Brief Survey on Computational Solutions for Bayesian Inference

José Domingos Alves[a], João Filipe Ferreira[a], Jorge Lobo[a], Jorge Dias[a,b]

Abstract— **In this paper, we present a brief review of research work attempting to tackle the issue of tractability in Bayesian inference, including an analysis of the applicability and trade-offs of each proposed solution. In recent years, the Bayesian approach has become increasingly popular, endowing autonomous systems with the ability to deal with uncertainty and incompleteness. However, these systems are also expected to be efficient, while Bayesian inference in general is known to be an NP-hard problem, making it paramount to develop approaches dealing with this complexity in order to allow the implementation of usable Bayesian solutions. Novel computational paradigms and also major developments in massively parallel computation technologies, such as multi-core processors, GPUs and FPGAs, provide us with an inkling of the roadmap in Bayesian computation for upcoming years.**

I. Introduction

In recent years, Bayesian models and inference techniques have been used in different areas of science and technology, such as physics, biology, economy and artificial intelligence (AI). In this particular field, in fact, namely in robotics, the Bayesian approach has been shown to be particularly suitable to deal with incompleteness and uncertainty in models of perception, decision and cognition [1], [2], [3], [4]. Unfortunately, even when the computations performed to implement this approach are made using closed-form expressions, they may still be intractable for the applications under consideration (e.g. when real-time performance is required, such as in the case the AI examples mentioned above).

In this paper, we present a brief review of research work attempting to tackle the issue of tractability in Bayesian inference, including an analysis of the applicability and trade-offs of each proposed solution, starting with a summary of inference algorithms and general implementations (section II-A), then an overview of probabilistic programming languages (section II-B), followed by a survey of state-of-the-art massively parallel implementations of inference using recent advances in technologies such as multi-core processors, GPUs and FPGAs (section II-C), and finishing with a discussion on the probable roadmap in Bayesian computation for upcoming years (section III).

This work was supported by the European Commission collaborative FET project BAMBI – Bottom-up Approaches to Machines Dedicated to Bayesian Inference – FP7-ICT-2013-C, project number 618024, and also partially supported by the Portuguese Foundation for Science and Technology (FCT) and by the European Commission via the COMPETE programme [project grant number FCOMP-01-0124-FEDER-028914, FCT Ref. PTDC/EEI-AUT/3010/2012].

[a]AP4ISR team, Institute of Systems and Robotics (ISR) Dept. of Electrical & Computer Eng., University of Coimbra. Pinhal de Marrocos, Pólo II, 3030-290 COIMBRA, Portugal `jfilipe@isr.uc.pt`.

[b]Jorge Dias is also with Khalifa University of Science, Technology, and Research Abu Dhabi 127788, UAE.

II. Bayesian Inference in Practice – Bayesian Computation

The application of the Bayesian approach to modelling consists in establishing a joint distribution describing a *generative model*, therefore encoding the probability of any set of data D being *generated* from any given hypothesis H. *Bayesian inference* is therefore the process of determining information based on the conditional probabilities of any hypothesis given the available data by applying Bayes' rule

$$\begin{aligned} P(H \mid D) &= \frac{P(H)P(D \mid H)}{P(D)} \\ &= \frac{P(H)P(D \mid H)}{\sum_{H \in \mathscr{H}} P(H)P(D \mid H)}, \end{aligned} \tag{1}$$

where "information", in this context, can mean the exact values or estimates of (1) the probability of a specific hypothesis or set of hypotheses $[H = h]$ given a specific instantiation of the data $[D = d]$, expressed as $P(h|d)$; (2) the full posterior distribution, $P(H \mid d)$; or it may also mean (3) a probabilistic decision based on the posterior distribution in favour of specific hypotheses, $f_h(P(H \mid d))$. In the text that follows, unless otherwise stated, we will be referring to case (2), given that cases (1) and (3) are partial results of the former.

We can therefore define *Bayesian computation* in generic terms as the act of *executing Bayesian inference*. If H and D are continuous random variables, there are in general no closed-form expressions for computing inference (with the notable exception of Kalman filters); if, on the other hand, the generative model is based on discrete random variables, although inference is computable, it is frequently intractable due to the so-called *curse of dimensionality* caused by the cardinality of the space of hypotheses $\mathscr{H}$. Moreover, determining the actual computational expression for inference in the discrete case has been shown to be in general NP-hard [1], [2].

In the following text, we will be analysing the approaches that have been proposed for Bayesian computation so as to overcome these problems.

