An Exact Algorithm for the Boolean Connectivity Problem for k-CNF[☆]

Kazuhisa Makino

Graduate School of Information Science and Technology, University of Tokyo

Suguru Tamaki

Graduate School of Informatics, Kyoto University

Masaki Yamamoto

Dept. of Informatics, Kwansei-Gakuin University

Abstract

We present an exact algorithm for a PSPACE-complete problem, denoted by CONNkSAT, which asks if the solution space for a given k-CNF formula is connected on the n-dimensional hypercube. The problem is known to be PSPACE-complete for $k \geq 3$, and polynomial solvable for $k \leq 2$ [6]. We show that CONNkSAT for $k \geq 3$ is solvable in time $O((2-\epsilon_k)^n)$ for some constant $\epsilon_k > 0$, where ϵ_k depends only on k, but not on n. This result is considered to be interesting due to the following fact shown by [5]: QBF-3-SAT, which is a typical PSPACE-complete problem, is not solvable in time $O((2-\epsilon)^n)$ for any constant $\epsilon > 0$, provided that the SAT problem (with no restriction to the clause length) is not solvable in time $O((2-\epsilon)^n)$ for any constant $\epsilon > 0$.

Keywords:
exponential-time algorithms, Boolean connectivity, CNF satisfiability

[☆]A preliminary version of this paper appeared in the Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT 2010), LNCS 6302, pp. 172–180, 2010.

Email addresses: makino@mist.i.u-tokyo.ac.jp (Kazuhisa Makino), tamak@kuis.kyoto-u.ac.jp (Suguru Tamaki), masaki.yamamoto@kwansei.ac.jp (Masaki Yamamoto)

1. Introduction

There are so many NP-hard problems around the world, which are considered to be intractable. To deal with those intractable problems, efficient algorithms with good approximation ratio or working well on average, have been proposed. Another approach to dealing with intractable problems is to develop algorithms that exactly solve the problems, so-called *exact algorithms*, where exact algorithms usually run in super-polynomial time, but exponentially faster than trivial ones. See [20, 17] for surveys on this topic. A number of exact algorithms for typical NP-hard problems have been proposed, and novel techniques for bounding the running time have been found: E.g., [1, 7, 2] for the traveling salesman problem, [4, 9, 3] for graph partitioning problems such as the graph coloring problem, and [14, 13, 16, 8, 15] for the satisfiability problem.

Viewing this approach in terms of computational complexity, we are concerned with the following question: Given an NP-hard problem of solution length n or witness length n, (for example, n denotes the number of vertices of a graph for the traveling salesman problem, or the number of variables of a formula for the satisfiability problem) is there an exact algorithm for the problem in time $O(2^n)$, or $O((2-\epsilon)^n)$ for some constant $\epsilon > 0$? Here, we assume that the length of instances with solution length n or witness length n is bounded by a polynomial in n. Moreover, as usual, we omit the polynomial factor in the O-notation when concerning with an upper bound of exponential-time.

The oldest result for this kind of questions is for the traveling salesman problem by Bellman [1] and by Held and Karp [7]. Given an undirected graph $G = (V, E)$ and a length function $\ell : E \to R_+$, the problem asks for finding a shortest Hamilton cycle. It is easy to see that the problem is solvable in time $O(n!)$. However, it is indeed not so easy to see that it is solvable in time $O(2^n)$. These two papers [1, 7] gave an affirmative answer to this question. There are several results that give such an affirmative answer: for example, [4, 9] for the k-coloring problem showed that it is solvable in time $O(2^n)$ for any k (not necessarily constant) while it is trivially solvable in time $O(k^n)$, and [19] for the maximum satisfiability problem where the clause length of an instance is at most two showed that it is solvable in time $O(1.731^n)$ while it is trivially solvable in time $O(2^n)$.

