Algorithms for Satisfiability Using Independent Sets of Variables

Ravi Gummadi[1], N.S. Narayanaswamy[1,⋆], and R. Venkatakrishnan[2]

[1] Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai-600036, India
gravi@peacock.iitm.ernet.in, swamy@shiva.iitm.ernet.in
[2] Department of Information Technology,
Crescent Engineering College, Vandalur, Chennai-600048, India
coolvenk@sancharnet.in

Abstract. An *independent set* of variables is one in which no two variables occur in the same clause in a given instance of k-SAT. Instances of k-SAT with an independent set of size i can be solved in time, within a polynomial factor of 2^{n-i}. In this paper, we present an algorithm for k-SAT based on a modification of the *Satisfiability Coding Lemma*. Our algorithm runs within a polynomial factor of $2^{(n-i)(1-\frac{1}{2k-2})}$, where i is the size of an independent set. We also present a variant of Schöning's randomized local-search algorithm for k-SAT that runs in time which is with in a polynomial factor of $(\frac{2k-3}{k-1})^{n-i}$.

1 Introduction

The Propositional Satisfiability Problem (SAT) is one of significant theoretical and practical interest. Historically, SAT was the first problem to be proven $\mathcal{NP}$-complete. No polynomial-time algorithm for a k-SAT problem ($k \geq 3$) is known, and no proof of its non-existence has been proposed, leaving open the question of whether $\mathcal{P} = \mathcal{NP}$?. The Satisfiability problem has important practical applications. For instance, in circuit design problems, a circuit that always produces an output of 0, can be eliminated from a larger circuit. This would reduce the number of gates needed to implement the circuit, thereby reducing cost. This problem naturally motivates the question of whether a given formula is satisfiable. Further, all the problems in the class $\mathcal{NP}$ can be reduced in polynomial-time to the Satisfiability problem. There are many practically important problems in this class. Therefore, a fast algorithm for SAT can also help to solve these problems efficiently. However, the existence of polynomial-time algorithms for $\mathcal{NP}$-complete problems is believed to be unlikely. Consequently, current research on SAT is focused on obtaining non-trivial exponential upper-bounds for SAT algorithms. For example, an algorithm running in $O(2^{n/r})$ for

⋆ Partly Supported by DFG Grant No. Jo 291/2-1. Part of Work done when the author was at the Institut Für Informatik, Oettingenstrasse 67, 80538, Munich, Germany.

H.H. Hoos and D.G. Mitchell (Eds.): SAT 2004, LNCS 3542, pp. 133–144, 2005.

instance, with large r could prove useful in solving many practical problems. Current research on SAT is focused on obtaining non-trivial exponential upper-bounds for SAT algorithms.

Algorithms for SAT. SAT algorithms are classified into *Splitting algorithms* and *Local Search algorithms* [DHIV01]. *Splitting algorithms* reduce the input formula into polynomially many formulae. The two families of Splitting algorithms are DPLL-like algorithms and PPSZ-like algorithms. *DPLL-like algorithms* [DP60, DLL62] replace the input formula F by two formulas $F[x]$ and $F[\neg x]$. This is done recursively. Using this technique, Monien and Speckenmeyer [MS85] gave the first non-trivial upper bounds for k-SAT. *PPSZ-like algorithms* [PPZ97, PPSZ98] use a different approach: variables are assigned values in a random order in the hope that the value of many variables can be obtained from the values of variables chosen prior to it. *Local Search algorithms* work by starting with an initial assignment and modifying it to come closer to a satisfying assignment. If, after a certain number of steps no satisfying assignment is found, the algorithm starts again with a new initial assignment. After repeating a certain number of times, if the algorithm does not find a satisfying assignment, it halts reporting "Unsatisfiable". *Greedy algorithms* [KP92] may be used to modify the current assignment in Local Search algorithms. These algorithms change the current assignment such that some function of the assignment increases as much as possible. *Random walk algorithms* [Pap91], on the other hand, modify the current assignment by flipping the value of a variable at random from an unsatisfied clause. 2-SAT can be solved in expected polynomial-time by a random walk algorithm [Pap91]. [Sch99] shows that k-SAT can be solved in time $(2 - 2/k)^n$ up to a polynomial factor. From the literature [Sch99, PPZ97, PPSZ98], it is clear that the current asymptotic performance of the local search algorithms is better than PPSZ-like algorithms.

