Horn Upper Bounds and Renaming*

Marina Langlois mirodo1@uic.edu
Robert H. Sloan sloan@uic.edu
György Turán† gyt@uic.edu
University of Illinois at Chicago
Chicago, IL 60607
USA

Abstract

Selman and Kautz [22] proposed an approach to compute a Horn CNF approximation of CNF formulas. We extend this approach and propose a new algorithm for the Horn least upper bound that involves renaming variables. Although we provide negative results for the quality of approximation, experimental results for random CNF demonstrate that the proposed algorithm improves both computational efficiency and approximation quality. We observe an interesting behavior in the Horn approximation sizes and errors which we call the "Horn bump": Maxima occur in an intermediate range of densities below the satisfiability threshold.

Keywords: *Horn approximation, renaming, random satisfying assignment*

Submitted August 2008; revised July 2009; published September 2009

1. Introduction

The reasoning problem in propositional logic can be considered to be the following question: Is a clause C entailed by a CNF expression φ? Typically φ is a fixed *knowledge base* and the number of *queries* C to be answered may be large. This reasoning problem is known to be hard; thus it is worthwhile to transform the original knowledge base φ into a new knowledge base φ', such that answering the question "Is a clause C entailed by φ'?" is tractable, but φ' is only approximately equivalent to φ. Selman and Kautz proposed such an approach in an important article [22] (see also [5, 7, 24]) and called the approach *knowledge compilation*.

The idea of Selman and Kautz's algorithm is to approximate the original knowledge base φ from above and below by *Horn formulas*, and then use these formulas to answer the queries efficiently. Specifically, they proposed an algorithm (outlined in Section 4) for computing a *Horn least upper bound (Horn-LUB)* of φ that is equivalent to the conjunction of all its Horn prime implicates.[1.]

There is an interesting combinatorial characterization of the set of truth assignments satisfying the Horn-LUB of φ. Define the *intersection* of two truth assignments to be their

* This material is based upon work supported by the National Science Foundation under grant CCF-0431059. A preliminary version of portions of this work appeared in [16].

† Also affiliated with Hungarian Academy of Sciences and University of Szeged, Research Group on Artificial Intelligence, Szeged, Hungary.

1. We omit the definition of Horn *greatest lower bounds*, as those will not be discussed in this paper.

componentwise conjunction.[2.] The set of truth assignments satisfying the Horn-LUB of φ is the *closure under intersection of the set of satisfying truth assignments of* φ.

Entailment queries to Horn formulas can be answered in polynomial time, but the knowledge compilation approach can still have several drawbacks: The size of the Horn approximation itself may be large, making the approach inefficient; not all queries may be answered correctly by the Horn upper bound (those implied by the Horn lower bound, but not implied by the upper bound). Selman and Kautz [22] and del Val [9] proved such negative results on the worst-case performance of the approach.

Another important question is the performance of the algorithms on random examples. Initial results in this direction were obtained by Kautz and Selman [14]. Boufkhad [4] extended the approach for Horn lower bounds by using renameable Horn formulas. The primary test cases were random 3-CNF formulas with density around 4.2, which is known to be hard for satisfiability algorithms.[3.] In other related work, van Maaren and van Norden [25] considered the connection between the efficiency of satisfiability algorithms and the size of a largest renameable sub-CNF for random 3-CNF.

In this paper we consider several new versions of the Horn-LUB algorithm and present both theoretical and experimental results on their performance. In contrast to the original approach, the proposed versions of the algorithm use a preprocessing step that involves renaming some variables in the initial 3-CNF (i.e., switching some variables and their complements). The idea behind the renaming is to create as many Horn clauses as possible in the original 3-CNF. By doing so, we hope to obtain a (possibly smaller) Horn-LUB faster and with better approximation quality. In general, the problem of finding the largest renameable Horn sub-CNF of a given CNF is NP-hard. Therefore, we use an approximation algorithm proposed by Boros [3] to find a large renameable subformula of the initial 3-CNF. Another variation of the Horn-LUB algorithm we consider is the use of resolvents of limited size only during the calculation. This is also expected to speed up the algorithm and decrease the size of the Horn upper bound, but possibly reduce the approximation quality. We explore the trade-offs and find an optimal size for the intermediate resolvents.

The combinatorial characterization of the Horn-LUB previously discussed carries over to the case of renaming: if an arbitrary vector is chosen as the 'bottom' of the hypercube instead of the standard all 0's vector then there is a new orientation of the hypercube. A renaming corresponds to such a reorientation. To obtain the Horn-LUB after the renaming of the original formula, intersections have to be taken with respect to this new chosen vector (orientation).

The theoretical results for the renaming variation of the algorithm show worst-case behavior similar to that of the original Selman–Kautz algorithm. We show that there are 3-CNF formulas with only a polynomial number of satisfying truth assignments such that for every renaming, the obtained Horn-LUB has superpolynomially many satisfying truth assignments. Another result shows that there is a polynomial size CNF expression such that for every renaming, only a superpolynomially small fraction of the prime implicates

2. E.g., $(1,1,1,0,0) \cap (1,1,0,0,1)$ is $(1,1,0,0,0)$. The intersection of two truth assignments is also the greatest lower bound of these assignments in the componentwise partial ordering of the hypercube.

3. Kautz and Selman also considered a class of planning problems, and Boufkhad also considered 4-CNF formulas.

are Horn clauses, and thus most of the prime implicate queries are answered incorrectly by the renamed Horn-LUB.