One of the most notable questions of this kind, which are still open, is for the satisfiability problem (SAT). This problem asks if there is a satisfying

assignment for a given conjunctive normal form (CNF) formula φ with *no* restriction to the clause length. It is clear that the problem is solvable in time $O(2^n)$. However, it is still open whether it is solvable in time $O((2-\epsilon)^n)$ for some constant $\epsilon > 0$. Another well-known open question is to ask whether the traveling salesman problem is solvable in time $O((2-\epsilon)^n)$ for some constant $\epsilon > 0$.

While developing exact algorithms for NP-complete problems and their optimization problems, we *rarely* see exact algorithms for decision problems in complexity classes beyond NP, such as the second and higher levels of PH, PSPACE, EXP, etc. There is one exceptional problem as far as we know: the quantified Boolean formula (QBF) problem, that is a typical PSPACE-complete problem, even if given Boolean formulas are restricted to 3-CNF. Williams [18] proposed an exact algorithm for this problem. However, he analyzed the running time with respect to *the number of clauses*, but not the number of variables. (Apart from decision problems, there are several problems solvable in time $O((2-\epsilon)^n)$, e.g., #k-SAT problem, which is #P-complete for $k \geq 2$. For this problem, we easily obtain an $O((2-\epsilon))^n$-time exact algorithm, by using a simple backtracking algorithm for k-SAT. The best upper bound for #3-SAT, for example, can be found in [10].)

In this paper, we show that the following PSPACE-complete problem, denoted by CONNkSAT, is solvable in time $O((2-\epsilon)^n)$ for n variables: given a k-CNF formula φ over n variables, decide whether the solution space of φ is connected on the n-dimensional hypercube. (See the next section for the precise definition.) This problem was proposed by Gopalan, Kolaitis, Maneva, and Papadimitriou to investigate connectivity properties on Boolean formulas. It is known that the problem is PSPACE-complete for $k \geq 3$, while it is in P for $k \leq 2$ [6]. Moreover, it is known to be coNP-complete, if given formulas are restricted to Horn 3-CNF [12]. We show that CONNkSAT for $k \geq 3$ is solvable in time $O((2-\epsilon_k)^n)$ for some constant $\epsilon_k > 0$, where ϵ_k depends only on k, but not on n. It seems to be the first nontrivial result that gives an $O((2-\epsilon)^n)$-time algorithm for a certain *PSPACE-complete problem* in terms of the number of *variables*. Furthermore, this result is considered to be interesting because Calabro, Impagliazzo, and Paturi [5] recently showed the following fact on Π_2-3-SAT: this problem, which is a typical Π_2^P-complete problem, is the QBF problem over 3-CNF formulas, where the quantifier starts with $\forall$, and the number of changes between two types of consecutive quantifiers is at most one. They showed that Π_2-3-SAT is *not* solvable in time $O((2-\epsilon)^n)$ for any constant $\epsilon > 0$, provided that the

SAT problem (with no restriction to the clause length) is not solvable in time $O((2-\epsilon)^n)$ for any constant $\epsilon > 0$. It means that the (general) QBF over 3-CNF formulas, which is a typical PSPACE-complete problem, is not solvable in time $O((2-\epsilon)^n)$ for any constant $\epsilon > 0$ under the same assumption.

2. Preliminaries

In this paper, we deal with k-CNF formulas, where the length of each clause of a formula is at most k. Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. An *assignment* to X is an element of $\{0,1\}^n$. A *partial* assignment to X is an element of $\{0,1,*\}^n$, where we regard a variable assigned $*$ as *unassigned*. We alternately express partial assignments by pulling out coordinates assigned 0 or 1, e.g., a partial assignment $(x_1 = 1, x_2 = *, x_3 = 0, x_4 = *, \ldots, x_n = *) \in \{0,1,*\}^n$ is denoted by $(x_1 = 1, x_3 = 0)$. For two assignments $t_1, t_2 \in \{0,1\}^n$ to X, the *Hamming distance* d between t_1 and t_2 is $d(t_1, t_2) = |\{i \in [n] : t_1(i) \neq t_2(i)\}|$. We extend this notion to partial assignments as follows [1]: for two partial assignments $t_1, t_2 \in \{0,1,*\}^n$, the *Hamming distance* d between t_1 and t_2 is

$$d(t_1, t_2) \stackrel{\text{def}}{=} \left| \left\{ i \in [n] \ : \ \begin{array}{l} t_1(i) \neq *,\ t_2(i) \neq *, \text{ and} \\ t_1(i) \neq t_2(i) \end{array} \right\} \right|.$$

