Journal on Satisfiability, Boolean Modeling and Computation 1 (2007) 143-145

Research Note

A Note on the Use of Independent Sets for the k-SAT Problem

Konstantin Kutzkov kutzkov@gmail.com
Naroden geroy 6,
1618 Sofia,
Bulgaria

Abstract

An independent set of variables is one in which no two variables occur in the same clause in a given k-SAT instance. Recently, independent sets have obtained more attention. Due to a simple observation we prove that a k-SAT instance over n variables with independent set of size i can be solved in time $O(\phi_{2(k-1)}(n-i))$ where $\phi_k(n)$ denotes an upper bound on the complexity of solving k-SAT over n variables.

KEYWORDS: *k-SAT problem, upper bounds, independent sets of variables*

Submitted January 2007; revised March 2007; published March 2007

1. Introduction

Independent sets of variables for the k-SAT problem have recently obtained attention since their usefulness has been practically shown [3]. In [3] the authors prove that a modification of the PPZ-algorithm [6] yields a bound of $O(2^{(n-i)(1-\frac{1}{2k-2})})$ and a modification of Schönings local search algorithm [7] needs $O((\frac{2k-3}{k-1})^{n-i})$ steps. We show that these bounds can be achieved by an obvious simplification of the formula. Further, by applying the currently best algorithms for k-SAT these bounds are significantly improved. Due to its simplicity the method could find an application in SAT solvers which explore independent sets of variables.

2. Preliminaries

A k-SAT formula is a conjunction of m clauses with at most k literals per clause. As usual, n denotes the number of the variables in a given k-SAT instance. A variable is written as c_i, $1 \leq i \leq n$, a literal as c_i and its negation as $\neg c_i$. V denotes the set of variables and $L := V \cup \overline{V}$ the set of literals. A k-clause $C = (c_1 \vee .. \vee c_k)$ is a disjunction of k literals.
An independent set of variables $I \subseteq V$ contains at most one variable pro clause. Its cardinality is written as $i := |I|$.
$\phi_k(n)$ denotes an upper bound on the complexity of solving k-SAT over n variables.

3. Main result

Theorem 1. A k-SAT instance F over n variables with independent set $I \subseteq V$ of size i can be solved in $O(\phi_{2(k-1)}(n-i))$ steps .

Proof: In order to simplify the reading we color the variables $v \in I$ in red and the rest variables $u \in V \backslash I$ in blue. Then the following **Algorithm 1** solves the problem: (The input is k-CNF formula F over n variables and an independent set $I \subseteq V$ of size i. We assume that the formula is not empty and $n > 0$. It is obvious that $1 \leq i \leq n$.)

while $(I \neq \emptyset)$
    choose a red variable $v \in I$
    resolute(F, v)
    simplify(F)
solve the $2(k-1)$-SAT problem for F

The procedure *resolute* (SAT instance F, variable v) implements the well-known *resolution rule* which doesn't affect the satisfiability of a given SAT formula [2]:

for each pair of clauses $(v \vee C)$, $(\neg v \vee D)$
    add the clause $(C \vee D)$ to the clause database
delete each clause containing the variable v

The procedure *simplify* (SAT instance F) deletes clauses containing a literal and its negation and clauses subsumed by another clauses. This guarantees that during the execution of the algorithm the size of the formula remains bounded by $O((n)^{2(k-1)})$.

Since we resolute only on red variables and each clause contains at most one red variable the clauses added by the resolution consist of blue variables exclusively. Therefore, the new clauses have at most $2(k-1)$ literals and all these are blue (so we don't resolute on a variable occurring in a $2(k-1)$-clause). Thus, the resulting formula is a $2(k-1)$-SAT instance over $n-i$ variables. □

In table 1 we give the running times of the best known $2(k-1)$-SAT algorithms w.r.t. to the number of variables n, deterministic as well probabilistic, for $k \geq 3$. The bounds for a k-SAT instance over n variables with independent set I of size i can be immediately concluded:

Table 1. Best known running times for $2(k-1)$-SAT algorithms

$2(k-1)$	probabilistic	deterministic
4	$O(1.474^n)$ [4]	$O(1.6^n)$ [1]
6	$O(1.638^n)$ [5]	$O(1.714^n)$ [1]
8	$O(1.725^n)$ [5]	$O(1.778^n)$ [1]
...	...	...

References

[1] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. M. Kleinberg, C. H. Papadimitriou, P. Raghavan, U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, **289** (1): 69–83, 2002.

[2] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, **7**: 201–215, 1960.

[3] R. Gummadi, N. S. Narayanaswamy, V. Ramaswamy. Algorithms for Satisfiability Using Independent Sets of Variables. *SAT 2004 LNCS*, **3542**: 133–144, 2005.

[4] K. Iwama, S. Tamaki. Improved Upper Bounds for 3-SAT. *Electronic Colloquium on Computational Complexity*, **53**: 2003.

[5] R. Paturi, P. Pudlak, M. E. Saks, F. Zane. An improved exponential-time algorithm for k-SAT *J. ACM*, **52** (3): 337–364, 2005.

[6] R. Paturi, P. Pudlak, F. Zane. Satisfiability Coding Lemma. *Chicago J. Theor. Comput. Sci.* 1999: 1999.

[7] U. Schöning. A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. *FOCS* 1999: 410–414, 1999.