1 Definitions and Notations

proved by Chvatal and Reed [3].

Theorem 1.1 *For any random 2-SAT formula F with N variables and αN clauses, where N tends to infinity, F is satisfiable with probability $1 - o(1)$ for $\alpha < 1$, and F is unsatisfiable with probability $1 - o(1)$ for $\alpha > 1$.*

2 Local Heuristics

First of all, we are going to start by presenting the most common and yet the simpliest local heuristics, namely Pure Literal (PL) and Unit Clause (UC). Their behaviour can be described in terms of free and forced steps [1]. More precisely, forced steps are those decisions that cannot go wrong, that is we can satisfy a certain variable, while preserving the satisfiability of the formula. On the other hand, free steps are those decisions that do not have these guarantees.

The forced steps taken by the Pure Literal and Unit Clause are universally good, and that is why the combination of these two heuristics is a basic ingredient of much more elaborate heuristics. However, the more advanced make a better use of the free steps, which significantly boosts their performance, but makes them much more difficult to analyse.

2.1 Pure Literal

First, let's call a literal l *pure*, if there is no clause containing $\neg l$. Observe that if given CNF ϕ contains a pure literal l, then we can satisfy l in ϕ without changing the satisfiability of ϕ, because we delete all occurances of $var(l)$ from ϕ, which results in a less constrained formula. This observation is the driving factor behind the Pure Literal heuristic, which is presented in Algorithm 1.

Broder et al. [2] showed that the Pure literal almost surely succeeds in finding a satisfying assignment when the ratio of clauses to variables is $\alpha \leq 1.63$.

Algorithm 1: Pure Literal Heuristic

```
Input: A k-SAT formula φ
Output: True if φ is satisfiable, False otherwise
while Vars(φ) ≠ ∅ do
    if exists pure literal l then
        // Forced Step
        satisfy l in φ;
    else
        // Free Step
        x ← randomly chosen variable in φ ;
        Choose uniformly at random l ∈ {x, ¬x} and satisfy it in φ;
    end
end
if φ contains empty clause then
    return False
else
    return True
end
```

The above algorithm runs in time $O(Nk\alpha)$, as the existence of pure literal can be checked in $O(1)$ by maintaining for each variable x the number of clauses containing x and $\neg x$. Moreoever, pruning ϕ after satysfying a variable, amortizes to $O(k\alpha)$ per iteration.

2.2 Unit Clause

Simirarly to PL, UC exploits a very simple observation. Namely, if any clause C contains only one literal l, then any satisfying assignment to ϕ must satisfy l, otherwise it would contain an empty clause, yielding an unsatisfiable formula. The algorithm for UC is presented below:

Algorithm 2: Unit Clause Heuristic

```
Input: A k-SAT formula φ
Output: True if φ is satisfiable, False otherwise
while Vars(φ) ≠ ∅ do
    if φ contains unit clause {l} then
        // Forced Step
        satisfy l in φ;
    else
        // Free Step
        x ← randomly chosen variable in φ ;
        choose uniformly at random l ∈ {x, ¬x} and satisfy it in φ;
    end
end
if φ contains empty clause then
    return False
else
    return True
end
```

Similarly to PL, UC runs in time $O(Nk\alpha)$, beacause after satisfying a varaible x, the only new clauses are the ones that previously contained x. Therefore, the detection of new unit clauses amortizes to $O(k\alpha)$.

Analysis

Despite the simplicity of the UC heuristic, the analysis of its performance on random k-SAT formulas is far from trivial. Achlioptas [1] presents a very elegant framework for analysing the performance of a family of heuristics that includes UC. In the following section, we will present the framework by applying it to the UC heuristic on a random 4-SAT formula.

First, let's start with new notation. Let $C_i(t)$ denote the number of clauses in ϕ that contain i literals after t steps of UC algorithm. Similarly, let $V(t)$ denote the set of variables present in ϕ after t steps. Then, we can present a crucial lemma proved by Achlioptas [1] that is the centre of the whole framework:

Lemma 2.1 *For every $0 \leq t \leq n$, any clause after t steps contains a uniformly random set of i distinct non-complementary literals from $L(V(t))$.*

First, let's start with the observation that the algorithm fails only on forced steps, more precisely when ϕ contains $\{l\}$ and $\{\neg l\}$. One way, to guarantee that the algorithm does not encounter such

a situation, is to assert that on each iteration the expected number of unit clauses generated is less than 1. This way, in expectation, at each step there is at most one unit clause, hence a contradiction cannot be derived. Formally, if $\mathbb{E}[\Delta C_1(t)] = C_2(t)/(n-t) < 1-\gamma$, then w.h.p. UC succeeds

