Simple but Hard Mixed Horn Formulas

Gayathri Namasivayam and Mirosław Truszczyński

Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
gayathri|mirek@cs.uky.edu

Abstract. We study simple classes of mixed Horn formulas, in which the structure of the Horn part is drastically constrained. We show that the SAT problem for formulas in these classes remains NP-complete, and demonstrate experimentally that formulas randomly generated from these classes are hard for the present SAT solvers, both complete and local-search ones.

1 Introduction

We study some simple classes of mixed Horn formulas and show that randomly generated formulas from these classes are hard for the present SAT solvers. A conjunctive normal form (CNF) formula F is a *mixed Horn formula* (an MHF, for short) if each clause in F is either a positive 2-clause (a clause of the form $a \vee b$, where a and b are propositional variables), or is Horn clause.

MHFs have received much attention recently [1, 2]. Researchers proved that many NP-complete problems have simple encodings as MHFs [2], showed that the satisfiability of MHFs remains NP-compete even under additional restrictions of the structure of input MHFs [2], and developed satisfiability algorithms for MHFs with good worst-case behavior lower bounds [1, 3].

Due to their simplicity on the one hand, and the expressive power on the other, MHFs are attractive as possible benchmarks for SAT solvers. Our goal in this paper is to propose some models of simple random MHFs of particularly constrained structure, and to show that these models yield instances that are hard for SAT solvers. In the process, we find interesting connections to a class of random logic programs that we recently identified as consisting of instances that are hard for answer-set solvers [5].

2 Preliminaries

Let $\mathcal{V} = \{v_1, v_2, \ldots\}$ be a fixed set of propositional variables. We define the class $MH_n(k, m)$, where $k \geq 1$ and $m \geq 0$, to consist of MHFs F such that

1. the set of atoms occurring in F is $\{v_1, \ldots, v_n\}$
2. F contains m positive 2-clauses
3. for every $v \in \mathcal{V}$, F contains a negative clause $C_v = \neg v \vee \neg w_1 \vee \ldots \vee \neg w_k$, where $w_1, \ldots w_k \in \mathcal{V}$
4. there are no other clauses in F.

We also define $MH_n(k) = \bigcup_m MH_n(k,m)$ (here m ranges from 0 to $\binom{n}{2}$), and $MH(k) = \bigcup_{n=0}^{\infty} MH_n(k)$.

Thus, formulas in $MH(k)$ contain only positive 2-clauses and negative $(k+1)$-clauses. The key aspect of the model is, though, that there is an additional constraint imposed on the set of negative $(k+1)$-clauses of a formula $F \in MH(k)$: the set of variables of F must be a system of distinct representatives for the family of the sets of variables of $(k+1)$-clauses of F.

Other classes of MHFs we consider in the paper impose additional connections between the negative and positive parts. Namely, we consider the class $MH^1_n(k)$, which we define as follows: an MHFs $F \in MH_n(k)$ belongs to $MH^1_n(k)$ if and only if its set of positive 2-clauses is given by $\{v \vee w \mid w \in Var(C_v)$, where $C_v \in F\}$. In the case of MHFs in $MH^1_n(k)$, there is a strong connection between the sets of positive and negative clauses: if $F \in MH^1_n(k)$, then F is *entirely* determined by its negative part. We note that the number of 2-clauses in formulas in $MH^1_n(k)$ is not fixed and ranges between $kn/2$ and kn. We write $MH^1(k)$ for $\bigcup_{n=0}^{\infty} MH^1_n(k)$, and MH^1_n for $\bigcup_{k=1} MH^1_n(k)$.

Despite constraints on the form of MHFs that form the classes $MH(k)$ and $MH^1(k)$, for each of them the satisfiability remains NP-complete.

Proposition 1. *For each of the classes $MH(k)$ and $MH^1(k)$, with $k \geq 2$, the satisfiability problem restricted to that class of formulas is NP-complete.*

Thus, the classes of formulas discussed above are on the one hand extremely simple, and on the other hand, as expressive as the class of (unrestricted) CNF formulas. Moreover, it is clear that in the case of each class $\mathcal{C}$ of formulas we introduced above, there are straightforward algorithms to generate formulas from $\mathcal{C}$ uniformly at random. These properties suggest that these classes be considered as possible models of random CNF formulas for use as benchmarks for SAT solvers.

