Journal on Satisfiability, Boolean Modeling and Computation 2 (2006) 221-227

*Research Note*

# Locality and Hard SAT-Instances

**Klas Markström** Klas.Markstrom@math.umu.se
*Department of Mathematics, UmeåUniversity,*
*SE-901 87 Umeå, Sweden*

## Abstract

In this note we construct a family of SAT-instance based on Eulerian graphs which are aimed at being hard for resolution based SAT-solvers. We discuss some experiments made with instances of this type and how a solver can try to avoid at least some of the pitfalls presented by these instances. Finally we look at how the density of subformulae can influence the hardness of SAT instances.

Keywords: *satisfiability, hard instances, resolution, graphs, density*

*Submitted October 2005; revised December 2005; published March 2006*

## 1. Introduction

Most current programs for solving SAT-instances in CNF are based on the DPLL method [5, 6]. This method performs what is essentially a depth-first search combined with propagation of unit clauses. The method is polynomial time equivalent to tree-like resolution, see e.g. [11] for a textbook discussion of this and other basic facts about proof systems. Tree-like resolution is a fairly weak proof system and for a long time the performance of solvers were quite limited. However, during the 90's the basic DPLL method was extended by using conflict driven learning of new clauses, see e.g. [20], and the experimental performance of the solvers improved drastically. The addition of learning leads to a new proof system which is exponentially stronger than tree-like resolution, but still weaker than general resolution [2]. The next step was to introduce restarts of the DPLL procedure, while still keeping the learnt clauses, and as shown in [2] this combination leads to a method which is polynomial time equivalent to general resolution.

The described combination of methods has lead to impressive performance on many types of instances but since there are examples [17] of families of formulae for which any resolution refutation have exponential size there are also known limitations.

The aim of this note is to construct a family of CNF formulae which are specially tuned to make use of the structure of current solvers to produce small hard examples. Several instances from the constructed family were submitted as benchmarks for the 2005 SAT solver competition, none of which could be solved by the competing solvers. In connection with this we will also give a short discussion of the importance of local dense subformulae for the performance of resolution based solvers. Here we will also comment on some structural similarities between our family of formulae and formulae from the random $k$-SAT distribution.

## 2. The problem class

Recall that a simple graph $G$ is *Eulerian* if there exist a closed walk which uses every edge exactly once and that a graph is Eulerian if and only if all vertices have even degree. See e.g. [18] for general facts about graph theory.

Let us say that an Eulerian graph has an *even colouring* if the edges can be labeled 0 and 1 in such a way that half of the edges incident with any vertex have label 1.

**Lemma 2.1.** *An Eulerian graph $G$ has an even colouring if and only if it has an even number of edges.*

*Proof.* If $G$ is Eulerian and has an even number of edges we can construct an even colouring by starting at a vertex $v$ and walking along an Eulerian walk labeling the edges alternatingly 0 and 1. If the number of edges is odd this procedure will produce a colouring which is even at all vertices except the last one along the walk

Furthermore, it follows from a theorem of Kotzig [12] on Eulerian walks with restricted transitions that if $G$ has an even colouring there exists an Eulerian walk such that the above method generates the colouring. □

If $G$ has an odd number of edges the walk procedure will produce a labeling which is even at all vertices except the starting vertex $v$, for which the last edge along the walk will get a label creating an imbalance.

We now define a set of CNF formulae encoding the existence of an even colouring.

**Definition 2.2.** Given an Eulerian graph $G$ we define a boolean formula $EC(G)$ in CNF encoding the existence of an even colouring of $G$. For every edge $e$ in $G$ we create a boolean variable $x_e$ and for every vertex $u$ we create set of clauses encoding that half the variables connected to the edges incident with $u$ are true.

So, a vertex of degree 2 which is incident with two edges $a$ and $b$ will give the two clauses

$$\{(x_a \vee x_b),\ (\bar{x}_a \vee \bar{x}_b)\},$$

and a vertex of degree 4 which is incident with edges $a, b, c, d$ give rise to the ten clauses

$$\{(x_a \vee x_b \vee x_c \vee x_d),\ (\bar{x}_a \vee x_b \vee x_c \vee x_d),\ (x_a \vee \bar{x}_b \vee x_c \vee x_d),\ (x_a \vee x_b \vee \bar{x}_c \vee x_d),\ (x_a \vee x_b \vee x_c \vee \bar{x}_d),$$
$$(x_a \vee \bar{x}_b \vee \bar{x}_c \vee \bar{x}_d),\ (\bar{x}_a \vee x_b \vee \bar{x}_c \vee \bar{x}_d),\ (\bar{x}_a \vee \bar{x}_b \vee x_c \vee \bar{x}_d),\ (\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c \vee x_d),\ (\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c \vee \bar{x}_d)\}$$