Given a partial assignment t, we simplify φ in the standard way, that is, eliminating any clause from φ if a literal of the clause is assigned 1 under t, and eliminating any literal from φ if the literal is assigned 0 under t. The resulting formula is denoted by $\varphi|_t$. For later use, we present a typical algorithm for k-SAT, denoted by `simple-sat`, in Fig. 1 below.

Proposition 2.1. *Given a k-CNF formula φ, the running time of `simple-sat`(φ) is $O(\beta_k^n)$ for some constant $\beta_k < 2$ depending only on k.*

As shown in section 2 of [11], β_k is the largest real number $x > 0$ that satisfies $x^k - x^{k-1} - \cdots - x^2 - x - 1 = 0$. (For example, $\beta_3 = 1.840$.)

We slightly modify this algorithm for our purpose. First, we omit the second "return `YES`" from the algorithm, that is, we just run `simple-sat`$(\varphi|_t)$ for each partial assignment $t \in S$. Second, we therefore omit the second

[1] It might be better to give it another term since the extension is no longer "distance": it does not satisfy the triangle inequality.

`simple-sat`(φ) // φ is a k-CNF formula

if $\emptyset \in \varphi$ (i.e., $\varphi \notin$ SAT), return `NO`
if $\varphi = \{\}$ (i.e., $\varphi \in$ SAT), return `YES`

Choose a clause $(\ell_1 \vee \cdots \vee \ell_{k'}) \in \varphi$ arbitrarily ($k' \leq k$)
Let $S = \{(\ell_1 = 0, \ldots, \ell_{i-1} = 0, \ell_i = 1) : 1 \leq i \leq k'\}$
$\subset \{0, 1, *\}^{k'}$

for each partial assignment $t \in S$
    if `simple-sat`($\varphi|_t$) returns `YES`, then return `YES`
end-for-each

return `NO`

Figure 1: A simple backtracking algorithm for k-SAT

"return `NO`" from the algorithm. Note that Proposition 2.1 also holds for this modified algorithm. This modification comes from our strategies for solving CONNkSAT: we enumerate *all* satisfying partial assignments. In what follows, we call this modified algorithm `simple-sat`.

Given a k-CNF formula, a *binary decision diagram* is constructed by the execution of `simple-sat`(φ). It is viewed as a rooted *binary* tree shown in Fig. 2: we only depict one part of the binary tree, where a recursive call of `simple-sat`($\varphi|_t$) with $t = (l_1 = 0, \ldots, l_{k'-1} = 0, l_{k'} = 1)$ for some $k' \leq k$ is executed. In such a binary tree, each non-leaf vertex represents a variable, and each edge is labelled with 0 or 1. Alternatively, in such a representation, every vertex can be viewed as a partial assignment. The *depth* of a vertex v in a binary tree is the number of ancestors of v.

Given a k-CNF formula, let T_φ be the rooted binary tree obtained by running `simple-sat`(φ). Let SAT$_\varphi$ be the set of leaves of T_φ that satisfy φ. We alternatively view SAT$_\varphi$ as the set of partial satisfying assignments. For simplicity, we assume that every leaf of T_φ corresponds to a partial satisfying assignment so that SAT$_\varphi$ is exactly the set of leaves of T_φ. This is because such a tree is constructed by erasing from T_φ all sub-trees every leaf of which are not satisfying assignments. Moreover, this construction is done in time $O(\beta_k^n)$, where $\beta_k \leq 2 - \epsilon_k$ for our target bound $O((2 - \epsilon_k)^n)$. Then, we note the following two facts about SAT$_\varphi$.

