On Exponential Time Algorithm for k-SAT

Subhas Kumar Ghosh

Abstract. In this work we present and analyze a simple algorithm for finding satisfying assignments of k-**CNF** (Boolean formulae in conjunctive normal form with at most k literals per clause). Our work is motivated by a simple question: Are there any structural property of the k-**CNF** which could help us to understand if a formula accepts isolated assignment? And can we deterministically find such isolated assignment if formula has one? In this work we show such a property exists in almost all non-trivial k-**CNF** formula, and we call it rigidity of a clause. Informally, rigidity of a clause can be defined to be how well connected a clause is to other clauses having same literals. If we satisfy a rigid clause for most of its literals then some of the variables are forced to take fixed values. Since, we can force such property and still get a satisfiable assignment; we save some of the decision paths in the algorithm and reduce its time complexity. Our main lemma shows that the number of branches in a depth n decision tree for k-**CNF** will be at least $2^{n-\left(\frac{k^2}{2^{k^2}}\right)\left(\frac{1}{\beta}\right)\left(\frac{n}{k-\frac{n}{k\beta}}\right)}$, with, β, a constant depending on k. This is at least $2^{n-\epsilon_1\left(\frac{n}{k-\epsilon_2}\right)}$, where $0 < \epsilon_1 \leq 1$ and $0 < \epsilon_2 < 0$ are constants. As a result we obtain a deterministic algorithm for k-**SAT** with bound on running time approaching $\mathbf{O}(poly(n) \cdot 2^{n-n/k})$, when formula has large number of overlapping clauses.

Keywords. Computational and structural complexity, Satisfiability of Boolean formulae, Conjunctive normal form, Deterministic algorithm.

1. Introduction

The problem of finding a satisfiable assignment for a propositional formula in k-**CNF** (conjunctive normal form with at most k literals per clause) is notably the most important problem in theory of computation. The decision problem for $k \geq 3$ was one of the first problem shown to be **NP**-complete[1, 3]. The obvious algorithm can solve the problem in time poly$(n)2^n$ using exhaustive search, where, n is the number of variables in the formula.

In this paper we look at certain structural properties of the k-**CNF**, and its corresponding set of variable assignment. For $\sigma \in \{0,1\}^n$, we let σ_i denote the ith bit of σ, and by $x_i^{\sigma_i}$, we denote the ith variable of a formula F taking the ith bit of σ as its assigned value. We say $\sigma \in \{0,1\}^n$ satisfies F, iff by assigning the ith bit σ_i of σ to ith variable x_i of F makes $F(x_1^{\sigma_1}, x_2^{\sigma_2}, \ldots, x_n^{\sigma_n}) = true$, we denote it as $\sigma \vDash F$. If σ does not satisfy F, then we denote it as $\sigma \nvDash F$.

The study of exponential time algorithm for k-**CNF** is long and rich. We focus on worst case running time of algorithms which is provably better than poly$(n)2^n$. There exists several works that analyzes the structure of the assignment set of k-**CNF** and uses its structural properties to suggest better exponential time algorithm, see [6, 4, 7, 2, 5]. We briefly discuss the structural properties used in [6].

An assignment $\sigma \vDash F$ is isolated in the direction j if flipping the jth bit of σ makes σ no more a satisfiable assignment for F. An assignment of F, that satisfies F will be called i-isolated if it has

exactly $(n-i)$ neighbors which satisfies F. Observation made in [6] was, if an assignment is isolated in the direction i, then there exist at least one clause in F that is satisfied only by the ith bit of σ for the ith variable in corresponding ordering of variables in σ. Such clause is called critical for assignment σ. Consider a rooted tree for assigning values to the variables of F. In ith step we will select x_i and assign values in $\{0,1\}$, and branch on those two assignment. If we let the tree grow up to depth n then there will be all possible assignment at the bottom most leaves, and if F is satisfiable, some of the leaves will satisfy F. However, to figure-out which of these are indeed satisfying F one need to check all 2^n assignments. If one of the assignment is isolated in the direction i, then at level $i-1$ once we have assigned values to variables $x_1, \ldots, x_{i-1}$ a clause will appear that is still not satisfied under one of these partial assignments, and has only one literal corresponding to variable x_i left to be assigned a value, making the value of x_i fixed. If we can effectively figure out when this happens we can fix some of the variables and reduce the number of leaves at the bottom most level, and reduce the exponential factor in the running time of the exhaustive search algorithm. In general we do not know if F has isolated assignment. In [6] authors bound this property over all possible ordering of variables, and make an observation, that either there is an isolated assignment, or there are many assignments. If F has isolated assignment then we can save some time or if it has many assignments then one can get success by trying to choose one at random. Their randomized algorithm has $\mathbf{O}(poly(n)2^{n-n/k})$ bound on the running time. One of the open problem identified by authors of [6], is to find a deterministic algorithm for k-**SAT** that runs in time $\mathbf{O}(poly(n)2^{n-n/k})$, and mentions that additional insight into the structure of large set accepted by k-**CNF** will be necessary for that. Our work is motivated by a simple question: Are there any structural property of the k-**CNF** which could help us to understand if a formula accepts isolated assignment? And can we deterministically find such isolated assignment if formula has one? In this work we show such a property exists in almost all non-trivial k-**CNF** formula, and we call it *rigidity* of a clause. Informally, rigidity of a clause c can be defined to be how well connected a clause is to other clause having same literal. If a clause c shares many variables with other clauses, and we assign values to the variables of c, we will also satisfy some of the other clauses along with c that has same literals. Then this clause and its assignment becomes rigid if it is satisfying many other clauses, and can not be changed, however those clauses sharing negated literals with that of c are now forced towards few remaining variables and their assignments, and forced to become critical. In the remaining sections we establish this intuitively appealing idea and provide a deterministic algorithm for k-**SAT**. In many ways our algorithm can be seen as a derandomization of main algorithm in [6]. In [4, 5] the algorithm in [6] has been improved introducing bounded time proof resolution. In almost all papers the essential ingredient is a restart mechanism, where our method uses the internal structural property, rigidity of clause, which we introduce in this paper, and build the decision tree deterministically.

