



# Exact Algorithms and Hardness Result for the Boolean Connectivity Problem of $k$ -Horn Formulas

Takashi Horiyama 

Faculty of Information Science and Technology, Hokkaido University, Sapporo, Japan

Yuto Okura 

Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan

Kazuhisa Seto 

Faculty of Information Science and Technology, Hokkaido University, Sapporo, Japan

Junichi Teruyama 

Graduate School of Information Science, University of Hyogo, Kobe, Japan

---

## Abstract

The Boolean connectivity problem asks whether the set of satisfying assignments of a given Boolean formula forms a connected subgraph in the  $n$ -dimensional hypercube. This problem is known to be coNP-complete, even when restricted to  $k$ -Horn formulas for  $k \geq 3$ , as shown by Makino, Tamaki, and Yamamoto. In this paper, we further investigate the complexity of the Boolean connectivity problem for  $k$ -Horn formulas, referred to as CONN  $k$ -HORN. We first present an exact exponential-time algorithm for CONN  $k$ -HORN without any structural restrictions. Our algorithm builds on the deterministic PPZ algorithm proposed by Paturi, Pudlák, and Zane. It runs in  $O^*(2^{(1-1/2k)n})$  time, achieving an exponential improvement over the previously known algorithm for the Boolean connectivity problem of  $k$ -CNF formulas, shown by Makino, Tamaki, and Yamamoto. We then examine both algorithmic and hardness results for CONN 3-HORN under bounded variable occurrences. On the algorithmic side, we propose a polynomial-time algorithm for CONN 3-HORN when each clause contains exactly three literals and each variable appears at most three times. This result generalizes to CONN  $k$ -HORN under the same structural constraints, in which each clause contains exactly  $k$  literals and each variable appears at most  $k$  times. On the hardness side, we prove that CONN 3-HORN remains coNP-complete even when restricted to instances in which each variable appears exactly four times.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Parameterized complexity and exact algorithms

**Keywords and phrases**  $k$ -Horn, Boolean connectivity, bounded variable occurrence, hardness, exact algorithm, satisfiability

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2025.25

**Funding** *Takashi Horiyama*: JSPS KAKENHI Grant Number 20H05964 and 23K24806

*Kazuhisa Seto*: JSPS KAKENHI Grant Number 22H00513, 23K24806 and 24K02898

*Junichi Teruyama*: JSPS KAKENHI Grant Number 22K11910, 23K28039, 23K28040

## 1 Introduction

The Boolean connectivity problem asks whether the satisfying assignments of a given Boolean formula are connected over the  $n$ -dimensional hypercube. The structure and connectivity of the satisfying assignments are related to analyzing the satisfiability algorithm and the threshold of satisfiability. The satisfiability problem is characterized by the famous Schaefer dichotomy theorem [12]. Schaefer's class denotes the class of formulas, e.g., 2-CNF, (dual) Horn, and affine formulas. Ekin, Hammer, and Kogan [5] showed that the Boolean connectivity of DNF is solvable in time linear in the size of DNF, while determining the connectivity of falsifying assignments of DNF is coNP-hard, i.e., the Boolean connectivity of CNF is



© Takashi Horiyama, Yuto Okura, Kazuhisa Seto, and Junichi Teruyama;  
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Parameterized and Exact Computation (IPEC 2025).

Editors: Akanksha Agrawal and Erik Jan van Leeuwen; Article No. 25; pp. 25:1–25:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

coNP-hard. Gopalan, Kolaitis, Maneva, and Papadimitriou [6] deeply investigated the computational complexity of the Boolean connectivity problem. They showed that it is either coNP-complete or PSPACE-complete for non-Schaefer's class, and in coNP for Schaefer's class. They also conjectured that it is solvable in polynomial time for Schaefer's class. Makino, Tamaki, and Yamamoto [7] and Schwerdtfeger [15] gave a negative answer by proving the coNP-completeness of the Boolean connectivity problem for  $k$ -Horn and other related formulas.

Although the trichotomy theorem of the Boolean connectivity problem has been fully established, only a few algorithms are known for instances beyond P. For the connectivity of  $k$ -CNF formulas (CONN  $k$ -CNF), it is solvable in polynomial time when  $k = 2$  [6]. Makino et al. [8] presented a moderately exponential-time algorithm for CONN  $k$ -CNF ( $k \geq 3$ ) over  $n$  variables and  $m$  clauses which is PSPACE-complete. Their algorithm constructs the solution graph over partial satisfying assignments and then determines the connectivity of the graph. It runs in  $O^*(2^{(1-c_k)n})$  time and exponential space, where  $c_k = \log \beta_k / (1 + \log \beta_k)$  and  $\beta_k (< 2)$  is the largest positive real number that satisfies  $x^k - x^{k-1} - \dots - x - 1 = 0$ . The  $O^*$  notation suppresses polynomial factors in  $n$  and  $m$ . Throughout this paper, the base of the logarithm is 2. Note that  $c_k = 1/2^{O(k)}$ .

In this paper, we deeply investigate the complexity of the Boolean connectivity of  $k$ -Horn formulas (CONN  $k$ -HORN). We first consider an exact algorithm for CONN  $k$ -HORN without any structural restrictions. The algorithm of Makino et al. can also solve CONN  $k$ -HORN in the same running time since  $k$ -Horn formulas are subsets of  $k$ -CNF formulas. Hence, it is a natural question whether there exists a faster algorithm for CONN  $k$ -HORN. Makino et al. [7] presented an exact algorithm in polynomial time in terms of the number of variables and the size of the characteristic set of a given Horn formula. Unfortunately, the efficiency of finding the characteristic set of a given Horn formula is not yet well-known. We give an affirmative answer by utilizing the deterministic PPZ algorithm for  $k$ -SAT shown by Paturi, Pudlák, and Zane [10]. The deterministic PPZ algorithm can find not only a satisfiability assignment but also all locally minimal satisfying assignments. Moreover, CONN  $k$ -HORN can be solved to find a locally minimal satisfying assignment with Hamming weight at least one [6]. These help us solve CONN  $k$ -HORN exponentially faster than the algorithm of CONN  $k$ -CNF.

► **Theorem 1.** *Given a  $k$ -Horn formula over  $n$  variables, there exists a deterministic  $O^*(2^{(1-1/2k)n})$  time and polynomial-space algorithm for CONN  $k$ -HORN.*

The slightly modified algorithm above can count the number of connected components in the solution graph of a given  $k$ -Horn formula.

► **Corollary 2.** *Counting the number of connected components in the solution graph of a given  $k$ -Horn formula over  $n$  variables can be done in deterministic  $O^*(2^{(1-1/2k)n})$  time and  $O^*(2^{(1-1/2k)n})$  space.*

We next consider CONN 3-HORN with a bounded number of occurrences of each variable. We present both algorithmic and hardness results.