Analysing $C_2(t)$ on its own is a hard task. However, Achlioptas [1] proposes a clever idea of modelling $\Delta C_2(t)$ instead, together with the change in clauses of size 3, $\Delta C_3(t)$, and of clauses of size 4, $\Delta C_4(t)$. Here, if we condition on $C_2(t), C_3(t), C_4(t)$, then by Lemma 2.1 we get that:

$$\begin{aligned} \Delta C_4(t) &= -A \\ \Delta C_3(t) &= B - C \\ \Delta C_2(t) &= D - E \end{aligned} \tag{1}$$

where

$$\begin{aligned} A &= Bin(C_4(t), \frac{4}{n-t}) \\ B &= Bin(C_4(t), \frac{2}{n-t}) \\ C &= Bin(C_3(t), \frac{3}{n-t}) \\ D &= Bin(C_3(t), \frac{3}{2(n-t)}) \\ E &= Bin(C_2(t), \frac{2}{n-t}) \end{aligned} \tag{2}$$

In order to give a little bit of intuition behind the above equations, let's consider the case of $\Delta C_3(t)$. Suppose, we are satisfying literal l, then $C_3(t)$ is increased by the number of clauses which previously were of size 4 and contained $\neg l$, and decreased by the number of clauses of size 3 that contained l or $\neg l$. Note that the probability of the former is $\frac{2}{n-t}$, and the probability of the latter is $\frac{3}{(n-t)}$, yielding the equation for $\Delta C_3(t)$. The same reasoning applies to the other cases.

Now, we observe 2 things:

- The probability that $\Delta C_i(t)$ deviates from its expected value is small, due to the properties of Binomial distribution.
- $\Delta C_2(t)$ is smooth in t and $C_i(t)$ (k-Lipschitz smooth), that is small change in either t or $C_i(t)$ does not change significantly the value of $\Delta C_i(t)$

This observations hints that the values $C_i(t)$ do not diverge significantly from their mean trajectories, which can be rigorously proven using Wormland's Theorem. To do so, we consider $C_i(t)$ in a continuous setting, by defining $c_i(x) = C_i(xn)/n$ for $x \in [0,1]$. By doing so, $\Delta C_i(t)$ translates to $\frac{d}{dx}c_i(x)$, yielding a system of differential equations describing the evolution of $c_i(x)$. Moreover, here the second observation of smoothness of $\Delta C_i(t)$ can be formalized, i.e. we require that $\frac{d}{dx}c_i(x)$ is k-Lipschitz smooth, for some constant k. Having done that, we can finally apply Wormland theorem, that states that for any $\epsilon > 0$ the solution to the system of differential yields a trajectory of $c_i(x)$, that w.h.p. closely approximates the actual values of a random process for $x \in [0, 1-\epsilon]$.

This way, we obtain following equations for $C_i(t)$ when $t \leq (1-\epsilon)n$:

$$\begin{aligned} C_2(t) &= \frac{3}{2}\alpha(\frac{t}{n})^2(1-\frac{t}{n})^2 + o(n) \\ C_3(t) &= 2\alpha(1-\frac{t}{n})^3\frac{t}{n} + o(n) \\ C_4(t) &= \alpha(1-\frac{t}{n})^4 + o(n) \end{aligned} \tag{3}$$

Recall, that our fomrula is w.h.p satisfiable if $C_2(t)/(n-t) < 1-\gamma$, which implies if $\alpha < 4.5$, then UC within its first $n(1-\epsilon)$ iterations, will not encounter a contradiction. However, Wormland theorem does not tell us anything about $C_i(t)$ when $t > (1-\epsilon)n$. But after setting $\epsilon = 0.1$ and $t_e = \lfloor(1-\epsilon)n\rfloor$, we get that $C_2(t_e) + C_3(t_e) + C_4(t_e) < \frac{2}{3}(n-t_e)$. Therefore, after t_e steps, if we erase from every clause of size 3 a randomly chosen variable, and from every clause of size 4 a randomly chosen pair of variables, then we will obtain a random 2-CNF formula with ratio of clauses to variables less than $\frac{2}{3}$. This way, we can apply Theorem 1.1 to conclude that w.h.p. the formula is satisfiable.

On the other hand, if $\alpha > 4.5$, then $C_2(t)/(n-t) > 1+\gamma$, then again by Theorem 1.1 we conclude that w.h.p. the formula is unsatisfiable.

References

[1] Dimitris Achlioptas. Lower bounds for random 3-sat via differential equations. *Theoretical Computer Science*, 265(1):159–185, 2001. Phase Transitions in Combinatorial Problems.

[2] E. Upfal A.Z. Broder, A.M. Frieze. On the satis,ability and maximum satis,ability of random 3-cnf formulas. *Annual ACM-SIAM Symp. on Discrete Algorithms*, 4th, 1993.

[3] V. Chvatal and B. Reed. Mick gets some (the odds are on his side) (satisfiability). In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627, 1992.