An Efficient Algorithm for 3-SAT

Cristian Dumitrescu

Abstract. In this article I describe an efficient, randomized algorithm (section 4) that solves the 3- SAT problem (known to be NP complete) with high probability, and a bit of the history of the problem under consideration. In the last section I present an interesting application, based on an idea that belongs to Godel.

Keywords. The Satisfiability Problem, Hamming distance, random walk with absorbing barriers, genetic algorithms.

Section 1. Useful notions that are used for the analysis of the algorithm.

A Boolean expression is said to be in conjunctive normal form (CNF) if it is of the form $E_1 \wedge E_2 \wedge E_3 \wedge \dots \dots .. \wedge E_k$, and each E_i, called a clause (or conjunct), is of the form $\alpha_{i1} \vee \alpha_{i2} \vee \alpha_{i3} \dots \dots .. \vee \alpha_{ir}$, where each α_{ij} is a literal, either x or $\urcorner$x, for some variable x.

A Boolean expression is said to be in disjunctive normal form (DNF) if it is of the form $F_1 \vee F_2 \vee F_3 \dots \dots .. \vee F_k$, and each F_j, called a clause (or disjunct), is of the form $\beta_{j1} \wedge \beta_{j2} \wedge \beta_{j3} \wedge \dots \dots .. \wedge \beta_{jr}$, where each β_{jk} is a literal, either y or $\urcorner$y, for some variable y.

A Boolean expression in CNF form is called satisfiable if there is some assignment of 0's and 1's to the variables that gives the expression the value 1.

The satisfiability problem is to determine, given a Boolean expression, whether it is satisfiable.

An expression is said to be 3 - CNF if each clause has exactly three distinct literals.

Theorem 1 (see reference [1]). L_{3SAT}, the satisfiability problem for 3 - CNF expressions, is NP - complete.

The Hamming distance $d_H(\mathbf{x}, \mathbf{y})$ between two vectors **x**, **y** is the number of components in which they differ. It is known that the Hamming distance $d_H(\mathbf{x}, \mathbf{y})$ satisfies the conditions for a metric.

Related to the theory of symmetric random walks (in one dimension), we have the following theorem.

Theorem 2 (see reference [2]). Limit theorem for first passages. For fixed t, the probability that the first passage through r occurs before epoch $t \cdot r^2$ tends to $P = \sqrt{\frac{2}{\pi}} \cdot \int_{\frac{1}{\sqrt{t}}}^{\infty} e^{-\frac{1}{2}s^2} ds = 2 \cdot \left(1 - N\left(\frac{1}{\sqrt{t}}\right)\right)$, as $r \to \infty$, where N is the normal distribution function. We note that when $t \to \infty$, then P tends to 1.

Section 2. The description of Schoning’s algorithm.

```
Input: a formula in 3-CNF with n variables.

Guess an initial assignment for the n variables, uniform at random.

Repeat 3n times:

    If the formula is satisfied by the actual assignment: stop and accept.

    Let C be some clause not being satisfied by the actual assignment.

    Pick one of the 3 literals in the clause at random, and flip its value in the
    current assignment.
```

Schoning proves (see reference [3]) that the complexity of k-SAT (with this algorithm) is within a polynomial factor of $\left(2 \cdot \left(1 - \frac{1}{k}\right)\right)^n$. This means that this algorithm does not have direct practical value, since the expected time needed to hit a solution grows exponentially with the number of variables.

Section 3. The WalkSAT algorithm, and GenWalkSAT, a genetic algorithm based on WalkSAT.

Description of the WalkSAT algorithm.

```
Input: a formula in 3-CNF with n variables.

Guess an initial assignment for the n variables, uniform at random.

Repeat A(n) times (where A(n) is polynomial):

    If the formula is satisfied by the actual assignment: stop and accept.

    For each variable, temporarily flip its value and count the number of clauses
    that are not satisfied in this case.

    Choose a variable, such that when its value is flipped temporarily, that leads
    to the minimum number of unsatisfied clauses in the 3-CNF expression. If
    several variables (when temporarily flipped) lead to the same number of
    unsatisfied clauses (even when this number is bigger than the current number
    of unsatisfied clauses), then choose one from them at random, and flip it
    permanently.
```

We note that this is a combination of RWalkSAT and hill climbing algorithm GSAT, which starts with a random assignment and repeatedly flips the value of the variable which results in the greatest increase in the number of clauses satisfied (or the greatest decrease in the number of clauses unsatisfied).

