On the Hardness of Approximate Reasoning

Dan Roth*
Aiken Computation Laboratory, Harvard University
33 Oxford St., Cambridge, MA. 02138
U. S. A.
danr@das.harvard.edu

Abstract

Many AI problems, when formulated, reduce to evaluating the probability that a propositional expression is true. In this paper we show that this problem is computationally intractable even in surprisingly restricted cases and even if we settle for an approximation to this probability.

We consider various methods used in approximate reasoning such as computing degree of belief and Bayesian belief networks, as well as reasoning techniques such as constraint satisfaction and knowledge compilation, that use approximation to avoid computational difficulties, and reduce them to model-enumeration problems over a propositional domain

We prove that counting satisfying assignments of propositional languages is intractable even for Horn and monotone formulae, and even when the size of clauses and number of occurrences of the variables are extremely limited. This should he contrasted with the case of deductive reasoning, where Horn theories and theories with binary clauses are distinguished by the existence of linear time satisfiability al gorithms. What is even more surprising is that, as we show, even approximating the number of satisfying assignments (i.e., "approximating" approximate reasoning), is intractable for most of these restricted theories.

We also identify some restricted classes oi propositional formulae for which we develop efficient algorithms for counting satisfying assignments.

1 Introduction

Investigating the computational cost of tasks that are of interest to AI has been argued [Levesque, 1986, Valiant, 1984] to be essential to our understanding and our ability to characterize these tasks and to finding knowledge representation systems adequate for them.

•Supported by NSF grants CCR-89 02500 and CCR-92 00884 and by DARPA AFOSR-F4962-92-J- 0466.

The problem discussed most extensively in this context is the problem of propositional satisfiability, the typical NP-hard problem, which is of special concern to AI because of its direct relationship to deductive reasoning. Many other forms of reasoning, including default reasoning, planning and others which make direct appeal to satisfiability, have also been shown to be NP-hard In practice, there is a fundamental disagreement about the implications of this. There is no debate that something has to be given up: restrict the form of the statements in the knowledge base, settle for approximate output and so on. One consequence of the intensive research in that direction is the identification of restricted languages for which propositional satisfiability can be solved efficiently (e.g.. Horn).

In this paper we consider a related problem, that of enumerating satisfying assignments of propositional formulae. We argue that the role played by satisfiability problems in many AI problems in which *deduction* is of special concern, is replaced by that of counting satisfying assignments when *approximate reasoning* techniques are used. To support this argument we show that various methods used in approximate reasoning, such as computing degree of belief and Bayesian belief networks, as well as reasoning techniques that, use approximation to avoid computational difficulties such as constraint satisfaction and knowledge compilation, can be reduced to solving enumeration problems.

We analyze the computational complexity of counting satisfying assignments of propositional languages, and prove that this is intractable even for Horn and monotone formulae, and even when the size of clauses and number of occurrences of a variable in the formula are extremely limited This should be contrasted with the case of deductive reasoning, where Horn theories and theories with binary clauses are distinguished by the existence of linear time algorithms for their satisfiability. What is even more surprising is that, as we show, even approximating the number of satisfying assignments (that is, "approximating" approximate reasoning), is intractable for most of those[1] restricted theories. We identify some restricted classes of propositional formulae for which we develop efficient algorithms for counting satisfying assignments While we show that our positive results can sometimes be used to find tractable languages for the approximate reasoning technique discussed, the Implica-

tions of our fairly surprising and widely applicable hardness results are not fully clear.

In the next section we briefly give background material from computational complexity. Section 3 summarizes and sketches the proofs of our technical results, which we put in the context of various approximate reasoning techniques in Section 4.

2 The Computational Complexity of Counting Problems

We give in this section a brief overview of the computational complexity of enumeration problems and the related problems of approximate enumeration and random generation of solutions. For a detailed discussion consult [Valiant, 1979a; Valiant, 1979b; Carey and Johnson, 1979; Jerrurn *et al.*, 1986].

