Approximation Techniques for Space-Efficient Compilation in Abductive Inference

Gregory Provan
Department of Computer Science, University College Cork, Cork, Ireland
Tel: +353 21 4901816, Fax: +353 21 4274390, e-mail: g.provan@cs.ucc.ie

Abstract

We address the problem of approximately compiling propositional abduction problems (PAPs). We show intractability of compiling a PAP into a fixed-size representation, and of compiling a PAP to within a factor $\epsilon > 0$ of the compilation of minimal size. Although generating an approximate compilation is intractable in general, we describe a preference-based PAP for which order-of-magnitude smaller compilations can be generated. We show that by restricting the distribution of solutions of a boolean function f to be power-law or exponential, we can compile a representation that approximates the solution coverage within a factor $\epsilon > 0$ yet requires orders-of-magnitude less space than that of complete compilations. We present empirical results for the compilation languages of DNNF and prime implicants.

1 Introduction

A broad range of compilation approaches have been proposed in the literature, such as prime implicates (de Kleer 1986), DNNF (Darwiche 2002), Ordered Binary Decision Diagrams (OBDDs) (Bryant 1992), cluster-trees (Pargamin 2003), and finite-state automata (Amilhastre, Fargier, & Marquis 2002). The benefit of these approaches is that they enable linear-time inference using the compiled representation; the drawback is that the size of the compiled representation can be exponentially larger than the original function. For example, the number of prime implicants of a set m of arbitrary clauses is $O(3^m)$ (Chandra & Markowsky 1978), and the size-complexity of compiled representations (e.g., DNNF, OBDD) for problems in propositional logic, Bayesian networks and constraint satisfaction is exponential in the treewidth of their interaction graph (Darwiche 2001; Dechter 2003; Jensen, Lauritzen, & Olesen 1990).

We focus on the *size* of a compilation, whereas most previous studies of compilation have focused on a variety of other questions, such as the existence of solutions, or of relevance and necessity of hypotheses (Eiter & Gottlob 1995). For propositional abduction problems, we show some results on the complexity of generating a compilation of fixed size, and of approximating a compilation. In particular, we show that the following problems are intractable.

- Generating a compilation *of fixed size* for a boolean function f.
- Generating from f an approximate compilation whose size is bounded by a fixed factor of the size of f.

To reduce the size of the compiled representation of a propositional abduction problem (PAP) $\mathcal{P}$, $\zeta(\mathcal{P})$, we propose to approximate $\zeta(\mathcal{P})$ using a valuation function to identify the most likely models (or satisfying assignments) of $\mathcal{P}$. We propose a model-based approach which generates approximate (sound but incomplete) representations that trade off, in a pre-specified manner, solution coverage for space and inference efficiency.

By introducing particular distributions over the solutions of a PAP (represented as a boolean function f), we can provide guarantees that we can compute compilations which are orders of magnitude smaller than complete compilations. We choose particular distributions that occur in important abduction problems. For example, exponential and power-law distributions occur within model-based diagnosis and optimal control instances of abduction problems. The intuition is that a small subset of the models contribute the vast majority of the total probability mass of the full set of models. In model-based diagnosis, it is the set Λ of most-likely diagnoses that play this role, with the vast majority of diagnoses being significantly less likely than the diagnoses in Λ. Hence the diagnosis distribution has a heavy tail of highly-unlikely diagnoses, and we take advantage of this heavy-tailed distribution. For these distributions over the solutions of a boolean function f, we can approximate *the size of a compilation* within a factor $\epsilon > 0$ in polynomial time.

2 Related Work

This section reviews prior work in related areas.

Roth (1996) has addressed the problems of abductive inference, and of approximating such inference. Roth focuses on counting the number of satisfying assignments for a range of AI problems, including some instances of PAPs. In addition, Roth shows that approximating the number of satisfying assignments for these problems is intractable.

Selman and Kautz (1996) adopt an alternative approximation approach, introducing upper and lower bounds to a compilation, in order to simplify the compilation process. This approach, though efficient, may in fact also generate a compiled representation that is exponential in the size of the original function f; in contrast, we address a different problem, that of compiling with size restrictions.

The complexity of compilation is typically intractable in the worst case; average-case results may be encouraging, given particular problem encodings. The complexity of PAPs has been thoroughly studied in (Eiter & Gottlob 1995), who showed that, for simple definitions of propositional abduction, the corresponding decision problems are complete for complexity classes at the second level of the polynomial hierarchy; moreover, Eiter and Gottlob showed that introducing prioritization increased the problem complexity. This work has been extended in (Hermann & Pichler 2007), where the complexity of counting solutions and computing minimal solutions is addressed.

The complexity of compilation has been addressed in many papers, including (Cadoli *et al.* 2002a; 1996; Liberatore 2001). Cadoli et al. (2002a) introduce a hierarchy of compilation complexity classes. Ferrara et al. (2007) have proven that for temporal logic model checking, preprocessing cannot reduce complexity, given polynomial space bounds (in the size of the input) on the output.

In the case of compiling using BDDs, the asymptotic worst-case complexity results indicate that many important BDD operations are NP-complete, NP-hard or tend to require unaffordable amounts of memory (Hachtel & Somenzi 2000). The size of a BDD is determined both by the function being represented and the variable ordering. There are functions that have only exponentially sized BDDs (Bryant 1991), although this is not true for the average boolean function. Using variable ordering, one cannot guarantee that a BDD will not be of size exponential in the number of variables, since the problem of finding the best variable ordering is NP-hard (Bollig & Wegener 1996). Further, the approximation problem is also hard, as for any constant $c > 1$ it is NP-hard to compute a variable ordering resulting in an OBDD with a size that is at most c times larger than optimal (Sieling 2002).

In the case of other compilation targets, like DNNF, the size of the DNNF generated for problems in propositional logic, Bayesian networks and constraint satisfaction is exponential in the *treewidth of their interaction graph* (Darwiche 2001; Dechter 2003; Jensen, Lauritzen, & Olesen 1990).[1] Recent work (Zabiyaka & Darwiche 2006) proposes functional treewidth as a better complexity bound than treewidth.

It is important to note that we focus on the *size* of the compilation, which few previous papers have addressed. We show the intractability of compiling a PAP within a fixed size, and how a preference-based PAP allows space-bounded approximate solutions for particular distributions.