A. Algorithms for Bayesian Computation

Guo and Hsu presented a comprehensive survey of algorithms for Bayesian inference in [15], where they provide the widely accepted taxonomy reproduced in Fig. 1. *Exact inference* consists of the process of computing the exact values of probabilities from the posterior distribution as accurately as possible (limited, of course, by the numerical precision available), and is generally considered in the context of the

TABLE I

WORKS ON EXACT ALGORITHMS.

Algorithm	Reference	Topology	Complexity*
Polytree	[5], [6]	for polytree only	polynomial complexity (n)
Conditioning	[6]	generic BN	NP-hard
Clustering (Clique/Junction tree)	[7]	sparse BNs	exponential (size of the largest clique)
Arc Reversal/Node Reduction	[8]	generic BN	-
Variable/Bucket Elimination	[9]	generic BN	NP-Complete
Symbolic Inference	[10]	generic BN	-
Differential Method	[11]	BN as polynomial	$O(n^2 \cdot exp(w))$
Successive Restriction Algorithm	[12], [13]	generic Bayesian models	-
Adaptive inference	[14]	generic BN	$O(d^w)$
Adaptive conditioning	[14]	generic BN	$O(n \cdot exp(w))$

*n is the number of nodes, w the tree-width and d the domain size of the variables

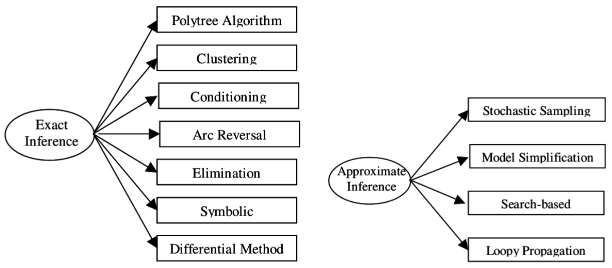

Fig. 1. Categories for exact and approximate inference algorithms (adapted from [15].

exhaustive computation of the full probability distribution. In theory, this would be the natural way of regarding the inference process, except that, as we have seen above, in many cases exact inference is not feasible. In those cases, Bayesian practitioners have resorted to *approximate inference*, for which exactness and exhaustiveness are sacrificed and replaced by an approximate description of the posterior distribution.

Therefore, based on this taxonomy, exact inference algorithms attempt to make the exhaustive computation of the probabilities composing the posterior distribution tractable by simplification of the corresponding computational tree (a first attempt in tackling the curse of dimensionality problem), in an effort to compute the full posterior distribution with a minimum amount of computational burden. Ever since the seminal work by Pearl [16], introducing Bayesian networks (BNs) and belief updating and propagation processors, with few exceptions, these have consisted mostly of graphical and tree-based methods. Approximate inference algorithms, on the other hand, attempt to generate the "best" possible approximation (according to a set of optimisation objectives) of the posterior distribution.

In Tables I and II, a summary of the research work on each of these types of algorithms is presented. The description of most of these algorithms can be found in [15]. In this survey, the authors found that most of the algorithms for real-time inference are *anytime algorithms*. *Anytime algorithms*, as their name indicates, return a result at any time during execution [17], while still producing results of a guaranteed quality [15]. They are classified as iterative refinement algorithms that are able to quickly

TABLE II

WORKS ON APPROXIMATE INFERENCE ALGORITHMS.

Algorithm	Reference
Stochastic Sampling/Monte Carlo	[19], [20], [21], [22], [23]
Model Simplification	[24], [25]
Search-Based Methods	[26], [27]
Loopy Belief Propagation	[28]

generate an imprecise answer and able to refine it using a number of subsequent iterations. It is noted in [15] that most of the approximate inference algorithms can be implemented to run under real-time constraints as they can be used as *anytime algorithms* by applying them iteratively. Examples of more recent developments presented in Tables I and II include [12] and [13], where an algorithm called *Successive Restrictions Algorithm* is presented, a goal-oriented approach that attempts to find an efficient marginalisation ordering for an arbitrary joint probability distribution. Additionally, in [14] an adaptive exact inference approach for graphical models is presented that takes advantage of previously computed quantities to perform inference more rapidly than from scratch (*adaptive inference*). Finally, Ramos and Cozman [18] describe an investigation on methods that balance time and space constraints against the quality of Bayesian network inferences is presented. The result of this investigation is an adaptive conditioning algorithm, that works by dividing a Bayesian network into sub-networks and processing each sub-network with a combination of exact and *anytime algorithms* strategies.