Our Work and Results. Our main motivation is to explore further directions towards improving the performance of PPSZ-like algorithms. While the algorithm in [PPZ97] computes the values of variables in a random order, in the process shrinking the search space based on the formula, we observe that variable sets which have a special property with respect to the formula naturally shrink the search space. For example, if I is a set of variables in which no two of them occur in a clause, then the values to the variables of I can be computed very easily given an assignment to the variables outside I. Consequently, we could spend our effort on trying to find an assignment to variables outside I that can be extended to variables of I to obtain a satisfying assignment for the formula. This is precisely the approach of this paper. We first consider the brute force algorithm, and then modify the Satisfiability Coding Lemma, and Schöning's randomized algorithm to obtain an assignment to variables outside I. While we have not obtained an improved algorithm in general, we observe that our algorithms guarantee to be faster in the case when I is large enough in a given formula. On the other hand, it is also quite easy to construct formulae in

which I is very small in which case the performance is the same as the algorithm in [PPZ97]. So the motivation for our work is our conjecture that random satisfiable formulae have large independent sets. This is reiterated by the benchmarks for satsolvers which have independent sets of size $\frac{n}{4}$.

Independent set like structures in the formula have been used to obtain better algorithms for 3-sat. In particular, the paper by [SSWH02] uses a set of disjoint clauses to identify the initial starting point of Schöning's randomized algorithm [Sch99]. Indeed the disjoint clauses form an independent set in the set of clauses. On the other hand, we use independent sets of variables in our attempt to improve the performance of algorithms for k-sat based on the Satisfiability Coding Lemma [PPZ97, PPSZ98].

Roadmap. Section 2 presents the preliminaries, the brute force algorithm in Section 2.1. The modified Satisfiability Coding Lemma is presented in Section 3, and the algorithm based on it is presented and analyzed in Section 4. The random walk algorithm is presented in Section 5. A discussion in Section 6, and a construction of formulae with small independent sets in Section 6.1 concludes the paper.

2 Preliminaries

We have used the usual notions associated with the k-SAT problem. The reader is referred to [DHIV01] for this. Let V denote the set of variables in an instance F of k-SAT. An *Independent Set* $I \subseteq V$, is a set of variables such that each clause contains at most one element of the set. In this paper, we consider independent sets that are maximal with respect to inclusion. I denotes a fixed maximal independent set of cardinality i in F. Given an assignment a' to the variables of $V - I$, we can check whether it can be extended to a satisfying assignment in polynomial time: when we substitute a' into the formula, then we get a conjunction of literals. Every variable in this conjunction is an element of I. Further, testing if a conjunction of literals is satisfiable is a trivial issue. A truth assignment to the variables of $V - I$ is said to be *extensible* if there is a truth assignment to the elements of I such that the resulting assignment to $\{x_1, \ldots, x_n\}$ is a satisfying assignment. An assignment that cannot be extended to a satisfying assignment is called a *non-extensible* assignment. An extensible assignment is said to be *isolated* along a direction $j, x_j \notin I$, if flipping the value of x_j results in a non-extensible assignment.

Isolated Extensible Assignments. For a truth assignment a, a_i is said to be *critical* for a clause C if the corresponding literal is the only true literal in C under the assignment a. Without loss of generality, let us consider the variables of $V - I$ to have indices from $[n - i] = \{1, \ldots n - i\}$. Further, for an assignment a' to the variables outside I, $F(a')$ is a conjunction of literals from I. Let $b = b_1 \ldots b_{n-i}$ be an extensible assignment that is isolated along directions indexed by the elements of $J \subseteq [n - i]$. Let b' be the assignment obtained by flipping $b_r, r \in J$. b' is non-extensible for one of the following two reasons:

1. The formula is falsified by b' as there is a clause with all its variables from $V - I$, and b_r is critical for this clause. An assignment is said to be *easy isolated* along x_r if this property is satisfied.
2. There exists an $x_l \in I$, two clauses C_1, C_2 such that $x_l \in C_1$, $x_r, \neg x_l \in C_2$, and only x_l occurs in $F(b)$, but both x_l and $\neg x_l$ occurs in $F(b')$. An assignment that is not easy isolated along x_r is said to be *hard isolated* along x_r if this condition is satisfied. We refer to the two clauses C_1 and C_2 as *falsifying clauses* for b along direction r. We refer to them as falsifying clauses, leaving out b and r, when there is no ambiguity. Clearly, if an extensible assignment is hard isolated along x_r, there exist two falsifying clauses.