2. Background

2.1. Preliminaries

Throughout this paper we fix our alphabet as $\Sigma = \{0,1\}$. Let A be a set, then by $|A|$ we denote its cardinality, while for a string $x \in A$, $|x|$ denotes the length of string x. For $\sigma \in \{0,1\}^n$, we let σ_i denote the ith bit of σ, and by $x_i^{\sigma_i}$, we denote the ith variable of a formula F taking the ith bit of σ as its assigned value. We say $\sigma \in \{0,1\}^n$ satisfies F, iff by assigning the ith bit σ_i of σ to ith variable x_i of F makes $F(x_1^{\sigma_1}, x_2^{\sigma_2}, \ldots, x_n^{\sigma_n}) = \mathit{true}$, we denote it as $\sigma \vDash F$. If σ does not satisfy F, then we denote it as $\sigma \nvDash F$. The index i for the ith bit of an assignment are termed as coordinates of σ. Given two assignments $\sigma, \sigma' \in \{0,1\}^n$, the Hamming distance between σ and σ' is the number of coordinates in which σ and σ' are different.

Here we recall some of the basic definitions and terminologies from the graph theory. A graph G consists a set of vertices $V(G)$ and a set $E(G)$ of pairs of vertices. Elements of $E(G)$ are called edges of graph G. $u, v \in V(G)$ are called neighbors or adjacent iff $\{u,v\} \in E(G)$. The set of vertices

$\mathbf{N}(v)$ are called neighbors of $v \in V(G)$ iff $\forall u \in \mathbf{N}(v), \{u, v\} \in E(G)$. The degree of vertex $v \in V(G)$, $deg(v) = |\mathbf{N}(v)|$.

For a clause c in formula F, let $var(c)$ denote the set of variables that appear as literals, and also let $lit(c)$ denote the set of literals in c. $|var(c)|$ is the length of clause c, and $|F|$ is the sum of the length of all clauses of F. We will denote the set of clause in a formula F by $clause(F)$.

Definition 2.1. *A family of clause graph $\mathbf{G}_c$ can be defined as follows:*

$$V(\mathbf{G}_c) = \{c | c \text{ is a clause}\} \wedge \forall c_i, c_j \in V(\mathbf{G}_c), \{c_i, c_j\} \in E(\mathbf{G}_c) \Leftrightarrow var(c_i) \cap var(c_j) \neq \emptyset$$

A graph is said to be Δ-regular if $\forall v \in V(G), deg(v) = \Delta$. A subset $K \subseteq V(G)$, is a clique, if every vertex in K is neighbor of all other vertices in K. For a fixed positive integer $n, k \leq n$, let $\mathbf{G}_{n;k}$ denote the graphs in the family of *clause graph* $\mathbf{G}_c$, such that for each $c \in V(\mathbf{G}_{n;k})$, $var(c) \subset \{x_1, x_2, \ldots, x_n\}$, the set of n variables and $|var(c)| \leq k$. A k-**CNF** formula can be thought of a subgraph of $\mathbf{G}_{n;k}$. Let $\mathbf{F}_m$ be the family of formulas that chooses exactly m such vertices. For a formula $F \in \mathbf{F}_m$, we can talk about the induced subgraph $\mathbf{G}_{n;k,m}$ of $\mathbf{G}_{n;k}$ with m vertices and edges of $\mathbf{G}_{n;k}$ such that $\{c_i, c_j\} \in E(\mathbf{G}_{n;k,m})$ iff $c_i, c_j \in V(\mathbf{G}_{n;k,m})$ and $\{c_i, c_j\} \in E(\mathbf{G}_{n;k})$.