► **Theorem 3.** *CONN 3-HORN is solvable in polynomial time when each clause has length exactly three and each variable appears at most three times.*

The key idea of our algorithm is based on the relationship between the solution graph of a Horn formula and a non-empty maximal self-implicating set, as shown in [15]. Any variable in a self-implicating set is forced to be true when all the other variables in the set are

true. The solution graph is disconnected if and only if there exists a non-empty maximal self-implicating set and no clauses with only negative literals of these variables. We show that such a set can be found in polynomial time.

► **Theorem 4.** *CONN 3-HORN is coNP-complete even when each variable appears exactly four times.*

We first prove the complement of CONN 3-HORN remains NP-complete even if each variable appears at most four times by a polynomial-time reduction from MONOTONE NOT-ALL-EQUAL 3-SAT with each variable appearing exactly four times. This problem is known to be NP-complete [3]. We next show Theorem 4 by providing a polynomial-time reduction from the above problem.

## Related Work

The connectivity problem between two satisfying assignments (ST-CONN) has also been extensively studied. Gopalan et al. [6] showed that ST-CONN is solvable in polynomial time for Schaefer's and a part of non-Schaefer's class and is PSPACE-complete otherwise. Scharpfenecker [13] proved that the complexity of SAT is equivalent to that of ST-CONN in Schaefer's class. This implies that ST-CONN is P-complete for Horn formulas and is NL-complete for 2-CNFs. Cardinal, Demaine, Eppstein, Hearn, and Winslow [2] showed that ST-CONN of PLANAR MONOTONE NOT-ALL-EQUAL 3-SAT is PSPACE-complete.

The problem of finding the shortest path in the solution graph of Boolean formulas has also been investigated. Mouawad, Nishimura, Pathak, and Raman [9] investigated the complexity of this problem. They established a trichotomy theorem stating that the problem is either in P, NP-complete, or PSPACE-complete. They also showed that some classes of Boolean formulas exist where the shortest path can be found in polynomial time, even though its length is not equal to the symmetric difference of the values of each variable in  $s$  and  $t$ . Bonsma, Mouawad, Nishimura, and Raman [1] showed the problem is  $W[1]$ -hard when parameterized by the path length  $\ell$  from  $s$  to  $t$ .

The complexity of the isomorphism of two solution graphs was studied in [14]. This problem is PSPACE-hard and lies in EXP for general formulas, and is  $C=P$ -complete for 2-CNF. The class  $C=P$  is a subclass of PSPACE and is defined in terms of exact counting.

## 2 Preliminaries

Let  $x$  be a Boolean variable that takes true (1) or false (0). A *literal* is a Boolean variable  $x$  (positive literal) or its negation  $\bar{x}$  (negative literal). A *clause* is a disjunction of literals. The *length* of clause  $C$  is defined as the number of literals in  $C$ . A *unit clause* is a clause of length one. A conjunctive normal form (CNF) formula is a conjunction of clauses, and a  $k$ -CNF formula is a CNF formula if the length of each clause is at most  $k$ . A CNF formula is *monotone* if all clauses in the formula consist of only positive (or negative) literals. In this paper, we use Monotone CNF as CNF formulas with all positive literals unless otherwise stated. A *Horn clause* is a clause including at most one positive literal. A Horn formula is a CNF formula whose all clauses are Horn, and a  $k$ -Horn formula is a  $k$ -CNF formula and a Horn formula. Let  $X = \{x_1, x_2, \dots, x_n\}$  be a set of Boolean variables. An *assignment* of a Boolean formula  $\varphi$  over  $X$  is  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \{0, 1\}^n$  such that  $x_i = \alpha_i$  for all  $i$ . A *partial assignment* of a Boolean formula  $\varphi$  over  $X$  is  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \{0, 1, *\}^n$  such that  $x_i = \alpha_i$  for all  $i$ . A partial assignment with  $\alpha_i = *$  means variable  $x_i$  is not assigned a 0/1 value. For a variable  $x$  and an assignment  $\alpha$ , we denote by  $\alpha(x)$  the value of  $x$  under

an assignment  $\alpha$ . We can simplify a Boolean formula  $\varphi$  by a partial assignment  $\alpha$  in a natural way, and the resulting formula is denoted by  $\varphi|_\alpha$ . An (partial) assignment  $\alpha$  is a *(partial) satisfying assignment* of a Boolean formula  $\varphi$  if  $\varphi|_\alpha$  is true. The satisfiability problem is to determine whether there exists a satisfying assignment of a given Boolean formula  $\varphi$ .

Many excellent exponential-time algorithms for the satisfiability problem of  $k$ -CNF formulas ( $k$ -CNF SAT) are known. One of the famous algorithms is shown by Paturi, Pudlák, and Zane [10].

► **Theorem 5** ([10]). *There exists an algorithm that can solve  $k$ -CNF SAT over  $n$  variables in a deterministic  $O^*(2^{(1-1/2k)n})$  time and polynomial space.*

For an assignment  $\alpha \in \{0, 1\}^n$ , the *Hamming weight* of  $\alpha$  is  $|\{i : \alpha_i = 1 \ (1 \leq i \leq n)\}|$ . For two assignments  $\alpha, \alpha' \in \{0, 1\}^n$ , the *Hamming distance* between  $\alpha$  and  $\alpha'$ , denoted by  $d(\alpha, \alpha')$ , is  $|\{i : \alpha_i \neq \alpha'_i \ (1 \leq i \leq n)\}|$ . We define the *solution graph* of a Boolean formula  $\varphi$  as  $G_\varphi := (V_\varphi, E_\varphi)$ , where  $V_\varphi$  is a set of all satisfying assignments of  $\varphi$ , that is,  $V_\varphi = \{\alpha : \varphi|_\alpha = 1, \alpha \in \{0, 1\}^n\}$ , and  $E_\varphi = \{(\alpha, \alpha') : d(\alpha, \alpha') = 1 \text{ and } \alpha, \alpha' \in V_\varphi\}$ .

► **Definition 6.** *The Boolean connectivity problem (CONN) is to ask whether the solution graph  $G_\varphi$  of a given Boolean formula  $\varphi$  over  $n$  variables is connected.*

Makino, Tamaki, and Yamamoto [7] showed that CONN 3-HORN is coNP-complete, thus the following lemma holds.