2.1 The Brute Force Approach

The idea is to find the largest independent set I, and search through all possible assignments to $V - I$. If an assignment is extensible, we report that F is satisfiable, otherwise report unsatisfiable when all assignments to $V - I$ have been tried. This algorithm runs in $O(2^{n-i}poly(|F|))$. While finding a large enough independent set is a problem by itself, we propose to find the maximum independent set by using the algorithm due to Beigel [Bei99] that runs quite efficiently. The other approach is to permute the variables at random and consider the independent set obtained by considering variables all of whose neighbours occur to their left in the random permutation. Two variables are said to be neighbours, if they occur together in a clause. This approach yields an independent set whose expected size is $\frac{n}{\Delta+1}$, where each variable has at most Δ neighbours.

3 A Variant of Satisfiability Coding Lemma

In the Section 2.1 we have observed a simple brute force algorithm that finds extensible solutions given an independent set I. We now improve this brute force algorithm by modifying the satisfiability coding lemma suitably. The approach that we take is similar to the approach in [PPZ97]. We first consider the issue of encoding isolated extensible solutions and bound the expected length of an encoding. We then show that this encoding process is reversible and it does prune our search space yielding a randomized algorithm that performs better than the brute force approach in Section 2.1. However, this does not better the performance of [PPZ97] unless I is a sufficiently large set.

Encoding. We consider the set of j-isolated extensible solutions for a fixed independent set of variables I. Let $x_1, \ldots, x_{n-i}$ be the variables of $V - I$ in a k-SAT formula. Let σ be a permutation on the set $\{1, \ldots, n-i\}$. Let $A = a_1 \ldots a_{n-i}$ be a binary string visualized as an assignment of a_r to $x_r, 1 \leq r \leq n-i$. Let A_σ denote the string $a_{\sigma(n-i)} a_{\sigma(n-i-1)} \ldots a_{\sigma(1)}$. In other words, A_σ is a permutation of A, according to σ. From A and σ, we construct an encoding $E(A, \sigma)$ as follows:

```
E(A,σ) is the empty string.
for(r = n − i; r ≥ 1; r − −)
begin
      if  A is isolated along  σ(r)
            AND all other variables in a critical clause for  x_σ(r)
                  occur to the left of  x_σ(r) in  A_σ
            OR the variables ∉ I in two falsifying clauses occur to
                  the left of x_σ(r) in  A_σ
            then do not add  a_σ(r) to  E(A,σ).
      else
            add a_σ(r) to  E(A,σ).
end
```

The operation of adding a bit to $E(A, \sigma)$ is equivalent to concatenating to the right end. The bits of this string are assumed to be indexed from left to right starting with 1 for the leftmost bit, and using consecutive indices. The output of the loop is $E(A, \sigma)$. Another point of view on $E(A, \sigma)$ is that it is obtained from A_σ by deleting some bits which can be computed from *previous* information in A_σ. Obviously, its length cannot exceed $n - i$.

Reconstruction. Given a k-SAT formula F, an independent set I, a bit string E, and a permutation σ, we find a bit string A such that $E(A, \sigma) = E$, if such an A exists. To obtain A we find $A_\sigma = a_{\sigma(n-i)} a_{\sigma(n-i-1)} \dots a_{\sigma(1)}$. The bit string E is considered from the leftmost bit. Each bit of E is assigned to at most one corresponding bit of A_σ. At each step the value of a bit of A_σ is inferred. It is inferred either by substituting the previously computed bits into the formula, or the current bit of E is assigned to A_σ.

