

Algorithm based on Lovasz Local Lemma

Theorem 1. *Given a CNF formula of m clauses, each clause has a suitably large constant k number of literals, each variable appears in up to $2^{\alpha k}$ clauses, for a sufficiently small constant $\alpha > 0$. Then there is an algorithm that finds a satisfying assignment to the formula in expected time that is polynomial in m .*

Two phase algorithm:

Phase 1: A subset of the variables are assigned random values. The remaining variables are deferred to the second phase. Using the Lovasz local lemma show that the random partial solution can be extended to a full solution without changing any of the already fixed values.

Phase 2: Fix the values of the deferred variables by doing exhaustive search. If the dependency graph H between events defined by the deferred variables has only small connected components (w.h.p) then this phase will take only polynomial time.

Algorithm

Assume a CNF formula F , with m clauses, ℓ variables, each clause has k literals, each variable appears in no more than $T = 2^{\alpha k}$ clauses. Assume that k is even; the case k is odd is similar.

Let x_1, \dots, x_ℓ be the variables and C_1, \dots, C_m be the clauses of F .

First Part:

A clause is **Dangerous** at a given step if both the following conditions hold:

1. The clause is not satisfied;
2. At least $k/2$ of its variables were fixed.

For $i = 1$ to ℓ

If x_i is not in a dangerous clause assign it a random value in $\{0, 1\}$.

A **surviving clause** is a clause that is not satisfied at the end of phase one.

A surviving clause has no more than $k/2$ of its variables fixed.

A **deferred** variable is a variable that was no assigned value in the first part.

Lemma 1. *There is an assignment of values to the deferred variables such that all the surviving clauses are satisfied (thus the formula is satisfied).*

Proof

At the end of the first phase we have m' “surviving clauses” (all the rest are satisfied), each surviving clause has at least $k/2$ deferred variables.

Consider a random assignment of the deferred variables.

Let E_i be the event clause i (of the surviving clauses) is not satisfied.

$$p = \Pr(E_i) \leq 2^{-k/2}.$$

The degree of the dependency graph is bounded by

$$d = kT \leq k2^{\alpha k}.$$

For a sufficiently small constant $\alpha > 0$,

$$4dp = 4k2^{\alpha k/2}2^{-k/2} \leq 1$$

there is a satisfying assignment of the deferred variables that (together with the assignment of the other variables) satisfies the formula.

Part Two:

Using exhaustive search assign values to the deferred variables to complete the truth assignment for the formula.

If a connected component has $O(\log m)$ clauses it has $O(k \log m)$ variables. Assuming $k = O(1)$ we can check all assignments in polynomial in m number of steps.

Lemma 2. *Let G' be the dependency graph (with degree at most $d = kT$) on the surviving clauses. With high probability all connected components in G' have size $O(\log m)$.*

Proof: Consider a connected component R of size $r = |R|$ in G (the original dependency graph). If R is a connected component in G' , then all its r nodes are surviving clauses.

A clause survives the first part if it is either a dangerous clause or it shares at least one deferred variable with a dangerous clause, i.e., it has a neighbor in G' that is dangerous.

The probability that a given clause is dangerous is at most $2^{-k/2}$.

The probability that a clause survives is at most $(d + 1)2^{-k/2}$, where $d = kT > 1$.

However, the survival of clauses are not independent.

4-tree

A **4-tree** S of a connected component R in G is defined as:

1. S is a rooted tree.
2. any two nodes in S are at distance at least 4 in H .
3. there can be an edge in S only between two nodes with distance exactly 4 between them in G .
4. any node of R is either in S or is at distance at most 3 from a node in S .

The survival of clauses in a 4-tree S are independent.

This is because, for each clause in S there is a **distinct** dangerous clause, and these dangerous clauses are at distance 2 from each other.

Thus for any 4-tree S , the probability that all the nodes survive is at most

$$((d + 1)2^{-k/2})^{|S|}.$$

Maximal 4-tree

A maximal 4-tree S of a connected component R is the 4-tree with the largest possible number of vertices.

Since the degree of a vertex in R is bounded by d , there are no than

$$d + d(d - 1) + d(d - 1)(d - 1) \leq d^3 - 1$$

nodes at distance 3 or less from any given vertex.

Thus a maximal 4-tree of R must have at least r/d^3 vertices.

We show that there is no 4-tree of size $r \geq c \log m$, for some constant c , that survives with probability $1 - o(1)$.

This will imply that there is no surviving connected component of size $r \geq c \log m$, with probability $1 - o(1)$.

How many 4-trees of size $s = r/d^3$ are in G ?

Counting 4-trees

Lemma 3. *There are no more than md^{8r/d^3} 4-trees of size r/d^3 in G .*

Proof.

We can choose a “root” of the 4-tree in m ways.

A tree with root v is uniquely defined by an Euler tour that starts and ends at v and traverses each edge in the tree twice, once in each direction.

At each node in the 4-tree the tour can continue in up to d^4 ways.

Thus the number of 4-trees of size $s = r/d^3$ is bounded by

$$m(d^4)^{2s} = md^{8r/d^3}.$$

□

The probability that the nodes of each such 4-tree survive in G' is at most

$$((d + 1)2^{-k/2})^s = ((d + 1)2^{-k/2})^{r/d^3}.$$

Hence the probability that at the end of the first phase there is a connected component of size r is bounded by

$$\begin{aligned} & md^{8r/d^3} ((d + 1)2^{-k/2})^{r/d^3} \\ & \leq mk^{8r/d^3} 2^{8\alpha kr/d^3} (k2^{\alpha k - k/2 + 1})^{r/d^3} \\ & \leq m2^{(r/d^3)(9 \lg k + 9\alpha k - k/2 + 1)} \end{aligned}$$

$$\leq m2^{(rk/d^3)(9(\lg k)/k + 9\alpha - 1/2)} = o(1)$$

for $r \geq c \log m$, for a suitably large constant c and a sufficiently small constant α , and a suitably large k .