In order to present the genetic algorithm based on WalkSAT, we need the following definitions.

Definition 1. An assignment (basically, an n-bit binary string) α is fitter (higher performance) than an assignment β if α leaves a smaller number of unsatisfied clauses than β.

Definition 2. We are given two assignments α and β. Randomly generate an m less than n. Create a new assignment γ by taking m bits from α (chosen at random), and n - m bits from β (in the corresponding positions). The two assignments α and β will generate assignment γ. We say that γ is the result of crossover between α and β. We can consider a simplified version of this by always taking $= \frac{n}{2}$.

Description of GenWalkSAT, the genetic algorithm based on WalkSAT (in the following, A(n), B(n), and C(n) are polynomial functions).

Input: a formula in 3 – CNF with n variables.

Start with a population of A(n) randomly selected assignments (n – bit strings).

Repeat B(n) times:

> *If any of the assignments satisfy the 3 – CNF formula, then accept and stop.*
>
> *The parent assignments are paired, crossed and mutated, in order to produce offspring.*
>
> *From the total of* $A(n) + \binom{A(n)}{2} + C(n)$ *n-bit strings(parent assignments, or generated by crossover or mutation), select the fittest A(n) assignments (so the population stays constant), and repeat the cycle.*

We note that the fittest assignments leave the minimum number of unsatisfied clauses (following definition 1).

There are some challenging open problems related to genetic algorithms in general. Simulation results show that the limiting distribution entropy decreases with population size, which suggests that for large populations, the probability associated with nonoptimal states can be made as small as required. It is also an open question if the speed of the algorithm for finite time improves with crossover.

In our case, we have to find linear or at most polynomial functions A(n), B(n), and C(n), such that GenWalkSAT finds a solution in polynomial time with high probability.

Section 4. The proposed dual expression algorithm (DEA).

We also notice that 2-SAT can be solved in linear time (one of the proofs is based on theorem 2 above). We can then look at any clause E in a 3 – CNF expression of the form $x_i \vee x_j \vee x_k$, where each x_i is a literal, either x or ˥x, for some variable x. If we then write $z_{ij} = x_i \vee x_j$, then we can write E as $z_{ij} \vee z_{ik} \vee z_{jk}$. An unsatisfied clause has at most one of the z_{ij}'s set to 1. In a satisfied clause, written as $z_{ij} \vee z_{ik} \vee z_{jk}$, at least two of the z_{ij} 's are 1. That means that if I choose at random one of the z_{ij}'s, and flip its values from 0 to 1, assign it the value 1, then this is the right assignment with probability greater than $\frac{1}{2}$. This is an essential observation, because it might allow us to use theorem 2 in the analysis. Before the presentation of the algorithm, we need some notation. We write Test2CNF for the function that tests if a certain 2 – CNF expression has a solution (and we know that it works in linear time). We start with n variables x_i, with the negations ˥x_i, we have 2n symbols. That means that the z_{ij}'s must represent no more than $\binom{2n}{2}$ symbols. We will call the z_{ij}, the **dual variables**. When given a 3 – CNF expression, we can always write it with the help of the dual variables. In this form, we will call it the dual 3 – CNF expression (not to be confused with duality between CNF and DNF).

Here is the algorithm, the **dual expression algorithm (DEA)**:

Input: a formula in 3-CNF with n variables x_i.

*We write the corresponding dual 3 – CNF expression in the z_{ij} variables. There will be at most $\binom{2n}{2}$ z_{ij}- variables involved in the dual expression. We can form a binary vector with these z_{ij} - variables, call it the **dual vector**.*

Guess an initial assignment for the z_{ij}- variables , uniform at random.