3 Notation and Preliminaries

We adopt a simplified version of the formal framework introduced in (Cadoli *et al.* 2002b). In particular, we adopt the generic notion of compilation as the transformation of a language S into another language, S', which is the compiled representation of S. Using this framework, we describe compilation using the "language of pairs" (Cadoli *et al.* 2002b): we assume that we are given as input the pair $\langle f, \sigma \rangle$, where f is the fixed part, and σ the varying part of the problem. This pair $\langle f, \sigma \rangle$ captures problem-solving in a variety of abductive applications. For example, in diagnosis f consists of the model of the system to be diagnosed, and σ corresponds to the sensor data from the system captured in f; the task is to compute if σ corresponds to faulty or normal behaviour of the system.

3.1 Languages

We assume that we are dealing with languages that are generated from a finite, nontrivial alphabet Σ. We define the length of a string $f \in \Sigma^*$ by $|f|$, and the cardinality of a set S as $\parallel S \parallel$. We assume that we are given as input the pair $\langle f, \sigma \rangle$, i.e., the problems we are interested in can be formally defined as sets of pairs of strings:

Definition 1 (Language of pairs)**.** *A language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ is such that, for any pair $\langle f, \sigma \rangle$, f is the fixed part, and σ the varying part of the problem.*

3.2 Propositional Logic

We assume a standard propositional logic in this paper. We use a finite alphabet of propositional symbols, $Z = \{z_1, z_2, ..., z_n\}$, using the usual boolean connectives $\wedge, \vee, \neg$, and $\Rightarrow$ for constructing well-formed formulae from Z. A literal is a propositional symbol or its negation. A clause $z_1 \vee z_2 \vee \cdots \vee z_k \vee \neg z_{k+1} \vee \cdots \vee \neg z_n$ is a disjunction of literals. A clause is Horn (definite Horn) if $k \leq 1$ ($k = 1$). A function (or formula) f is a conjunction of clauses; in this article we assume that a formula is defined over n symbols, unless stated otherwise. The size of a formula f is $|f|$, and $|f^*|$ is the size of the minimal formula equivalent to f. f_1^* is the minimal function equivalent to 1 (the tautology).

An interpretation γ for Z is a mapping from Z to {true, false}, where interpretations can be extended to boolean formulae in the usual recursive way. Γ is the set of all interpretations. A *model* of a formula f is an interpretation M that maps f to true (written $M \models \mathsf{true}$).

3.3 Propositional Abduction Problem

We adapt the notation of (Liberatore & Schaerf 2007) to describe propositional abduction. A propositional abduction problem (PAP) can be defined using a triple $\langle H, \mu, f \rangle$, where H and μ are sets of variables, while f is a propositional formula. H is typically referred to as the hypotheses, and μ as the manifestations. The set of solutions, given manifestations μ, is defined as follows: $\Lambda(f, \mu) = \{H' \subseteq H | H' \cup f \text{ is consistent and } H' \cup f \models \mu\}$.

By introducing a preference function $\preceq$ we can define a preference-based PAP using $\langle H, \mu, f; \preceq \rangle$. One can then use an ordering $\preceq$ over the subsets of H in order to compute the set of preferred ("minimal") solutions of the problem, given manifestations μ: $\Lambda_{\preceq}(f) = \min(\Lambda(f), \preceq)$.

As noted in (Liberatore & Schaerf 2007), the ordering $\preceq$ captures the intuitive notion of plausibility of a solution to $\langle H, \mu, f; \preceq \rangle$. In other words, $H' \preceq H''$ holds if H' is more likely to be the "real" cause of the manifestations than H''.

[1] It turns out that many real-world problems, e.g., the ISCAS benchmark circuits (Brglez, Bryan, & Kozminski 1989), do not have treewidths bounded by some relatively small integer, in order to ensure compact DNNF compilations.

3.4 Compilability

We address abduction problems that can be expressed as the pair $\langle f, \sigma \rangle$, where f is the fixed part and σ the varying part. We assume that the fixed part, f, is compiled (preprocessed), and the varying part, σ, is only accessible on-line. A problem S is compilable if, given an instance $\langle f, \sigma \rangle$, the fixed part, f, can be preprocessed into an alternative representation, $\zeta(f)$.[2]

Definition 2 (Compilation)**.** *A language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ denotes a compilation if and only if there exists a function ζ and a language of pairs S' such that for all $\langle f, \sigma \rangle \in \Sigma^* \times \Sigma^*$, we have $\langle \zeta(f), \sigma \rangle \in S'$ if and only if $\langle f, \sigma \rangle \in S$.*

The function ζ represents the compilation of the fixed part. It is clear from the definition that a compilation S' of S preserves the models of S. In order to decide whether $\langle f, \sigma \rangle \in S$, we process the fixed part f off-line, thus obtaining $\zeta(f)$, and then we decide whether $\langle \zeta(f), \sigma \rangle \in S'$. Compilation is worthwhile if deciding $\langle \zeta(f), \sigma \rangle \in S'$ is easier than deciding $\langle f, \sigma \rangle \in S$. Note that we do not restrict the time needed to compute the function ζ.

This definition of compilation captures all of the compilation approaches that have been proposed in the literature, such as prime implicates (de Kleer 1986), DNNF (Darwiche 2002), Ordered Binary Decision Diagrams (OBDDs) (Bryant 1992), cluster-trees (Pargamin 2003), and finite-state automata (Amilhastre, Fargier, & Marquis 2002). These approaches all make space/time tradeoffs, i.e., they typically generate compilations, from which a variety of classes of inference can be done in time linear in the size of the compiled representation, although the compilation may require significant space. However, these typical compilation approaches are space intensive.

3.5 ϵ-Approximate Compilation

We define ϵ-equivalency of boolean functions as follows:

Definition 3. *$f(\gamma)$ is ϵ-equivalent to $g(\gamma)$ if $Pr_{\gamma \in \Gamma}\{g(\gamma) \neq f(\gamma)\} \leq \epsilon$, i.e., the fraction of n-bit strings γ for which $g_n(\gamma) \neq f_n(\gamma)$ is less than ϵ.*

Using these notions, we explore an alternative compilation approach, one in which we can generate all but a fraction ϵ of the solutions. In some cases that can lead to a space-efficient compilation. We represent this as follows:

Definition 4 (ϵ-Approximate Compilation)**.** *Given a parameter $0 < \epsilon < 1$, a language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ denotes an ϵ-approximate compilation if and only if there exists a function ζ and a language of pairs S' such that $S(\gamma)$ is ϵ-equivalent to $S'(\gamma)$, i.e., for all $\langle f, \sigma \rangle \in \Sigma^* \times \Sigma^*$ there exists at most a fraction ϵ of encodings $\zeta(f)$ that do not satisfy: $\langle \zeta(f), \sigma \rangle \in S'$ if and only if $\langle f, \sigma \rangle \in S$.*

4 Complexity Results

4.1 Review of Complexity Theory

We define a complexity class in terms of a set of languages. The two best-known complexity classes are P and NP: P is the set of languages possessing algorithms that run in time that is a polynomial in the length of the input; NP is the set of languages possessing algorithms that run in nondeterministic polynomial time.