To conclude, Korbinian [29] presented a relevant comparison of exact static and dynamic Bayesian context inference methods for activity recognition we strongly recommend the interested reader to refer to.

B. Probabilistic Modelling Languages

Implementing Bayesian models and inference for cognitive applications implies an identification of the Bayesian approach as an alternative to traditional logic [1], [2], [36]. Therefore, to provide a reasonable way of applying the algorithms presented in the previous section in this context, Bayesian practitioners realistically need a way to universally combine the definition and specification of generative models and resulting probabilistic representations with first-order logic [37] in a user-friendly fashion. Moreover, since Bayesian computation has generally been implemented using

TABLE III
EXAMPLES OF PROBABILISTIC PROGRAMMING LANGUAGES.

Language	Reference	Style/paradigm	Inference Algorithm	Models
IBAL	[30]	Functional Programming	Variable Elimination	BN/hidden Markov models
Church	[31]		Markov chain Monte Carlo	
PRISM	[32]	Logic-based		BN/hidden Markov models
Blog	[33]		Markov chain Monte Carlo	
Factorie	[34]	Imperative-style	Markov chain Monte Carlo	Relational factor graphs
Figaro	[35]	Object-oriented paradigm	Variable Elimination/Metropolis-Hasting	Bayesian Models (broad sense)
ProbT	[13], [2]	Declarative style	Successive Restriction Algorithm (exact)/sampling methods (approximate)	Bayesian Models (broad sense)

software, the research community has provided this service in the form of *probabilistic programming languages*. In general, these languages are supported by:

1) an application programming interface (API) that allows the programmer to specify his/her model and provides an interface for learning model parameters as easily as possible;
2) an inference engine that parses the model and establishes a computational process that implements algorithms such as those presented in the previous section.

A number of probabilistic programming languages have been developed using different programming approaches. Some of them are based on functional programming [38], such as IBAL [30] and Church [31], while others are logic-based, such as the PRISM [32] and Blog [33]. An imperative style programming called *Factorie* is presented in [34]. A more recent approach is presented in [30] and [38], where an object-oriented paradigm for probabilistic programming called Figaro is proposed, which combines the programming styles and paradigms of the previous examples, in an attempt to capitalise on their advantages and at the same time avoid their drawbacks. Another recent example, ProBT by ProBAYES [13], uses a more general formalism for Bayesian model representation named *Bayesian Programming* [2], [1]. The Bayesian programming paradigm allows a natural way to specify Bayesian models in a broad sense and probabilistic questions (i.e. inference goals) to “ask” those models.

Table III presents an overview of probabilistic programming languages and their respective styles or paradigms, built-in inference algorithms, and the type of models they support. Note that, however useful these languages might be, for medium-to-complex models they are still unable to provide real-time performance required by applications such as those mentioned in the introductory section.

C. Massively Parallel Inference

Bayes inference, be it either to exact or approximate, implies the computation of a large number of independent factors. Additionally, conditional independence and hierarchical structures [1] makes it possible to also deal with different parts of most models independently of one another. These two facts combined, *data and structural parallelism*, respectively, mean that Bayesian inference is, in fact, an excellent candidate for massively parallel computation. As a consequence, the availability of emerging massively parallel computational architectures, such as multi-core CPUs, GPUs and FPGAs, have resulted in a large amount of research that attempts to take advantage of the parallel trait of Bayesian inference to drastically lower execution times.

From 2006 to 2011, the research group lead by Professor Viktor Prasanna at the University of Southern California produced a vast body of work contributing with solutions for the implementation of exact inference in multi/many-core CPUs and GPUs. Starting in 2006, Namasivayam et al. presented a study on parallel implementations of inference for generic BNs [39], with prior transformation into junction trees [40], using OpenMP and MPI. Performance for three different types of junction trees was evaluated: linear junction trees, balanced trees and random junction trees. Xia et al. [45], [42], [43], [46], [41], [44] followed up in the study of parallel implementations for exact inference in junction trees deployed in CPUs until 2011. In 2010, Jeon, Xia et al. [47] also presented an adapted implementation of [45] and [42] on a heterogeneous CPU-GPGPU system.

A similar approach was proposed in 2012 by Zheng and Chong in a joint effort from Carnegie Mellon University and the University of California, Berkeley, [48], where Bayesian network models are also converted in junction trees for exact inference, but using a GPU system. They developed data structures and algorithms that extend existing junction tree techniques, and specifically develop a novel approach to computing each belief propagation message in parallel. They implemented the approach on an NVIDIA GPU and tested it using BNs from several applications.