Consider the case when A_σ has been computed up to the r+1-th bit, $n-i-1 \geq r \geq 1$. We substitute this partial assignment into F and consider the resulting formula. There are three cases:

$x_{\sigma(r)}$ *can be inferred from the previous values:* This can happen in two ways. The first, when $x_{\sigma(r)}$ occurs as a single literal. This means that there is a corresponding critical clause in which all other literals have been set to 0. $x_{\sigma(r)}$ is set appropriately to make the literal true. The second case is when a variable $x \in I$ occurs as $(x)(\neg x \vee y)$, where y is a literal of $x_{\sigma(r)}$. In this case, the value assigned to $x_{\sigma(r)}$ is inferred from the value that makes y true.

$x_{\sigma(r)}$ *takes its value from E:* This happens when $x_{\sigma(r)}$ does not satisfy either of the two conditions mentioned above. In this case, $x_{\sigma(r)}$ is to be assigned the current bit of E. If all the bits of E have been *used up* then halt reporting failure. At each step of the reconstruction, we keep track of whether a variable and its complement occur as single clauses. If this happens, we halt reporting failure. If A_σ is computed successfully, then it means that we have found an extensible assignment.

3.1 Quality of the Encoding

Here we discuss the expected length of $E(A, \sigma)$ when σ is chosen from a class of distributions on S_n, the set of permutations of $\{1, \ldots, n\}$. These distributions are characterized by γ and satisfy the following property

$$| \, Pr_{\pi \in \mathcal{F}}(\min\{\pi(X)\} = \pi(x)) - \frac{1}{|X|} \, | \leq \frac{\gamma}{|X|} \tag{1}$$

Here, $X \subseteq \{1, \ldots, n\}$ and $\pi(X)$ is the image of the set X under a permutation π. Clearly, the required probability is $\frac{1}{|X|}$ when π is chosen uniformly from the set of all permutations. The goal of identifying a smaller family of permutations that guarantee this property is well motivated and is studied by Charikar et. al in [MBFM00]. For each γ, D_γ is a probability distribution on S_n and $D_\gamma(\sigma)$ denotes probability of choosing σ in the distribution D_γ.

σ Chosen from D_γ. We now compute the average length of $E(A, \sigma)$ averaged over all $\sigma \in D_\gamma$. Clearly, the only directions that get eliminated are those along which A is either easy isolated or hard isolated. Let us assume that A is an extensible solution, easy isolated along j_e directions, and hard isolated along j_h directions. For a direction r along which A is easy isolated, we lower bound the probability that a_r is eliminated in the encoding of A with a randomly chosen permutation.

Since A is easy isolated along r, there is a corresponding critical clause all of whose variables are from $V - I$. a_r will be eliminated if all the $k-1$ literals in the critical clause occur to the left in a randomly chosen permutation. This event happens with probability at least $\frac{1-\gamma}{k}$. It follows from the linearity of expectation that, for an A which is easy isolated along j_e directions, the expected number of variables eliminated is at least $\frac{j_e(1-\gamma)}{k}$. Similarly a direction r, along which A is hard isolated, will be eliminated if all the variables belonging to $V - I$ from corresponding falsifying clauses occur to the left of x_r in a randomly chosen permutation. The number of such variables from two falsifying clauses is at most $2k - 3$. Consequently, this event happens with probability at least $\frac{1-\gamma}{2k-2}$. By linearity of expectation, the expected number of hard isolated directions that get eliminated is at least $\frac{j_h(1-\gamma)}{2k-2}$. Therefore, the expected value of $E(A, \sigma)$ is at most $n - i - (1 - \gamma)(\frac{j_e}{k} + \frac{j_h}{2k-2})$.

Existence of a Good Permutation. We now use the above argument to show that there is a permutation $\sigma \in D_\gamma$ for which the average length $E(A, \sigma)$, over all extensible solutions A isolated along $j = j_e + j_h$ directions, is upper bounded by $n - i - (1 - \gamma)(\frac{j_e}{k} + \frac{j_h}{2k-2})$. For this we consider the following average,

$$\sum_{\sigma} D_\gamma(\sigma) \sum_{A \in J} \frac{1}{|J|} E(A, \sigma) = \sum_{A \in J} \frac{1}{|J|} \sum_{\sigma} D_\gamma(\sigma) E(A, \sigma) \tag{2}$$

This is upper bounded by $n - i - (1 - \gamma)(\frac{j_e}{k} + \frac{j_h}{2k-2})$ since we know from the above calculation that $\sum_\sigma D_\gamma(\sigma)E(A, \sigma) \leq n - i - (1 - \gamma)(\frac{j_e}{k} + \frac{j_h}{2k-2})$.