Since we will be considering problems that are harder than those in P and NP, we define the polynomial hierarchy as follows:

Definition 5. *A language $\mathcal{L}$ is in the class Σ_i^P iff there is another language $\mathcal{L}'$ in the class P and an integer k for which*

$$\mathcal{L} = \{x : (\exists y_1)(\forall y_2)(\exists y_3) \cdots (Q y_i), |y_i| = |x|^k \text{ for all } i, [(x, y_1, y_2, \cdots, y_i) \in \mathcal{L}']\}, \quad (1)$$

where the sequence of quantifiers alternates, ending with $Q = \exists$ if i is odd or $Q = \forall$ if i is even.

According to this definition, $P = \Sigma_0^P$ and $P = \Sigma_1^P$. We can also define the complementary hierarchy Π_i^P of problems, which denote the problems defined by co$\mathcal{L} = \{L : \bar{L} \in \mathcal{L}\}$, or in other words, $co\Sigma_i^P = \Pi_i^P$, for $i = 0, ..., \infty$.

4.2 Complexity of General Problem

In this section we will show the complexity of the general problem of compilation with a size limitation. We define this problem as follows.

In our analysis, it suffices to consider just the fixed part f of a PAP. This is because the size of the compiled PAP depends only on the compilation of the fixed part. As a consequence, we restrict our attention to f, which we consider in this article to be a Boolean function. We assume that all manifestations μ will be conjunctions of literals.

Problem 1 (EQUIV-COMPILATION)**.** *Given a boolean function f and an integer k, does there exist a compiled representation $\zeta(f)$ of length at most k?*

It is straightforward to show that EQUIV-COMPILATION is in Σ_2^P, which implies that generating a fixed-length compilation is intractable. We can define a reduction to EQUIV-COMPILATION from the well-known problem EQUIVALENT FORMULAS (Umans).

Problem 2 (EQUIVALENT FORMULAS)**.** *The language* EQUIVALENT FORMULAS *consists of those pairs (k, f) where k is an integer and f is a Boolean function for which there exists an equivalent Boolean expression f' of length at most k.*

Further, we conjecture that EQUIV-COMPILATION is Σ_2^P-complete, since many people believe that EQUIVALENT FORMULAS is Σ_2^P-complete, although no proof has been demonstrated as yet.

This result is more general than the Σ_2^P-hardness of deciding if there exists an explanation for a set of manifestations (Eiter & Gottlob 1995), since we are now showing equivalence of *languages*, from which a much broader range of decisions problems can be defined. For example, our notion of compiled representation $\zeta(f)$ can be used to find an explanation, the set of all explanations, etc.

Given that EQUIV-COMPILATION is in Σ_2^P, it is unlikely that there exists a poly-time compilation algorithm to

[2] Note that this definition differs from the original definition of (Cadoli *et al.* 2002b), where a compilation is assumed to be of size polynomial in $|f|$.

generate a fixed-size $\zeta(f)$ from f. We can further extend this result to include the definition of compilability in terms of poly-size functions (Liberatore & Schaerf 2007); i.e., we would have $k = |\zeta(f)|$, where ζ is a poly-size function.

4.3 Space-Bounded Compilation Results

This section states two results concerning the existence of poly-time compilation algorithms that achieve particular size bounds. In particular, we state that there exists no poly-time algorithm, unless $P = NP$, that can compile from f a Boolean function $f' = \zeta(f)$ such that f' (a) is within some function-dependent factor of the size of f, or that (b) approximates f (in terms of ϵ-equivalency) within some function-dependent factor of the size of f.

We first state a theorem about exact compilation.

Theorem 1. *Given a Boolean function f and an integer $\chi(f)$, there is no polynomial time algorithm, if $P \neq NP$, which compiles from f an equivalent representation f' such that $|f'| = \chi(f) \cdot |f|$, where $\chi(f) = \left(\frac{|f^*|}{(|f_1^*|+1)}\right)$.*

Our next theorem concerns an inapproximability result, defined in terms of ϵ-equivalent compilations. Recall from Definition 3 that $f(\gamma)$ is ϵ-equivalent to $g(\gamma)$ if $Pr_{\gamma \in \Gamma}\{g(\gamma) \neq f(\gamma)\} \leq \epsilon$.

Theorem 2. *Given a Boolean function f and an integer $\chi(f)$, there is no polynomial time algorithm, if $P \neq NP$, which compiles from f an ϵ-equivalent representation f' such that $|f'| = \chi(f) \cdot |f|$, where $\chi(f) = \left(\frac{|f^*|}{(|f_1^*|+1)}\right)$, $\epsilon = O(2^{-n^c})$, and $0 < c < 1$ is a constant.*

Together, these two theorems state that the task of generating a bounded-size compilation or ϵ-approximate compilation, such as a poly-size compilation (Liberatore & Schaerf 2007), is intractable.

5 Preference-Based Compilation

This section introduces the notion of preference-based compilation, and of particular distributions for preference-based compilation.

5.1 Preference-Based Ranking

Assume that we have a full compilation $\zeta(f)$ such that we can specify the space of solutions, Λ. We can rank-order the solutions of Λ according to a given preference relation ϕ into a set of equivalence classes, in which each equivalence class is characterised by the same ϕ value.

We now consider the case where we use a preference criterion ϕ to guide the choice of compilation target, i.e., we aim to compile the *most-preferred* solutions.

Definition 6 (Preference Function)**.** *A preference function ϕ defined over over the solutions Λ of f defines a partial ordering $\succ$ over Λ. We say that solution $\lambda_1 \in \Lambda$ is preferred to solution $\lambda_2 \in \Lambda$, written $\lambda_1 \succ \lambda_2$, if $\phi(\lambda_1) > \phi(\lambda_2)$.*

In this article, we assign a preference function (valuation) to variables, and then use this valuation to compute most preferred solutions.[3]

We now specify three important preference-criteria that are widely used in the literature for PAP compilation:

Subset-inclusion $\boldsymbol{\phi_{\subseteq}}$: $2^H \rightarrow \lambda$; $\lambda_1 \succ \lambda_2$ if $\phi_{\subseteq}(\lambda_1) \subseteq \phi_{\subseteq}(\lambda_2)$.