With a large number of decision problems we can naturally associate a counting problem. For example, counting the number of satisfying assignments of a Boolean formula, counting the number of perfect matchings in a bipartite graph and counting the number of cycles in a graph. Clearly, the counting version is at least as hard as the decision problem but in many cases, even when the decision problems is easy, no computationally efficient method is known for counting their number. The class of #P was introduced by Valiant. [Valiant, 1979a, Valiant, 1979b] in an effort to explain this phenomena.

In particular, is was shown that counting the number of satisfying assignments of a CNF formula as well as the counting versions of many other NP-complete problems are complete for #P, but counting versions of some problems in P are also complete for #P. Examples include counting the number of satisfying assignments of a DNF formula, counting the number of cycles in a graph and many other problems [Valiant, 1979a; Valiant, 1979b; Provan and Ball, 1983].

Problems that are #P-complete are at least as hard as NP-complete problems, but probably much harder Evidence to the hardness of problems in #P is sup plied by a result of [Toda, 1989] which implies that one call to a #P oracle suffices to solve any problem in the polynomial hierarchy in deterministic polynomial time. This may serve also as indication that #P is outside of the polynomial hierarchy. It is therefore natural to consider the problem of approximate counting. The notion of approximation we use is that of *relative approximation* [Karp and Luby, 1983; Stockmeyer, 1985; Jerrurn *et ai,* 1986]. We say that *M' approximates M* within t iff

$$M'/(1+\epsilon) \leq M \leq M'(1+\epsilon)$$

Indeed, approximating a solution to a #P problem might be easier than finding an exact solution In fact, it is no harder than solving NP hard problems [Stockmeyer, 1985]. For example, there exists a polynomial time randomized algorithm that approximates the number of satisfying assignments of a DNF formula within any constant ratio [Karp and Luby, 1983; Jerrurn *et* al., 1986]. It is possible, though, for a #P-complete problem, even if its underlying decision problem is easy, to resist even an efficient approximate solution. An example for that was given in [Jerrurn *et a/.*, 1986], and in this paper we exhibit a similar phenomena. We prove, for various propositional languages for which solving satisfiability is easy, that it is NP-hard to approximate the number of satisfying assignments even in a very weak sense.

We note that a related class of problems of interest to AI, that of randomly generating solutions from a uniform distribution, was shown in [jerrurn *et al,* 1986] to be equivalent to randomized approximate counting, for a wide class of problems. (All natural problems considered here, e.g. finding satisfying assignments of Boolean formulae and various graph problems are in this class.) It is therefore enough, from the computational complexity point of view to consider the problems of exact and approximate counting, as we do here.

3 Summary of Counting Results

Let $\#(SAT, \mathcal{L})$ ($\tilde{\#}(SAT, \mathcal{L})$) denote the problem of counting (approximating) the number of satisfying assignments of a given formula of the propositional language $\mathcal{L}$. Given the problem $\#(SAT, \mathcal{L})$, a problem hierarchy is obtained whenever we place additional restrictions or relaxations on the allowed instances. Given propositional languages $\mathcal{L}_1$ and $\mathcal{L}_2$, define $\mathcal{L}_1 \subseteq \mathcal{L}_2$ if every instance of $\mathcal{L}_1$ is also an instance $\mathcal{L}_2$. (e.g., HORN $\subseteq$ CNF.) Clearly, if we can solve the problem $\#(SAT, \mathcal{L}_2)$ we can solve the problem $\#(SAT, \mathcal{L}_1)$; to prove hardness results it is enough therefore to consider the most restricted languages. The same argument holds for the corresponding approximation problem. Figure 1 summarizes our results; it presents a hierarchy of propositional languages along with a classification according to the complexity of $\#(SAT, \mathcal{L})$ and $\tilde{\#}(SAT, \mathcal{L})$. Based on the above comment these results imply similar results on other, less restricted languages. The following notations are used for propositional languages: **SAT** - Boolean formulae; **MON** - Boolean formulae in which all variables are unnegated (monotone formulae); **CNF** - Boolean formulae in Conjunctive Normal Form; **MONCNF** - monotone CNF; **HORN** - A CNF in which clauses have up to one unnegated variable (Horn clauses); **2BPMONCNF** - A 2MONCNF in which the set of variables can be divided into two sets, and every clause contains one variable from each, **Acyclic-2MONCNF** - A 2MONCNF with the following property: the graph in which nodes correspond to variables and edges connect any two nodes which correspond to variables in the same clause, is a tree. If $LANG$ is a class of Boolean formulae, k, l, integers, then kLANG denotes the subclass of formulae in LANG in which a clause consists of up to k literals; $l\mu$-LANG denotes the class of all LANG formulae in which no variable occurs more than l times. l is the *degree* of the formulae. It is noticeable that for various propositional languages having efficient algorithms for satisfiability, and even for very restricted versions of these (e.g., 3μ-2HORN), exact counting is complete for #P. In fact, for the case of Horn theories, the situation is fully understood, and we