► **Lemma 7** ([7]). *CONN  $k$ -HORN is coNP-complete when  $k \geq 3$ .*

Given two assignments  $\alpha, \alpha' \in \{0, 1\}^n$ , we denote  $\alpha \leq \alpha'$  as the coordinate-wise partial order if  $\alpha_i \leq \alpha'_i$  for each  $1 \leq i \leq n$ . A satisfying assignment  $\alpha$  for a given formula  $\varphi$  is *locally minimal* if  $\alpha$  has no neighboring satisfying assignment  $\alpha'$  with  $d(\alpha, \alpha') = 1$  and  $\alpha' \leq \alpha$ . For  $v, v' \in V_\varphi$ , a *monotone path* from  $v$  to  $v'$  is a path in  $G_\varphi$ ,  $v \rightarrow u_1 \rightarrow \dots \rightarrow u_r \rightarrow v'$  such that  $v \leq u_1 \leq \dots \leq u_r \leq v'$ . Gopalan, Kolaitis, Maneva, and Papadimitriou [6] showed a notable connection between locally minimal satisfying assignments and the connectivity of Horn formulas.

► **Lemma 8** ([6]). *Given a Horn formula  $\varphi$ , every connected component of  $G_\varphi$  has a unique locally minimal satisfying assignment. Moreover, there exists a monotone path from the locally minimal satisfying assignment to each satisfying assignment in the same connected component.*

The following corollary follows from that any Horn formula without unit clauses over  $n$  variables has an all-zero satisfying assignment ( $0^n$ ) and Lemma 8.

► **Corollary 9** ([7]). *Given a Horn formula  $\varphi$  without unit clauses, the solution graph  $G_\varphi$  is connected if and only if no locally minimal non-zero satisfying assignment exists.*

Let  $P(C)$  (resp.  $N(C)$ ) denote the set of variables that appear in positive (resp. negative) literals in a clause  $C$  of  $\varphi$ . Makino et al. [7] introduced the following Boolean formula  $\Phi_\varphi$ .

$$\begin{aligned} \Phi_\varphi &= \varphi \wedge \bigwedge_{i=1}^n D_i, \\ D_i &= \overline{x_i} \vee \bigvee_{C \in \varphi: P(C)=\{x_i\}} \bigwedge_{y \in N(C)} y. \end{aligned} \tag{1}$$

For example, given a Horn formula

$$\varphi = (x_1 \vee \overline{x_2} \vee \overline{x_3})(x_1 \vee \overline{x_3} \vee \overline{x_4})(\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})(\overline{x_2} \vee x_3 \vee \overline{x_4})(\overline{x_1} \vee \overline{x_3} \vee x_4),$$

then

$$D_1 = \overline{x_1} \vee x_2x_3 \vee x_3x_4, D_2 = \overline{x_2}, D_3 = \overline{x_3} \vee x_2x_4, D_4 = \overline{x_4} \vee x_1x_3.$$

Note that each  $D_i$  is either a unit clause with a negative literal or a conjunction of one negative literal and disjunctions of positive literals. The satisfiability of  $\Phi_\varphi$  is strongly related to the connectivity of the solution graph of a Horn formula  $\varphi$  because a satisfying assignment of  $\Phi_\varphi$  is a locally minimal satisfying assignment of  $\varphi$ .

► **Lemma 10** ([7]). *Given a Horn formula  $\varphi$  without unit clauses over  $n$  variables, the solution graph  $G_\varphi$  is disconnected if and only if  $\Phi_\varphi$  has a non-zero satisfying assignment.*

A *restraint clause* is a clause with only negative literals, and a *restraint set* is a set of variables included in restraint clauses. An *implication clause* is a clause that has one positive literal and one or more negative literals. A Boolean variable  $x$  is said to be *implied* by a set of variables  $U$  if setting all variables in  $U$  to true forces  $x$  to be true. A *self-implicating set*  $U$  is a set of variables, where every  $x \in U$  is implied by  $U \setminus \{x\}$ . We also say that  $U$  is a *maximal self-implicating set* if  $U$  is a set of all variables implied by  $U$ . For maximal self-implicating sets, there exists a relation between locally minimal satisfying assignments and maximal self-implicating sets.

► **Lemma 11** ([15]). *For every Horn formula  $\varphi$  without positive unit clauses, there is a bijection correlating each connected component  $\varphi_i$  with a maximal self-implicating set  $U_i$  containing no restraint set, i.e.,  $U_i$  consists of the variables assigned true in the minimum assignment of  $\varphi_i$  (the ‘lowest’ component is correlated with the empty set).*

The following corollary follows from Lemma 11 and Corollary 9.

► **Corollary 12** ([15]). *The solution graph of a Horn formula  $\varphi$  without positive unit clauses is disconnected if and only if  $\varphi$  has a non-empty maximal self-implicating set containing no restraint set.*

### 3 Exact Algorithm for the Boolean Connectivity of $k$ -Horn formulas based on the deterministic PPZ Algorithm

In this section, we present a faster algorithm for CONN  $k$ -HORN based on the deterministic PPZ algorithm [10]. Before presenting our new algorithm, we describe the notations and behaviors of the deterministic PPZ algorithm, referred to as DETPPZ. Let  $\varphi$  be a  $k$ -CNF formula with  $n$  variables and  $m$  clauses, and let  $\alpha$  be a satisfying assignment. A clause  $C_{(\alpha,i)}$  is a *critical clause* for the variable  $x_i$  at the solution  $\alpha$  if  $C_{(\alpha,i)}$  is to be false by flipping  $\alpha_i$ . A variable  $x_i$  is a *critical variable* if there exists a critical clause  $C_{(\alpha,i)}$ . The DETPPZ first checks whether there exists a satisfying assignment of  $\varphi$  to check all assignments with Hamming weight at most  $\epsilon n$  where  $0 < \epsilon < 1/2$ . If there is no such satisfying assignment, the DETPPZ tries to find a locally minimal satisfying assignment  $\alpha$  with Hamming weight at least  $\epsilon n$ . Note that  $\varphi$  is falsified by any assignment obtained by flipping any 1 in  $\alpha$  to 0. Thus,  $\alpha$  must have at least  $\epsilon n$  critical variables. Let  $\sigma = (\sigma_1, \dots, \sigma_n)$  be a permutation of  $\{1, \dots, n\}$ . The DETPPZ picks up a permutation  $\sigma$  in a “good” small sample space and branches by assigning a 0/1 value to  $x_{\sigma_i}$  according to the order in  $\sigma$ . If any critical