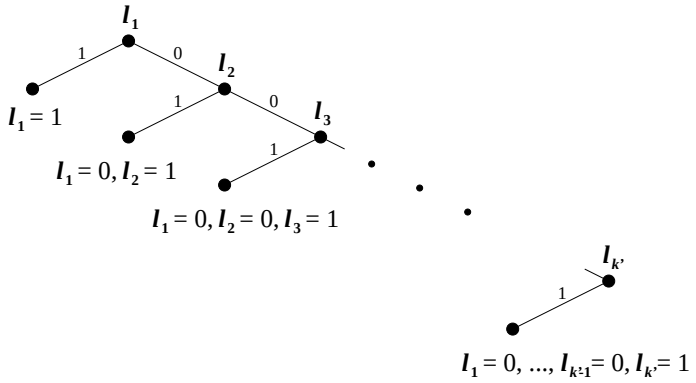

Figure 2: A binary tree

Note 2.2. *For any pair of distinct vertices $u, v \in SAT_\varphi$, $d(u, v) \geq 1$.*

Note 2.3. *The vertex set SAT_φ is a partition of the set of all satisfying assignments of φ, that is, for any satisfying assignment $t \in \{0,1\}^n$ of φ, there is a unique vertex $v \in SAT_\varphi$ (i.e., v is a satisfying partial assignment) such that $d(t, v) = 0$.*

Given a k-CNF formula φ over n variables, let SAT_φ be as above, and let H_φ be the graph induced from the n dimensional hypercube by SAT_φ. The *solution space induced by $S \subset SAT_\varphi$* is the graph induced from H_φ by S, that is, by $\{t \in \{0,1\}^n : \exists s \in S[d(s, t) = 0]\}$. We here note the following two facts, which are easily shown.

Note 2.4. *The solution space induced by a single vertex of SAT_φ is connected.*

Note 2.5. *Let v_1, v_2 be distinct vertices of SAT_φ. Suppose that $d(v_1, v_2) = 1$. Then, the solution space induced by $\{v_1, v_2\}$ is connected.*

Given a k-CNF formula φ over n variables, the connectivity problem which we study, denoted by *CONNkSAT*, is to ask if the graph H_φ is connected.

Theorem 2.6 (Gopalan et al. [6])**.** *CONNkSAT is PSPACE-complete for $k \geq 3$. On the other hand, CONNkSAT is in P for $k \leq 2$.*

3. An Exact Algorithm for CONNkSAT

We present an exact algorithm for CONNkSAT, and show the running time is $O((2-\epsilon_k)^n)$ for some constant $\epsilon_k > 0$. The algorithm, denoted by `conn-sat`(φ) given a k-CNF formula φ, is shown in Fig. 3, where β_k is the constant specified in the preliminary section.

`conn-sat`(φ) // parameter α is a real number that satisfies $(2\beta_k)^{\alpha n} = \beta_k^n$

Run `simple-sat`(φ)
Let T_φ and SAT_φ be as defined above
Let $V_\varphi = V(T_\varphi)$ be the set of vertices of T_φ,

Let $E_\varphi = \emptyset$
Construct an undirected graph $G_{\mathrm{SAT}} = (\mathrm{SAT}_\varphi, E_\varphi)$
as follows:
(1) for each pair of vertices $u, v \in \mathrm{SAT}_\varphi$ with depth at most $(1-\alpha)n$
if $d(u,v) = 1$, then add (u,v) to E_φ
(2) for each $u \in \mathrm{SAT}_\varphi$ with depth more than $(1-\alpha)n$
Visit $v \in V_\varphi$ in the depth-first search manner starting from the root of T_φ so that
if $d(u,v) \geq 2$, then do not visit vertices which are descendants of v any longer
else if $v \in \mathrm{SAT}_\varphi$ and $v \neq u$, then add (u,v) to E_φ

if $G_{\mathrm{SAT}} = (\mathrm{SAT}_\varphi, E_\varphi)$ is connected, output `YES`,
else output `NO`