The second half of the paper is devoted to experimental results and to comparisons to previous work. We consider three new algorithms for calculating various forms of approximations to the Horn-LUB and find that they give improvements in both efficiency and approximation quality over Selman and Kautz's original idea. The best approach turns out to be the one that combines both modifications: renaming and bounded size resolvents. To compare these algorithms we use the performance over all truth assignments as an evaluation measure. Due to running-time constraints, we have chosen to use formulas with 20 variables so we can exhaustively enumerate all vectors. In order to consider formulas with a large number of variables, it would be necessary to efficiently generate a sample of random satisfying truth assignments of a Horn formula. We show that this is not possible unless RP = NP. In the other direction, we show that Google's relsat[4.] can be used to generate random satisfying assignments for Horn formulas with at least up to 60 variables.

Unlike previous work, which concentrated simulation effort around density 4.2, we have considered 3-CNF formulas of *different densities*, including densities below the famous threshold. For lower densities we notice an interesting, and apparently new, phenomenon—the *Horn bump*: the performance of each algorithm is the worst in an intermediate range of densities, far below the critical density 4.2.

2. Preliminaries

A clause is a disjunction of literals; a clause is Horn (resp., definite, negative) if it contains at most one (resp., exactly one, no) unnegated literal. A CNF is a conjunction of clauses; it is a 3-CNF if each clause contains exactly 3 literals. A clause C is an *implicate* of a CNF expression φ if every *truth assignment* or *vector* in $\{0,1\}^n$ satisfying φ also satisfies C; it is a *prime* implicate if none of its sub-clauses is an implicate. An n-variable *random 3-CNF* formula of *density* α is obtained by selecting $\alpha \cdot n$ clauses of size 3, selecting each clause from the uniform distribution over all such clauses. A *Horn formula* or *Horn-CNF* is a conjunction of Horn clauses.

The set of satisfying truth assignments of a formula φ is denoted by $T(\varphi)$. The weight of a 0-1 vector is the number of its 1 components. The *intersection* of vectors $(x_1, \ldots, x_n)$, $(y_1, \ldots, y_n) \in \{0,1\}^n$ is $(x_1 \wedge y_1, \ldots, x_n \wedge y_n)$. A Boolean function can be described by a Horn formula if and only if its set of satisfying truth assignments is closed under intersection [12, 19]. The *(ordinary) Horn closure* $\mathcal{H}(S)$ of any set S of truth assignments is the smallest intersection-closed set of truth assignments containing S.

A Horn least upper bound of φ (Horn-LUB(φ)) is any conjunction of Horn clauses logically equivalent to the conjunction of all Horn prime implicates of φ. The set of satisfying truth assignments of Horn-LUB(φ) is $\mathcal{H}(T(\varphi))$, the Horn closure of $T(\varphi)$. Selman and Kautz [22] give an algorithm for computing a Horn-LUB(φ).

Renaming a variable x in a CNF is the operation of simultaneously *switching* every occurrence of x to $\bar{x}$ and of $\bar{x}$ to x. A *renaming (function) with respect to vector* $d \in \{0,1\}^n$, denoted by $\mathcal{R}_d$, maps a CNF formula φ to $\mathcal{R}_d(\varphi)$, obtained by switching every pair of literals x_i and $\bar{x}_i$ such that $d_i = 1$. The following easily verified proposition shows that

4. relsat (version 2.02) is Google's exact solution counter for propositional satisfiability instances.

the operation on truth assignments corresponding to renaming w.r.t. vector d is taking the exclusive or with d, and one can use a renamed version of a CNF φ to solve the reasoning problem formulated in the introduction.

Proposition 1. a) *A truth assignment a satisfies CNF φ iff the truth assignment $a \oplus d$ satisfies $\mathcal{R}_d(\varphi)$.*

b) *For any CNF φ, clause C, and vector d, we have $\varphi \models C$ if and only if $\mathcal{R}_d(\varphi) \models \mathcal{R}_d(C)$.*

Proof: *a)* The proof is based on the following fact: for any clause C of φ and some literal x_i from C, if a_i makes x_i positive (negative), then after renaming, $a_i \oplus d_i$ makes variable $\mathcal{R}_{d_i}(x_i)$ positive (negative).

b) Can be proved using a definition of an implicate and Part *a*) of this Proposition. □

A CNF φ is *Horn renameable* if $\mathcal{R}_d(\varphi)$ is a Horn formula for some vector d. It can be decided in polynomial time if a CNF is Horn renameable [1, 18], but finding a largest Horn renameable sub-CNF of a given CNF is NP-hard [6]. Boros [3] gave an approximation algorithm for finding a large Horn renameable sub-CNF in an arbitrary CNF in linear time. Given a direction $d \in \{0,1\}^n$, the *d-Horn closure of a set S of truth assignments* is

$$\mathcal{H}_d(S) = \mathcal{H}(\{a \oplus d : a \in S\}).$$

With an abuse of notation, we refer to $\mathcal{H}_d(T(\varphi)) = \mathcal{H}(T(\mathcal{R}_d(\varphi)))$ as the *d-Horn closure* of φ.

3. Bounds on Horn closure sizes with renaming

In this section we present negative results for Horn closures and Horn-LUB with renaming, analogous to those for the ordinary Horn closure and Horn-LUB.

Theorem 2. *There are 3-CNF formulas φ with a polynomial number of satisfying truth assignments such that for every direction d, the size of the d-Horn closure of φ is super-polynomial.*

Proof: The construction uses the following lemma.