An assignment $\sigma \vDash F$ is isolated in the direction j if flipping the jth bit of σ makes σ no more a satisfiable assignment for F. An assignment of F, that satisfies F will be called i-isolated if it has exactly $(n-i)$ neighbors which satisfies F. Variable x_i corresponding to an isolated assignment in the direction i is called critical, and the clause that is satisfied only due to assignment of x_i is called critical clause. Let $[n] = \{1, 2, \ldots, n\}$, be n coordinates of a formula defined over n variables. We let extend the notion of $var()$ in two ways: for a collection of clauses $C = \{c_1, c_2, \ldots, c_\alpha\}$ of F, as $var(C) = \cup_{c \in C} var(c)$, and with respect to a set of coordinates $P \subseteq [n]$, as $var(c, P) = var(c) \cap (\cup_{i \in P} x_i)$.

Definition 2.2. *For any set of clauses $C = \{c_1, c_2, \ldots, c_\alpha\}$, with $|C| = \alpha$, a class $\hat{C} = \left\{\hat{C}_1, \ldots, \hat{C}_\beta\right\}$ with $\left|\hat{C}\right| = \beta$, is a* decomposition *of C iff $C = \cup_{1 \leq i \leq \beta} \hat{C}_i$ and for any $1 \leq i, j \leq \beta, i \neq j, var(\hat{C}_i) \cap var(\hat{C}_j) = \emptyset$. Let $P \subseteq [n]$, and let F be a k-**CNF**. Then we define a decomposition $\hat{C}$ of F with respect to P, iff $\hat{C}$ satisfies the following:*

1. $\hat{C} = \left\{\hat{C}_1, \ldots, \hat{C}_\beta\right\}$ *is a decomposition of set of all clauses of F.*
2. *There exists clauses $\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_\beta$ such that:*
   (a) $(\forall i : 1 \leq i \leq \beta) \left[\hat{c}_i \in \hat{C}_i\right]$
   (b) $(\forall i : 1 \leq i \leq \beta) \left(\forall c \in \hat{C}_i\right) [var(\hat{c}_i, [n] - P) \cap var(c, [n] - P) \neq \emptyset]$
   (c) $(\forall i, j : 1 \leq i, j \leq \beta, i \neq j) \left[var(\hat{C}_i, [n] - P) \cap var(\hat{C}_j, [n] - P) = \emptyset\right]$

For each class $\hat{C}_i$, $\hat{c}_i$ is a special clause appears as a maximum degree vertex, and we will call it a *root* of decomposition class $\hat{C}_i$. If $\hat{C}_i$ forms a clique (e.g. in the decomposition of F w.r.t $[n] - x_i$ for any variable x_i), then any vertex can be the root. Intuitively, for any $P \subseteq [n]$, a decomposition of F is a "view" of F from set of coordinates $[n] - P$.

Lemma 2.1. *Let $P \subseteq [n]$, then there exists a decomposition of F with respect to P, and such decomposition can be computed in time $\mathbf{O}(|F|^2 h)$, where $h = |[n] - P|$.*

Proof: Consider the following algorithm:

```
1: procedure DECOMPOSE(Input: F, P ⊆ [n])
2:     for all c_i ∈ F do                          ▷ let N(c_i) be c_i's neighborhood
3:         for all c_j ∈ N(c_i) do
4:             if var(c_i, [n] − P) ∩ var(c_j, [n] − P) = ∅ then
5:                 Remove edge between c_i and c_j
6:             end if
```

```
        end for
    end for
    for each clause c_i ∈ F in arbitrary order do
        Find the vertex with maximum degree in N(c_i) ∪ {c_i}
        This vertex with maximum degree will be ĉ_i and Ĉ_i = N(ĉ_i) ∪ {ĉ_i}
        if ∃c_j ∈ Ĉ_i such that ∃c_k ∈ N(c_j) and c_k ∉ Ĉ_i then
            Remove edge between c_j and c_k
        end if
    end for
end procedure
```

It is easy to observe that the algorithm above will compute the decomposition and running time of this procedure is at most $\mathbf{O}(h\,|F|^2)$. ■

Next we define the rigidity of a clause.

Definition 2.3. *For a clause c, we denote its rigidity by $\mathbf{R}(c)$, and define it as:*

$$\mathbf{R}(c) = \sum_{\forall c' \in clause(F), c \neq c'} |lit(c) \cap lit(c')|$$

With respect to a set of coordinates $P \subseteq [n]$, we define rigidity of a clause as

$$\mathbf{R}(c, P) = \sum_{\forall c' \in clause(F), c \neq c'} |lit(c) \cap lit(c') \cap (\cup_{i \in P}\{x_i, \neg x_i\})|$$

rigidity of a clause c can be defined to be how much connected a clause is to other clause having same literal. It is easy to see that, rigidity of a clause can be computed in time $\mathbf{O}(n\,|F|)$.

A k-**CNF** formula has clauses of at most k literals. Since we assume that all literals in any clause are of distinct variables, if $\sum_{c \in clause(F)} |var(c)| > n$, then there exists a variable in F, which has been used in more than $\left\lfloor \left(\sum_{c \in clause(F)} |var(c)| - n \right) / n \right\rfloor$ many clauses.