Cardinality $\boldsymbol{\vartheta}$: $2^H \rightarrow |\lambda|$; $\lambda_1 \succ \lambda_2$ if $\vartheta(\lambda_1) < \vartheta(\lambda_2)$.

Probability $\boldsymbol{Pr}$: $2^H \rightarrow [0,1]$; $\lambda_1 \succ \lambda_2$ if $Pr(\lambda_1) > Pr(\lambda_2)$.

Preference-based compilation occurs often in abductive inference; for example, in diagnosis, we denote cardinality in terms of the number of faulty components in a solution, and we are interested in computing the min-cardinality solutions. In configuration, we specify a preference relation over the attributes, such as colour, size, shape, etc., and we want the most-preferred configurations. Consider the function f defined over variables $\{z_1, z_2, ..., z_n\}$ We represent a valuation of a variable z_i using $\phi(z_i)$. Hence, we have a weighted-pair (z_i, ϕ_i) for each constraint and valuation.

The most preferred solution in a compilation can be computed based on the specific preference relation, using the composition function of the preference relation. We give an example of a probabilistic preference relation below.

Example 1. Consider the case where we define our PAP as a diagnosis problem for discrete circuits, using the model-based diagnosis (MBD) framework of (Reiter 1987). Recall that a PAP is defined using a triple $\langle H, \mu, f \rangle$, where hypotheses H and manifestations μ are sets of variables, and f is a propositional formula. Each circuit consists of a set of gates (representing OR-, NOT-, AND-, or NAND-functions); we defined the variables denoting the health of gate i, h_i, $i = 1, ..., m$, as the hypotheses. The manifestations μ consist of conjunctions of the input and output circuit values, and the formula f denotes the functionality of the circuit, in terms of H, μ and a set of additional variables denoting internal inputs and outputs. Note that we defined only the normal functionality of each gate.

In addition, we defined a preference function over H. This corresponds to the probability that a gate is operating normally or not. We defined the probability that gate i is functioning, $Pr(h_i = 1) = p$, by randomly sampling the value for p from a uniform distribution over some range R. The failure probability, $Pr(h_i = 0)$, is just the complement, $1 - p$. The solutions, or diagnoses, given a manifestation μ, thus have an assigned probability, since each diagnosis consists of a conjunction of mutually-independent hypotheses $H' \subseteq H$, where $\lambda = \bigwedge_i \{h_i | h_i \in H'\}$, and $Pr(\lambda) = \prod_{h_i \in H'} Pr(h_i)$.

5.2 Preferred Compilations

Definition 7 (Preference Preservation)**.** *A compilation $\zeta(f)$ preserves a preference relation ϕ if, given $\langle f, \sigma \rangle$, for any pair of solutions λ_1, λ_2 such that $\lambda_1, \lambda_2 \in \Lambda_f$ and $\lambda_1, \lambda_2 \in \Lambda_{\zeta(f)}$, $\lambda_1 \succ \lambda_2$ is valid in $\zeta(f)$ iff $\lambda_1 \succ \lambda_2$ is valid in f.*

[3] We could define a more general form of valuation, where we assign valuations to clauses. In the clause-based viewpoint, a variable can be viewed as a unary clause.

We use a notion of *preferential* approximate equivalency given a weighted boolean function. Given a threshold ϱ, we compile all solutions with weight greater than ϱ.

Definition 8. *$f(\gamma)$is (ϕ, ϵ)-equivalent to $g(\gamma)$ if, given a valuation threshold ϕ, $\phi_{\gamma\in\Gamma}\{g(\gamma) \neq f(\gamma)\} \leq \epsilon$, i.e., ϵ is the cumulative valuation of solutions where $g(\gamma) \neq f(\gamma)$.*

In this case, we are interested in the *weighted* fraction of n-bit strings γ for which $g_n(\gamma) \neq f_n(\gamma)$ is less than ϵ.

We can now use these notions to define ϵ-approximate compilations based on our preference relations.

Definition 9 (Preferred ϵ-Approximate Compilation)**.** *Given a parameter $0 < \epsilon < 1$ and a preference function ϕ over f, a language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ denotes an ϵ-approximate compilation if and only if there exists a function ζ and a language of pairs S' such that for all $\langle f, \sigma\rangle \in \Sigma^* \times \Sigma^*$,*

- *there exists at most a fraction ϵ of encodings $\zeta(f)$ that do not satisfy: $\langle\zeta(f), \sigma\rangle \in S'$if and only if $\langle f, \sigma\rangle \in S$;*
- *there is no solution $M \in \Lambda_f \setminus \Lambda_{\zeta(f)}$ such that $M \succ M'$, where $M' \in \Lambda_{\zeta(f)}$.*

6 Approximate Compilations using Model Distributions

So far, we have seen the intractability associated with approximating a compilation of an abduction problem. This section shows that, if we have particular classes of preference function, we can have *stochastic* guarantees on the size of such a compilation. In particular, we show that, if we can define particular classes of Zipf- or Pareto-distributions over the solutions of f, then we can obtain a memory-bounded, ϵ-approximate compilation for the most-preferred solutions. It is important to note that the degree of memory savings depends on the parameters of the Zipf/Pareto distribution, as we will show.

The advantage of this approach is that, given certain problem characteristics, we can predict value of ϵ given μ. The model-based diagnosis (MBD) problem is an example of an important abduction problem that satisfies these distribution requirements. In the following, we describe the power-law and exponential distributions on which we focus. We then show how such distributions lead to memory-bounded, ϵ-approximate compilations that capture the majority of the most-preferred solutions.

6.1 Power-Law Distributions

Power-law distributions have been observed in many natural systems, such as physical (biological or man-made) and sociological systems (Newman 2003; 2005).

Definition 10 (Power Law)**.** *A probability density function (PDF) for a random variable X follows a power-law if $Pr(X > x) \sim x^{-\beta}$, as $x \to \infty$, $0 < \beta < 2$.*

Two of the most important power-law distributions are the Zipf and Pareto distribution.