Lemma 3. *There is a set $S \subseteq \{0,1\}^m$ with $|S| = 2m$ such that for every direction $d \in \{0,1\}^m$ it holds that*

$$|\mathcal{H}_d(S)| \geq 2^{\lceil m/2 \rceil}.$$

Proof: Let S be the set of vectors of weight 1 and $(m-1)$ and let $d \in \{0,1\}^m$ be any direction. Then d has at least $\lceil m/2 \rceil$ 0's or $\lceil m/2 \rceil$ 1's. Assume w.l.o.g. that the first $\lceil m/2 \rceil$ components of d are 0 (resp., 1). Consider those vectors from S which have a single 0 (resp. 1) in one of the first $\lceil m/2 \rceil$ components. All possible intersections of these vectors (resp., the complements of these vectors) are contained in the d-Horn closure of S. Thus all possible vectors on the first $\lceil m/2 \rceil$ components occur in the d-Horn closure and the bound of the lemma follows. □

Now to prove Theorem 2, consider the 4-CNF φ formed by taking the conjunction of all possible clauses of size 4 containing two unnegated and two negated literals over m variables. The vectors satisfying this formula are those in the set S in the proof of Lemma 3 plus the all 0's and the all 1's vectors. Hence by Lemma 3, φ's Horn closure with respect to any direction has size at least $2^{\lceil m/2 \rceil}$.

In order to obtain a 3-CNF ψ, introduce a new variable z for each clause $(a \vee b \vee c \vee d)$ in φ, and replace the clause by five new clauses: $(a \vee b \vee \bar{z}), (c \vee d \vee z), (\bar{a} \vee \bar{b} \vee z), (\bar{a} \vee b \vee z)$ and $(a \vee \bar{b} \vee z)$. It follows by a standard argument (omitted for brevity) that ψ has the same number of satisfying truth assignments as φ, and every truth assignment of φ has a unique extension to a satisfying truth assignment of ψ. Hence the Horn closure of ψ in any direction has size at least $2^{\lceil m/2 \rceil}$. Thus ψ has $n = \Theta(m^4)$ variables, $\Theta(m)$ satisfying truth assignments and its Horn closure in every direction has size at least $2^{\lceil m/2 \rceil}$, so the theorem follows. □

It may be of interest to note that the bound of Lemma 3 is fairly tight.

Theorem 4. *For every polynomial p and every $\epsilon > 0$, for all sufficiently large m, for every set S of at most $p(m)$ binary vectors of length m, there exists a direction d such that the size of the d-Horn closure of S is at most $2^{\frac{m}{2}(1+\epsilon)}$.*

Proof: We show that a randomly chosen direction $d \in \{0,1\}^m$ has nonzero probability of having the desired property. For every vector $a \in S$, the probability that $a \oplus d$ has more than $\frac{m}{2}(1 + \frac{\epsilon}{2})$ 1's is at most $e^{-\epsilon^2 m/8}$ using a Chernoff bound (see, e.g. [15, Additive Form, page 190]). If m is sufficiently large then $p(m)e^{-\epsilon^2 m/8} < 1$. In this case there is a direction d such that $a \oplus d$ has at most $\frac{m}{2}(1 + \frac{\epsilon}{2})$ 1's for every $a \in S$. Every vector in the d-Horn closure of S is below one of the vectors $a \oplus d$. Hence the size of the d-closure is at most $p(m)2^{\frac{m}{2}(1+\frac{\epsilon}{2})}$, which is less than $2^{\frac{m}{2}(1+\epsilon)}$ for all sufficiently large m. □

The following result shows the existence of CNF formulas for which the Horn-LUB in every direction d gives an incorrect answer to a large fraction of the prime implicate queries. The construction is based on a construction of Levin [17] of a DNF formula with a bounded number of terms, having the maximal number of prime implicants. In [23], we showed that all bounded term DNF with the maximal number of prime implicants can be obtained as a natural generalization of this example.

Theorem 5. *There are polynomial size CNF formulas φ such that for every direction d, the ratio of the number of non-Horn and Horn prime implicates of $\mathcal{R}_d(\varphi)$ is superpolynomial.*

Proof: To construct φ, we begin with a complete binary tree of the height k and put $n = 2^k$. The variables of φ are $x_1, \ldots, x_{n-1}$ and $y_1, \ldots, y_n$. Each internal node of the tree is labeled with a distinct x variable and the ith leaf is labeled with $\bar{y}_i$. The formula φ has n clauses, one for each leaf. The clause corresponding to a leaf is the disjunction of all the variables on the root-to-leaf path to the leaf, with each x variable being negated if and only if the path went left when leaving that node. Thus the depth-1 tree pictured in Figure 1 corresponds to $(\bar{x}_1 \vee \bar{y}_1) \wedge (x_1 \vee \bar{y}_2)$.

The formula φ has a distinct prime implicate for each of the $2^n - 1$ nonempty subsets of the leaves [17, 23]. The prime implicate corresponding to a particular subset S of leaves is

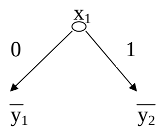

Figure 1. Tree of depth 1

the disjunction of x variables corresponding to any inner node such that *exactly one* of the two subtrees of the node contains a leaf in S, and the negated y variables corresponding to the leaves in S. An x variable in the prime implicate is negated iff its left subtree is the one containing leaves in S. Thus, for example, the formula corresponding to the tree in Figure 1 has three prime implicates, one for each nonempty subset of the two leaves. For $\{\bar{y}_1\}$ we have $(\bar{x}_1 \vee \bar{y}_1)$; for $\{\bar{y}_2\}$ we have $(x_1 \vee \bar{y}_2)$; for $\{\bar{y}_1, \bar{y}_2\}$ we have $(\bar{y}_1 \vee \bar{y}_2)$.