Lemma 2.2. *If the number of clause in F is m then there exists at least one clause with rigidity r, and*

$$\left\lfloor \frac{mk - n}{n} \right\rfloor \geq r \geq \left\lfloor \frac{\left(\sum_{c \in clause(F)} |var(c)| - n \right)}{2n} \right\rfloor \geq \left\lfloor \frac{(m-1) + k - n}{2n} \right\rfloor$$

Proof: Lower bound is considering only one clause of length k and all others are of only one literal. Also note that the $2n$ factor in denumerator comes from the fact that in a group of clause sharing literals of same variable, rigidity due to one literal is least when group could be splitted to equal halves. The upper bound comes from considering all clauses are of length k, and they form clique while sharing same literals. ■

2.2. Outline of Proof

If we compute the decomposition of F with respect to $P \subseteq [n]$, $\hat{C} = \left\{ \hat{C}_1, \ldots, \hat{C}_\beta \right\}$, then we get a collection of clauses in each decomposition class $\hat{C}_i$ with a root clause $\hat{c}_i$ with certain rigidity. Let us concentrate on one such class $\hat{C}_i$ with a root clause $\hat{c}_i$. Let $\alpha = \{x_{\alpha_1}, x_{\alpha_2}, \ldots, x_{\alpha_m}\}$ be the set of variables that are present in the root, such that all $c \in \hat{C}_i$ are connected to $\hat{c}_i$ for one or more variables in α. If we make a partial assignments to the variables in $\alpha' \subseteq \alpha$ such that all the clauses that are contributing to the rigidity of $\hat{c}_i$ are satisfied, along with $\hat{c}_i$, then: clauses that are connected to $\hat{c}_i$ for variables only in $\alpha - \alpha'$, or, clauses that has only negated literals that of $\hat{c}_i$ are forced to be satisfied for variables not in α'. This way we might be able to make some of the clause critical for an assignment, but assigning values to remaining variables might not lead to a satisfiable assignment.

Thus, question will be to what extent we can "design" such forced isolated assignments? Here we state our key observations, we omit the proof as they are quite direct:

Observation 1. *In one satisfiable assignment a clause can not be critical for more than one variable* [6].

Observation 2. *If σ' and σ'' are two partial assignments of variable set α, with Hamming distance between σ' and σ'' is at least 1. If σ' forces a clause c to become critical and c has a literal for a variable in α then σ'' satisfies c.*

Observation 3. *If σ' is a partial assignments of variable set α, forcing a clause c to become critical for variable x_i, and assignment $x_i^{\sigma_i}$ satisfies c, then σ' and σ_i can be part of a $\sigma \vDash F$, iff, there is no other clause that is critical for x_i and satisfied with $x_i^{1-\sigma_i}$.*

Observation 4. *Under a partial assignment of F, a clause and corresponding variable becomes critical under a ordering of assignments, the same clause may not be critical under a different ordering of assignments.*

Our main technical result can be seen the following way. Every, non-trivial k-**CNF** has a rigid clause. Otherwise, consider a decomposition of F w.r.t $[n]-i$, for any coordinate i. The decomposition obtained is a clique K with r vertices. We can have r odd (or even) and rigidity of every clause can be $\lfloor r/2 \rfloor$ in worst case, and formula is trivially satisfiable (or unsatisfiable). Now consider a k-**CNF** with a rigid clause, and an algorithm for finding its assignment. We form a decision tree by assigning values to variables of F. We assign values to one variable in each step and branch on two possible assignments and form a labeled rooted tree of depth n with 2^n branches. Our main lemma shows that the number of branches in a depth n decision tree be at least $2^{n-\left(\frac{k^2}{2^{k^2}}\right)\left(\frac{1}{\beta}\right)\left(\frac{n}{k-\frac{n}{k\beta}}\right)}$, with, β, a constant depending on n and k. This is at least $2^{n-\epsilon_1\left(\frac{n}{k-\epsilon_2}\right)}$, where $0 < \epsilon_1 \leq 1$ and $0 < \epsilon_2 < 0$ are constants depending on k. As a result we obtain a deterministic algorithm for k-**SAT** with bound on running time approaching $\mathbf{O}(poly(n) \cdot 2^{n-n/k})$, when formula has large number of overlapping clauses..

3. Main Lemma

In this section we obtain a bound on the number of branches of depth n decision tree, where we deterministically force few variables to become critical by satisfying a rigid clause. While we do force a set of clause to become critical we must ensure that if the formula is satisfiable we must obtain such satisfiable assignment. First, we show that behind every i-isolated assignment there is a rigid clause.

Lemma 3.1. *Let σ be a satisfiable assignment of F isolated in direction i, then there is rigid clause in F.*

Proof: Let σ be a isolated satisfiable assignment of F for variable x_i and clause c_i. Let $C = \{c_{\alpha_1}, c_{\alpha_2}, \ldots, c_{\alpha_m}\}$ be set of other clauses in F having one or more variables in $\beta = var(c_i) - x_i$. Since σ is a satisfiable assignment of F, a subset $C' \subseteq C$ of clause are satisfiable for variables in β, and has negated literals that of c_i. There is a decomposition of F where clause of C' form a decomposition class with a rigid root. ■

Before we could present our main result on decision tree encoding, we need to present some properties of root of decomposition. Unless indicated, we will say the maximum rigid root as the root of decomposition.