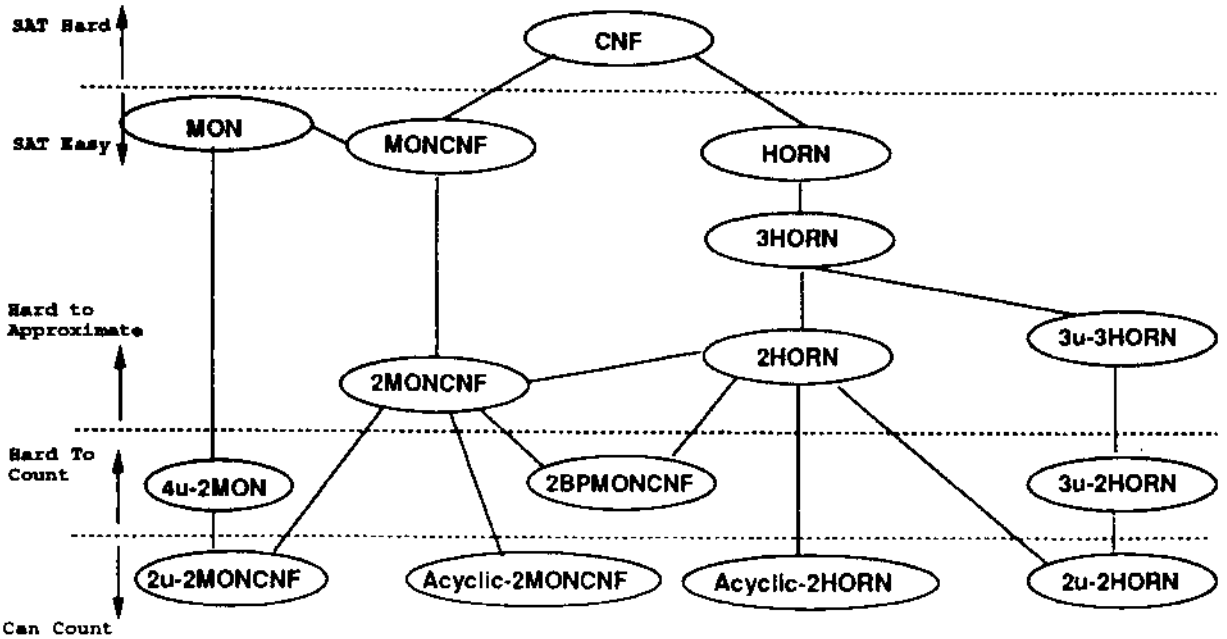

Figure 1: Complexity of (Approximately) Counting Satisfying Assignments

give an efficient algorithm for the only possible case, 2μ-2HORN. The situation for approximate counting is even more surprising: for very restricted classes of Horn theories (e.g., 3μ-3HORN) it is NP-hard to approximate the number of satisfying assignments even within a factor of $2^{\sqrt{n}}$. (See Section 3.1 for the exact, stronger result.) Similar results hold for 2MONCNF theories, for which the bounded degree case is open. Our positive results complete the complexity picture and can be directly applied in some of the reasoning techniques considered.

3.1 Statements of Results

We now formally state the technical results outlined above. We state the results only for some of the important languages; results for other languages can be easily deduced by inclusion, as described in Section 3.