It now follows by the pigeon hole principle that for some σ, $\sum_{A \in J} \frac{1}{|J|} E(A, \sigma) \leq n - i - (1-\gamma)(\frac{j_e}{k} + \frac{j_h}{2k-2})$. We state these bounds in the following theorem.

Theorem 1. *Let A be an extensible solution which is easy isolated along j_e directions, and hard isolated along j_h directions. The expected value of $E(A, \sigma)$ is at most $n - i - (1-\gamma)(\frac{j_e}{k} + \frac{j_h}{2k-2})$. Consequently, for J, the set of extensible solutions, easy isolated along j directions,there is a permutation $\sigma \in D_\gamma$ such that $\sum_{A \in J} \frac{1}{|J|} E(A, \sigma) \leq n - i - (1-\gamma)(\frac{j_e}{k} + \frac{j_h}{2k-2})$. Here i is the size of an independent set I.*

4 Algorithm to Find Satisfying Assignments

For a k-CNF $|F|$ with an independent set I, we use the result in Theorem 1 to design a randomized algorithm. Further, we set $\gamma = 0$, that is we use a family of permutations that guarantees exact min-wise independence. From now on, $\gamma = 0$. The effectiveness of this algorithm over the one presented in [PPZ97] depends on how large an independent set there is in the formula, and how much time is needed to find a reasonably large independent set. The algorithm that we present here, is quite similar to the randomized algorithm presented in [PPZ97]. In the description below, the word *forced* is a property of a variable whose value is determined by the values the previous variables. For example, a variable x_r is forced if it occurs as a single literal in $F(a_1, \ldots, a_{r-1})$. Here $a_1, \ldots, a_{r-1}$ are the assignments to the variables $x_1, \ldots, x_{r-1}$, respectively. x_r could also be forced if two falsifying clauses occur in $F(a_1, \ldots, a_{r-1})$.

```
Find an independent set I
Repeat n^2 2^{(n-i)(1-1/(2k-2))} times
      While there is an unassigned variable in V - I
            select an unassigned variable y from V - I at random
            If y is forced, then set y as per the forcing
            Else set y to true or false at random
      end while
  If the assignment can be extended
    then output the satisfying assignment
End Repeat
```

We state the following lemma, a special case of the isoperimetric inequality, which is used to prove our main theorem. We present a complete proof here.

Lemma 1. *Let $S \subseteq \{0,1\}^n$, be a non-empty set. For $x \in S$, define $I_n(x)$ be the number of distance-1 neighbours of x that are not in S. Define $value(x) = 2^{(I_n(x)-n)}$. Then, $\Sigma_{x \in S} value(x) \geq 1$.*

Proof. The proof is by induction on n. The base case is when $n = 1$. If $I_1(x) = 0$, then we observe that $\Sigma_{x \in S} value(x) = 1$. If $I_1(x) = 1$, $\Sigma_{x \in S} value(x) = 1$.

For $n > 1$, and $i \in \{0,1\}$, let S_i be a subset of $\{0,1\}^{n-1}$ such that for each $x \in S_i, xi \in S$. Now we consider two cases:

Case I: If one of the two sets is empty, then we have a direction along which each element of S is isolated. Let us consider S' to be a subset of $\{0,1\}^{n-1}$ obtained by projecting along the rightmost bit. By induction, $\Sigma_{x \in S'} value(x) \geq 1$. That is, $\Sigma_{x \in S'} 2^{I_{n-1}(x)-(n-1)} \geq 1$. Clearly, the number of directions along which an $x \in S$ is isolated in $\{0,1\}^n$ is one more than the number of directions along which it's projection(along the rightmost bit) is isolated in $\{0,1\}^{n-1}$. Consequently, $\Sigma_{x \in S'} 2^{I_{n-1}(x)+1-(n-1)}$ is exactly $\Sigma_{x \in S} 2^{I_n(x)-n}$. Hence the induction hypothesis is proved in this case.