From 2010 to 2012, at UC Berkeley, the team led by Professor John Wawrzynek fronted a research effort focussed on implementing Bayesian inference using multi-core architectures based on FPGAs. From this team, Lebedev et al. [50] and [51] presented an architecture coined MARC – Many Core Approach to Reconfigurable Computing for inference in Bayesian networks. These works describe a methodology that combines a many-core architectural template and a high-level imperative programming model to efficiently target FPGAs for general-purpose compute-intensive applications. This system enables applications to be expressed using a parallel programming model such as OpenCL and to be targeted to a MARC machine implemented on FPGA. A MARC prototype machine with 48 processing nodes was implemented in a Xilinx Virtex-5 FPGA for a Bayesian network inference problem using Markov Chain Monte Carlo Algorithm. Another work from this group exploiting the FPGA’s distributed memories and its abundant hardware structures was presented in [52]. An FPGA was used to construct a high throughput *“Bayesian Computing Machine”*

TABLE IV

COMPARISON BETWEEN MASSIVELY PARALLEL IMPLEMENTATIONS OF BAYESIAN INFERENCE.

Technology	Reference	Implementation	Inference Type & Optimisation	Input Models	Reported Speedup
multi-core CPU	[39], [40]	MPI/OpenMP	exact (structure-level), pointer jumping	generic BNs	$\approx \times 200$ (1,024 nodes, 256 processors)
	[41]	MPI/OpenMP	exact (structure-level), pointer jumping	junction trees	N.A. (scalability study)
	[42]	C++ (Cell Ext.)	exact (structure-level), junction tree decomposition	junction trees	$\approx 7\times$ (1,024 nodes, 8 Cell BE SPE)
	[43]	MPI	exact (structure-level), junction tree decomposition	junction trees	$\approx 100\times$ (100 cliques, 64 processors)
	[44]	Pthreads	exact (structure-level), junction tree rerooting	junction trees	$\approx 7\times$ (8 cores)
	[45]	MPI/OpenMP	exact (data-level), table operations optimisation	junction trees	N.A. (scalability study)
	[46]	MPI/CGM	exact (data-level), table operations optimisation	junction trees	$\approx 100\times$ (1,024 nodes, 128 processors)
GPU	[47]	CUDA	exact (data- and structure-level), ad. of [45] and [42]	junction trees	$30\times/2\times$ (vs hom./heter. multi-core CPU)
	[48]	CUDA	exact (data-level) cluster-sepset belief propagation	junction trees	≈ 10 (36 cliques)
	[49]	OpenCL	exact (data-level) processors for exact inference	processing nodes	≈ 100 (searched var. card. $\approx 1 \times 10^9$)
FPGA	[50],[51]	OpenCL	approximate, sampling (MCMC)	generic BNs	N.A. (area vs performance study)
	[52]	-	data-level processor for exact & approx. inference	processing nodes	$\approx 100\times/\approx 200\times$ (vs GPU/CPU)
	[53]	-	data-level stochastic processor for exact inference	processing nodes	N.A. (proof-of-concept)

(BCM) for computing numerous important algorithms, including the forward or backward algorithm, the Viterbi algorithm, the iterative "turbo" decoding algorithm, Pearl's belief propagation algorithm, the Kalman filter, and certain fast Fourier transform (FFT) algorithms.

The research team led by Professor Jorge Dias at the Institute of Systems and Robotics (University of Coimbra) have presented recent work on Bayesian inference exploiting data-level parallelism. In particular, Duarte et al. [53] presented proof-of-concept work on a general-purpose "Bayesian Machine" (BM) implemented on FPGA, a processing node that takes as input soft evidence from observations and prior probability distributions on the conjunction of searched variables and returns as a resulting output the posterior distribution. Although this approach allows for the specification of a complete model using ProBT that then outputs a computational tree that is mapped to a corresponding BM, the focus of the work was on designing and implementing the actual stochastic processing units that perform approximate computations of the probabilities that exhaustive Bayesian inference needs for full posterior probability retrieval. Finally, Ferreira et al. [49], in a sister publication also submitted to this workshop, present SIMD principles of tractable computation at data-level of models with high-dimensional variables, implemented using OpenCL on GPU (structure-level parallelisation is performed by hand). These SIMD techniques are put to test on a hierarchical Bayesian framework for gaze estimation for human-robot interaction, and is proven to achieve the real-time performance required by this particular application.