Definition 11 (Zipf distribution)**.** *A discrete random variable Z follows a Zipf distribution if the r^{th} largest value has probability density function (PDF) $p(r; \beta) = Kr^{-\beta}$, for $\beta, r > 0$, and constant K.*

The simplest continuous power-law distribution is the Pareto distribution, which has PDF and cumulative distribution function (CDF), respectively:

Definition 12 (Pareto distribution)**.**

$$\begin{aligned} p(x; \alpha, k) &= \alpha k^{\alpha} x^{-\alpha-1}, \textit{ for } \alpha, k > 0,\ x \geq k. \\ F(x; \alpha, \eta) &= Pr[X \leq x] = 1 - \left(\frac{\eta}{x}\right)^{\alpha}, \end{aligned}$$

where η represents the smallest value the random variable X can take.

The closely related exponential distribution has also been used for modeling many natural systems (Newman 2005).

6.2 Distributions for Preference-Based Compilations

Every preference-based compilation has an associated distribution function. For example, consider the cardinality preference criterion. If the number of solutions of cardinality k is given by $|\vartheta_k(\Lambda)|$, and the total number of solutions is given by $|\Lambda|$, then we can define a probability density function (PDF) using this preference criterion as follows:

$$f(\vartheta = k) = \frac{\parallel \vartheta_k(\Lambda) \parallel}{\parallel \Lambda \parallel}.$$

In this case, $f(\vartheta)$ defines the distribution over solution cardinality.

Further, we can define the cumulative density function (CDF) for this preference criterion as follows:

$$F(\vartheta \leq k) = \sum_{i \leq k} f(\vartheta_i) = \sum_{i \leq k} \frac{\parallel \vartheta_i(\Lambda) \parallel}{\parallel \Lambda \parallel}.$$

In an analogous manner, we can define the PDF and CDF for the subset-inclusion preference criterion as follows:

$$\begin{aligned} f(\phi_{\subseteq} = k) &= \frac{\parallel \vartheta_k(\Lambda) \parallel}{\parallel \Lambda \parallel}, k = 1, ..., n. \\ F(\phi_{\subseteq} \leq k) &= \sum_{i \leq k} \phi_{\subseteq}(\vartheta_i) = \sum_{i \leq k} \frac{\parallel |\vartheta_i(\Lambda) \parallel}{\parallel \Lambda \parallel}. \end{aligned}$$

Example 2. *Figure 1 shows a PDF and CDF of a typical diagnosis problem, such as that described in Example 1. Using a probabilistic preference function with $Pr(z = OK) = 0.9$, the PDF and CDF are clearly Pareto-distributed. In the CDF the x-axis denotes the fraction of total memory required by a compilation containing diagnoses up to cardinality k, i.e., containing up to k broken components.*

In the following, we show how restricting these distributions can provide stochastic guarantees over the size of ϵ-approximate compilations covering the most-preferred solutions.

6.3 Restrictions on Model Distributions

This section shows how we can compile a representation that contains the approximate solutions of f using the distribution of the solutions of f.

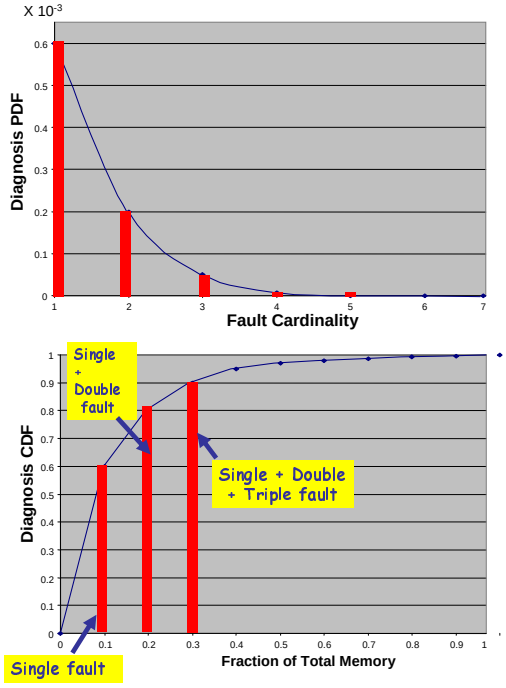

Figure 1: PDF and CDF of diagnosis distribution.

Consider the case where we adopt a cardinality preference function over solutions, $\vartheta(M)$, which is exponentially-distributed; in this case, we know that the PDF and CDF are given, respectively, by $f(x) = \xi e^{\xi x}$, $F(x) = 1 - e^{\xi x}$, for $x \in \Lambda, x = 1, 2, 3, .., n,\ \xi > 0$.

Assume that we want to find the subset of solutions such that the weighted sum of the solutions is $(1 - \epsilon)$ of the total solution weight. This can be found using $F(x) = 1 - e^{\xi x} = (1 - \epsilon)$, which results in $x = -\frac{1}{\xi} ln\epsilon$. This means that given the parameters (ξ, ϵ), we can estimate the maximum cardinality κ of the solutions that need to be generated; hence we can ignore solutions with cardinality $\vartheta(\lambda) > \kappa$, since these solutions do not need to be generated given that we have already generated sufficient solutions to achieve our ϵ-bounds.

6.4 Stochastic Guarantees for Compact Compilations

We can use $f(\boldsymbol{\xi})$ to specify a distribution over the space of solutions of f. Depending on the assumptions that we make for the f_i, we obtain different distributions for $f(\xi)$.

Lemma 1. *Given a preference-based Horn PAP $\langle H, \mu, f; \preceq \rangle$ for which the solutions $\Lambda(f)$ satisfy a power-law (or exponential) distribution under $\preceq$, a preferred ϵ-approximate compilation has size polynomially bounded in $|f|$.*

Example 3. Assume that we have a PAP function f with $n = 100$ variables and $|H|$=30 hypotheses, together with an exponential distribution ϕ over the solutions $\Lambda(f)$.

If $\epsilon \leq 0.01$, then $x \geq \frac{-1}{\lambda} ln 0.01 = \frac{4.6}{\lambda}$. For $n = 100$ and $\lambda \simeq 1$ (as is typical in real-world systems (Newman 2003)), then $x \geq 4.6$. From this, we know that we only need to generate solutions with at most 5 hypotheses; this subset of solutions is clearly polynomial in n. □

Given that this approximate stochastic compilation $\zeta'(f)$ has polynomially-bounded space, it is simple to show that $\zeta'(f)$ takes only a fraction of the memory of the full compilation $\zeta(f)$. The memory ratio that we are interested in computing is $\Omega = |\zeta'(f)|/|\zeta(f)|$.