Fact 3.1. *Let $\hat{C}_i$ be a decomposition class in the decomposition of F with respect to $P \subseteq [n]$. Let $P' \subset P$, then there exists a decomposition class $\hat{C'}_i$ in decomposition of F with respect to P' such that $\hat{C'}_i \subseteq \hat{C}_i$.*

Proof: Follows from the fact that $([n] - P) \subset ([n] - P')$ ■

Fact 3.2. *Let $P_1, P_2 \subseteq [n]$ and $P_1 \cap P_2 = \emptyset$. Then following holds:*

- *$|P_1| + |P_2| > k$ and there is no clause c that appear as a root in all P_1, P_2 and $P = P_1 \cup P_2$*
- *There is a clause c that appear as a root in all P_1, P_2 and $P = P_1 \cup P_2$, then $2 \leq |P_1| + |P_2| \leq k$*

Proof: Follows from the fact that there is no clause of length greater than k and $P_1 \cap P_2 = \emptyset$. ■

Combining the above three facts we claim that, if F has isolated solutions, then there always exists $P_k \subset P_{k-1} \subset \ldots \subset P_1 \subseteq [n]$ and $|P_k| \geq 1, |P_1| \leq k$, such that the rigidity of a root is non-decreasing over $P_k \subset P_{k-1} \subset \ldots \subset P_1 \subseteq [n]$, we will say c is monotonic root over $[P_1, P_k]$. Intuitively, it is easy to see that if $P_{i-1} \subset P_i$ and c is a root in P_{i-1} then introducing more variable in P_i will at least keep is rigidity same.

We now present an algorithm to encode and reduce the size of a decision tree. Let T be a deterministic labeled rooted decision tree. Root of T is labeled with F. Let $F \restriction_{x_1^{\sigma_1}, x_2^{\sigma_2}, \ldots, x_i^{\sigma_i}}$ denote a partial assignment of variables of F. In step i we extend a internal vertex of the tree labeled in step $i-1$ with partial assignment $F \restriction_{x_1^{\sigma_1}, x_2^{\sigma_2}, \ldots, x_{i-1}^{\sigma_{i-1}}}$, with two branches as assignment x_i^0 and x_i^1. There can be at most 2^{i-1} vertices after step $i-1$, thus if we let extend all of them we will have 2^i vertices after step i. We will show in steps how we can use the existence of a rigid clause to fix the assignment of some variables, either as x_i^0 or x_i^1, and obtain lesser number of vertices after step i. In sequel we first provide an algorithm for encoding a decision tree.

- W.o.l.o.g. for the procedure, let $x_1, x_2, \ldots, x_n$ be the ordering of the assignments of variables in the tree over all possible permutation π of the set $[n]$. Let T be the decision tree of depth n, with vertices t at level i labeled with $F \restriction_{x_1^{\sigma_1}, x_2^{\sigma_2}, \ldots, x_i^{\sigma_i}}$, and having 2^i such vertices in each level $1 \leq \forall i \leq n$.
- Let γ be set of variables for which assignment σ_γ has been fixed till step $i-1$ as they have been forced to become critical, we will call γ a cover of $[n]$. Thus, when we start $\gamma = \emptyset$.
- In step i we are considering the set of variables $\alpha_i = \{x_1, x_2, \ldots, x_i\}$, thus we check which of these variables are already being assigned a value, and compute $DECOMPOSE(F, [n] - \alpha)$, where $\alpha = \alpha_i - \gamma$, the remaining unassigned variables in $\alpha_i = \{x_1, x_2, \ldots, x_i\}$.
- After computing decomposition, we will try to fix the assignments of roots. Let $\hat{C} = \left\{\hat{C}_1, \ldots, \hat{C}_\beta\right\}$ be decomposition of set of all clauses of F with $\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_\beta$ as roots. Let $\sigma_{\alpha'} = \sigma_{\hat{c}_1}, \sigma_{\hat{c}_2}, \ldots, \sigma_{\hat{c}_\beta}$, with $\sigma_{\hat{c}_j} = \sigma_{\hat{c}_{j_1}} \sigma_{\hat{c}_{j_2}} \ldots \sigma_{\hat{c}_{j_p}}$ for some p be such that each $\sigma_{\hat{c}_j}$ is a partial assignment of root $\hat{c}_j, 1 \leq j \leq \beta$, and this partial assignment satisfies all those literals of root $\hat{c}_j$, that contributes to $\hat{c}_j$'s rigidity. Let $\alpha' \subseteq \alpha$ be corresponding variables for this partial assignment. Also note that there will not be any conflict in partial assignment as decomposition classes are disjoint over variable set α.
- Now in each decomposition class determine if there is a critical clause forced by assignment $\sigma_{\alpha'}$. Let c_p be critical for variable x_p in class $\hat{C}_p$. Check decomposition of F w.r.t. $[n] - p$. This decomposition can have a single clause c_p, or a clique K with r vertices. If it is a clique and rigidity of c_p is less than $r-1$, then do not force the assignment of $\hat{c}_p$, and remove $\sigma_{\hat{c}_p}$ from $\sigma_{\alpha'}$. We continue to do this in the while loop. But notice that this loop will run for at most n.
- clearly the algorithm runs in time polynomial in n and $|T|$.