Table IV summarises and compares the research work surveyed in this section. Given the diversity of this body of work – exploitation of either data- or structure-level parallelism (or both), different types of input models, different performance metrics, etc. – it is nearly impossible to fairly compare these implementations with one another directly. Moreover, in most cases, using the proposed approaches as off-the-shelf solutions is infeasible, since the implementations are in general not packaged to be open-source. However, all of the proposed solutions bring to light astonishing advances in terms of Bayesian inference implementation, opening up several paths of a future roadmap for Bayesian modelling and computation.

III. DISCUSSION

The brief survey presented in this paper shows that solutions for Bayesian computation have evolved dramatically since the late twentieth century. We have witnessed the development of a solid body of work concerning algorithms that have dramatically improved the efficiency of computational solutions for Bayesian inference. Bayesian modellers currently have at their disposal probabilistic programming languages that provide them with APIs that conceal the complexity of these computational solutions through high-level abstractions. Moreover, many algorithms are currently being adapted for efficient implementations in specialised hardware, taking full advantage of the data- and structure-level parallelisms inherent to Bayesian inference.

Nonetheless, the NP-hard nature of Bayesian computing still makes time-critical applications in many cases impractical, and despite the effort put into providing user-friendly probabilistic programming languages, they are not yet efficient enough for generalised use – in fact, most practitioners still tend to develop proprietary solutions for their inference needs. An important reason for this is currently believed to be that common von Neumann architectures are not natively designed to perform Bayesian inference. In fact, Bayesian computations overload standard von Neumann machines due to the large dimensionality of the underlying entities, leading to slow computations. For many practical applications for which inference is needed, von Neumann machines present performance, power and area bottlenecks, making them costly and inefficient.

As works such as [53], [49], [52] imply, the current roadmap for Bayesian computation seems to be research on a solution for what might be called a universal Bayesian inference machine (Fig. 2). As a matter of fact, currently there are numerous projects that propose novel computing hardware natively capable of performing inference, in an attempt to take a step forward towards energy-efficient and limited resources hardware but able to compute any inference problem in real-time, surpassing the main drawback of probabilistic computations on standard von Neumman architectures.

Analysing these projects and related research, one might identify three different (albeit sometimes overlapping) trends: (1) neuromorphic hardware development, with the argument

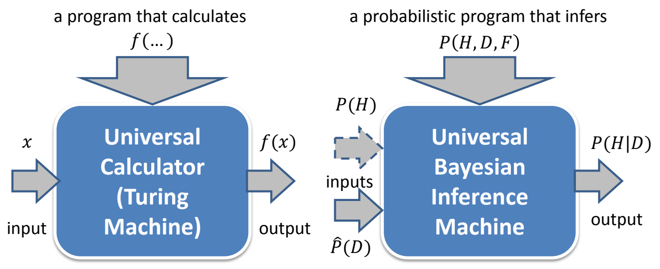
a program that calculates
$f(\dots)$
x
input
Universal Calculator (Turing Machine)
$f(x)$
output
a probabilistic program that infers
$P(H,D,F)$
$P(H)$
inputs
$\hat{P}(D)$
Universal Bayesian Inference Machine
$P(H|D)$
output

Fig. 2. The universal Bayesian inference machine as a generalisation of Turing's universal calculator (adapted from [54]). Such a machine would be able to take a probabilistic program specifying a joint distribution $P(H,D,F)$ on hypotheses, observed data and free variables (cf equation (1)), and respective structural information (i.e. its decomposition), apply it to inputs consisting of (soft evidence concerning) observed data D (and perhaps prior distributions on hypotheses H resulting as outputs from other Bayesian inference machines), and obtain the posterior distribution $P(H \mid D)$ after marginalising out the free variables (or perhaps an approximation of the posterior, or even only a sample of interest).

that the brain seems to natively perform probabilistic inference; (2) stochastic hardware for sampled, approximate inference, as proposed for example by Mansinghka [54], who argues that "[c]ritically, inference machines do not make it easy to calculate probabilities, but only to simulate (or sample) good guesses"; and also (3) stochastic hardware for approximate exhaustive probability computation, with the argument that "local" precision may be sacrificed for the sake of tractability.