Theorem 3.1 *[Hardness of Exact Counting] Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, counting the number of satisfying assignments of Σ is complete for $\#P$:*
(1) $\mathcal{L}$ = 2MONCNF [Valiant, 1979b]
(2) $\mathcal{L}$ = 2BPMONCNF [Provan and Ball, 1983]
(3) $\mathcal{L}$ = 2HORN
(4) $\mathcal{L}$ = 3μ-2HORN
(5) $\mathcal{L}$ = 4μ-2MON

Theorem 3.2 *[Hardness of Approximation] Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, approximating the number of satisfying assignments of Σ to within a factor of $2^{n^{1-\epsilon}}$, for every ϵ, is NP-hard:*
(1) $\mathcal{L}$ = 2MONCNF
(2) $\mathcal{L}$ = 3μ-3HORN

Proof: (Outline) The main step in the proof is the following lemma[1]:

Lemma 3.3 *For any ϵ, approximating the number of independent sets of a graph on n vertices within $2^{n^{1-\epsilon}}$ is NP-hard.*

Proof: (Outline) We use the "blow-up" technique introduced in [Jerrum *et al.* 1986], to reduce the problem of approximating the number of independent sets in G to the k-INDEPENDENT-SET problem ([Garey and Johnson, 1979]). Given $G(V, E)$, we construct a graph $G'(V', E')$ such that an approximation of the number independent sets in G to within $2^{n^{1-\epsilon}}$ can be used to solve k-INDEPENDENT-SET. ■

To get (1) from Lemma 3.3 we construct a 1-1 correspondence between 2MONCNF formulae and graphs, such that satisfying assignments of the formula correspond to independent sets of the graph. ■

Theorem 3.4 *[Positive Results] Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, then exists an efficient algorithm for counting the number of satisfying assignments of Σ:*
(1) $\mathcal{L}$ = 2μ-2MONCNF
(2) $\mathcal{L}$ = 2μ-2HORN
(3) $\mathcal{L}$ = Acyclic-2MONCNF
(4) $\mathcal{L}$ = Acyclic-2HORN

[1] We thank Mark Jerrum for pointing out to us that the blow-up technique can be used to prove this result, as was discovered by Sinclair [Sinclair, 1988]. We use this same technique here with a slightly different argument.

4 Reducing Approximate Reasoning to Counting

In this section we consider various techniques for approximate reasoning and show that in each case inference is equivalent to solving a counting problem. Thus, the results in Section 3 apply to all these methods. Due to space limitations the techniques description and the interpretation of the results are brief. We elaborate only in the case of computing degree of belief, the underpinning of approximate reasoning, since the results there have wider implications.

4.1 Degree of Belief

The inference of a degree of belief is a generalization of deductive inference, and can be used in case the knowledge base is augmented by, e.g., statistical information, or as an effort to avoid the computationally hard task of deductive inference.

Consider a KB consisting of a propositional theory Σ and assume we would like to assign a *degree of belief* to a particular statement α. This situation is natural in various AI problems such as planning, expert systems and others, where the actions an agent takes may depend crucially on this degree of belief. In [Nilsson, 1986] it is suggested that the kind of reasoning used in expert system is the following: "we are given a knowledge base of facts (possibly, with their associated probabilities); we want to compute the probability of some sentence of interest. ... According to *probabilistic logic,* the probability of a sentence is the sum of the probabilities of the sets of possible worlds in which that sentence is true...

Indeed, the general .approach to computing degree of belief is that of assigning equal degree of belief to all basic "situations" consistent with the knowledge base, and computing the fraction of those which are consistent with the query. Much work has been done on how to apply this principle, and how to determine what are the basic situations [Bacchus, 1990; Bacchus *et al.,* 1992].

We consider here the question of *computing* the degree of belief in a restricted and simpler case, in which the knowledge base consists of a propositional theory and contains no statistical information2. Using the above approach, all possible models of the theory are given equal weight and we are interested in the computational complexity of computing the degree of belief of a propositional formula i.e., the fraction of models that are consistent with a propositional query.