3 Phase transition for $MH_n(k,m)$

For every fixed k and fixed sufficiently large n, the class $MH_n(k)$ demonstrates the classical phase-transition behavior. That is, when m (we recall that m stands for the number of 2-clauses in a formula) is small, formulas in $MH_n(k)$ are almost certainly satisfiable, when m is large, they are almost certainly unsatisfiable, and the transition from satisfiability to unsatisfiability occurs rapidly (the rate of change increases with n). Figures 1(a) and (b) show the phase transition for $k = 5$ and $k = 10$, respectively (with $n = 225$ and 200 instances).

The location of the phase-transition region expressed in terms of the ratio m/n, where m is the number of positive 2-clauses for which the phase transition occurs, grows with k. Our experimental results for $n = 200$ and $k = 3, \ldots 25$, and based on 200 instances, show that the location of the phase transition grows slightly slower than k. Figure 2 shows the dependence.

As in the best studied case of random 3-CNF formulas, there is a strong correlation between the phase transition region and the hardness of the instances generated from that region. The graph given in Figure 3 is representative. It gives the average running time in seconds for *clasp* on instances from the model $MH_n(k)$, for $k = 5$, $n = 200$,

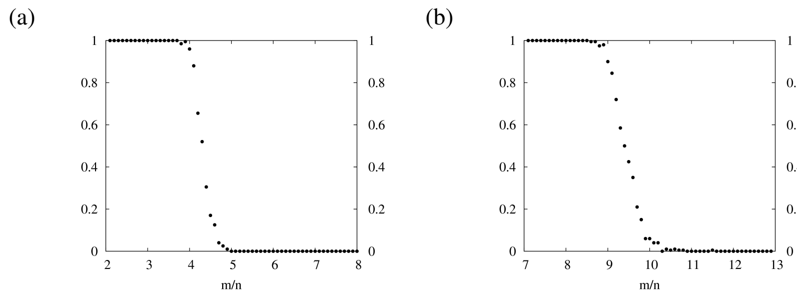

Fig. 1. The phase transition $k = 5$ and $k = 10$, $n = 225$, 200 instances

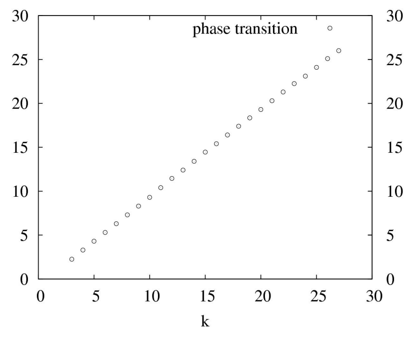

Fig. 2. The location of the phase transition in the model $MH_n(k)$ as a function of k

as the density of 2-clauses m/n grows (for each density 100 instances were generated). It shows that the instances requiring on average the most time come from the phase transition region. The results for other solvers and other values of k were similar.

4 Easy-hard-easy behavior

We pointed out above that for a fixed k, as the number of 2-clauses, m, grows, instances from $MH_n(k)$ are initially getting harder and then, after passing the area of the phase transition, start getting easier again.

However, the framework of the classes $MH_n(k)$ we consider reveals yet another interesting phenomenon. Being parameterized with k, it allows us to compare hardness of instances generated from the phase transition region for *different* values of k. Somewhat surprisingly, it turns out that as we increase k, the easy-hard-easy pattern emerges again. We observed an easy-hard-easy pattern using several SAT solvers that performed well in the SAT 2009 competition [4], including *precosat*, *glucose*, *clasp* and *march hi* (each was the winner in at least one of the categories of that competition). Initially, as k grows, the phase-transition instances are getting harder at an increasing rate. The hardness peaks when $k \approx 15, 16$, and from that point on the instances are becoming increasingly easier. Figure 4 illustrates that pattern observed for *clasp* for $n = 200$, and k ranging from 1 to 50.