Example 4. Consider an approximate stochastic compilation $\zeta'(f)$ in which we have to generate solutions with at most 10 hypotheses. In this case, we have $\Omega = \frac{\zeta'}{\zeta} = \frac{\sum_{i=1}^{10} \binom{n}{i}}{2^n}$. For n large, $|\zeta'(f)| \ll |\zeta(f)|$. □

The interesting thing to point out is that the "value" of the approximate compilation method, measured in terms of memory savings required to obtain $1 - \epsilon$ of the probability mass of solutions, increases as n increases. Hence, the larger the system under analysis, the more valuable is this approximate compilation method.

Finally, we can show that we can generate an ϵ-approximate compilation for Horn formulae in polynomial time, using a two-step process. This compilation is equivalent to a set of most-likely solutions. In step one, we enumerate the ranked hypothesis-sets by computing the weight of each assignment to H. Since we have shown that, given some threshold ϕ guaranteeing an ϵ-approximation we need only a polynomial number of subsets of H, this step can be done in polynomial time. In step two, we extend each partial instantiation (as defined by the assignment to H) to a full instantiation, or solution, using a SAT oracle. Given a Horn propositional function f, this can be done in linear time. Hence our overall algorithm is polynomial-time.

7 Empirical Analysis

We have implemented this approximation approach, and now show how it can be applied to generating compilations for two target representations, DNNF and prime implicants.[4]

All our experiments were run with a set of formulae (expressed in CNF format) representing a suite of digital circuits. The digital circuits were generated randomly by a circuit generator program (Provan & Wang 2007), such that the circuits have properties similar to those of the ISCAS circuit benchmarks (Brglez, Bryan, & Kozminski 1989).

In the following we report results for the probabilistic preference criterion. The other two criteria (subset and cardinality) show similar results, and we omit them for reasons of space. Experiments were conducted on a 3MHz Pentium4 with 2GB of RAM.

All prime implicant experiments were conducted using the algorithm by Shiny and Pujari (2002), using a *trie* data structure. All DNNF experiments were conducted using the algorithm by Darwiche (2001). All algorithms generate the compiled representations *incrementally*. For example, for prime implicants, we generate prime implicants up to those that exceed the preference threshold ϕ. This guarantees that our compilation can be used to generate solutions whose valuation is below ϕ. In addition, this incremental approach

[4]These results are a summary of the results from (Venturini & Provan 2007).

Table 1: Properties of function f for circuits

Gates	# Clauses	# Variables	# Literals in f
7	40	14	136
8	46	16	156
9	52	18	176
10	58	20	196
11	64	22	216
12	74	24	260
13	78	26	268
14	84	28	288
15	90	30	308
16	98	32	348
17	104	34	360
18	108	36	368
19	118	38	412

ensures that we never have to generate the full compilation (which may be of exponential size), but generate only the desired subset of the compilation.

We designed our experiments to test the compilation of preference-based Horn PAPs on diagnosis models as described in Example 1. We used circuits of a range of sizes, as described in Table 1. For a circuit C with k gates (components), this table shows the properties of the boolean function defining C.

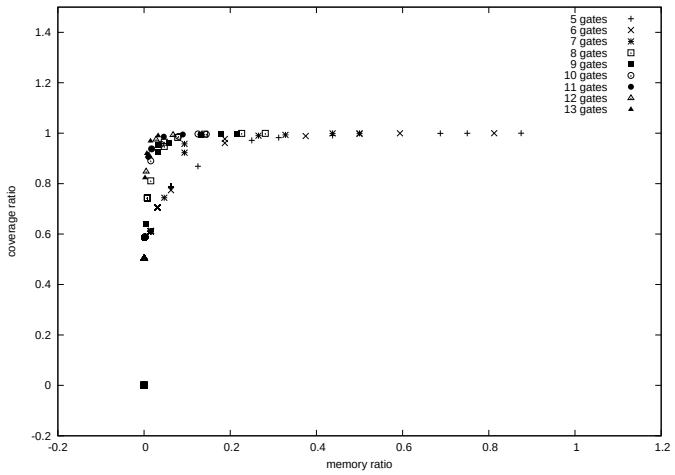

Figure 2: CDF of diagnosis distribution for compilation using DNNF.

The other key parameter in our experiments is the solution threshold ϕ, which denotes the minimal probability of a diagnosis that is of interest. By compiling only diagnoses with weights above ϕ, we generate a compilation that is sound but incomplete with respect to ϕ.

Figure 2 shows the CDFs of diagnosis distributions for compilation using DNNF, in the case where the range $R = [0.9, 1]$, for circuits with 5 to 13 gates. Figure 3 shows the CDFs of diagnosis distributions for compilation using prime implicants, in the case where the range $R = [0.99, 1]$, for circuits with 10 to 18 gates. Here we see that, as the circuits grow in size, the approximate compilation achieves greater memory savings for very high coverage ratio ($> 99\%$). For example, the 18-gate circuit achieves 4 orders of magnitude of memory savings with $> 99\%$ coverage. Our experimental evidence to date indicates that the memory savings increase with larger functions f, meaning that this approach will enable approximate inference with systems larger than can be handled by any existing exact compilation approach, with little loss of query coverage.

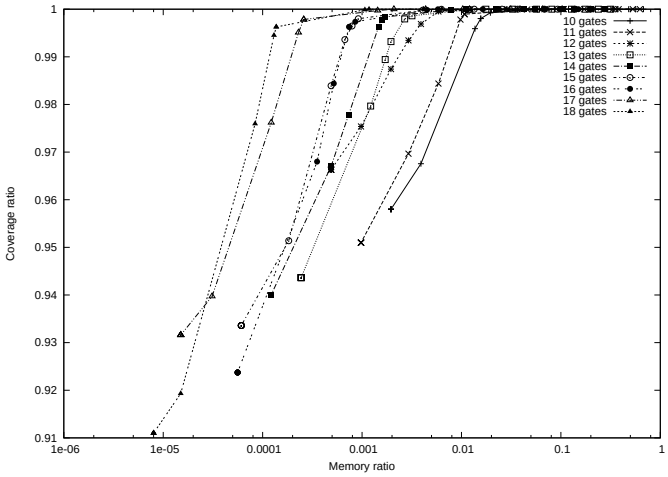

Figure 3: CDF of diagnosis distributions for compilation using prime implicants. This figure focuses on the upper left quadrant of the distributions to better identify the differences in the distributions.