---

**Algorithm 1** PPZ-CONN- $k$ HORN( $\varphi$ ).

---

```

1 Construct the set of permutations  $S$  in Lemma 13
2 for all assignments  $\alpha$  with at most  $\epsilon n$  1's without all 0's do
3   if  $\alpha$  satisfies  $\varphi$  and  $\alpha$  is locally minimal then
4     return NO
5 for all permutations  $\sigma$  in  $S$  do
6   for all strings  $s$  of  $n(1 - \epsilon/k) + 1$  bits do
7     for  $i = 1$  to  $n$  do
8       Let  $x_j$  be the  $i$ -th variable according to  $\sigma$  (i.e.,  $j = \sigma_i$ ).
9       if there exists a unit clause  $x_j$  or  $\bar{x}_j$  then
10        set  $x_j$  to make that clause true
11      else if there exists an unused bit from  $s$  then
12        set  $x_j$  equal to the next unused bit from  $s$ 
13      else
14        go to the next string in the for loop of lines 6–17.
15        ▷ there are no unused bits from  $s$ 
16    if  $\varphi$  is satisfiable and the satisfying assignment is locally minimal then
17      return NO
18 return YES

```

---

variable appears last among the variables in the corresponding critical clause, then it can be assigned correctly. It helps to reduce the number of branches in the algorithm. Modifying the DETPPZ such that it does not halt even if it finds a satisfying assignment with Hamming weight at most  $\epsilon n$ , the DETPPZ can find all locally minimal satisfying assignments. From Corollary 9, CONN  $k$ -HORN can be solved by checking whether there exists some locally minimal satisfying assignment except for  $0^n$  assignment. Thus, by combining the (modified) DETPPZ and the procedure checking locally minimality, we solve CONN  $k$ -HORN in the same running time as the DETPPZ. We describe our algorithm, PPZ-CONN- $k$ HORN, in Algorithm 1.

The following “good” sample space of permutations using  $k$ -wise independent variables is useful.

► **Lemma 13** ([10]). *One can construct the set of permutations  $S$  of size  $O(n^{3k})$  that satisfies the following condition: for any set  $Y$  of up to  $k$  variables, any variable  $y \in Y$ , and for a randomly chosen permutation from  $S$ , the probability that  $y$  appears last among the variables in  $Y$  is at least  $1/|Y| - 1/n$ .*

The following lemma immediately follows from the proof of correctness of the deterministic PPZ algorithm shown in [10]. We provide almost the same proof for self-containedness in this paper.

► **Lemma 14.** *Let  $S$  be the set of permutations constructed by Lemma 13. Let  $\alpha$  be an arbitrary satisfying assignment with at least  $\epsilon n$  critical clauses. There exists a permutation in  $S$  that produces  $\alpha$ .*



**Proof.** Let  $\alpha$  be a satisfying assignment with at least  $\epsilon n$  critical clauses. We now choose a permutation  $\sigma$  from  $S$  at random. For each critical clause at  $\alpha$ , the probability that the critical variable  $x$  appears last in  $\sigma$  among the variables in the clause is at least  $1/k - 1/n$  by Lemma 13. Since the number of critical clauses is at least  $\epsilon n$ , the expected number of times this event appears is at least  $\epsilon n(1/k - 1/n) = \epsilon n/k - \epsilon$ , and there exists some  $\sigma$  in  $S$  that achieves at least the expectation. With respect to such  $\sigma$ , we assign at most  $n - (\epsilon n/k - \epsilon) < n(1 - \epsilon/k) + 1$  variables to satisfy  $\varphi$ . Thus, by checking all strings of  $n(1 - \epsilon/k) + 1$  bits, the permutation  $\sigma$  can produce a satisfying assignment  $\alpha$ . ◀

We next show that the locally minimality of satisfying assignments can be efficiently checked.

► **Lemma 15.** *Let  $\alpha$  be a satisfying assignment of  $\varphi$  with Hamming weight  $w$ . One can check whether  $\alpha$  is locally minimal in  $O(mw + n)$  time.*

**Proof.** From the definition of locally minimality, if any satisfying assignment  $\alpha$  of  $\varphi$  is locally minimal, then any assignment  $\alpha'$  obtained by flipping any 1's bit is not a satisfying assignment of  $\varphi$ . It can be done by checking whether  $\alpha'$  satisfies  $\varphi$ . Since the Hamming weight of  $\alpha$  is  $w$ , we can check the locally minimality of  $\alpha$  in  $O(mw + n)$  time. ◀

We present the main lemma, which establishes the correctness of our algorithm.

► **Lemma 16.** *The deterministic PPZ algorithm can find all locally minimal satisfying assignments of  $\varphi$ .*

**Proof.** The deterministic PPZ algorithm first checks all assignments with Hamming weight at most  $\epsilon n$ . Hence, any satisfying assignment  $\alpha$  with Hamming weight at most  $\epsilon n$  can be found. We can check whether  $\alpha$  is locally minimal from Lemma 15.

We next consider that any locally minimal satisfying assignment  $\alpha$  has Hamming weight at least  $\epsilon n$ . Since  $\alpha$  is a locally minimal satisfying assignment and has at least  $\epsilon n$  1's, there exists at least  $\epsilon n$  critical clauses at  $\alpha$ . From Lemma 14,  $\alpha$  must be produced by some permutation in the sample space  $S$  constructed in the algorithm. The algorithm checks all permutations in  $S$ , then it can find  $\alpha$ . We can check whether  $\alpha$  is truly locally minimal from Lemma 15. ◀

We are now ready to prove Theorem 1.

► **Theorem 1.** *Given a  $k$ -Horn formula over  $n$  variables, there exists a deterministic  $O^*(2^{(1-1/2k)n})$  time and polynomial-space algorithm for CONN  $k$ -HORN.*

**Proof.** We can solve CONN  $k$ -HORN by checking whether there exists some locally minimal non-zero satisfying assignment from Corollary 9. Lemma 16 shows that the deterministic PPZ algorithm can find all locally minimal satisfying assignments. Thus, PPZ-CONN- $k$ HORN in Algorithm 1 solves CONN  $k$ -HORN correctly. The deterministic PPZ algorithm runs in  $O^*(2^{(1-1/2k)n})$  time and polynomial space. Checking locally minimality can be done  $O(mn)$  time for each satisfying assignment from Lemma 15. Thus, our algorithm runs in  $O^*(2^{(1-1/2k)n})$  time and polynomial space. ◀

By adding the procedure of counting the number of distinct locally minimal satisfying assignments (see Algorithm 2), we obtain the following Corollary.