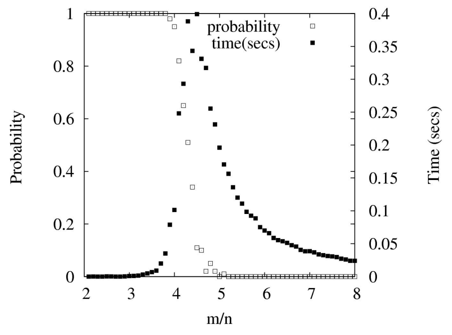

Fig. 3. The correlation of the instance hardness and the phase transition regions for the model $MH_n(k)$; $k = 5, n = 200$

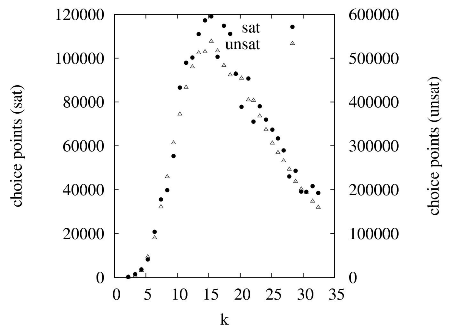

Fig. 4. The easy-hard-easy pattern of instances generated from the phase transition region of classes $MH_n(k)$ as a function of k.

While it is rather natural that the hardness of instances from the phase transition in the model $MH_n(k)$ initially grows with k, it may seem surprising that at some point it peaks and then starts to decrease. Providing a formal explanation to this phenomenon is an interesting open problem.

5 Hard Benchmarks for SAT Solvers

Our results suggest that MHFs randomly generated from the phase transition region for the class $MH_n(k)$ for $k = 15$ or 16 (located when $m \approx (k - 0.5)n$, where m stands for the number of 2-clauses) can provide challenging instances for SAT solvers. It is indeed so. We randomly generated 50 instances from $MH_n(k, m)$, with $n = 350$, $k = 15$ and $m = 14.5$. Given the timeout limit of 1800 seconds, *clasp* and *march hi* solved fewer than 20% of the satisfiable instances and none of the unsatisfiable ones.

We stress that the instances in the set $MH_n(15, 14.5)$ are small (350 atoms and 5425 clauses), and more importantly, that most of their clauses (5025) are 2-clauses. Since they pose a challenge for the state-of-the-art complete solvers, the class $MH_n(15, 14.5)$ is important for the design and testing solvers performance.

The classes $MH_n^1(k)$ offer even harder instances. While they can also serve as benchmarks for complete solvers, even for relatively small values of n, satisfiable instances from $MH_n^1(15)$ become very hard also for local-search solvers! The selection of $k = 15$ is not accidental. Our experiments showed that when we vary k, we observe the easy-hard-easy pattern, with the peak for $k \approx 15$. We also found (a property important below) that in the maximum hardness area, the percentage of instances that are satisfiable exceeds 90%. Figure 5 illustrates these claims.

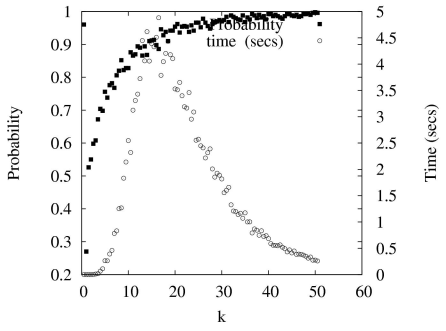

Fig. 5. The easy-hard-easy pattern for the model $MH_n^1(k)$, and the probability of satisfiability ($n = 100$)

We generated 100 random CNF formulas from each of the sets $MH_n^1(15)$, where $n = 450$ and 550. Given our experiments, the expected number of satisfiable instances in these two sets of formulas is at least 90. We ran *TNM* [4] on these formulas. *TNM* is currently one of the best local-search solvers. It won in the random category (satisfiable instances only) at the SAT 2009 competition. The solver does not require any parameters, as it adaptively selects them. We observed that for $n = 450$, *TNM* could still solve 86% of the instances in less than 1800 seconds (yet, already likely missing some satisfiable instances). The larger value of n, $n = 550$, resulted in many hard instances. Indeed, for $n = 550$, *TNM* solved only 53 of the 100 instances within 1800 seconds while we expect about 90 instances to be satisfiable in this sample.