Case II: If both S_i are non-empty. Then, by induction, $\Sigma_{x \in S_i} value(x) \geq 1$. Observe that, here $value(x)$ is defined with respect to S_i, for each i. Due of the induction hypothesis,

$$\begin{aligned} 2 &\leq \Sigma_{x \in S_0} 2^{I_{n-1}(x)-(n-1)} + \Sigma_{x \in S_1} 2^{I_{n-1}(x)-(n-1)} \\ &\leq 2\Sigma_{x \in S_0} 2^{I_n(x0)-n} + 2\Sigma_{x \in S_1} 2^{I_n(x1)-n} \\ &= 2\Sigma_{x \in S} 2^{I_n(x)-n} \end{aligned} \tag{3}$$

The equation 3 follows from the previous equation due to the fact that $I_{n-1}(x) \leq I_n(xi), i \in \{0,1\}$. The induction hypothesis holds in this case too, and hence the lemma is proved. □

The following theorem is proved using the Lemma 1 along the lines of a similar theorem in [PPZ97].

Theorem 2. *Let I be an independent set of variables in F, a satisfiable instance of k-SAT. The randomized algorithm in Section 4 finds a satisfying assignment with very high probability in time $O(n^2|F|2^{(n-i)(1-\frac{1}{2k-2})})$.*

Proof. Let S denote the set of extensible assignments. Let us assume that x is a j-isolated extensible solution of F. Among these let $j_e(x)$ and $j_h(x)$ be easy and hard isolated directions, respectively. The probability that x is output by the algorithm is the sum over all $d \geq 0$, probability that for a randomly chosen permutation, d directions are forced, and the remaining directions are chosen correctly. This is at least the probability that for a randomly chosen permutation, at least $\frac{j_e}{k} + \frac{j_h}{2k-2}$ directions are forced, and the remaining directions are guessed correctly. Recall that $\frac{j_e}{k} + \frac{j_h}{2k-2}$ is a lower bound expected number of directions that are eliminated by the process of encoding x. The probability of finding x is dependent on two events, one is to find a permutation that eliminates $\frac{j_e}{k} + \frac{j_h}{2k-2}$ directions, and the second is to make the correct choices on the remaining values. We now lower bound this probability by estimating the probability of finding a right permutation, and then conditioned on this event, estimate the probability of making the correct choices.

Probability of Finding a Right Permutation. Recall that a right permutation is one using which the process of encoding x eliminates at least $\frac{j_e}{k} + \frac{j_h}{2k-2}$ directions. We can now partition the permutation into the following sets: for $r < \frac{j_e}{k} + \frac{j_h}{2k-2}$, P_r consists of those permutation that eliminate r variables, and P_{av} consists of those permutations that eliminate at least $\frac{j_e}{k} + \frac{j_h}{2k-2}$ variables. The number of sets in this partition is at most $n-i$. Therefore, by the pigeon hole principle, one of these sets must have at least $\frac{1}{n-i}$ of the permutations. Following the argument in [PPZ97], P_{av} has at least $\frac{1}{n-i}$ of the permutations. Therefore, the probability of picking the right permutation is at least $\frac{1}{n-i} > \frac{1}{n}$.

Probability of Making the Right Choices on the Unforced Bits. Conditioned on the fact that a right permutation is chosen, we now estimate the probability that the right choices are made on the unforced bits so that we get x. The number of unforced bits is at most $n - i - \frac{j_e}{k} - \frac{j_h}{2k-2}$. The probability of making the correct choices is at least $2^{-(n-i-\frac{j_e}{k}-\frac{j_h}{2k-2})}$.