We give an upper bound for the number of Horn prime implicates under any renaming d. Notice that by symmetry, renaming internal nodes does not change the number of Horn or non-Horn prime implicates. At the leaves, making all the y's negated maximizes the number of the prime implicates that are Horn. Thus it is in fact sufficient to estimate the number of Horn prime implicates of the original formula.

Let H_k (resp., N_k, D_k) be the number of Horn (resp., negative, definite) prime implicates of the formula built from a binary tree of height k. Then $H_k = N_k + D_k$. The numbers H_k satisfy the following recurrence: $H_1 = 3$ and

$$H_k = H_{k-1} + N_{k-1} + H_{k-1} \cdot N_{k-1} + N_{k-1} \cdot D_{k-1} \ . \tag{1}$$

Here the first item is the number of Horn implicates of the left subtree. The second term is the number of negative Horn implicates of the right subtree: by adding the unnegated variable from the root, those correspond to definite Horn implicates. The third (resp, the fourth) term corresponds to prime implicates obtained from an arbitrary Horn (resp., a negative) prime implicate of the left subtree and a negative (resp., a definite) one form the right subtree. Note that two definite prime implicates from the two subtrees will form a non-Horn prime implicate. In order to use (1) to get an upper bound on H_k, we must bound N_k. Similarly to (1), one can derive the following recurrence: $N_1 = 2$ and

$$N_k = (N_{k-1})^2 + N_{k-1}.$$

It can be shown that $N_k < 2^{\frac{11}{16}n}$ and $H_k \leq 3^{k-4} \cdot 2^{14} \cdot 2^{\frac{11}{16}n}$. □

4. Computational results

The initial algorithm for Horn-LUB proposed by Selman and Kautz takes a set of clauses (i.e., a CNF) as input and derives new clauses by resolving two clauses where at least one of them must be non-Horn. If the new resolvent subsumes a clause already in the set, then that subsumed clause is removed and the resolvent is included in the set. When no new resolvent that is not subsumed by a clause already in the set can be found, the process terminates. The Horn least upper bound is the collection of all Horn clauses in the final set.

We took Selman and Kautz's original algorithm and considered three new modifications of it to compute different Horn approximations:

Renamed-Horn-LUB uses a preprocessing step to find a renaming of the variables of a given CNF using a heuristic algorithm of [3]. This extra step takes only linear time. Then the algorithm follows the Horn-LUB algorithm.

4-Horn-UB computes the Horn upper bound in the same way as the original algorithm, except that it keeps only resolvents of size at most 4 and discards the rest.

Renamed-4-Horn-UB is a combination of the first two modifications: preprocessing step with renaming and then keeping only resolvents of size at most 4.

We have found that the last modification, Renamed-4-Horn-UB, outperforms all of the proposed algorithms. In Table 1 we give its running time and output size compared to the original Horn-LUB approach.

Table 1. Mean running time in CPU seconds (on a Dell laptop with a 2.40 GHz CPU and 256 MB RAM) and number of clauses in the output for Horn-LUB and Renamed-4-Horn-UB on random 3-CNF formulas on 20 variables as a function of density α, averaged over 50 runs.

	Original LUB		Renamed-4	
α	Time	Size	Time	Size
1	1.0	96.1	0.0	28.2
2	50.5	1044.7	0.2	236.2
3	126.8	889.8	1.5	704.8
4	224.6	409.3	0.9	452.7

As we stated in the introduction, we limit ourselves to formulas on $n = 20$ variables while comparing our algorithms to the original approach. We do this for two main reasons: First, the time required to compute the Horn-LUB using Selman and Kautz's original algorithm increased by approximately a factor of 5–10 for every two variables added to the set. Even for $n = 25$, the time for computing the Horn-LUB was large enough to make it somewhat inconvenient to perform a few hundred such computations, and it would have been computationally impossible to work with formulas with, say, 75 variables.

(Selman and Kautz themselves reported experimental results on only the *unit clauses within the Horn-LUB*. That can be computed more efficiently by using a SAT solver, rather than the complete Horn-LUB itself.)

Second, in order to compare the algorithms we enumerated all 2^n Boolean vectors, and this becomes infeasible for values significantly larger than $n = 20$. (More discussion of this issue is given at the end of this section.)

It turned out that the running time of Renamed-4-Horn-UB is much smaller than that of Horn-LUB. The size of the Horn upper bounds produced by Renamed-4-Horn-UB is smaller for density $\alpha \leq 3$, and modestly larger for density 4. We observed that the output sizes for all algorithms have unimodal behavior as a function of the initial 3-CNF density. More detailed data show that the maximum ("Horn bump") occurs around density 2.5.