8 Conclusions

We have examined the problem of compiling propositional abduction problems (PAPs), and in particular compiling representations that are not exponential in the PAP parameters. We have shown that no poly-time algorithms exist that can compile a PAP to produce a representation of fixed size, or to approximate the PAP within a factor $\epsilon > 0$ of the full compilation. Given this intractability, we examined restricted problems that occur in real-world PAPs. For example, in diagnosis, the diagnoses can be ranked according to a power-law or exponential distribution, given ranking functions like $\leq, \subseteq$, or Pr. Using this motivation, we showed that by restricting the distribution of solutions of a boolean Horn function f to be power-law or exponential, we can compile a representation that approximates the solution coverage within a factor $\epsilon > 0$ yet requires orders-of-magnitude less space than that of complete compilations. We empirically demonstrated this space efficiency for compilation targets DNNF and prime implicants.

9 Appendix: Proofs

This appendix presents our proofs and proof sketches.

9.1 Complexity of General Problem

This section shows that EQUIV-COMPILATION is in Σ_2^P. We first show a mapping from EQUIVALENT FORMULAS to EQUIV-COMPILATION, and then show that EQUIVALENT FORMULAS is in Σ_2^P.

As defined in Section 3.3, compilation generates a function $f' = \zeta(f)$ such that $f' \models \mu$ iff $f \models \mu$. Hence it is trivial to demonstrate a mapping from EQUIVALENT FORMULAS to EQUIV-COMPILATION.

EQUIVALENT FORMULAS is in Σ_2^P because we can define a language $\mathcal{L} \in P$ as follows: $\mathcal{L}$ accepts those tuples $((f,k),f',\gamma)$ for which f and f' are Boolean formulas, f' has length of at most k, and f agrees with f' on interpretation γ. We then see that $(f,k) \in$ EQUIVALENT FORMULAS $\Leftrightarrow (\exists f')(\forall\gamma)[((f,k),f',\gamma) \in \mathcal{L}]$, as is required by Equation 1 for $i = 2$.

9.2 Space-Bounded Compilation Complexity

Proof of Theorem 1

To prove this theorem, we need to identify a growth function $\chi(f)$, and show that no poly-time algorithm can compile a representation satisfying this growth function. We use Theorem 1 of (Prokopyev & Pardalos 2004)), as sketched below, to establish both requirements.

Theorem:*If $P \neq NP$ there is no polynomial-time algorithm, which for a given Boolean function f builds a Boolean function f' equivalent to f with size less than $|f| \cdot \frac{|f^*|}{(|f_1^*|+1)}$.*

Proof: Assume that there exists a polynomial-time algorithm, which for a given Boolean formula f builds a Boolean formula f' equivalent to f with the size less than $|f| \cdot \frac{|f^*|}{(|f_1^*|+1)}$. Here, $|f_1^*|$ is the size of minimal function equivalent to **1** and $|f^*|$ is the size of the minimal function equivalent to f.

We will prove this by contradiction, by using formula f to build another formula $g = f \vee (y_1 \wedge y_2 ... \wedge y_l)$, where l is some natural number, which we will define later, and $y_1, ..., y_l$ are new Boolean variables, which do not appear in formula f. We can apply our algorithm (which exists due to our assumption) to the formula g. It is easy to notice that if f is a tautology, then as the output of the algorithm we get the formula with size at most $\frac{(|f|+l)\cdot|f_1^*|}{(|f_1^*|+1)}$. If f is not a tautology then all variables $y_1, ..., y_l$ must appear in the result formula. So the size of the result formula is not less than l. Hence we can distinguish the two cases, whether f is a tautology or not, if

$$(|f|+l)/(|f_1^*|+1)|f_1^*| < l \quad (2)$$

$$|f||f_1^*| + l|f_1^*| < l|f_1^*| + l \quad (3)$$

$$|f||f_1^*| < l. \quad (4)$$

Therefore, according to equation 4, we can choose l equal to $|f|\cdot|f_1^*|+1$ to find out if f is a non-tautology or not (Problem 3 below). Notice that our transformation is polynomial (the size of g is polynomial in the size of f). As a result, we can use our algorithm to solve, in polynomial time, the Non-tautology problem. However, this problem is known to be NP-complete (Garey & Johnson 1978), so our assumption that the algorithm exists is false. □

Problem 3 (NON-TAUTOLOGY)**.** *Given a Boolean expression f over a set Z of variables, using the connectives $\neg, \vee, \wedge, \Rightarrow$, is f not a tautology, i.e., is there a truth assignment for Z that makes f false?*

Proof Sketch of Theorem 2

The proof for this theorem follows a similar structure to that of theorem 1, so we provide just a sketch here. We again need to identify a growth function $\chi(f)$, and show that no poly-time algorithm can compile a representation satisfying this growth function. However, in this case we admit compiled functions f' which are ϵ-equivalent to f.

We use [Theorem 3 (Prokopyev & Pardalos 2004)], as sketched below, to establish both requirements.

Theorem:*Given $\epsilon = O(2^{-n^c})$, where $0 < c < 1$ is a constant, there is no polynomial-time algorithm, which for a given Boolean function f builds a Boolean function f' ϵ-equivalent to f with the size less than $\frac{|f|}{(|f_1^*|+1)\cdot|f^*|}$, if $P \neq NP$.*

Proof: As in the previous proof, we assume that there exists a polynomial-time algorithm A, but which admits ϵ-equivalency. Hence, for a given Boolean formula f, A builds a Boolean formula f' ϵ-equivalent to f with size less than $|f^*| \cdot \frac{|f|}{(|f_1^*|+1)}$, where $\epsilon = O(2^{-n^c})$ and c is some constant such that $0 < c < 1$. We prove this result using this algorithm to solve the ϵ-TAUTOLOGY problem in polynomial time.

Problem 4 (ϵ-TAUTOLOGY)**.** *Given a Boolean expression f, such that f is equivalent to 1, or at least a fraction ϵ of n-bit interpretations γ are such that $f(\gamma)$ is equal to 0, is f not a tautology, i.e., is there a truth assignment for Z that makes f false?*

As before, we use proof by contradiction, using formula f to build another formula $g = f \vee (y_1 \oplus y_2 ... \oplus y_l)$, where l is some natural number, which we will define later, $y_1, ..., y_l$ are new Boolean variables, which do not appear in f.