Therefore, the probability of picking x is at least $\frac{1}{n}2^{-(n-i-\frac{j_e}{k}-\frac{j_h}{2k-2})}$. The probability that the algorithm outputs some solution of F is given by the following:

$$\begin{aligned}
\Sigma_{x\in S}\Pr(x \text{ is output}) &\geq \Sigma_{x\in S}\frac{1}{n}2^{-(n-i-\frac{j_e(x)}{k}-\frac{j_h(x)}{2k-2})} \\
&\geq \frac{1}{n}2^{-(n-i)(1-\frac{1}{2k-2})}\Sigma_{x\in S}2^{-(n-i)+I(x)} \\
&\geq \frac{1}{n}2^{-(n-i)(1-\frac{1}{2k-2})} \qquad (4)
\end{aligned}$$

The last inequality follows from Lemma 1. The repetition of the **while** loop $n^2 2^{(n-i)(1-\frac{1}{2k-2})}$ increases the probability of finding a satisfying assignment to a constant. Hence the theorem is proved. □

Comparison with the Randomized Algorithm in [PPZ97]. The randomized algorithm presented in [PPZ97] has a running time of $O(n^2|F|2^{n-n/k})$, and ours has a running time of $O(n^2|F|2^{(n-i)(1-\frac{1}{2k-2})})$. Our algorithm does better than the algorithm in [PPZ97] when $(n-i)(1-\frac{1}{2k-2}) < n(1-\frac{1}{k})$. This happens when $i > \frac{n(k-2)}{k(2k-3)}$. For $k = 3$, our algorithm does better when $i > \frac{n}{9}$.

5 Extensible Solutions via Local Search

In this section, we analyze a modification of Schöning's local search algorithm to find an extensible solution. As usual, let I denote an independent set of cardinality i. The algorithm is as follows:

```
   Let I be a maximal independent set of variables.
For numtries times
   Select a random partial assignment a ∈ {0,1}^{n-i}
      Repeat 3n times
         Consider F(a) by substituting partial assignment a.
         if (C(a) = 0 for some C ∈ F)
            Randomly, flip the value of one of the literals from C
         else if (C_1(a) = x and C_2(a) = ¬x for C_1, C_2 ∈ F, x ∈ I)
            Randomly, flip one of the variables from C_1 ∪ C_2 − x
         else
            Extend a to a satisfying assignment s of F; return s;
      EndRepeat
EndFor
Return 'unsatisfiable'
```

Let a^* be an extensible assignment. We lower bound the probability that the above algorithm finds a^*, or some other extensible assignment. Let $a \in \{0,1\}^{n-i}$ be the initial random assignment, at a Hamming distance of j from a^*. To analyze the algorithm, we consider a walk on a Markov Chain, whose states are labelled $\{0, 1, 2, ..., n-i\}$. Each state represents the Hamming distance between the current assignment and a^*. Initially, the walk starts at state j. We now observe that at each step of the algorithm, the probability of moving one step closer to the state 0 is at least $\frac{1}{2k-2}$. This is easy to see, as we know that if a is not extensible, then either there is a clause $C, var(C) \subseteq V - I$, such that $C(a) = 0$, or there is an $x \in I$, and two clauses C_1, C_2 such that $C_1(a) = x$ and $C_2(a) = \neg x$. In the former case, the algorithm moves to an assignment with a lesser Hamming distance from a^* with probability at least $\frac{1}{k}$, and in the latter, with probability at least $\frac{1}{2k-2}$. The reasoning is that the values assigned to the variables in C by a and a^* have to differ at at least one variable. Similarly, the values assigned to variables in $C_1 \cup C_2 - x$ by a and a^* must differ at at least one variable. Consequently, the size of C and $C_1 \cup C_2 - x$ give the claimed probabilities. The probability of finding a^* from the chosen a (Hamming distance between a and a^* is j) in one iteration of the outer loop is at least the probability that the process moves from state j to state 0. This probability, denoted by q_j, is at least $(\frac{1}{2k-3})^j$. See [Sch99] for the derivation of this probability. Further, the success probability for one iteration of the outer loop is

$$p \;\geq\; (\tfrac{1}{2})^{n-i} \textstyle\sum_{j=0}^{n-i} \binom{n-i}{j} (\frac{1}{2k-3})^j \;=\; (\frac{1}{2}(1 + \frac{1}{2k-3}))^{n-i}$$

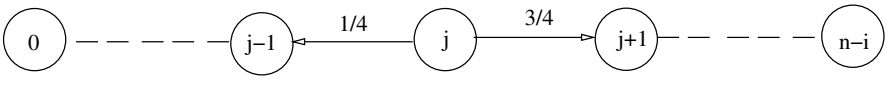

Fig. 1. Random Walk: Analysis of Local Search with Independent Set for 3-SAT

For $k = 3$, if the size of the independent set is high ($i \geq 0.3n$), then the algorithm works better than Schöning's randomized algorithm.