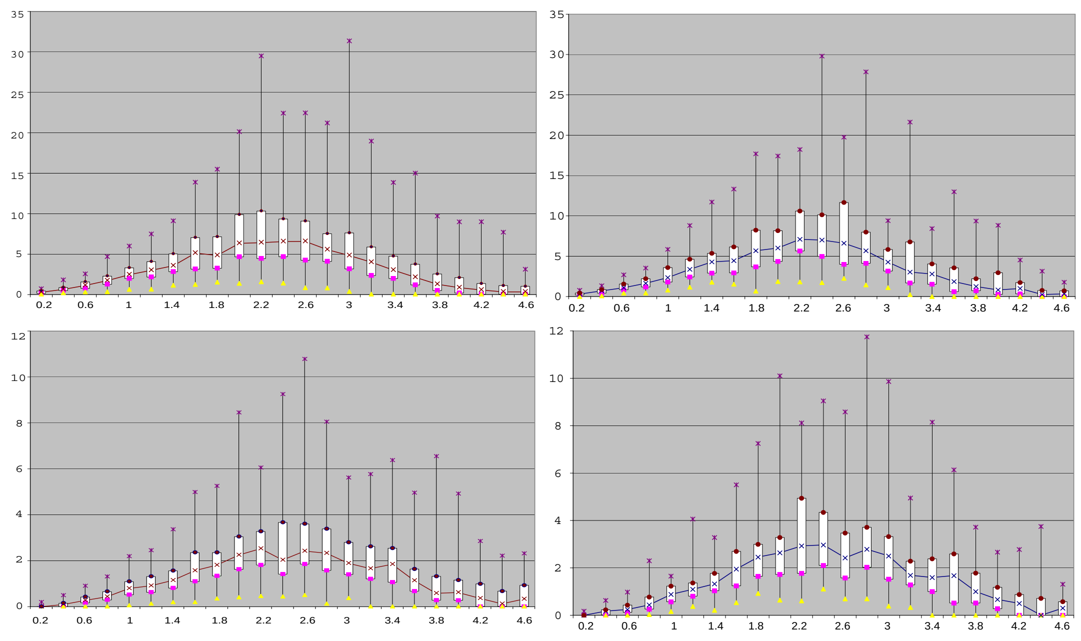

Figure 2. Relative errors $r_A(\varphi)$ of the algorithms for random 3-CNF with 20 variables as function of density. Measured by exhaustive examination of all length 20 vectors. Averaged over 100 runs. From top to bottom: Horn-LUB, 4-Horn-UB, Renamed-Horn-LUB, and Renamed-4-Horn-UB. *The scales for the relative error run from 0–35 for the first two algorithms, but from 0–12 for the Renamed variants.*

4.1 Accuracy of the approximations

For all four algorithms, the Horn upper bound is a conjunction of some clauses that are implicates of the original 3-CNF φ (i.e., their output is implied by φ); in other words, each algorithm's output has a one-sided error. The *relative error* of such an algorithm A on an input formula φ is measured by

$$r_A(\varphi) = \frac{|T(A(\varphi))| - |T(\varphi)|}{|T(\varphi)|} \ ,$$

where $A(\varphi)$ denotes the formula output by A on φ.

Figure 2 shows computational results for the relative errors of the four Horn upper bound algorithms for 3-CNF's with different densities on 20 variables. Statistical values on all the figures are median, max and min values; the values at the ends of the white bars are 25% and 75%. Note that error curves have a unimodal behavior again, with the maximum value around density 2.4. The same experiments for a smaller number of variables show similar values of the maxima.

As we discussed above, the two main modifications of the Horn-LUB algorithm are renaming and restricting resolvent size to four. Taken alone, the former improves the relative error more than the latter; notice that the relative errors were sufficiently different that Figure 2 uses two different scales. We conclude that all in all, Renamed-4-Horn-UB is the best algorithm among those proposed for 20 variables.[5.]

In terms of running time, Renamed-4-Horn-UB is considerably faster than both Horn-LUB and Renamed-Horn-LUB, and somewhat faster than 4-Horn-UB. In terms of output size, its output is considerably smaller than Horn-LUB or Renamed-Horn-LUB, but larger than that of 4-Horn-UB. The relative error for Renamed-4-Horn-UB is just slightly worse than that of Renamed-Horn-LUB, which has the smallest relative error of all. Modifying Renamed-4-Horn-UB either by decreasing the allowed resolvent size from 4 to 3, or by using all implicates of size at most 3 results in a large increase in the relative error for densities below the satisfiability threshold.

It is to be expected, and it is supported by some experimental evidence, that as the number of variables increases, the limit on the resolvent size required for producing reasonable relative error will also increase.

The second measure that can be used to evaluate the algorithms is to compare the number of queries that are answered *incorrectly* by their output. For the renaming case, by Proposition 1 we can just query the renamed clause. We will use the prime implicates of the initial formula φ as our test set of clauses.

By the definition of a Horn least upper bound, the original Horn-LUB algorithm gives the correct answer for any Horn clause query, and the wrong answer for any non-Horn prime implicate query. Therefore, renaming is expected to improve the query-answering accuracy of the LUB. On the other hand, for the case of restricted size resolvents in computing the upper bound, the query-answering accuracy will worsen because some Horn prime implicates may receive the wrong answer.

The performance of the four algorithms is shown in Figure 3; these are error ratios, and all are fairly high for densities significantly below the critical density of $\alpha \approx 4.2$. We again observe unimodal behavior, with the maximum around a density of 1.6, with some variation by algorithm. The best performance is indeed for Renamed-Horn-LUB, but Renamed-4-Horn-UB is only a bit worse than Renamed-Horn-LUB.

4.2 Random sampling of satisfying assignments

In order to perform our experiments, we needed to exhaustively check every truth assignment, which is not feasible for formulas with a larger number of variables. One way to estimate the relative error without generating all possible vectors is to use random sampling by repeatedly generating a random satisfying truth assignment of the Horn upper bounds. This raises the question whether it is possible to generate a random satisfying truth assignment of a Horn formula (almost) uniformly in polynomial time. The answer is "no":

Theorem 6. *Unless $RP = NP$, there is no fully polynomial almost uniform generator for Horn formulas.*

Proof: This follows directly from combining previous results [10, 13, 21]. Jerrum et al. [13] showed that a fully polynomial almost uniform generator exists if and only if there exists

5. Preliminary experiments show Renamed-4-Horn-UB also performing relatively well for up to at least 40 variables, but for 40 variables, simply *measuring* performance is computationally expensive.