Figure 3: An exact algorithm for CONNkSAT

The idea of this algorithm is to enumerate all satisfying partial assignments using `simple-sat`, and to construct a graph over those assignments such that there is an edge between two satisfying partial assignments if and only if the Hamming distance between them is *exactly* one. (Recall Note 2.2 that $d(u,v) \geq 1$ for any pair of distinct vertices $u, v \in \mathrm{SAT}_\varphi$.) Then, we can

easily check the connectivity of the graph in linear time with respect to its size, i.e., the number of vertices and edges of the graph.

Before proceeding to the formal analysis of the algorithm, we shall give intuition on how to upper-bound the running time. The crucial point is to show that the size of the constructed graph is at most $O((2-\epsilon)^n)$ for some constant $\epsilon > 0$. We can see that the number of vertices is at most $O(\beta_k^n)$ by Proposition 2.1 since the set of vertices consists of leaves in the tree obtained by running `simple-sat`. However, it is not obvious to bound the number of edges. One trivial upper bound, the square of the number of vertices, may exceed 2^n. (For example, $(\beta_3^n)^2 > 3.385^n$.) Another trivial upper bound, the sum of the degree of the vertices, may also exceed 2^n since each vertex corresponds to a partial assignment and can have exponentially large degree (up to 2^s where s is the number of $*$'s in the assignment).

Our key observation is the following two facts; (i) the number of "shallow" vertices is small and (ii) the degree of "deep" vertices is small. Here shallow and deep mean the depth of a vertex in the tree associated with the execution of `simple-sat`. Fact (i) essentially follows from Proposition 2.1 and fact (ii) holds since a deep vertex corresponds to an assignment with few $*$'s. Then we classify edges into two types, (shallow, shallow) and (deep,any). The number of the former and the latter types can be bounded by $O((2-\epsilon)^n)$ using the square of the number of the shallow vertices and the sum of the degree of the deep vertices, respectively.

Now we proceed to the formal analysis.

Lemma 3.1. *Given a k-CNF formula φ, let $G_{SAT} = (SAT_\varphi, E_\varphi)$ be the final G_{SAT} obtained by constructing E_φ. Let $v_1, v_2 \in SAT_\varphi$ be distinct vertices. Then,*

$$d(v_1, v_2) = 1 \iff (v_1, v_2) \in E_\varphi.$$

Proof. Note first that $d(v_1, v_2) \geq 1$, which comes from Note 2.2. It is easy to see that $(u, v) \in E_\varphi$ implies $d(u, v) = 1$ since our algorithm adds an edge (u, v) to E_φ only if $d(u, v) = 1$.

Suppose that $d(u, v) = 1$. We see that it means our algorithm adds an edge (u, v) to E_φ because of the following observation: if $d(u, v) = 1$, then $d(u, w) \leq 1$ for any ancestor w of v. □

Lemma 3.2. *Given a k-CNF formula φ, let $G_{SAT} = (SAT_\varphi, E_\varphi)$ be the final G_{SAT} obtained by constructing E_φ. Then,*

$$\varphi \in CONNkSAT \iff G_{SAT} \text{ is connected.}$$

Proof. We first consider the case of $|\mathrm{SAT}_\varphi| \leq 1$. In this case, it is obvious that $G_{\mathrm{SAT}} = (\mathrm{SAT}_\varphi, E_\varphi)$ is connected. Moreover, $\varphi \in \mathrm{CONN}k\mathrm{SAT}$ holds because of Note 2.4. Thus, this lemma holds for $|\mathrm{SAT}_\varphi| \leq 1$.