The formulae $EC(G)$ have several interesting properties:

1. We can easily determine wether they are satisfiable or not by counting the number of edges in $G$.

2. Given an Eulerian graph $G$ with an odd number of edges the formula $EC(G)$ still has very large satisfiable subformulae. In fact according to the Eulerian walk argument above we can satisfy all but one of the clauses in $EC(G)$. Since it is only the assignment of the variable $x_e$ corresponding to the last edge along the Eulerian walk which creates an unsatisfied clause we can also find an optimal MAX-SAT assignment in which any given clause is satisfied.

3. By the previous property formulae of this kind will not have a “back-bone” [15], thus avoiding one structure making formulae easy to solve.

4. If the graph $G$ has good expanding properties, i.e. all small sets of vertices have many neighbours outside the set, and $EC(G)$ is unsatisfiable, the methods of [17] can be used to prove exponential lower bounds for the length of resolution refutations.

5. As long as $G$ has low maximum degree $EC(G)$ will have fairly few and short clauses. A vertex of degree 2 give rise to 2 clauses of length 2 and a vertex of degree 4 to 10 clauses of length 4. Thus giving us formulae $EC(G)$ with low clause per variable density.

## 2.1 Generating hard instances

Considering the outline of methods used in current DPLL-based SAT-solvers we can now make an observation about graphs which should lead to hard instances $EC(G)$. First let us restrict our attention to graphs $G$ which are obtained by picking an edge in a 4-regular graph on $n$ vertices and subdividing it, i.e. introducing a new vertex as midpoint of the old edge. For such a graph $EC(G)$ will be unsatisfiable, have $2n+1$ variables and $10n+2$ clauses. This gives us unsatisfiable formulae with a density of close to 5 clauses per variable. For formulae with clauses of length 4 this is a density for which a formula from the random 4-SAT ensemble is expected to be satisfiable, see e.g. [1]. In this random model a formula is constructed by for each possible clause of length 4, for a set of $n$ variables, letting it be part of the formula with probability $p$. Random 4-SAT formulae with expected density 5 are also expected to be easy to solve.

What can we say about the smallest number of variables involved in a nontrivial conflict, allowing us to learn a new clause, in a formula of this kind? We can get a simple lower bound for this by observing that we must at least set all variables corresponding to the edges of some cycle $C$ in $G$ before a nontrivial conflict can arise. Getting more accurate lower bounds is harder, but this tells us that the girth of the graph can be used to control the size of conflict sets.

This immediately draws our attention to 4-regular graphs, keeping the clause length low, of high girth. If the girth of $G$ is $g$ we will at first not be able to learn any nontrivial clauses of length less than, at least, $g$. If the SAT-solver we are using has a cut–off length for the clauses it learns, i.e. it does no retain any conflict clauses with more than some number $k$ of literals, it will in fact be prevented from learning any new clauses at all, thus reducing it to a pure DPLL procedure.

If the solver uses restarts and removes long clauses when it performs a restart it will in a similar way risk changing into a DPLL procedure with learning, but with an effective run time corresponding to the time between consecutive restarts.

With this in mind it seems natural to use formulae of the form $EC(G)$ based on 4-regular graphs, as described, as challenging unsatisfiable instances for SAT-solvers. If the formulae are to be tuned for hardness then using graphs with high girth, or at least few short cycles, also seems natural. If hard solvable instances are desired one could apply the methods of [14] on $EC(G)$. The paper [14] presents a construction which, based on a hard unsatisfiable formula builds a solvable formula which is hard for a broad class of solvers.

| $g/n/N/C$ | 5/17/35/172 | 6/26/53/262 | 7/53/107/532 |
|---|---|---|---|
| Zchaff | 0.36s | 4.6s | more than 2h |
| Berkmin | 0.30s | 30.2s | more than 2h |
| Satzoo | 0.34s | 24.5s | 3450s |

**Figure 1.** Running time on 4-regular cages. $n$ is the number of vertices in the cage, $g$ the girth, N the number of variables and C the number of clauses

## 3. Experiments

I have written a `Mathematica`-notebook for converting a 4-regular graph $G$ into a formula of the type $EC(\hat{G})$, where $\hat{G}$ is obtained from $G$ by subdividing the lexicographically smallest edge in $G$. The notebook can be downloaded from the authors webpage [13]. The variable name in this implementation comes from lexicographical ordering of the edge in $\hat{G}$ and replacing this with a random numbering might make the instances harder.