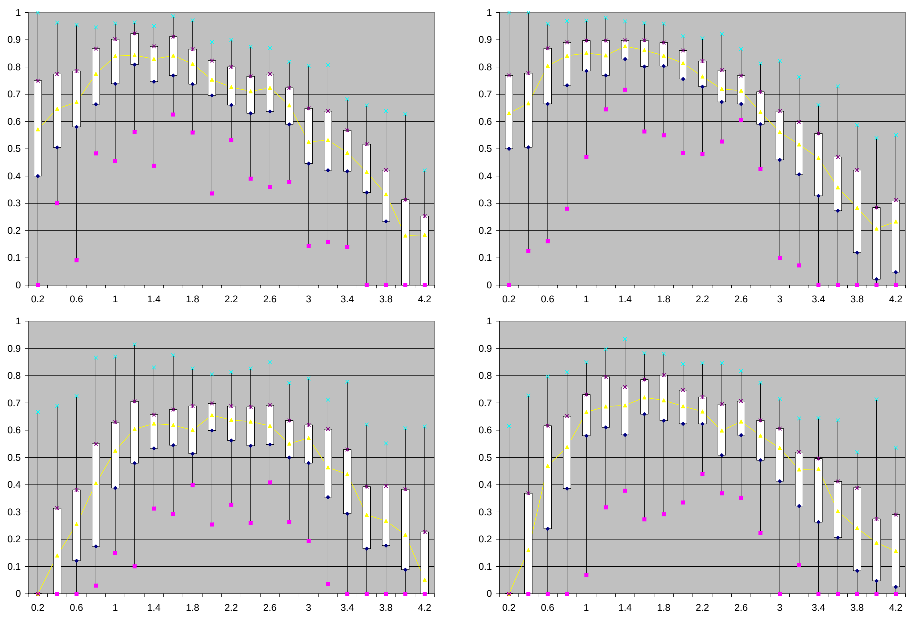

Figure 3. Fraction of all prime implicate queries to a random 3-CNF formula on 20 variables that would receive the wrong answer from the particular type of Horn upper bound, as a function of density. Averaged over 50 runs. From top to bottom: Horn-LUB, 4-Horn-UB, Renamed-Horn-LUB, Renamed-4-Horn-UB.

a fully polynomial randomized approximate counting scheme. Combining reductions given in [10, 21] we get that unless RP = NP, randomized polynomial-time approximate counting of assignments satisfying a Horn CNF is not possible, and hence fully polynomial almost uniform generation is not possible. □

We have started to do some initial experiments for randomly generating a satisfying truth assignment of a Horn formula using *relsat*. We used relsat to construct a uniformly random satisfying truth assignment by repeating the following procedure: Pick a variable x, set x to 1 with probability p and to 0 with probability $1-p$, where p is the fraction of satisfying truth assignments of the original formula where x is true.

This method runs very quickly on Horn formulas with only 20 variables. We tested how much time it takes to generate a random satisfying truth assignment for Horn formulas with a somewhat larger number of variables. To get large Horn formulas to sample, we used the Renamed-4-Horn-UB algorithm on 3-CNFs.

In Table 2, we report the running time in seconds for sampling different Horn formulas output by Renamed-4-Horn-UB, as a function of the number of variables and the density *of the original 3-CNF*. It can be observed that even for a rather small number of variables the time to create a single random satisfying assignment is large for intermediate densities. For density 4, the running time is better. This is unsurprising, because the size of the Horn approximation is smaller for density 4 as can be seen from Table 1. (Table 1 has values only for 20 variables but the general behavior is the same for larger numbers of variables.)

Table 2. Running time in seconds to find one random satisfying truth assignment using the relsat-based algorithm for the Horn sentence found by using the Renamed-4-Horn-UB algorithm, after Renamed-4-Horn-UB algorithm has been run on an 3-CNF initial formula of the indicated number of variables and density. Results are averaged over 10 runs.

	Density			
	1	2	3	4
20 variables	0.07	0.12	0.12	0.04
40 variables	2.4	16.3	16.6	1.3
60 variables	429.7	10270.2	9121.1	22.1

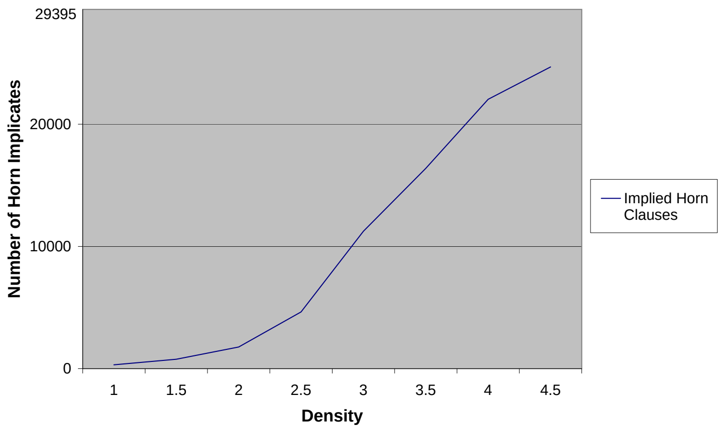

Figure 4. Number of Horn clauses up to size 4 (total 29395) over 20 variables that were implied by a random 3-CNF as a function of its density. Averaged over 20 runs.

4.3 Another measure of approximation

We have already introduced two tests that measure the quality of approximation: relative error and the fraction of all prime implicates queries answered incorrectly. From these tests one can see that Renamed-4-Horn-UB is the best algorithm for both efficiency and approximation quality.

Here we propose one more measure to evaluate the performance of Renamed-4-Horn-UB. The Horn-LUB is logically equivalent to the conjunction of all Horn prime implicates of the original formula, which allows the Horn-LUB to answer all Horn queries correctly.