6 Discussion

In this paper, we have introduced the notion of an independent set of variables and use maximum independent sets in algorithms to solve k-SAT. The problem of finding a maximum independent set is a terribly hard problem. Even to find a good approximate solution in polynomial time is equally hard. However, when we permit exponential running time, finding a maximum independent set in an undirected graph has a provably better running time than the best known algorithms for k-SAT. The algorithm to find a maximum independent set due to [Bei99] runs in time $2^{.290n}$ which is approximately 1.2226^n. On the other hand, one of the best algorithms for 3-SAT is randomized and runs in time 1.3302^n [SSWH02]. Based on this observation, our approach spends some of the exponential time finding a maximum independent set, and then uses it to find a satisfying assignment. This approach is faster than [PPZ97, Sch99, SSWH02] only if the maximum independent set is sufficiently large. While there are formulae with very small independent sets, as we show below, an important direction of research is to explore the size of independent sets in random satisfiable formulae.

6.1 Formulae with Small Independent Sets

Here we construct formulae which have a small maximum independent set, and the number of clauses is also small, contradicting the intuition that small number of clauses mean large independent sets. Consider the following construction for a formula with n variables, and a parameter $1 \leq b \leq n$:

Step 1: Partition the variables into sets of b variables. There are n/b such sets.
Step 2: For each set of b variables, construct $\binom{b}{3}$ clauses made up of variables of same parity.

This formula is trivially satisfiable. The formula has $\binom{b}{3}\frac{n}{b}$ clauses, and the size of any independent set is $\frac{n}{b}$. The following table shows the sample values for different values of b.

b	no. of clauses	ind. set size
9	$9.3n$	$\frac{n}{9}$
8	$7n$	$\frac{n}{8}$

Acknowledgments. The second author would like to thank Jan Johannsen for discussions on SAT algorithms.

References

[Bei99] R. Beigel, *Finding Maximum Independent Sets in Sparse and General Graphs*, Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999.

[DHIV01] E. Dantsin, E.A. Hirsch, S. Ivanov, M. Vsemirnov. *Algorithms for SAT and Upper Bounds on their Complexity*. Electronic Colloquium on Computational Complexity, Report No.12, 2001.

[DLL62] M.Davis, G. Logemann, D. Loveland, *A machine program for theorem-proving*, Communications of the ACM **5(7)** (1962), 394-397.

[DP60] M.Davis, H. Putnam, *A computing procedure for quantification theory*, Journal of the ACM **7(3)** (1960), 201-215.

[KP92] E. Koutsoupias, C.H. Papadimitriou, *On the Greedy algorithm for Satisfiability*,Information Processing Letters **43(1)** (1992), 53-55.

[MBFM00] A. Z. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. *Min-wise independent permutations*, In Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, 1998, pages 327–336, 1998.

[MS85] B. Monien, E. Speckenmeyer, *Solving Satisfiability in less than* 2^n *steps*, Discrete Applied Mathematics **10** (1985), 287-295.

[Pap91] C.H. Papadimitriou, *On selecting a satisfying truth assignment*, Proceedings of FOCS'91, 1991, 163-169.

[PPSZ98] R. Paturi, P. Pudlák, M.E. Saks, F. Zane, *An improved exponential-time algorithm for k-SAT*, Proceedings of FOCS'98, 1998, 628-637.

[PPZ97] R. Paturi, P. Pudlák, F. Zane, *Satisfiability Coding Lemma*, Proceedings of FOCS'97, 1997, 566-574.

[Sch99] U. Schöning, *A probabilistic algorithm for k-SAT and constraint satisfaction problems*, Proceedings of FOCS'99, 1999, 410-414.

[SSWH02] T. Hofmeister, U. Schöning, R. Schuler, O. Watanabe, *A Probabilistic 3-SAT Algorithm Further Improved*, Proceedings of STACS'02, 2002, LNCS 2285:193-202.