As a first test three different solvers were tested on the formulae obtained by using the smallest 4-regular cages, recall that a cage is a smallest possible graph of a given girth and degree. The known cages can be obtained from Gordon Royle's webpage [16]. The solvers were Zchaff 2004.11.15 [19, 20], Berkmin561 [9] and Satzoo 1.02 [7, 8]. In Figure 3 we see the runtime for each of the three solvers. The tests were done on a 2.4MHz Celeron. As we can see it quickly becomes very hard to determine the satisfiability of the formulae. For these cages the formula $EC(G)$, which is satisfiable, turns out to be extremely easy to solve. All three solvers solved the 9-cage case in less than a second, as did the local search solver Unitwalk [10].

The formulae based on the 4-regular cages with girth 7, 8 and 9 were submitted as benchmarks for the 2005 SAT competition and none of the competing solvers managed to solve any of them, within the competition time limit of 100 minutes for each formula. These formulae are available on the author's webpage [13].

In order to further demonstrate the effect of short cycles on the hardness of the instances a number of random 4-regular graphs were also generated, using the standard uniform pairing model [3]. In Figure 2 we see the average running time plotted against the number of triangles in the graph. Here the Minisat solver [7] was used for 2600 graphs on 32 vertices, the machine used a 1.25 GHz G4 processor. Here the running time clearly decreases as the number of short cycles increases. This agrees with what we expect since a large number of short cycles give us smaller conflict sets.

## 4. Conclusions and further directions

Both our discussion in 2.1 and the experiments in the previous section clearly indicate the importance of small dense structures, such as triangles in the Eulerian graphs, for the hardness of an instance of the type considered here. However we would expect this to be true in greater generality. Let us make this a bit more precise.

Given a k-SAT formula $F$ let us look at its clause-variable incidence graph, i.e., the graph with one vertex for each variable, one vertex for each clause, and an edge between a

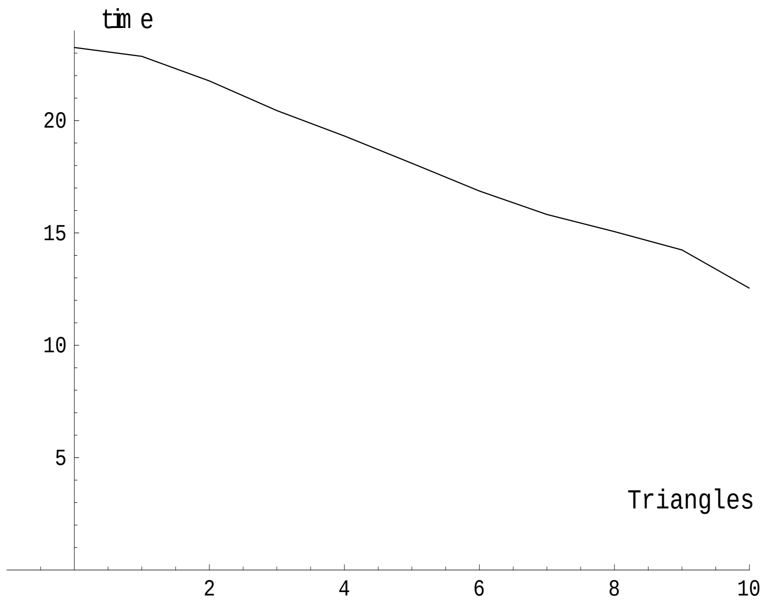

**Figure 2.** Running time in seconds versus the number of 3-cycles

variable-vertex and a clause-vertex if the variable is present in the clause. Given a subset $S$ of the variables we define $F[S]$, the induced subformula of $F$ on $S$, to be the set of clauses $C$ in $F$ such that $C$ only contains variables present in $S$. The density of the formula $F[S]$ is as usual the number of clauses in $F[S]$ divided by $|S|$.

If a formula $F$ has many dense subformulae a clause learning DPLL-solver has a good chance of learning new short conflict clauses due to conflicts within a dense local part. However for a given clause length $k$ there exist a density $r(k)$ such that a formula with clauses of length at most $k$ and density less than $r(k)$ is always satisfiable. So, if an unsatisfiable formula $F$ lacks such local dense parts a solver will initially have to assign values to many variables in order to find a conflict and learn a new clause. In this case the learnt clauses will also tend to be quite long.

