1 Definitions and Notations

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 basic ingredient of much more elaborate heuristics. More precisely, the more advanced heuristics make the forced steps of PL and UC, but make a better use of the free steps.

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 $\phi$
Output: True if $\phi$ is satisfiable, False otherwise
while $Vars(\phi) \neq \emptyset$ do
    if exists pure literal $l$ then
        // Forced Step
        satisfy $l$ in $\phi$;
    else
        // Free Step
        $x \leftarrow$ randomly chosen variable in $\phi$ ;
        choose uniformly at random $l \in \{x, \neg x\}$ and satisfy it in $\phi$;
    end
end
if $\phi$ contains empty clause then
    return False
else
    return True
end
```

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 $\phi$
Output: True if $\phi$ is satisfiable, False otherwise
while $Vars(\phi) \neq \emptyset$ do
    if $\phi$ contains unit clause $\{l\}$ then
        // Forced Step
        satisfy $l$ in $\phi$;
    else
        // Free Step
        $x \leftarrow$ randomly chosen variable in $\phi$ ;
        choose uniformly at random $l \in \{x, \neg x\}$ and satisfy it in $\phi$;
    end
end
if $\phi$ contains empty clause then
    return False
else
    return True
end
```

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 random 4-SAT formula.

First, we present a crucial theorem that is the centre of the whole framework, which was proved by Chvatal and Reed [3].

Theorem 2.1 *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* $\alpha > 1$*.*

Above theorem implies that if at any point during the runtime of the algorithm the density of clauses of size 2 is more than 1, then the formula is w.h.p. unsatisfiable. Our analysis will focus on finding the ratio α, for which we can guarantee that at any point the density of underlying 2-SAT formula is less than 1.

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.