1: **procedure** ENCODE(Input: F, T)
2: $\gamma = \emptyset$
3: **for** $1 \leq i \leq n$ **do**
4: $\alpha = \alpha_i - \gamma$
5: $DECOMPOSE(F, [n] - \alpha)$
6: $ASSIGN$ values to root such that this partial assignment satisfies all those literals of each root, that contributes to root's rigidity
7: $UPDATE$ γ with new variables $\alpha' \subseteq \alpha$ fixed

8: **while** There are variables that become critical **do**
9: Let x_p be the critical variable
10: **if** x_p is not conflicting **then**
11: $\gamma = \gamma \cup \{x_p\}$
12: **else**
13: REMOVE those variables in α' from γ which made x_p critical
14: **end if**
15: **end while**
16: **for** each vertex $t \in T$ in at level i **do**
17: **if** Assignment in t for variables in γ does not match with σ_γ **then**
18: REMOVE that sub-tree rooted at t
19: **end if**
20: **end for**
21: **end for**
22: **return** encoded T
23: **end procedure**

First we will show the correctness of the algorithm above. We need few notions. Under a forced assignment of the roots of decomposition of F, in every step of the algorithm we fix the assignment of a set of variables. This also forces some variables to become critical. In the procedure above γ is the set of variables, whose assignment is fixed in every step. We will call γ a cover of $[n]$. If $|\gamma| = d$ then we will call γ a d-cover of $[n]$. Since we fix the assignment of all variables in a cover, if the procedure above produces a d-cover of $[n]$, then number of branches in the encoded tree are at most 2^{n-d}. For the procedure to be correct, it is sufficient to prove that in any step of the algorithm, there is no clause having variables only in cover γ and is unsatisfied by assignment σ_γ. We will call such a cover γ a good cover of $[n]$.

Claim 3.2. *In the procedure above γ is always a good cover.*

Proof: We will prove it by induction on the invariant that γ is always a good cover. If γ is a good cover in step 0 (which is true as $\gamma = \emptyset$, and by induction hypothesis we assume that it is a good cover in step $i-1$, then it is sufficient to show that after step i it still remains a good cover.

Let us assume that in step $i-1$ we consider the set of variables $\alpha_{i-1} = \{x_1, x_2, \ldots, x_{i-1}\}$ and in step i we consider set of variables $\alpha_i = \{x_1, x_2, \ldots, x_i\}$, and $\alpha_i - \alpha_{i-1} = \{x_i\}$. Let γ after step $i-1$ be such that $x_i \notin \gamma$ and $\gamma \subseteq \alpha_{i-1}$. It may happen that γ is not a subset of α_{i-1} or α_i and it has few more variables outside α_i. However, this assumption on γ holds for the proof, because, the only way there can be a variable in γ which is outside α_i is that there exists a clause having variable in $\alpha_{i-1} \cap \gamma$ for which is is not satisfied and remaining one variable was critical and was forced by the algorithm.

The way cover γ is expanded at step i is by fixing the value of x_i to satisfy a root with x_i as a variable contributing to its rigidity, and if this makes another variable x_j critical, then we fix the assignment of variable x_j as well.

By assumption x_i is not in γ(otherwise, it is part of γ after step $i-1$ and claim follows). Thus x_i can be fixed as long as it does not force another variable x_j to become critical for more than one clause with conflicting literals (having x_j as literal in one clause and $\neg x_j$ in another and for both x_j is the only remaining critical variable to be assigned). This condition is checked in the while loop (line 10). ■

We have defined $|var(c)|$ as the length of a clause c. We will define a function that captures the distribution of clauses of different length in a formula F.

Definition 3.1. *Let $N_l(F)$ be the number of clauses of length l in F. For k-**CNF**, $N_0(F) = 0$ and $N_{k+1}(F) = 0$, and $N_l(F)$ has a distribution over $1 \leq l \leq k$, with $\sum_{1 \leq l \leq k} N_l(F) = m$, the number of clauses in F.*

Let us define another function that captures the distribution of the weight of a formula F over $[n]$ contributed by various clauses.

Definition 3.2. *Let $\omega_i(F)$ denote the number of clauses that has variable x_i, under a fixed ordering of the variables. $\omega_i(F) = \sum_{\forall c \in clause(F)} |var(c) \cap \{x_i\}|$, let $\mu_\omega = 1/m \sum_{i\in[n]} \omega_i(F)$.*

Clearly, $\sum_{i\in[n]} \omega_i(F) = \sum_{1\leq l\leq k} l \cdot N_l(F)$. We next prove our main lemma.