Repeat A(n) times (where A(n) is polynomial discussed later):

> *Call the routine* Test2CNF for the conjunction of all the z_{ij} that are currently assigned value 1 (this will be a 2 – CNF expression in the x_i - variables).
>
> *If Test2CNF finds that this* conjunction of all the z_{ij}'s (that are currently assigned value 1) is satisfied, and if all clauses are satisfied, then stop and accept.
>
> *If Test2CNF finds that this* conjunction of all the z_{ij}'s (hat are currently assigned value 1) is satisfied, but not all clauses are satisfied, then find that first unsatisfied clause, and flip a random z_{ij} from that clause that currently has value 0 (change its value from 0 to 1), and update all the clauses where it appears.
>
> *If Test2CNF finds that this* conjunction of all the z_{ij}'s (that are currently assigned value 1) is not satisfied, then choose a random z_{ij} that is currently set to 1 and flip its value to 0.
>
> *Repeat cycle.*

The difference between this and Schoning's algorithm is that when we choose at random a z_{ij}- variable and assign it the value 1 (flip its value from 0 to 1), we are right with probability at least $\frac{1}{2}$, in other words, the probability of decreasing the Hamming distance between this dual vector and the possible dual vector solution is at least $\frac{1}{2}$. The polynomial A(n) can be taken as $C \cdot \binom{2n}{2}^2$, where C is a large constant (it cannot be greater than this number, but in practical terms, it will be, in fact, quadratic in the number of clauses). We do not have more than $\binom{2n}{2}$ z_{ij}- variable involved in the dual expression. I have not been able to find a suitable mathematical model for this algorithm (in terms of random walks or Markov chains, for example). The problem is that when Test2CNF finds inconsistency, it pushes the random walk one unit (in terms of the Hamming distance) away from the possible solution, with high probability, but if Test2CNF does not find inconsistency too often (towards the final stages it will not), the chain will hit the target in polynomial time with high probability. I leave this problem as a challenging problem, at the moment.

Section 5. Godel's letter to von Newmann.

For general implications, related to efficiently solving NP – complete problems, see [4]. An interesting application is related to the problem of automated theorem proving using an efficient algorithm for NP – complete problems.

We know that we can solve the following problem in polynomial time:

Given two well formed formulas α and β, in a given axiomatic system (like ZFC), is β a ZFC – proof of α?

Therefore, the following problem is in NP:

Given a formula α, and a number n, is there a ZFC – proof of size at most n for α?

Any efficient solution for NP-complete problems would make automated theorem proving a reality. We can have an automated system that would tell us (with probability as close to 1 as we want) that no solution to a given problem exists, that can be written in (for example) less than 10000 pages, or hit upon (find) such a proof.

In a letter in 1956, Godel asked John von Newmann whether there was a general method to find proofs of size n, using time that increases only as n or n^2. If such a method existed, Godel argued that this *" would have consequences of the greatest magnitude. That is to say, it would clearly indicate that the mental effort of the mathematician in the case of yes or no questions could be completely replaced by machines. One would indeed have to simply select an n so large that, if the machine yields no result, there would then also be no reason to think further about the problem. "*.

This is the main reason why I wrote this article. This is not just a problem of optimization, or applied mathematics. I think that this problem should be the focus of attention for the core of the mathematicians, a problem the solution of which could transform mathematics and fulfill (to some extent) Hilbert's dream, by following an idea that belongs to Godel.

Another interesting path is to consider quantum algorithms, quantum random walks, in particular, but we will not go into this issue here.

Conclusions. If we can settle the challenging problems above, then for all practical purposes, we can assume that P = NP, even if the conjecture P ≠ NP might be true, if we exclude randomized algorithms. This article can be considered a review article. The ideas expressed in section 4, the DEA algorithm, and section 3, the GenWalkSAT algorithm are original though. The main motivation for writing this article is in drawing the attention of pure mathematicians (not just people working in applied mathematics) to this important problem.

References:

[1]. J. E. Hopcroft, J. D. Ullman, *" Introduction to Automata Theory, Langiages, and Computation "*, Addison - Wesley Publishing Company, 1979.

[2]. W. Feller, *" An Introduction to Probability Theory and Its Applications "*, John Wiley & Sons, 1968.

[3]. Uwe Schoning, *" A probabilistic Algorithm for k-SAT and Constraint Satifaction Problems "*, Research Supported by the ESPRIT Basic Research, 1991.

[4]. L. Fortnow, *" The Golden Ticket, P, NP, and The Search For The Impossible "*, Princeton University Press, 2013.

Cristian Dumitrescu,
119 Young St., Ap. 11,
Kitchener, Ontario N2H 4Z3,
Canada.

Email: cristiand43@gmail.com
cristiand41@hotmail.com

Tel : (519) 574-7026