► **Corollary 2.** *Counting the number of connected components in the solution graph of a given  $k$ -Horn formula over  $n$  variables can be done in deterministic  $O^*(2^{(1-1/2k)n})$  time and  $O^*(2^{(1-1/2k)n})$  space.*

---

**Algorithm 2** COUNT-CC- $k$ HORN( $\varphi$ ).

---

```

1 count  $\leftarrow$  0
2 Construct the set of permutations  $S$  in Lemma 13
3 for all assignments  $\alpha$  with at most  $\epsilon n$  1's do
4   if  $\alpha$  satisfies  $\varphi$ , and  $\alpha$  is locally minimal and has never been produced then
5     count  $\leftarrow$  count+1
6 for all permutations  $\sigma$  in  $S$  do
7   for all strings  $s$  of  $n(1 - \epsilon/k) + 1$  bits do
8      $\alpha \leftarrow 0^n$ 
9     for  $i = 1$  to  $n$  do
10      Let  $x_j$  be the  $i$ -th variable according to  $\sigma$  (i.e.,  $j = \sigma_i$ ).
11      if there exists a unit clause  $x_j$  or  $\bar{x}_j$  then
12        set  $\alpha_j$  to make that clause true
13      else if there exists an unused bit from  $s$  then
14        set  $\alpha_j$  equal to the next unused bit from  $s$ 
15      else
16        go to the next string in the for loop of lines 6–17.
17       $\triangleright$  there are no unused bits from  $s$ 
18    if  $\varphi$  is satisfiable, and the satisfying assignment  $\alpha$  is locally minimal and has
19      never been produced then
20      count  $\leftarrow$  count+1
21 return count

```

---

#### 4 Complexity on the Boolean Connectivity of 3-Horn Formulas with Bounded Variable Occurrence

In this section, we consider the Boolean connectivity of 3-Horn, whose variable occurrence is bounded. Section 4.1 presents a deterministic polynomial-time algorithm for CONN 3-HORN with each clause of length exactly three and each variable appearing at most three times (CONN E3-HORN-3). Section 4.2 shows the coNP-complete of CONN 3-HORN where each variable appears exactly four times (CONN 3-HORN-E4).

##### 4.1 Polynomial-time Algorithm

We present a deterministic polynomial-time algorithm for CONN E3-HORN-3. Our algorithm can solve the connectivity of  $k$ -Horn formulas with the length of each clause exactly  $k$  and each variable appearing at most  $k$  (CONN Ek-HORN- $k$ ). For a set of variables  $X$  and a CNF formula  $\varphi$ , we denote by  $C_{\varphi[X]}$  the set of all clauses in  $\varphi$  that consist of only variables in  $X$ .

► **Lemma 17.** *Let  $\varphi$  be an instance of CONN E3-HORN-3 over  $X = \{x_1, x_2, \dots, x_n\}$ . The solution graph  $G_{\varphi}$  is disconnected if and only if there exists a partition of the set  $X$  into two sets  $U$  and  $X \setminus U$  and a partition of the set  $C_{\varphi[X]}$  of clauses of  $\varphi$  into two sets  $C_{\varphi[U]}$  and  $C_{\varphi[X \setminus U]}$  such that  $U$  is a non-empty set of variables which appear once as a positive literal.*



**Proof.** We first prove the if-part, and assume that a variable set  $U$  satisfies the if-condition. We show that  $U$  is a non-empty maximal self-implicating set containing no restraint set, then  $G_\varphi$  is disconnected from Corollary 12. The number of implication clauses in  $C_{\varphi[U]}$  is  $|U|$  since each variable in  $U$  appears once as a positive literal. These implication clauses have  $3|U|$  literals since the length of each clause in  $C_{\varphi[U]}$  is three, and then the total number of literals of clauses in  $C_{\varphi[U]}$  is at least  $3|U|$ . However, the total number of occurrences of variables in  $U$  is at most  $3|U|$  since each variable appears at most three times. These lead to the total number of literals in  $C_{\varphi[U]}$  being exactly  $3|U|$ . Thus, all literals in  $C_{\varphi[U]}$  appear in implication clauses in  $C_{\varphi[U]}$ , i.e., there is no restraint set in  $U$ . For any implication clause  $C$ , the positive literal  $x$  in  $C$  is assigned true when the other negative literal(s) are assigned false. This implies that every  $x \in U$  is implied by  $U \setminus \{x\}$ . In addition, any variable in any clause in  $C_{\varphi[X \setminus U]}$  cannot be implied by  $U$  since  $U \cap (X \setminus U) = \emptyset$ . Hence,  $U$  is a non-empty maximal self-implicating set and contains no restraint set. This concludes the proof of the if-part.

We next prove the only-if part. Assume that  $G_\varphi$  is disconnected. From Corollary 12,  $\varphi$  has a non-empty maximal self-implicating set  $U$  containing no restraint set. This implies that every variable in  $U$  appears as a positive literal at least once. We claim that the number of implication clauses with only variables in  $U$  is  $|U|$ . There exists at least  $|U|$  implication clauses in  $C_{\varphi[X]}$  since  $U$  is a self-implicating set, i.e., there exists a clause  $u \vee \bar{x} \vee \bar{y}$  for every  $u \in U$ . Since the total number of occurrences of variables in  $U$  is at most  $3|U|$  and the length of each clause is three, there exist at most  $|U|$  implication clauses with only variables in  $U$ . If the number of such implication clauses with only variables in  $U$  is at most  $|U| - 1$ , there exists some  $u \in U$  appearing in an implication clause as positive literals with at least one variable  $x$  in  $X \setminus U$ . We denote  $(u \vee \bar{x} \vee \bar{y})$  as such a clause. In this case,  $u$  is implied by  $x$  and  $y$ . This contradicts the fact that  $U$  is a maximal self-implicating set. Thus, the number of implication clauses with only variables in  $U$  is  $|U|$ , and this implies that every variable in  $U$  appears as a positive literal only once. The number of literals of these clauses is  $3|U|$ , and it equals the maximum total number of occurrences of variables in  $U$ . This implies that the pair  $(U, X \setminus U)$  is a partition of  $X$ , and the pair of  $(C_{\varphi[U]}, C_{\varphi[X \setminus U]})$  is a partition of  $C_{\varphi[X]}$ . This concludes the proof of the only-if part.  $\blacktriangleleft$

Now, we present a polynomial-time algorithm based on Lemma 17.

► **Theorem 3.** *CONN 3-HORN is solvable in polynomial time when each clause has length exactly three and each variable appears at most three times.*

**Proof.** A given 3-Horn formula  $\varphi$  over  $X = \{x_1, x_2, \dots, x_n\}$  has at most  $3n$  literals and  $n$  clauses since the length of each clause is three and each variable appears at most three times. We find a non-empty variable set that satisfies Lemma 17 to determine the connectivity of the solution graph of  $\varphi$ .