6 Conclusions

We studied classes of simple MHF's: $MH_n(k)$ and MH_n^1. The key finding is that despite their simple form, randomly generated formulas from these classes (for the appropriate selections of parameters) are challenging benchmarks for the current generation of the state-of-the-art SAT solvers. Thus, formulas in these classes are relevant for the design of fast SAT solvers and deserve attention. We studied these classes experimentally, focusing on identifying phase transitions and hardness patterns, in order to facilitate generation of hard formulas. Interestingly, we found that the hardness of the instances from the phase transition region in the classes $MH_n(k)$ shows the easy-hard-easy pattern as a function of k, with the peak hardness for $k = 15$. We showed that instances generated from the phase transition region of $MH_n(15)$ (which occurs when the number of 2-clauses is about $14.5n$) pose a challenge instances for the current generation of SAT solvers. Similarly, the instances from MH_n^1 show the easy-hard-easy behavior (as the length k of purely negative clauses, grows), with the peak hardness when $k = 15$. The instances generated from $MH_n^1(15)$ are predominantly satisfiable (probability of a random formula generated from that class being satisfiable is at least 0.9). Those instances that are satisfiable pose a challenge for local-search solvers.

We note that the class MH_n^1 is closely related to a class R^- of logic programs studied in [5] and identified as containing programs that are especially hard for the current generation of the answer-set solvers. The class R^- consists of programs whose every clause is of the form $a \leftarrow not\ b$. The *completions* [6] of such programs (certain CNF theories whose models capture, in this particular case, the semantics of logic programs given by answer sets [7]) consist of positive 2-clauses and purely negative clauses (possibly of varying lengths). Completions of programs from R^-, in which every atom appears in the head of exactly k rules form precisely the class $MH_n^1(k)$.

This paper contains mostly experimental results. Yet it opens several interesting theoretical questions concerning tight bounds on the location of the phase transition in the model $MH_n(k)$, the properties of formulas in sets $MH_n(k)$ and $MH_n^1(k)$, when k grows with n (for instance, when $k = cn$, for some positive c), and possible reasons for the easy-hard-easy pattern demonstrated by formulas from the phase transition region of $MH_n(k)$ as k grows (or by formulas in $MH_n^1(k)$, as k grows).

References

1. Kottler, S., Kaufmann, M., Sinz, C.: A new bound for an NP hard subclass of 3-sat using backdoors. In: Proceedings of SAT 2008. Volume 4996 of Lecture Notes in Computer Science., Springer (2008) 161–167
2. Porschen, S., Schmidt, T., Speckenmeyer, E.: On some aspects of mixed horn formulas. In: SAT '09: Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, Berlin, Heidelberg, Springer-Verlag (2009) 86–100
3. Porschen, S., Speckenmeyer, E.: Worst case bounds for some NP-complete modified horn-sat problems. In: Proceedings of SAT 2004. Volume 3542 of Lecture Notes in Computer Science., Springer (2004) 251–262
4. : The SAT 2009 competition. http://www.satcompetition.org/ (2009)
5. Namasivayam, G., Truszczyński, M.: Simple random logic programs. In: Proccedings of LPNMR 2009. Volume 5753 of Lecture Notes in Cmputer Science., Springer (2009) 223–235
6. Clark, K.: Negation as failure. In Gallaire, H., Minker, J., eds.: Logic and data bases. Plenum Press, New York-London (1978) 293–322
7. Gelfond, M., Lifschitz, V.: The stable semantics for logic programs. In: Proceedings of the 5th International Conference on Logic Programming (ICLP 1988), MIT Press (1988) 1070–1080