By showing that the algorithm solves the ϵ-TAUTOLOGY in polynomial time, a contradiction arises, since ϵ-TAUTOLOGY is NP-complete. Hence our assumption that such an algorithm exists is false. □

9.3 Preference-Based Compilation

Lemma 2. *Given a preference-based Horn PAP $\langle H, \mu, f; \preceq \rangle$ for which the solutions $\Lambda(f)$ satisfy a power-law (or exponential) distribution under $\preceq$, a preferred ϵ-approximate compilation has size polynomially bounded in $|f|$.*

Proof Sketch: We sketch this proof for the Pareto and exponential distributions.

Pareto distribution: Assume a Pareto distribution ϕ over the solutions Λ. We want to identify the parameters of the CDF F such that $F \geq 1 - \epsilon$, i.e., the cumulative probability mass is at least $1 - \epsilon$ of the total probability mass.

The Pareto distribution has cumulative distribution function $F(x;\alpha,\eta) = Pr[X \leq x] = 1 - \left(\frac{\eta}{x}\right)^{\alpha}$, where η represents the smallest value the random variable X can take.

If we set $\delta = \frac{\eta}{x}$, then we must have $1 - \delta^{\alpha} = 1 - \epsilon$, from which we obtain $\delta = \epsilon^{\frac{1}{\alpha}}$. For ϵ small and α taking on real-world values ($\simeq 2$), it is straightforward to show that under ordering $\leq, \subseteq, Pr$ only a number of solutions polynomial in n is needed.

Exponential distribution: Assume that we have an exponential distribution ϕ over the solutions Λ, i.e. $F(x) = 1 - e^{-\lambda x}$. We can compute the required fraction of ϵ-approximate solutions for approximate compilation θ' as follows: If we set $F(x) = 1 - e^{-\lambda x}$ equal to $1 - \epsilon$, then we obtain $x = \frac{-1}{\lambda} ln\epsilon$. In a similar fashion to the Pareto distribution, it is straightforward to show that under ordering $\leq, \subseteq, Pr$ only a number of solutions polynomial in n is needed. □

Acknowledgments

This work has been supported by SFI grant 04/IN3/I524.

References

Amilhastre, J.; Fargier, H.; and Marquis, P. 2002. Consistency restoration and explanations in dynamic CSPs Application to configuration. *Artif. Intell.* 135(1-2):199–234.

Bollig, B., and Wegener, I. 1996. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers* 45(9):993–1002.

Brglez, F.; Bryan, D.; and Kozminski, K. 1989. Combinational profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems*, volume 3, 1929–1934.

Bryant, R. E. 1991. On the complexity of vlsi implementations and graph representations of boolean functions with application to integer multiplication. *IEEE Trans. Comput.* 40(2):205–213.

Bryant, R. 1992. Symbolic boolean manipulation with ordered binary-decision diagrams. volume 24. 293–318.

Cadoli, M.; Donini, F.; Liberatore, P.; and Schaerf, M. 1996. Feasibility and unfeasibility of off-line processing. *Proceedings of the Fourth Israeli Symposium on Theory of Computing and Systems (ISTCS96)* 100–109.

Cadoli, M.; Donini, F.; Liberatore, P.; and Schaerf, M. 2002a. Preprocessing of Intractable Problems. *Information and Computation* 176(2):89–120.

Cadoli, M.; Donini, F. M.; Liberatore, P.; and Schaerf, M. 2002b. Preprocessing of intractable problems. *Inf. Comput.* 176(2):89–120.

Chandra, A., and Markowsky, G. 1978. On the number of prime implicants. *Discrete Mathematics* 24:7–11.

Darwiche, A. 2001. Decomposable negation normal form. *Journal of the ACM (JACM)* 48(4):608–647.

Darwiche, A. 2002. A compiler for deterministic, decomposable negation normal form. In *AAAI/IAAI*, 627–634.

de Kleer, J. 1986. An assumption-based TMS. *Artif. Intell.* 28(2):127–162.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Eiter, T., and Gottlob, G. 1995. The complexity of logic-based abduction. *Journal of the ACM (JACM)* 42(1):3–42.

Ferrara, A.; Liberatore, P.; and Schaerf, M. 2007. Model Checking and Preprocessing. In *Proc. AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*. Springer.

Garey, M., and Johnson, D. S. 1978. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Hachtel, G. D., and Somenzi, F. 2000. *Logic Synthesis and Verification Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers.

Hermann, M., and Pichler, R. 2007. Counting complexity of propositional abduction. In *IJCAI*, 417–422.

Jensen, F.; Lauritzen, S.; and Olesen, K. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly* 4:269–282.

Liberatore, P., and Schaerf, M. 2007. Compilability of propositional abduction. *ACM Transactions on Computational Logic (TOCL)* 8(1).

Liberatore, P. 2001. Monotonic reductions, representative equivalence, and compilation of intractable problems. *Journal of the ACM (JACM)* 48(6):1091–1125.

Newman, M. 2003. The structure and function of complex networks. *SIAM Review* 45(2):167– 256.

Newman, M. E. J. 2005. Power laws, pareto distributions and zipf's law. *Contemporary Physics* 46:323.

Pargamin, B. 2003. Extending cluster tree compilation with non-boolean variables in product configuration. In *Proceedings of the IJCAI-03 Workshop on Configuration*.

Prokopyev, O., and Pardalos, P. 2004. On Approximability of Boolean Formula Minimization. *Journal of Combinatorial Optimization* 8(2):129–135.

Provan, G., and Wang, J. 2007. Evaluating the adequacy of automated benchmark model generators for model-based diagnostic inference. In *Proceedings of IJCAI-07*.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–95.

Roth, D. 1996. On the hardness of approximate reasoning. *Artificial Intelligence* 82(1-2):273–302.

Selman, B., and Kautz, H. A. 1996. Knowledge compilation and theory approximation. *J. ACM* 43(2):193–224.

Shiny, A., and Pujari, A. 2002. Computation of Prime Implicants using Matrix and Paths. *Journal of Logic and Computation* 8(2):135–145.

Sieling, D. 2002. The nonapproximability of OBDD minimization. *Inf. Comput.* 172(2):103–138.

Umans, C. Hardness of approximating Σ_2^p minimization problems. *Symp. Foundations of Computer Science*.

Venturini, A., and Provan, G. 2007. Empirical analysis of approximate compilation algorithms. *submitted*.

Zabiyaka, Y., and Darwiche, A. 2006. Functional treewidth: Bounding complexity in the presence of functional dependencies. *SAT* 116–129.