Given a propositional theory Σ, the *probability that* Σ *is satisfied,* P_Σ, is computed over the uniform distribution on a set of *n* variables.

$$P_\Sigma = Prob\{\Sigma \equiv T\} = |SAT(\Sigma)|/2^n$$

Given a propositional theory Σ and a propositional statement α, the *conditional probability* of a with respect to Σ (the *degree of belief in* α), $P_{\alpha|\Sigma}$, is the fraction of satisfying assignments of Σ that satisfy *a:*

$$P_{\alpha|\Sigma} = Prob\{\alpha \wedge \Sigma \equiv T | \Sigma \equiv T\} = \frac{|SAT(\alpha \wedge \Sigma)|}{|SAT(\Sigma)|}$$

Since $|SAT(\alpha)| = P_{\alpha|p \vee \not p}$ for any variable *p,* we have:

Observation 4.1 *Computing (approximating) the degree of belief in a propositional statement with respect to a propositional theory, is equivalent to computing (approximating) the number of models of the statement.*

Based on this observation, our results prove the intractability of computing and even approximating the degree of belief for restricted propositional languages such as Horn and monotone formulae of bounded degree and bounded size of clauses. The observation also implies that the positive results for, e.g., acyclic theories and theories of degree 2 can be applied directly.

4.2 Bayesian Belief Networks

Bayesian belief networks provide a natural method for representing probabilistic dependencies among a set of variables and are considered an efficient and expressive language for representing knowledge in many domains [Holtzman, 1989]. We consider here the class of *multiple connected belief network,* i.e., networks that contain at. least one pair of nodes (variables) that have more than one undirected path connecting them. It has been argued that the expressiveness of these networks is required for representing knowledge in several domains, like medicine. For definitions and an elaborate discussion of Bayesian belief networks, the expressiveness of this representation and the type of inference one can utilize using it see [Pearl, 1988].

The general *inference problem* using belief network is that of calculating the posterior probability *P(S1\S'2),* where *S1 (S2)* is either a single instantiated variable or a conjunction of instantiated variables. The most restricted form of probabilistic inference, determining *P(Y* = T) for some propositional variable *Y* (with no explicit conditioning information), was analyzed by [Cooper, 1990] who proved it is NP-hard. This hardness results for the exact inference problem shows that one cannot expect to develop general-purpose algorithms for probabilistic inference that have a polynomial running time and therefore there is a need to divert attention toward trying to construct *approximation algorithms* for probabilistic inference. Our results show that this is not the case; Cooper's argument can be modified and his results strengthen in the following way: we reduce the problem of counting satisfying assignments of a propositional formula (e.g., in 3SAT) to that of computing the probability that a node in a belief network is true. The results presented in Section 3 imply:

Theorem 4.2 *Computing the probability that a node in a Bayesian belief network is true, is complete for #P. Approximating this probability is NP-hard.*

The proof consists of reducing a counting problem to the inference problem, and is given in the full version of the paper. We note that based on the results in Section 3, formulae from restricted propositional languages can be

2This problem was considered in the first order case [Grove *et* a/., 1992] and it was shown that almost all problems one might want to ask are highly undecidable. In some cases, though, it was shown that the asymptotic conditional probabilities exist, and can be computed. The hardness results we get in the restricted just highlights the computational difficulties in the more general cases.

reduced to an inference problem in a similar way, resulting in even stronger results, in which the topology of the network is further restricted. Recently, [Dagum and Luby, 1991] have proved that even finding an absolute (additive) approximation of a solution to the inference problem is NP-hard.

4.3 Knowledge Compilation

The idea of *knowledge compilation* was introduced by [Selman and Kautz, 1991] as a new approach to developing fast and efficient knowledge representation systems. In this framework, statements represented in a general unrestricted representation language arc cornpiled by the system into a restricted language that allows for efficient inference. Since an exact translation into a tractable form is impossible in general, the system searches for the best approximation of the original information. This process is NP-hard and therefore the technique is called "compilation". The aim is to use approximations to speed up inference, without giving up correctness or completeness: computational costs are shifted from "run-time" to the off-line compilations process. In particular, in [Selman and Kautz, 1991] it is shown how propositional logical theories can be compiled into Horn theories that approximate the original information.