Next, we assume that $|\mathrm{SAT}_\varphi| \geq 2$. Suppose that G_{SAT} is connected. We will show that any pair of two satisfying assignments of φ is connected on H_φ. Let $t_1, t_2 \in \{0,1\}^n$ be distinct satisfying assignments of φ. Let $v_1 \in \mathrm{SAT}_\varphi$ (resp. $v_2 \in \mathrm{SAT}_\varphi$) be a vertex of G_{SAT} corresponding to t_1 (resp. t_2), that is, $d(v_1, t_1) = 0$ (resp. $d(v_2, t_2) = 0$). From Note 2.3, there is such a vertex which is unique. We may assume $v_1 \neq v_2$ since otherwise it is the same as the case of $|\mathrm{SAT}_\varphi| \leq 1$. Since G_{SAT} is connected, there is a path between v_1 and v_2 (on G_{SAT}). Consider any pair of adjacent vertices on the path, say, $u_1, u_2 \in \mathrm{SAT}_\varphi$, From the previous lemma, $d(u_1, u_2) = 1$ since $(u_1, u_2) \in E_\varphi$. Moreover, from Note 2.5, the solution space of H_φ induced by $\{u_1, u_2\}$ is connected. Applying this argument repeatedly to every pair of adjacent vertices on the path, we see that the solution space of H_φ induced by all vertices on the path is connected, and hence t_1 and t_2 are connected on H_φ. This holds for any pair of two satisfying assignments of φ. Thus, we conclude $\varphi \in \mathrm{CONN}k\mathrm{SAT}$.

Suppose that $\varphi \in \mathrm{CONN}k\mathrm{SAT}$. We will show that any pair of two vertices of SAT_φ is connected on G_{SAT}. Let $v_1, v_2 \in \mathrm{SAT}_\varphi$ be distinct vertices of G_{SAT}. Let t_1 (resp. t_2) be an arbitrary satisfying assignment of φ such that $d(t_1, v_1) = 0$ (resp. $d(t_2, v_2) = 0$). Since $\varphi \in \mathrm{CONN}k\mathrm{SAT}$, there exists a path $t_1 = a_0 \to a_1 \to \ldots \to a_\ell = t_2$ on H_φ. Consider any pair of a_i and a_{i+1}. There are two cases: (1) there is a vertex $u \in \mathrm{SAT}_\varphi$ such that $d(a_i, u) = d(a_{i+1}, u) = 0$, and (2) there are distinct vertices $u_1, u_2 \in \mathrm{SAT}_\varphi$ such that $d(a_i, u_1) = d(a_{i+1}, u_2) = 0$. Consider the second case. (We do not need to care for the first case.) Since $d(a_i, a_{i+1}) = 1$, we must have $d(u_1, u_2) = 1$. (We do not have $d(u_1, u_2) = 0$ since $u_1 \neq u_2$.) From the previous lemma, it means $(u_1, u_2) \in E_\varphi$. Applying this argument repeatedly to every pair of adjacent vertices on the path, we see that v_1 and v_2 are connected on G_{SAT}. This holds for any pair of two vertices of SAT_φ. Thus, we conclude that G_φ is connected. □

From this lemma, we conclude that the output of `conn-sat`(φ) is correct for any φ. It remains to show the upper bound on the running time of `conn-sat`(φ).

Lemma 3.3. *The running time of* `conn-sat`(φ) *is* $O((2 - \epsilon_k)^n)$ *for some constant* $\epsilon_k > 0$ *depending only on* k.

Proof. Given a k-CNF formula φ, let T_φ be the rooted binary tree obtained by running `simple-sat`(φ). Let $G_{\mathrm{SAT}} = (\mathrm{SAT}_\varphi, E_\varphi)$ be the final G_{SAT} obtained by constructing E_φ. Note here that the running time of constructing T_φ is $O(\beta_k^n)$, where β_k is the constant specified in the preliminary section. For showing the worst-case running time, it suffices to estimate an upper bound of $|E_\varphi|$. For any $\alpha : 0 \leq \alpha \leq 1$, let