Lemma 3.3. *Let F be a satisfiable k-**CNF**, and F has a rigid monotonic root over $[P, P'], P \subseteq [n], P' \subset P$ with rigidity r, such that P' is a good cover. Then there exists a cover of size at most*

$$\sum_{i=1}^{|P-P'|} \left[(\mu_\omega)^3 \cdot \left(\frac{1}{4rn^2}\right) \cdot \left(\frac{|P'|+i}{n}\right)^{k^2} \cdot \left(\frac{n\,(n-|P'|-i)}{(|P'|+i)^2}\right)^k \right]$$

over $[P, P']$.

Proof: Let $N_l(F)$ be the distribution of clauses of different length in a formula F over $[1, k]$, and let $\omega_i(F)$ denote the number of clauses that has variable x_i, under a fixed ordering of the variables over $[1, n]$. Also let p_i be the indicator binary random variable for ith variable being fixed with $p_i = 1$ if variable being fixed $p_i = 0$ otherwise. Then

$$\mathbf{E}\,(|\gamma|) = \mathbf{E}\left(\sum_{i\in P-P'} p_i\right) = \sum_{i\in P-P'} \mathbf{E}\,(p_i) \tag{1}$$

$\mathbf{E}\,(p_i) = \mathbf{Pr}\,[p_i = 1]$, and it will be sufficient for us to give a bound on this. There are two different events that fix the assignment of a variable. Either, it happens when we force the assignment of a root (we shall indicate this by random variable $p_{\mathbf{forced}}$), or it becomes critical(we shall indicate this by random variable $p_{\mathbf{critical}}$). Then we need to compute, assuming $p_{\mathbf{forced}}$ and $p_{\mathbf{critical}}$ are disjoint:

$$\mathbf{Pr}\,[p_i = 1] \leq \mathbf{Pr}\,[p_{\mathbf{forced}}] + \mathbf{Pr}\,[p_{\mathbf{critical}}] \tag{2}$$

First, we will compute $\mathbf{Pr}\,[p_{\mathbf{critical}}]$. Let γ' denote the cover in P'. We need to compute over all clause having variable only in $\{x_i\} \cup \gamma'$, that are unsatisfied by γ', and the remaining variable is x_i, which does not conflict. To simplify farther consider $|\gamma'| = |P'| = q$, then

$$\mathbf{Pr}\,[p_{\mathbf{critical}}] = \sum_{l=1}^{k} \frac{lN_l(F)}{2mn} \left(\frac{q}{n}\right)^{l-1} \tag{3}$$

To compute $\mathbf{Pr}\,[p_{\mathbf{forced}}]$, we want to find out the probability that we can force the value of x_i. Which is the probability of fixing the bit x_i while there exists a rigid clause with rigidity r, and forcing the value of x_i does not create any conflicting critical variable. Since γ' is already a good cover we look for clauses that has at least one variable outside the cover and that variable is x_i, and it does not force another variable x_j to become critical, which is conflicting. This gives us:

$$\mathbf{Pr}\,[p_{\mathbf{forced}}] = \left[\sum_{l=1}^{k} \frac{lN_l(F)}{rmn} \left(\frac{n-1}{n}\right)^{l-1}\right] \cdot \left[\sum_{l=1}^{k} \frac{lN_l(F)}{2} \left(\frac{q+1}{n}\right)^{l-1} \left(\frac{n-q-1}{n}\right)\right] \tag{4}$$

After simplification, we get,

$$\mathbf{Pr}\,[p_i = 1] \leq \left[\sum_{l=1}^{k} (lN_l(F)/m)^3\right] \cdot \left(\frac{q}{n}\right)^{k^2} \cdot \left(\frac{n\,(n-q)}{q^2}\right)^k \cdot \left(\frac{1}{4rn^2}\right) \tag{5}$$

Since the term $\frac{lN_l(F)}{m}$ is really the average weight distribution we can take $\left[\sum_{l=1}^{k} \left(\frac{lN_l(F)}{m}\right)^3\right]$, as a constant for a formula, and we denote it by μ_ω^3.

$$\mathbf{Pr}\,[p_i = 1] \leq (\mu_\omega)^3 \cdot \left(\frac{q}{n}\right)^{k^2} \cdot \left(\frac{n\,(n-q)}{q^2}\right)^k \cdot \left(\frac{1}{4rn^2}\right) \tag{6}$$

and,

$$\mathbf{E}\left(|\gamma|\right) \leq \sum_{i=1}^{|P-P'|} \left[\left(\mu_\omega\right)^3 \cdot \left(\frac{1}{4rn^2}\right) \cdot \left(\frac{|P'|+i}{n}\right)^{k^2} \cdot \left(\frac{n\left(n-|P'|-i\right)}{\left(|P'|+i\right)^2}\right)^k \right] \tag{7}$$

■

Corollary 3.4. *Every non-trivial satisfiable* k-**CNF**, *over* n *variables and* m *clauses has a decision tree with at least*

$$2^{n-\left(\frac{k^2}{2^{k^2}}\right)\left(\frac{1}{\beta(n)}\right)\left(\frac{n}{k-\frac{n}{k\beta(n)}}\right)} \tag{8}$$

branches, with, $\beta = N_l(F)$ *for all* $1 \leq l \leq k$.