It seems interesting to see how many Horn queries are answered incorrectly by Renamed-4-Horn-UB. We restrict attention to clauses of size up to 4 for simplicity. Figure 4 shows the number of Horn clauses of size up to 4 that are implied by a random 3-CNF as a function of its density for 20 variables. For a given 3-CNF formula we count the fraction of Horn implicates that are *not* implied by Renamed-4-Horn-UB on Figure 5. We again observe an unimodal behavior with a bump around density 2, where the worst-case disagreement of roughly 0.075% occurs. For this measure, Renamed-4-Horn-UB is close to Horn-LUB for every density.

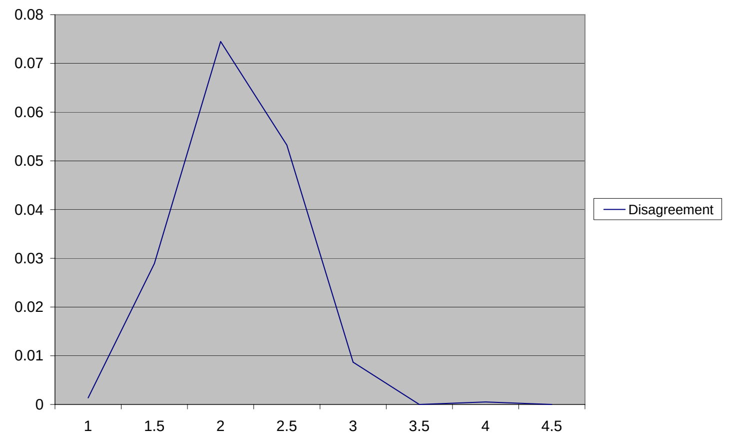

Figure 5. Percentage of all Horn clauses up to size 4 where original randomly chosen 3-CNF φ (over 20 variables) and Renamed-4-Horn-UB(φ) approximation disagree as a function of φ's density. Averaged over 20 runs.

4.4 SAT-solver versus Renamed-4-Horn-UB

Selman and Kautz's original idea of creating the LUB algorithm in the 1990s was to take advantage of approximation by formulas in Horn form. For their original and our modified Horn-LUB algorithms, all Horn queries can be answered almost immediately via forward chaining. The drawback is that in order to obtain this representation a time penalty for preprocessing must be paid.

Our last test is a comparison of the running times of Renamed-4-Horn-UB and a fast modern SAT solver, zChaff. This test was performed on a AMD Athlon MP 2000+ processor (1.6 GHz) running Linux. We consider several formulas over 75 and 200 variables with different densities. Table 3 below shows some statistics on the running times for 75 variables for both methods: the left column is for zChaff, and the right column is for Renamed-4-Horn algorithm. For zChaff we show how much time was needed to answer a Horn query, and for Renamed-4-Horn we report the running time to find a corresponding Renamed-4-Horn-UB. We generated 10^4 random Horn clauses up to size 4, and random 3-CNF formulas over 75 variables with different densities.

Thus according to these data, for 75 variables the average time required by zChaff to answer a query is not larger than the time required by Renamed-4-Horn-UB to answer a query. In addition, Renamed-4-Horn-UB has a non-negligible preprocessing time. Thus, Renamed-4-Horn-UB does not seem competitive in this range. (The comparison is, however, a little misleading, because we are comparing the highly optimized zChaff program to a simple non-optimized forward chaining program.)

The situation changes if we increase the number of variables to 200. Then, for density 4, zChaff answers queries in average time $\approx$ 0.3 CPU second and worst-case time $\approx$ 40 seconds. Renamed-4-Horn-UB still answers queries in $\leq 8 \cdot 10^{-3}$ seconds in the worst case. Thus Renamed-4-Horn-UB has a significantly better running time for answering a single query. Preprocessing time for Renamed-4-Horn-UB is quite large, $\approx$ 2 days on average and

Table 3. Different running times in CPU seconds for zChaff to answer a query and for calculation of the Renamed-4-Horn-UB on random 3-CNF formulas on 75 variables as a function of density α. Renamed-4-Horn-UB query answer time is $2 \cdot 10^{-3}$ sec on average.

	zChaff		Renamed-4-Horn	
α	Worst	Average	Worst	Average
1	0.02	$3 \cdot 10^{-4}$	2.39	0.05
2	0.02	$5 \cdot 10^{-4}$	0.32	0.213
3	0.021	$7 \cdot 10^{-4}$	24.83	3.62
4	0.03	$1 \cdot 10^{-3}$	$33 \cdot 10^{3}$	$24 \cdot 10^{3}$

$\approx$ 3 days in the worst case. Thus preprocessing would help if we need to answer 100,000 or more queries. Note that in a real application the algorithm would run on far more efficient computing platforms. This would maintain the comparison between the two approaches but it would significantly scale down running times. (In fact, things might be more favorable for the approximation approach, because the preprocessing might run on a more powerful computer than the query answering for either approach.) Thus we conclude that in certain ranges of the parameters Renamed-4-Horn-UB can be competitive with, or even faster than applying a SAT solver.[6.]

5. Concluding remarks

We have proposed several algorithms for generating Horn upper bounds. The paper contains both theoretical negative results on the approximation quality of Horn upper bounds using renaming and experimental results on the behavior of the algorithms. As the initial knowledge base we used random 3-CNF with 20 variables and different densities. Based on our experimental results, we have concluded that the best algorithm in terms of running time, relative error and output size is Renamed-4-Horn-UB. We observed an unusual behavior ("Horn bump") for the different performance measures in some intermediate range of densities. It would be interesting to obtain theoretical results on the phenomena observed.