We first enumerate the non-empty variable set  $U$  such that  $C_{\varphi[X]} = C_{\varphi[U]} \cup C_{\varphi[X \setminus U]}$  and  $C_{\varphi[U]} \cap C_{\varphi[X \setminus U]} = \emptyset$  as follows. We construct the graph  $G = (X, E)$ , where  $E = \{(x_i, x_j) : C_{\varphi[X]} \text{ has a clause containing } x_i, x_j \in X\}$ . It is easy to see that the variable set  $U'$  corresponding to the vertices in any connected component of  $G$  constructs  $C_{\varphi[U']}$ . Thus, we can enumerate the non-empty variable sets to enumerate the connected components of  $G$ . We use the depth-first search for  $G$  to enumerate all the connected components. It remains to check whether each  $U'$  is a set of variables that appear once as a positive literal. If at least one such  $U'$  exists, the solution graph is disconnected. Otherwise, the solution graph is connected, as shown in Lemma 17.

We now estimate the running time of this algorithm. Constructing the graph  $G$  can be done in  $O(n)$  time since the number of edges is at most  $3n$ . Enumerating the non-empty variable set  $U'$  can be done in time  $O(n)$  by the depth-first search. Note that the number of non-empty variable sets is  $O(n)$ . We can check whether each  $U'$  is a set of variables that appear once as a positive literal in  $O(n)$  by checking the clauses. Hence, our algorithm runs in time  $O(n^2)$ . ◀

We can show Corollary 18 since the discussion in Lemma 17 also holds for  $k \geq 3$ .

► **Corollary 18.** *Let  $\varphi$  be an instance of CONN Ek-HORN- $k$  over  $X = \{x_1, x_2, \dots, x_n\}$ . The solution graph  $G_\varphi$  is disconnected if and only if there exists a partition of the set  $X$  into two sets  $U$  and  $X \setminus U$  and a partition of the set  $C_{\varphi[X]}$  of clauses of  $\varphi$  into two sets  $C_{\varphi[U]}$  and  $C_{\varphi[X \setminus U]}$  such that  $U$  is a non-empty set of variables which appear once as a positive literal.*

Gopalan et al. [6] showed that CONN 2-CNF is solvable in polynomial time with no bounded variable occurrence. The proof of Theorem 3 also holds for  $k \geq 3$  from Corollary 18. Therefore, we can show Corollary 19.

► **Corollary 19.** *For  $k \geq 2$ , CONN Ek-HORN- $k$  is solvable in polynomial time.*

We can show the following corollary from Corollary 18.

► **Corollary 20.** *For  $k \geq 3$ , given a  $k$ -Horn formula  $\varphi$ , where each clause length and each variable occurrence are exactly  $k$ , the number of connected components in the solution graph of  $\varphi$  is at most  $2^{\lfloor n/k \rfloor}$ .*

**Proof.** Let  $\varphi$  be an instance of CONN  $k$ -HORN with each clause length exactly  $k$  and each variable occurrence exactly  $k$ , where the solution graph of  $\varphi$  is disconnected. A variable set of  $\varphi$  is denoted by  $X$ . Then, there exists at least one variable set  $U \subseteq X$  such that  $C_{\varphi[U]}$  and  $C_{\varphi[X \setminus U]}$  partition  $C_{\varphi[X]}$  and  $U$  is a non-empty set of variables that appear once as a positive literal. Now, we assume that  $\varphi$  has  $m$  variable sets  $U_i$  ( $1 \leq i \leq m$ ) and a variable set  $V$  such that  $C_{\varphi[X]} = \bigcup_{i=1}^m C_{\varphi[U_i]} \cup C_{\varphi[V]}$  and  $\bigcup_{i=1}^m U_i \cup V = X$ ,  $U_i \cap V = \emptyset$ , and  $U_i \cap U_j = \emptyset$  for any  $i, j$  ( $i \neq j$ ), where  $U_i$  is a non-empty set of variables which appear once as a positive literal. Then, we can construct all locally minimal satisfying assignments by setting the value 0 or 1 to all variables for each  $U_i$  ( $1 \leq i \leq m$ ) and setting the value 0 to all variables in  $V$  from Lemma 11. Therefore, there exist  $2^m$  locally minimal satisfying assignments of  $\varphi$ . Moreover,  $m$  is a multiple of  $k$  since the length of each clause and the number of occurrences of each variable are exactly  $k$ . Hence, the maximum value of  $m$  is  $\lfloor n/k \rfloor$ . ◀

## 4.2 Hardness

We prove the coNP-completeness of CONN 3-HORN where each variable appears at most four times (CONN 3-HORN-4), then we prove the coNP-completeness of CONN 3-HORN-E4 by a polynomial-time reduction from it. Instead of proving the coNP-completeness of CONN 3-HORN-4, we show the NP-completeness of the complement problem of CONN 3-HORN-4 (DISCONN 3-HORN-4) that asks whether the solution graph of a given 3-Horn formula with each variable appearing at most four times is disconnected. The proof is based on a polynomial-time reduction from MONOTONE NOT-ALL-EQUAL E3-SAT-E4 (for short, MONOTONE NAE E3-SAT-E4) by utilizing the framework of Makino et al. [7]. An *NAE-satisfying assignment* is a satisfying assignment in which each clause has at least one literal assigned false and at least one literal assigned true. Given a Monotone 3-CNF formula

$\phi$  with  $n$  variables and  $m$  clauses in which each clause has exactly three literals and each variable appears exactly four times, MONOTONE NAE E3-SAT-E4 asks whether there exists an NAE-satisfying assignment of  $\phi$ . It is known that this problem is NP-complete [3].

We first show the construction of a 3-Horn formula from an instance of MONOTONE NAE E3-SAT-E4. Let  $X = \{x_1, x_2, \dots, x_n\}$  be a set of variables. Let  $\phi$  be an instance of MONOTONE NAE E3-SAT-E4 over  $X$  with  $m$  clauses. Let  $C_\phi = \{C_1, C_2, \dots, C_m\}$  be the set of clauses of the formula  $\phi$ . For simplicity, if the variable  $x_j$  appears in the clause  $C_i$ , we write  $x_j \in C_i$ . We first replace the  $k$ -th occurrence of  $x_i \in X$  with a new variable  $x_{i,k}$  and we denote by  $X_\phi$  the set of new variables created from  $X$ . Let  $Y = \{y_i : 1 \leq i \leq m\}$ ,  $Z = \{z_i : 1 \leq i \leq m-2\}$  and  $Q = \{q_{i,j} : 1 \leq i \leq n, 1 \leq j \leq 4\} \cup \{q_0, q', q_{n+1,1}\}$ . We construct the formula  $\varphi_\phi$  over  $X_\phi \cup Y \cup Z \cup Q$  from  $\phi$  as follows.