If we look at random k-SAT instances of fixed clause to variable density we will only expect a small number of short cycles in the clause-variable incidence graph, actually an asymptotically size independent number, just as in the case of random regular graphs [3]. As a consequence, the density of $F[S]$ will typically be very small, often less than 1, for variables sets $S$ of small size. Likewise our formulae $EC(G)$ will lack small dense subformulae when the underlying graph $G$ has high girth. If we instead use random regular graphs $G$ we know, [3], that there will typically exist only a small number of cycles in $G$ and no small subgraphs denser than a cycle. In this case the formulae $EC(G)$ will thereby inherit expected properties much like those for random $k$-SAT.

Thus some of the experimentally observed hardness of random unsatisfiable instances may well come from their lack of local dense subformulae. Here an experimental study of running times for unsolvable random instances versus the number of short cycles in their incidence graphs could be interesting. In this context, it is interesting to note that for satisfiable instances this lack of local dense parts is considered to be responsible for some of the success of randomized algorithms like the survey propagation method [4].

An interesting possibility would be to make more explicit use of the structure of the clause-variable incidence graph of an instance in resolution methods as well. This could e.g. be used to control the choice of resolution variable in the DPLL procedure.

In order to reduced the risk of the kind of trap the $EC(G)$-formulae represent for clause learning SAT-solvers some simple modifications can be added. One addition which seems cost effective would be to keep track of a running mean of the size of the clauses discovered during the learning process and keep all clauses not much larger than the current mean. That way long clauses would still be kept for formulae in which no short clauses are possible to learn without first learning long clauses.

However, the general problem faced here is still the fundamental limitations of solvers which do not use a proof system stronger than general resolution. While producing more optimised resolution based solvers is undoubtedly a worthwhile undertaking it is becoming more and more important to find efficient algorithms based on stronger proof systems.

## References

[1] Dimitris Achlioptas and Yuval Peres. The threshold for random $k$-SAT is $2^k \log 2 - O(k)$. *J. Amer. Math. Soc.*, **17**(4):947–973 (electronic), 2004.

[2] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artificial Intelligence Res.*, **22**:319–351 (electronic), 2004.

[3] Béla Bollobás. *Random graphs*, volume **73** of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, second edition, 2001.

[4] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: an algorithm for satisfiability. *Random Structures Algorithms*, **27**(2):201–226, 2005.

[5] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Comm. ACM*, **5**:394–397, 1962.

[6] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. Assoc. Comput. Mach.*, **7**:201–215, 1960.

[7] Nicklas Een and Niklas Sörensson. Webpage for satzoo and minisat at chalmers university.
http://www.cs.chalmers.se/~een/Satzoo/.

[8] Nicklas Een and Niklas Sörensson. An extensible sat-solver. In *Lecture Notes in Computer Science, 2919,SAT 2003*, pages 502–518, 2004.

[9] Eugene Goldberg and Yakov Novikov. Webpage for berkmin.
http://heigold.tripod.com/BerkMin.html.

[10] Edward A. Hirsch and Arist Kojevnikov. UnitWalk: a new SAT solver that uses local search guided by unit clause elimination. *Ann. Math. Artif. Intell.*, **43**(1-4):91–111, 2005. Theory and applications of satisfiability testing.

[11] Hans Kleine Büning and Theodor Lettman. *Propositional logic: deduction and algorithms*, volume **48** of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1999. Translated from the 1994 German original by the authors.

[12] Anton Kotzig. Moves without forbidden transitions in a graph. *Mat. Časopis Sloven. Akad. Vied*, **18**:76–80, 1968.

[13] Klas Markström. Web page of current author, see combinatorial data and programs. http://www.math.umu.se/~klasm.

[14] Edward A. Hirsch Michael Alekhnovich and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. In *International Colloquium on Automata, Languages and Programming,ICALP 2004*, pages 84–96, 2004.

[15] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic "phase transitions". *Nature*, **400**(6740):133–137, 1999.

[16] Gordon Royle. Gordon royles homepages for combinatorial data. http://www.cs.uwa.edu.au/~gordon/remote/cubics/index.html.

[17] Alasdair Urquhart. Hard examples for resolution. *J. Assoc. Comput. Mach.*, **34**(1):209–219, 1987.

[18] Douglas B. West. *Introduction to graph theory.* Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[19] Lintao Zhang. webpage for zchaff at princeton university. http://www.princeton.edu/~chaff/zchaff.html.

[20] Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient conflict driven learning in boolean satisfiability solver. In *International Conference on Computer Aided Design, ICCAD 2001*, pages 279–285, 2001.