Graph minors. xiii. the disjoint paths problem. *Journal of Combinatorial Theory*, *63*(B), 65–110.

Rose, D. J. (1970). Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, *32*(3), 597–609.

Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo method*. John Wiley and Sons.

Shenoy, P., & Shafer, G. (1990). *Uncertainty in AI*, Vol. 4, chap. Axioms for Probability and Belief-Function Propagation. R. Shachter et al.

Shokhet, K., & Geiger, D. (1997). A practical algorithm for finding optimal triangulations. In *Proc. of AAAI'97*, pp. 187–190.

Smail, L. (2004). *Algorithmique pour les Réseaux Bayésiens et leurs Extensions*. Thèse de doctorat, Université de Marne-La-Valée, France.

Smith, A. F., & Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related monte carlo methods. *Journal of the Royal Statistical Society B*, *55*, 3–23.

Zhang, N. L., & Poole, D. (1994). A simple approach to bayesian network computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, pp. 171–178.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399