$$\begin{aligned} \varphi_\phi \stackrel{\text{def}}{=} & \bigwedge_{i=1}^m \left( \bigvee_{x_{j,k} \in C_i} \overline{x_{j,k}} \right) \wedge \bigwedge_{i=1}^m \left( \bigwedge_{x_{j,k} \in C_i} (y_i \vee \overline{x_{j,k}}) \right) \\ & \wedge (\overline{y_1} \vee \overline{y_2} \vee z_1) \wedge \bigwedge_{\ell=1}^{m-3} (\overline{z_\ell} \vee \overline{y_{\ell+2}} \vee z_{\ell+1}) \wedge (\overline{z_{m-2}} \vee \overline{y_m} \vee q') \\ & \wedge \bigwedge_{i=1}^n ((\overline{x_{i,1}} \vee x_{i,2} \vee \overline{q_{i,1}})(\overline{x_{i,2}} \vee x_{i,3} \vee \overline{q_{i,2}})(\overline{x_{i,3}} \vee x_{i,4} \vee \overline{q_{i,3}})(\overline{x_{i,4}} \vee x_{i,1} \vee \overline{q_{i,4}})) \\ & \wedge (q_0 \vee \overline{q_{1,1}}) \wedge \bigwedge_{i=1}^n ((q_{i,1} \vee \overline{q_{i,2}})(q_{i,2} \vee \overline{q_{i,3}})(q_{i,3} \vee \overline{q_{i,4}})(q_{i,4} \vee \overline{q_{i+1,1}})) \\ & \wedge (q_{n+1,1} \vee \overline{q_0} \vee \overline{q'}). \end{aligned}$$

Each clause in  $\bigwedge_{i=1}^m (\bigvee_{x_{j,k} \in C_i} \overline{x_{j,k}})$  has three negative literals since each clause  $C_i \in C_\phi$  has three positive literals. Other clauses in  $\varphi_\phi$  have at most one positive literal and at most two negative literals. Hence,  $\varphi_\phi$  is a 3-Horn formula. The number of occurrences of each variable  $x_{j,k} \in X_\phi$  and  $y_i \in Y$  is exactly four, and each variable  $q_{j,k} \in Q \setminus \{q_0, q', q_{n+1,1}\}$  appears exactly three times. Also, each variable  $z_i \in Z$  and  $q_0, q', q_{n+1,1}$  appears exactly twice. Therefore,  $\varphi_\phi$  is an instance of DISCONN 3-HORN-4.

We show the following claim about the satisfying assignment of  $\varphi_\phi$ .

▷ **Claim 21.** Let  $\alpha$  be an arbitrary NAE-satisfying assignment of  $\phi$ , and let  $\alpha'$  be a satisfying assignment of  $\varphi_\phi$  such that  $\alpha(x_{i,1}) = \alpha(x_{i,2}) = \alpha(x_{i,3}) = \alpha(x_{i,4}) = \alpha_i$  for all  $i$  ( $1 \leq i \leq n$ ), then  $\alpha'$  assigns

- (a) the value 1 to all the variables  $y_i \in Y$ ,
- (b) the value 1 to all the variables  $z_i \in Z$  and the variable  $q'$ ,
- (c) the same value to all the variables  $q_{i,j} \in Q$  and the variable  $q_0$ .

**Proof.** From  $\bigwedge_{i=1}^m (\bigwedge_{x_{j,k} \in C_i} (y_i \vee \overline{x_{j,k}}))$  and  $\alpha$  assigns 1 to at least one variable in each  $C_i \in C_\phi$ ,  $\alpha'$  must assign 1 to all the variables  $y_i \in Y$ . In this case, the second line of  $\varphi_\phi$  forces all the values of  $z_i \in Z$  and  $q'$  to be assigned 1 for satisfying  $\varphi_\phi$ . Furthermore,  $q' = 1$  forces the values of all variables  $q_0, q_{i,j} \in Q$  to be the same from the last two lines of  $\varphi_\phi$ . ◁

For simplicity of proving the correctness of our reduction, we further transform  $\varphi_\phi$  to  $\Phi_{\varphi_\phi}$  as follows, recalling Equation (1):

$$\begin{aligned}
\Phi_{\varphi_\phi} = & \varphi_\phi \wedge \bigwedge_{i=1}^m \left( \overline{y_i} \vee \bigvee_{x_{j,k} \in C_i} x_{j,k} \right) \\
& \wedge (\overline{z_1} \vee y_1 y_2) \wedge \bigwedge_{\ell=1}^{m-3} (\overline{z_{\ell+1}} \vee y_{\ell+2} z_\ell) \wedge (\overline{q'} \vee y_m z_{m-2}) \\
& \wedge \bigwedge_{i=1}^n ((\overline{x_{i,2}} \vee x_{i,1} q_{i,1}) (\overline{x_{i,3}} \vee x_{i,2} q_{i,2}) (\overline{x_{i,4}} \vee x_{i,3} q_{i,3}) (\overline{x_{i,1}} \vee x_{i,4} q_{i,4})) \\
& \wedge (\overline{q_0} \vee q_{1,1}) \wedge \bigwedge_{i=1}^n ((\overline{q_{i,1}} \vee q_{i,2}) (\overline{q_{i,2}} \vee q_{i,3}) (\overline{q_{i,3}} \vee q_{i,4}) (\overline{q_{i,4}} \vee q_{i+1,1})) \\
& \wedge (\overline{q_{n+1,1}} \vee q_0 q').
\end{aligned}$$

We need two auxiliary lemmas to prove the NP-hardness of DISCONN 3-HORN-4.

► **Lemma 22.**  $\Phi_{\varphi_\phi}$  has a non-zero satisfying assignment if  $\phi$  has an NAE-satisfying assignment.

**Proof.** Let  $\alpha$  be an NAE-satisfying assignment of  $\phi$ , and let us construct a satisfying assignment  $\alpha'$  of  $\Phi_{\varphi_\phi}$  from  $\alpha$ . For each  $i$  ( $1 \leq i \leq n$ ), we set the value of each variable  $x_{i,k}$  ( $1 \leq k \leq 4$ ) in  $\alpha'$  to the value of  $x_i$  in  $\alpha$ . Since  $\alpha$  is an NAE-satisfying assignment of monotone formulas, there exists at least one variable  $x_i$  assigned 1. Thus,  $x_{i,1}, x_{i,2}, x_{i,3}$ , and  $x_{i,4}$  are also assigned 1. This implies  $q_{i,1}, q_{i,2}, q_{i,3}$ , and  $q_{i,4}$  must be 1 from the third line of  $\Phi_{\varphi_\phi}$ . From Claim 21, all the variables except for  $x_{i,k}$  must be assigned 1 in  $\alpha'$ . Clearly,  $\alpha'$  is a non-zero assignment and satisfies  $\Phi_{\varphi_\phi}$ . ◀

► **Lemma 23.**  $\phi$  has an NAE-satisfying assignment if  $\Phi_{\varphi_\phi}$  has a non-zero satisfying assignment.