For example, the goal of the IBM SyNAPSE (Systems of Neuromorphic Adaptive Plastic Scalable Electronics) project[1] was the development of a brain-inspired computer. The resulting TrueNorth framework has a parallel, distributed, modular, scalable, fault-tolerant, flexible architecture that integrates computation, communication, and memory and has no clock. TrueNorth project used neurons as inspiration for computational building blocks (using standard fixed point arithmetic).

For DARPA's UPSIDE (Unconventional Processing of Signals for Intelligent Data Exploitation) project[2], on the other hand, instead of traditional CMOS-based electronics, the project team envisions arrays of physics-based devices performing the processing chores. Unlike traditional digital processors that operate by executing specific instructions to compute, UPSIDE arrays would rely on higher-level computational elements based on probabilistic inference embedded within a digital system.

The BAMBI (Bottom-up Approaches to Machines Dedicated to Bayesian Inference) collaborative project[3], funded by the European Commission's FP7 "Future Emerging Technologies" programme, takes inspiration from biological and physical systems to attain the long term goal of building an energy-efficient probabilistic computer. The three main axes in this project (theory of probabilistic computation, probabilistic inference in biology, and hardware implementation) imply an ambitious bottom-up approach, starting from the development of elementary components, followed by research on how to combine them to build more complex systems, as follows: (1) the study of Bayesian gates operating on probability distributions on binary variables as the building blocks of our probabilistic algebra, in fact a generalisation of logical operators in Boolean algebra; (2) the interpretation of elementary cell signalling pathways as biological implementation of these probabilistic gates, with the additional hope of gaining new insights for innovative probabilistic hardware implementation; (3) the association of conventional electronics and novel stochastic nano-devices to build the required hardware elements, leading to new artificial information processing systems, which could, in the future, outperform classical computers in tasks involving a direct interaction with the physical world.

The first steps to building dedicated hardware to perform fast probabilistic inference using energy-efficient hardware have been taken. For example, Vigoda [55] presented a system that performs exact inference using continuous time analog circuits. Also, Mansinghka [56], the creator of the Church programming language presented in section II-B, proposed the compilation of a specification written in this language into an electronic circuit that performs approximate inference using samplers to represent probability distributions and using Metropolis algorithms.

Nevertheless, although we are closer to building inference machines that perform Bayesian computation with the energy- and time-efficient requirements that Bayesian practitioners require, as we hope to have shown with this brief survey, there is still a lot of exciting ground to cover and paths to explore.

[1]`http://www.research.ibm.com/articles/brain-chip.shtml`
[2]`http://www.darpa.mil/program/unconventional-processing-of-signals-for-intelligent-data-exploitation`
[3]`https://www.bambi-fet.eu/`

References

[1] J. F. Ferreira and J. Dias, *Probabilistic Approaches for Robotic Perception*. Springer, 2014.

[2] P. Bessiere, E. Mazer, J. M. Ahuactzin, and K. Mekhnacha, *Bayesian programming*. CRC Press, 2013.

[3] P. BessiÃ¨re, C. Laugier, and R. Siegwart, eds., *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems*, vol. 46 of *Springer Tracts in Advanced Robotics*. Springer-Verlag, 2008.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge: MIT Press, 2005.

[5] J. H. Kim and J. Pearl, "A computational model for causal and diagnostic reasoning in inference engines," in *8th International Joint Conference on Artificial Intelligence*, (Karlsruhe, West Germany), pp. 190–193, 1983.

[6] J. Pearl, "Fusion, propagation and structuring in belief networks," *Artificial Intelligence*, vol. 29, no. 3, pp. 241–288, 1986.

[7] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their applications to expert systems,," *Proceedings of the Royal Statistical Society, Series B., 50, 154-227*, 1988.

[8] R. Shachter, "Intelligent probabilistic inference," *In J. F. Lemmer and L. N. Kanal, editors, Uncertainty in Artificial Intelligence, pages 371–382. North Holland, Amsterdam*, 1986.

[9] N. L. Zhang and D. Poole, "A simple approach to Bayesian network computations," in *Proc. Tenth Canadian Conference on Artificial Intelligence*, pp. 171–178, 1994.

[10] R. D. Shachter, B. D'Ambrosio, and B. D. Del Favero, "Symbolic probabilistic inference in belief networks," in *Proc. 8th National Conference on Artificial Intelligence*, (Boston), pp. 126–131, MIT Press, 1990.

[11] A. Darwiche, "A differential approach to inference in bayesian networks," *In UAI'00*, 2000.