$$\begin{aligned} U &\stackrel{\text{def}}{=} \{u \in \mathrm{SAT}_\varphi : \mathrm{depth}(u) \leq (1-\alpha)n\}, \\ W &\stackrel{\text{def}}{=} \{w \in \mathrm{SAT}_\varphi : \mathrm{depth}(w) > (1-\alpha)n\}, \end{aligned}$$

where $\mathrm{depth}(u)$ is the depth of u in T_φ. Then,

$$|E_\varphi| = |E_1| + |E_2|, \text{ where } \begin{cases} E_1 = E_\varphi \cap (U \times U), \\ E_2 = (E_\varphi \cap (U \times W)) \cup (E_\varphi \cap (W \times W)). \end{cases}$$

Claim 3.4. *For any $\alpha : 0 \leq \alpha \leq 1$,*

$$\begin{aligned} 1.\ |E_1| &\leq \left(\beta_k^{(1-\alpha)n}\right)^2 \quad \left(= \beta_k^{2(1-\alpha)n}\right), \\ 2.\ |E_2| &\leq \sum_{0 \leq t \leq \alpha n} \beta_k^{n-t}\left((n-t)2^t\right) \quad \left(\leq n^2 \cdot 2^{\alpha n} \cdot \beta_k^{(1-\alpha)n}\right). \end{aligned}$$

Proof. The first inequality comes from the fact that the number of vertices of T_φ with depth at most $(1-\alpha)n$ is at most $\beta_k^{(1-\alpha)n}$.

Fix t with $0 \leq t \leq \alpha n$ arbitrarily. Consider an arbitrary vertex $w \in W$ with depth $n-t$. We will estimate the possible number of edges $(w, v) \in E_\varphi$ where $v \in \mathrm{SAT}_\varphi$. Let r_i be the ancestor of w at depth i $(0 \leq i < n-t)$. Let r'_i be the child vertex of r_i that is not an ancestor of w. Let $T_{w,i}$ be the sub-tree of T_φ rooted at r'_i. Then, the number of assignments (not necessarily satisfying ones) $a \in \{0,1\}^n$ such that $d(r'_i, a) = 0$ and $d(w, a) \leq 1$ is exactly 2^t since the number of variables assigned $*$ under w is t. Let $A \subset \{0,1\}^n$ be the set of those assignments. Then, the number of leaves v of $T_{w,i}$ such that $d(w, v) \leq 1$ is at most 2^t since each assignment $a \in A$ corresponds to a unique leaf v if a is a satisfying assignment. Thus, for any $w \in W$ with depth $n-t$, the number of leaves v such that $d(w, v) \leq 1$ is at most $(n-t)2^t$. Since the number of vertices $w \in W$ with depth $n-t$ is at most β_k^{n-t}, the second inequality holds. □

From this claim, we have $|E_\varphi| \leq \beta_k^{2(1-\alpha)n} + n^2 2^{\alpha n} \beta_k^{(1-\alpha)n}$ for any $0 \leq \alpha \leq 1$. By fixing α to a constant satisfying $\beta_k^{2(1-\alpha)n} = 2^{\alpha n} \beta_k^{(1-\alpha)n}$, which is

equivalent to $(2\beta_k)^{\alpha n} = \beta_k^n$, we obtain an upper bound on $|E_\varphi|$ as follows:

$$|E_\varphi| \leq 2 \cdot \mathrm{poly}(n) \cdot 2^{\alpha n} \beta_k^{(1-\alpha)n}.$$

We see that the formula on the right-hand-side is $O((2-\epsilon_k)^n)$ for some constant $\epsilon_k > 0$ (depending only on k) since β_k is a constant less than 2. □

From Lemma 3.2, we see that our algorithm solves CONNkSAT. From Lemma 3.3, we see that our algorithm runs in time $O((2-\epsilon_k)^n)$ for some constant $\epsilon_k > 0$. Therefore, we obtain the following theorem:

Theorem 3.5. *The problem CONNkSAT is solvable in time $O((2-\epsilon_k)^n)$ for some constant $\epsilon_k > 0$ depending only on k. (For example, it is $O(1.914^n)$ for $k = 3$.)*

Lemma 3.2 actually shows that the number of connected components in H_ϕ is same as that in G_{SAT}. Thus, we have:

Corollary 3.6. *Given a k-CNF formula φ over n variables, the number of connected components in H_φ can be computed in time $O((2-\epsilon_k)^n)$ for the constant ϵ_k given in Theorem 3.5.*

4. Conclusion

We have presented an $O(2-\epsilon_k)^n$-time exact algorithm for CONNkSAT. One of our future work is to improve the analysis of the running time of our algorithm, and to obtain the upper bound $O(\beta_k^n)$ which is same as the running time of `simple-sat`: our bound is slightly worse than $O(\beta_k^n)$. Instead of doing that, we may be able to reduce the running time just by replacing `simple-sat` with a more sophisticated backtrack-type algorithm A that satisfies the following: all leaves of the rooted binary tree constructed by A constitute a partition of all satisfying assignments. However, we encounter the same problem as above: we cannot derive the running time as much as that of A from our analysis.

References

[1] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *Journal of the ACM*, 9(1):61–63, 1962.

[2] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, "The travelling salesman problem in bounded degree graphs," In *Proc. ICALP 2008 part I*, pp. 198–209, 2008.

[3] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, "Fourier meets Möbius: Fast subset convolution," In *Proc. STOC 2007*, pp. 67–74, 2007.

[4] A. Björklund and T. Husfeldt, "Inclusion-exclusion algorithms for counting set partitions," In *Proc. FOCS 2006*, pp. 575–582, 2006.

[5] C. Calabro, "The exponential complexity of satisfiability problems," Ph. D. thesis, University of California, San Diego, 2009.

[6] P. Gopalan, P. G. Kolaitis, E. N. Maneva, and C. H. Papadimitriou, "The connectivity of Boolean satisfiability: Computational and structural dichotomies," *SIAM J. Comput.*, 38(6):2330–2355, 2009.

[7] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," Operations Res., 18(6):1138–1162, 1970.

[8] K. Iwama and S. Tamaki, "Improved upper bounds for 3-SAT," In *Proc. SODA 2004*, pp. 328–329, 2004.

[9] M. Koivisto, "An $O(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion," In *Proc. FOCS 2006*, pp. 583–590, 2006.

[10] K. Kutzkov, "New upper bound for the #3-SAT problem," *Inf. Process. Lett.*, 105(1):1–5, 2007.

[11] B. Monien and E. Speckenmeyer, "Solving satisfiability in less than 2^n steps," Discrete Applied Mathematics, 10:287–295, 1985.

[12] K. Makino, S. Tamaki, M. Yamamoto, "On the Boolean connectivity problem for Horn relations," *Discrete Applied Mathematics*, to appear.

[13] 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.

[14] R. Paturi, P. Pudlák, and F. Zane, "Satisfiability coding lemma," *Chicago J. Theor. Comput. Sci.*, 1999.

[15] D. Rolf, "Improved bound for the PPSZ/Schöning-algorithm for 3-SAT," *JSAT*, 1:111–122, 2006.

[16] U. Schöning, "A probabilistic algorithm for k-SAT based on limited local search and restart," *Algorithmica*, 32(4):615–623, 2002.

[17] U. Schöning, "Algorithmics in exponential time," In *Proc. STACS 2005*, pp. 36–43, 2005.

[18] R. Williams, "Algorithms for quantified Boolean formulas," In *Proc. SODA 2002*, pp. 299–307, 2002.

[19] R. Williams, "A new algorithm for optimal constraint satisfaction and its implications," In *Proc. ICALP 2004*, pp. 1227–1237, 2004.

[20] G. J. Woeginger, "Exact algorithms for NP-hard problems: a survey," In *Combinatorial Optimization (Edmonds Festschrift)*, LNCS 2570, pp. 185–207, 2003.