Consider a propositional theory Σ, and let $\mathcal{M}(\Sigma)$ denote its set of models. The sets Σ_{lb}, Σ_{ub} of Horn clauses are a Horn lower-bound and Horn upper-bound, respectively, of Σ, iff

$$\mathcal{M}(\Sigma_{lb}) \subseteq \mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{ub})$$

. Σ_{glb} and Σ_{lub}, the greatest Horn lower-bound and least Horn upper-bound, respectively, of Σ, are called *Horn approximations* of the original theory Σ.

Horn approximation can be used in the inference procedure by testing if $\Sigma_{lub} \models \alpha$ and $\Sigma_{glb} \not\models \alpha$ rather than $\Sigma \models \alpha$. This procedure takes only linear time in the length of the approximations[3].

Taking the "counting" approach, as we suggest in this paper, can shed some light on the problems in knowledge compilation. In knowledge compilation, approximation is defined in terms of *containment*, and not in terms of proximity in the number of models. We show, for example, that it is intractable to even approximate the utility of an approximation, i.e., to determine how often will the approximation help in the deduction process.

Since the length of Σ_{lub} might be exponentially long with respect to the original theory [Kautz and Selman, 1992], which would nullify any gain reasoning with the approximation might provide, various techniques were considered to evaluate the approximation process, and abort it when it is good enough. Our approach rules out the existence of a general technique for that[4]. On the other hand, due to the tight relations between counting satisfying assignments and the quality of the approximation, it might be worthwhile to use our positive results and investigate the question of approximating a theory by languages for which we can efficiently count satisfying assignments.

4.4 Constraint Satisfaction Problems

Constraint satisfaction problems (CSP) provide a convenient way of expressing declarative knowledge, by focusing on local relationships among entities in the domain.

A *constraint satisfaction problem* [Dechter and Pearl, 1988] involves a set of n variables $x_1, \ldots x_n$ having domains $D_1, \ldots D_n$, where each D_i defines the set of values that variable x_i may assume. An *n-ary relation* on these variables is a subset of the Cartesian product $D_1 \times D_2 \times \ldots \times D_n$. A *constraint* between variables is a subset of the Cartesian product of their domains. The set of all n-tuples satisfying all the constraints are the solutions to the CSP. The problem is either to find all the solutions, or one solution.

It is clear that these problems are hard since a constraint satisfaction is a generalization of CNF formulae. In particular, if we consider a network of binary constraints[5] over $D_i = \{0, 1\}$, as is usually done, the problem can be represented as a 2SAT formula.

The counting point of view taken here gives insight into several questions in constraint satisfaction problems. Finding all the solutions is clearly an enumeration problem, and based on the results in Section 3, it is #P-complete for almost all non-trivial cases[6].

Search techniques were traditionally used to solve CSPs, and various heuristics for guiding the search [Dechter and Pearl, 1988] suggest that one rely on counting to evaluate the most probable path to take. Our results show that in general, even these heuristics are computationally intractable. On the other hand, the positive results, e.g., the result for Acyclic formulae, can be used to identify domains for which these problems can be solved efficiently.

5 Conclusions

We have put results on the complexity of counting and approximating the number of satisfying assignments of propositional formulae in the context of various approximate reasoning techniques. The significance of this approach was illustrated by showing that various, supposedly different methods in approximate reasoning can be reduced to counting.

[3]The implication problem for Horn theories can be solved in linear time in the combined length of the theory (KB) and the query. This remains true for even a broader class of queries such as DNF formulae where each disjunct contains at most one negative literal and arbitrary CNF formulae.

[4]Recent results use learning techniques to find a locally-optimal approximation [Greiner, 1992]. There is no guarantee however, that this approximation is "close" to the optimal one, nor that the optimal one approximates the original theory within any reasonable bound, as our techniques show

[5]Not every n.-ary relation can be represented by a network of binary constraints with n-variables [Montanari, 1974].

[6]We comment, though, that Valiant's results ([Valiant, 1979b], Fact 7) imply that under simple conditions (e.g., when finding one solution is easy and the problem satisfies a form of self-reducibility), enumerating the solutions is polynomial in their number even when the counting problem is hard. These conditions trivially hold for Horn formulae, and therefore for subclasses of CSP as well.