[12] L. Smail and J. P. Raoult, "Successive restrictions algorithm in bayesian networks," *A.F. Famili et al. (Eds.): IDA 2005, LNCS 3646, pp. 409–418/Springer-Verlag Berlin Heidelberg*, 2005.

[13] K. Mekhnacha, J.-M. Ahuactzin, P. Bessière, E. Mazer, and L. Smail, "Exact and approximate inference in ProBT," *Revue d'intelligence artificielle*, vol. 21, no. 3, pp. 295–331, 2007.

[14] O. e. a. Summer, "Adaptive exact inference in graphical models," *Journal of Machine Learning Research 12 (2011) 3147-3186*, 2011.

[15] H. Guo and W. Hsu, "A survey of algorithms for real-time bayesian network inference," in *Joint AAAI-02/KDD-02/UAI-02 Workshop on Real-Time Decision Support and Diagnosis Systems*, 2002.

[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc. (Elsevier), revised second printing ed., 1988.

[17] M. Boddy, "Anytime problem solving using dynamic programming," *In Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), p. 738-743. AAAI Press, San Mateo, CA*, 1991.

[18] F. Ramos and F. Cozman, "Anytime anyspace probabilistic inference," *International Journal of Approximate Reasoning*, vol. 38, pp. 53–80, 2005.

[19] M. Henrion, "Propagating uncertainty in bayesian networks by probabilistic logic sampling," *In Uncertainty in Artificial Intelligence 2, eds. J. Lemmer and L. Kanal, 149-163. New York: Elsevier Science*, 1988.

[20] R. Fung and K. C. Chang, "Weighting and integrating evidence for stochastic simulation in bayesian networks.," *In Uncertainty in Artificial Intelligence 5, pages 209-219, New York, N. Y. Elsevier Science Publishing Company, Inc.*, 1989.

[21] R. D. Shachter and M. A. Peot, "Simulation approaches to general probabilistic inference on belief networks," *In Proc. of the Conf. on Uncertainty in AI, volume 5*, 1990.

[22] S. Geman and G. D., "Stochastic relaxation,gibbs distribution and the bayesian restoration of images.," *IEEE Transactions on Pattern Analysis and Machine IntelligenceIntelligence, 6(6):721–741*, 1984.

[23] S. C. Ahsanul Haque and L. K. C. Aggarwal, "Distributed adaptive importance sampling on graphical models using mapreduce," *IEEE International Conference on Big Data*, 2014.

[24] F. Jensen and S. Andersen, "Approximations in bayesian belief universes for knowledge-based systems.," *In UAI-90, pp. 162–169*, 1990.

[25] T. S. Jaakkola and M. Jordan, "Variational probabilistic inference and the qmr–dt network.," *Journal of Artificial Intelligence Research, 10:291–322,*, 1999.

[26] G. F. Cooper, "Nestor: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge.," *Ph.D. Dissertation, Medical Informatics Sciences, Stanford University, Stanford, CA,*, 1985.

[27] D. Poole, "The use of conflicts in searching bayesian networks," *Proceedings of the Ninth Conference on Uncertainty in AI, Washington D.C., pages 359-367,*, 1993.

[28] K. P. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference: an empirical study," *In Proceedings of Uncertainty in AI, pages 467-475,*, 1999.

[29] F. e. a. Korbinian, "Comparison of exact static and dynamic bayesian context inference methods for activity recognition," *Pervasive Computing and Communications Workshops (PERCOM Workshops).*, 2010.

[30] A. Pfeffer, "The design and implementation of ibal: A general-purpose probabilistic language," *In Lise Getoor and Ben Taskar, editors, Statistical Relational Learning. MIT Press*, 2007.

[31] N. D. Goodman, V. Mansinghka, D. Roy, K. Bonawitz, and J. B. Tenenbaum, "Church: A language for generative models.," *In Conference on Uncertainty in Artificial Intelligence*, 2008.

[32] T. Sato, "A glimpse of symbolic-statistical modeling in prism," *Journal of Intelligent Information Systems*, 2008.

[33] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov, "Blog: Probabilistic models with unknown objects," *Statistical relational learning, p. 373*, 2007.

[34] A. McCallum, K. Rohanemanesh, M. Wick, K. Schultz, and S. Singh, "Factorie: Efficient probabilistic programming for relational factor graphs via imper- ative declarations of structure, inference and learning.," *In NIPS 2008 Workshop on Probabilistic Programming*, 2008.

[35] A. Pfeffer, *Inductive Logic Programming*, ch. Practical probabilistic programming, pp. 2–3. Springer, 2011.