Proof: In the main lemma 3.3, considering $|P'| = 0$, and summing over 1 to n, we get,

$$|\gamma| \leq \left(\mu_\omega\right)^3 \cdot \left(\frac{1}{r}\right) \cdot 2^{-k^2+k-2} \tag{9}$$

and by lemma 2.2, that there exists a rigid clause with rigidity $r \leq \frac{mk-n}{n}$, we get:

$$|\gamma| \leq \left(\mu_\omega\right)^3 \cdot \left(\frac{n}{mk-n}\right) \cdot 2^{-k^2+k-2} \tag{10}$$

with, $\beta = N_l(F)$ for all $1 \leq l \leq k$, and $m = k\beta$ and $\sum_{l=1}^{k} lN_l(F)/m = (k+1)/2$,we get,

$$|\gamma| \leq \left(\frac{k^2}{2^{k^2}}\right)\left(\frac{1}{\beta}\right)\left(\frac{n}{k-\frac{n}{k\beta}}\right) \tag{11}$$

and there exists constant $0 < \epsilon_1 \leq 1$ and $0 < \epsilon_2$ such that

$$|\gamma| \leq \epsilon_1 \left(\frac{n}{k-\epsilon_2}\right) \tag{12}$$

Thus, decision tree will have $2^{n-|\gamma|} = 2^{n-\epsilon_1\left(\frac{n}{k-\epsilon_2}\right)}$ branches. ■

4. Algorithm

In this section we provide a deterministic algorithm for determination of a satisfiable solution for a k-**CNF** formula. We start with F as a root, and form a tree in number of steps. In step i we consider ith variable. In this step we expand each node of the tree labeled by variable $\{x_1, x_2, \ldots, x_{i-1}\}$ to $\{x_1, x_2, \ldots, x_{i-1}, x_i\}$ such that x_i is the ith variable in this ordering. We will use the procedure $ENCODE$ to reduce the number of possible assignments at each step. Note, we are considering one ordering of the variables, and by main lemma 3.3 we will have at least $2^{n-\left(\frac{k^2}{2^{k^2}}\right)\left(\frac{1}{\beta(n)}\right)\left(\frac{n}{k-\frac{n}{k\beta(n)}}\right)}$ branches. Problem is what will be the ordering of variables that will ensure that the algorithm reaches running time close to expected by corollary-3.4? We will consider the following: consider decomposition of F with respect to $[n]-i, \forall i \in [n]$, each decomposition will be a clique. Let x_i be the variable corresponding to which we get a maximum rigid clause. In next step consider $\{x_i, x_j\}$, for all $x_j \neq x_i$, and again take the maximum rigid clause in the decomposition of F w.r.t $[n]-i \cup j$ for all $j \in [n]-i$. Continuing this way in n steps we will have an ordering of variables such that we start with maximum rigid clause in every step of the algorithm in the remaining set of unassigned variables.

Claim 4.1. *If a formula F has a satisfiable assignment which is isolated in the direction $P \subseteq [n]$, and $|P|$-isolated assignment is the maximum isolated assignment F can have, then above ordering of variables will ensure that algorithm will always find the maximum isolated assignment, and the constructed tree will have minimum number of branches.*

Sketch of proof: It indeed follows from the $ENCODE$ procedure that if we force the assignment of maximum rigid roots then size of the cover is bigger at a higher levels of the algorithm and will have the least number of branches, also from lemma 3.2 we can show that a good maximal cover will ensure the maximal isolated assignment as a leaf node. ■

The algorithm thus has two phases. Phase one creates the ordering of the variables as described above and phase two builds the decision tree while encoding the constructed tree in every step. Note that in corollary 3.4 we choose $N_l(F)$ to be uniform. This gives us the bound stated with β depending only on k. Our observation is that the complexity of k-**CNF** is closely related to the distribution $\omega_i(F)$ and $N_l(F)$, the distribution of weight over variables and distribution of clauses of various lengths. These two distribution governs the structural property we define here as rigidity, and our claim is rigidity implies isolated assignments. Deeper analysis of these factors will be our direction of future works after this preliminary report.

References

[1] S. A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[2] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-**SAT** based on local search. *Theor. Comput. Sci.*, 289(1):69–83, 2002.

[3] L. Levin. Universal'nyie perebornyie zadachi (universal search problems: in russian). *Problemy Peredachi Informatsii, English translation in* [8], 9(3):265–266, 1973.

[4] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-sat. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 628, Washington, DC, USA, 1998. IEEE Computer Society.

[5] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005.

[6] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, page 566, Washington, DC, USA, 1997. IEEE Computer Society.

[7] U. Schöning. A probabilistic algorithm for k-**SAT** and constraint satisfaction problems. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 410, Washington, DC, USA, 1999. IEEE Computer Society.

[8] B. A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

Subhas Kumar Ghosh
Honeywell Technology Solutions Laboratory
151/1, Doraisanipalya, Bannerghatta Road,
Bangalore, India, 560076
e-mail: `subhas.kumar@honeywell.com`