There are several other future directions that can be pursued. First, we are interested in showing that there are CNF expressions with superpolynomially large Horn-LUB for every direction (this is possible for the ordinary Horn closure and an example is given in [22]).

As it was mentioned in the introduction, the Horn least upper bound can be considered as the closure under intersection of the set of satisfying truth assignments for a Horn formula. This leads to a general theoretical question: What is the expected size of the intersection closure of a random subset of $\{0, 1\}^n$, given a probability distribution on the subsets? We are unaware of any results on this problem. Another closure problem, the dimension of the subspace spanned by a random set of vectors, has been studied in great detail. Perhaps

6. SAT-solver-based methods always return the correct answer. Horn knowledge compilation, if both upper and lower bounds are used together, returns either the correct answer or "I don't know." Thus an option, already suggested in the original work of Selman and Kautz [22], is to combine the two approaches by using knowledge compilation, and if an "I don't know" is returned then using a SAT solver. As we have not implemented Horn lower bounds, we have no data on the efficiency of this approach.

a first model to study the intersection problem would be to generate m random vectors, where each bit of a vector is set to 1 with probability p. The number of satisfying truth assignments of the Horn-LUB of a random 3-CNF with a given density is another special case of the general problem.

There has been a great deal of study of how a random 3-CNF evolves as its density increases. This work is the spirit of Erdős and Rényi's classic work [11] on the evolution of random graphs, and of work on the evolution of random Boolean functions (see, e.g., Bollobás et al. [2]). For instance, Mora et al. [8] and Mézard et al. [20] show some interesting behavior at densities below the critical density (clustering of the solutions in the case of Mora et al., and a particular behavior of a class of local search algorithms in the case of Mézard et al.). We wonder whether the Horn bump has any connection to any of these random 3-CNF phenomena.

Acknowledgment We would like to thank Eli Ben-Sasson, Tom Hayes, Bart Selman, and the referees for useful comments.

References

[1] Bengt Aspvall. Recognizing disguised NR(1) instances of the satisfiability problem. *Journal of Algorithms*, **1**:97–103, 1980.

[2] Béla Bollobás, Yoshiharu Kohayakawa, and Tomasz Łuczak. On the evolution of random Boolean functions. In P. Frankl, Z. Füredi, G. Katona, and D. Miklós, editors, *Extremal Problems for Finite Sets* (Visegrád), **3** of *Bolyai Society Mathematical Studies*, pages 137–156, Budapest, 1994. János Bolyai Mathematical Society.

[3] Endre Boros. Maximum renamable Horn sub-CNFs. *Discrete Appl. Math.*, **96-97**:29–40, 1999.

[4] Yacine Boufkhad. Algorithms for propositional KB approximation. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 280–285, 1998.

[5] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Communications*, **10**(3–4):137–150, 1997.

[6] Yves Crama, Oya Ekin, and Peter L. Hammer. Variable and term removal from Boolean formulae. *Discrete Applied Mathematics*, **75**(3):217–230, 1997.

[7] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Research (JAIR)*, **17**:229–264, 2002.

[8] Hervé Daudé, Marc Mézard, Thierry Mora, and Riccardo Zecchina. Pairs of SAT-assignments in random Boolean formulae. *Theoret. Comput. Sci.*, **393**(1-3):260–279, 2008.

[9] Alvaro del Val. First order LUB approximations: characterization and algorithms. *Artificial Intelligence*, **162**(1-2):7–48, 2005.

[10] Martin Dyer, Alan Frieze, and Mark Jerrum. On counting independent sets in sparse graphs. *SIAM J. Comput.*, **31**(5):1527–1541, 2002.

[11] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, **5**:17–61, 1960.

[12] Alfred Horn. On sentences which are true on direct unions of algebras. *J. Symb. Logic*, **16**:14–21, 1951.

[13] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, **43**:169–188, 1986.

[14] Henry Kautz and Bart Selman. An empirical evaluation of knowledge compilation by theory approximation. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 155–161, 1994.

[15] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, Cambridge, Massachusetts, 1994.

[16] Marina Langlois, Robert H. Sloan, and György Turán. Horn upper bounds and renaming. In *Proc. SAT 2007: Tenth Int. Conf. Theory and Applications of Satisfiability Testing*, **4501** of *Lecture Notes in Computer Science*, pages 80–93, 2007.

[17] A. A. Levin. Comparative complexity of disjunctive normal forms. *Metody Discret. Analiz.*, **36**:23–38, 1981. In Russian.

[18] Harry R. Lewis. Renaming a set of clauses as a Horn set. *Journal of the ACM*, **25**:134–135, 1978.

[19] J. C. C. McKinsey. The decision problem for some classes without quantifiers. *J. Symb. Logic*, **8**:61–76, 1943.

[20] Marc Mézard and Riccardo Zecchina. The random K-satisfiability problem: from an analytic solution to an efficient algorithm. *Physical Review E*, **66**:056126, 2002.

[21] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, **82**:273–302, 1996.

[22] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, **43**:193–224, 1996.

[23] Robert H. Sloan, Balázs Szörényi, and György Turán. On k-term DNF with the maximal number of prime implicants. *SIAM J. Discret. Math.*, **21**(4):987–998, 2008.

[24] Klaus Truemper. *Effective Logic Computation.* Wiley-Interscience, 1998.

[25] Hans van Maaren and Linda van Norden. Hidden threshold phenomena for fixed-density SAT-formulae. In *Proc. Int. Conf. Theory and Applications of Satisfiability Testing (SAT 2003)*, **2219** of *Springer LNCS*, pages 135–149, 2004.