[36] E. T. Jaynes, *Probability Theory: the Logic of Science*. Cambridge University Press, 2003.

[37] C. Howard and M. Stumptner, "A survey of directed entity-relation-based first-order probabilistic languages," *ACM Computing Surveys (CSUR) 47.1*, vol. 47, no. 1, pp. 4:1–4:40, 2014.

[38] A. Pfeffer, "Figaro: An object-oriented probabilistic programming language," *http://www.cs.tufts.edu/ nr/cs257/archive/avi-pfeffer/figaro.pdf*.

[39] V. Namasivayam and V. Prasanna, "Scalable parallel implementation of exact inference in Bayesian networks," in *12th International Conference on Parallel and Distributed Systems – (ICPADS'06)*, IEEE, 2006.

[40] V. Namasivayam, A. Pathak, and V. Prasanna, "Scalable Parallel Implementation of Bayesian Network to Junction Tree Conversion for Exact Inference," in *18th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'06)*, IEEE, Oct 2006.

[41] N. Ma, X. Xia, and V. Prasanna, "Exact Inference On Manycore Processors Using Pointer Jumping," in *IASTED International Conferences on Informatics*, Actapress, 2010.

[42] Y. Xia and V. K. Prasanna, "Parallel exact inference on the Cell Broadband Engine processor," in *2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, Nov 2008.

[43] Y. Xia and V. K. Prasanna, "Junction tree decomposition for parallel exact inference," *2008 IEEE International Symposium on Parallel and Distributed Processing*, Apr 2008.

[44] Y. Xia, X. Feng, and V. Prasanna, "Parallel Evidence Propagation on Multicore Processors," *Journal of Supercomputers*, vol. 57, pp. 189–202, 2011.

[45] Y. Xia and V. K. Prasanna, "Node Level Primitives for Parallel Exact Inference," in *19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07)*, IEEE, Oct 2007.

[46] Y. Xia and V. K. Prasanna, "Scalable Node-Level Computation Kernels for Parallel Exact Inference," *IEEE Transactions on Computers*, vol. 59, pp. 103–115, Jan 2010.

[47] H. Jeon, Y. Xia, and V. K. Prasanna, "Parallel Exact Inference on a CPU-GPGPU Heterogenous System," in *39th International Conference on Parallel Processing*, IEEE, Sep 2010.

[48] L. Zheng, O. Mengshoel, and J. Chong, "Belief Propagation by Message Passing in Junction Trees: Computing Each Message Faster Using GPU Parallelization," Feb. 2012.

[49] J. F. Ferreira, P. Lanillos, and J. Dias, "Fast Exact Bayesian Inference for High-Dimensional Models," in *Workshop on Unconventional computing for Bayesian inference (UCBI), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[50] I. Lebedev, S. Cheng, A. Doupnik, J. Martin, C. Fletcher, D. Burke, M. Lin, and J. Wawrzynek, "MARC: A Many-Core Approach to Reconfigurable Computing," in *2010 International Conference on Reconfigurable Computing and FPGAs*, IEEE, Dec 2010.

[51] I. Lebedev, C. Fletcher, S. Cheng, J. Martin, A. Doupnik, D. Burke, M. Lin, and J. Wawrzynek, "Exploring Many-Core Design Templates for FPGAs and ASICs," *International Journal of Reconfigurable Computing*, vol. 2012, pp. 1–15, 2012.

[52] M. Lin, I. Lebedev, and J. Wawrzynek, "High-Throughput Bayesian Computing Machine with Reconfigurable Hardware," in *FPGA'10 (ACM 978-1-60558-911-4/10/02)*, 2010.

[53] R. Duarte, J. Lobo, J. F. Ferreira, and J. Dias, "Synthesis of Bayesian Machines on FPGAs using Stochastic Arithmetic," in *2nd International Workshop on Neuromorphic and Brain-Based Computing Systems (NeuComp 2015), Design Automation Test Europe (DATE2015)*, 2015.

[54] V. K. Mansinghka, "Beyond calculation: Probabilistic computing machines and universal stochastic inference," in *NIPS Philosophy and Machine Learning Workshop*, 2011.

[55] B. Vigoda, *Analog logic: Continuous-time analog circuits for statistical signal processing*. PhD thesis, Massachusetts Institute of Technology, 2003.

[56] V. K. Mansinghka, *Natively probabilistic computation*. PhD thesis, Massachusetts Institute of Technology, 2009.