Our hardness results seem to indicate that computing degree of belief, as well as other approximate reasoning techniques, are intractable for almost all propositional languages. Moreover, even an approximate computation of the probability was proved to be intractable for a wide class of propositional languages. The fact that most applications are believed to require much more than propositional calculus just highlights these computational difficulties. These results do not rule out the possibility for efficient algorithms that apply in restricted cases, as our positive results suggest; identifying more positive results and investigating how they apply to various techniques might be one direction to extend this work.

On the other hand, the extent to which the hardness results apply calls for a more profound investigation of the implication of these computational difficulties.

Acknowledgments

1 am very grateful to Les Valiant for very helpful discussions and for his comments on an earlier draft of this paper. I would also like to thank Karen Darnels for her comments on earlier drafts of this paper.

References

[Bacchus ei ai, 1992] F. Bacchus, A. Grove, J. V. Halpern, and D. Koller. From statistics to beliefs. In National Conference on Artificial Intelligence, pages 602 608,1992.

[Bacchus, 1990] F. Bacchus. Representing and Reasoning With Probabilistic Knowledge A Logical Approach to Probabilities. MIT Press, 1990.

[Cooper, 1990] F. G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence, 42:393-405, 1990.

[Dagum arid Luby, 1991] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is np-hard. Technical Report KSL-91-51, Knowledge Systems Laboratory, Stanford University. 1991.

[Dechter and Pearl, 1988] R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. Artificial Intelligence, 34:1 38, 1988.

[Garey and Johnson, 1979] M. Garey and D Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco. 1979.

[Greiner, 1992] R. Greiner. Learning near-optimal Horn-approximation. In Proceedings of the 3rd international conference on principles of knowledge representation and reasoning, 1992.

[Grove et ai, 1992] A. Grove, J. Y. Halpern, and D. Koller. Asymptotic conditional probabilities for first-order logic. In ACM Symp. of the Theory of (Computing, number 24, 1992.

[Holtzman, 1989] S. Holtzman. Intelligent Decision Systems. Addison-Wesley, Reading, MA, 1989.

[Jerrum ei ai, 1986] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science, 43:169-188, 1986.

[Karp and Luby, 1983] R.M. Karp and M. Luby. Montecarlo algorithms for enumeration and reliability problems. In IEEE Symp. of Foundation of Computer Science, number 24, pages 56-64, 1983.

[Kautz and Selman, 1992] H. Kautz and B. Selman. Forming concepts for fast inference. In National Conference on Artificial Intelligence, 1992.

[Levesque, 1986] H. Levesque. Making believers out of computers. Artificial Intelligence, 30:81-108, 1986.

[Montanari, 1974] U. Montanari. Networks of constraint: Fundamental properties and applications to picture processing. Inf. Sci., 7:95-132, 1974.

[Nilsson, 1986] N. J. Nilsson. Probabilistic logic. Artificial Intelligence, 28:71 87, 1986.

[Pearl, 1988] J. P earl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman, 1988.

[Provan and Ball, 1983] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. SIAM Journal of Computing, 12(4):777-788, November 1983.

[Selman and Kautz, 1991] B. Selman and H. Kautz. Knowledge compilation using Horn approximations. In National Conference on Artificial Intelligence, 1991.

[Sinclair, 1988] A. Sinclair. Randomized Algorithms for Counting and Generating Combinatorial Structures. PhD thesis, Department of Computer Science, University of Edinburgh, 1988.

[Stockmeyer, 1985] L. Stockmeyer. On approximation algorithms for #P. SIAM Journal of Computing, 14:849 861, 1985.

[Toda, 1989] S. Toda. On the computational power of PP and Op. In IEEE Symp. of Foundation of Computer Science, number 30, pages 514519, 1989.

[Valiant, 1979a) L. G. Valiant. The complexity of computing the permanent. Theoretical Computer Science, 8:189 201, 1979.

[Valiant, 1979b] L. G. Valiant. The complexity of enumeration and reliability problems. SI AM Journal of Computing, 8:410 421, 1979.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. Communications of the ACM, 27(11):1134-1142, November 1984.