**Proof.** Let  $\alpha$  be an arbitrary non-zero satisfying assignment of  $\Phi_{\varphi_\phi}$ . We consider two cases for assigning to the variable  $q_0$ .

(a) In the case  $\alpha(q_0) = 0$ , we show that  $\Phi_{\varphi_\phi}$  cannot have any non-zero satisfying assignments. We consider the formula  $\Phi'_{\varphi_\phi}$  by removing the formula  $\varphi_\phi$  from  $\Phi_{\varphi_\phi}$ . From the clauses  $(\overline{q_0} \vee q_{1,1}) \wedge \bigwedge_{i=1}^n ((\overline{q_{i,1}} \vee q_{i,2}) (\overline{q_{i,2}} \vee q_{i,3}) (\overline{q_{i,3}} \vee q_{i,4}) (\overline{q_{i,4}} \vee q_{i+1,1})) \wedge (\overline{q_{n+1,1}} \vee q_0 q')$  in  $\Phi'_{\varphi_\phi}$  and  $\alpha(q_0) = 0$ , all variable  $q_{i,j} \in Q$  must be assigned 0 under  $\alpha$ . Then, the formula  $\Phi'_{\varphi_\phi}$  is simplified as follows.

$$\begin{aligned}
& \bigwedge_{i=1}^m \left( \overline{y_i} \vee \bigvee_{x_{j,k} \in C_i} x_{j,k} \right) \wedge (\overline{z_1} \vee y_1 y_2) \wedge \bigwedge_{\ell=1}^{m-3} (\overline{z_{\ell+1}} \vee y_{\ell+2} z_\ell) \wedge (\overline{q'} \vee y_m z_{m-2}) \\
& \wedge \bigwedge_{i=1}^n (\overline{x_{i,2}} \wedge \overline{x_{i,3}} \wedge \overline{x_{i,4}} \wedge \overline{x_{i,1}})
\end{aligned}$$

To satisfy this formula,  $\alpha$  assigns

- (1) the value 0 to all variables  $x_{i,k} \in X_\varphi$  from  $\bigwedge_{i=1}^n (\overline{x_{i,2}} \wedge \overline{x_{i,3}} \wedge \overline{x_{i,4}} \wedge \overline{x_{i,1}})$ ,
- (2) the value 0 to all variables  $y_i \in Y$  from  $\bigwedge_{i=1}^m (\overline{y_i} \vee \bigvee_{x_{j,k} \in C_i} x_{j,k})$ ,
- (3) the value 0 to all variables  $z_i \in Z$  from  $(\overline{z_1} \vee y_1 y_2) \wedge \bigwedge_{\ell=1}^{m-3} (\overline{z_{\ell+1}} \vee y_{\ell+2} z_\ell)$ ,
- (4) the value 0 to  $q'$  from  $(\overline{q'} \vee y_m z_{m-2})$

Hence,  $\alpha$  must be the all-zero satisfying assignment. Therefore,  $\Phi_{\varphi_\phi}$  cannot have any non-zero satisfying assignment.

- (b) In the case  $\alpha(q_0) = 1$ , from the clauses  $(\overline{q_0} \vee q_{1,1}) \wedge \bigwedge_{i=1}^n ((\overline{q_{i,1}} \vee q_{i,2})(\overline{q_{i,2}} \vee q_{i,3})(\overline{q_{i,3}} \vee q_{i,4})(\overline{q_{i,4}} \vee q_{i+1,1})) \wedge (\overline{q_{n+1,1}} \vee q_0 q')$  in  $\Phi_{\varphi_\phi}$  and  $\alpha(q_0) = 1$ , all variables  $q_{i,j}$  and  $q' \in Q$  must be assigned 1 under  $\alpha$ . Then,  $(\overline{z_1} \vee y_1 y_2) \wedge \bigwedge_{\ell=1}^{m-3} (\overline{z_{\ell+1}} \vee y_{\ell+2} z_\ell) \wedge (\overline{q'} \vee y_m z_{m-2})$  is simplified to  $(\overline{z_1} \vee y_1 y_2) \wedge \bigwedge_{\ell=1}^{m-3} (\overline{z_{\ell+1}} \vee y_{\ell+2} z_\ell) \wedge (y_m z_{m-2})$ . To satisfy these clauses, the values of all variables  $y_i \in Y$  and  $z_i \in Z$  are forced to 1 under  $\alpha$ . Then, the formula  $\Phi_{\varphi_\phi}$  is simplified as follows:

$$\begin{aligned} & \bigwedge_{i=1}^m \left( \left( \bigvee_{x_{j,k} \in C_i} \overline{x_{j,k}} \right) \wedge \left( \bigvee_{x_{j,k} \in C_i} x_{j,k} \right) \right) \\ & \wedge \bigwedge_{i=1}^n ((\overline{x_{i,1}} \vee x_{i,2})(\overline{x_{i,2}} \vee x_{i,3})(\overline{x_{i,3}} \vee x_{i,4})(\overline{x_{i,4}} \vee x_{i,1})) \\ & \wedge \bigwedge_{i=1}^n ((x_{i,1} \vee \overline{x_{i,2}})(x_{i,2} \vee \overline{x_{i,3}})(x_{i,3} \vee \overline{x_{i,4}})(x_{i,4} \vee \overline{x_{i,1}})). \end{aligned}$$

Using  $(x \vee \overline{y})(\overline{x} \vee y) = (x \leftarrow y)(x \rightarrow y) = (x \leftrightarrow y)$ , then the above formula is

$$\begin{aligned} & \bigwedge_{i=1}^m \left( \left( \bigvee_{x_{j,k} \in C_i} \overline{x_{j,k}} \right) \wedge \left( \bigvee_{x_{j,k} \in C_i} x_{j,k} \right) \right) \\ & \wedge \bigwedge_{i=1}^n ((x_{i,1} \leftrightarrow x_{i,2})(x_{i,2} \leftrightarrow x_{i,3})(x_{i,3} \leftrightarrow x_{i,4})(x_{i,4} \leftrightarrow x_{i,1})). \end{aligned}$$

To satisfy this formula, each variable  $x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}$  must be assigned the same value from  $\bigwedge_{i=1}^n ((x_{i,1} \leftrightarrow x_{i,2})(x_{i,2} \leftrightarrow x_{i,3})(x_{i,3} \leftrightarrow x_{i,4})(x_{i,4} \leftrightarrow x_{i,1}))$ . We now construct an assignment  $\alpha'$  by setting  $\alpha'_i$  to the value of  $x_{i,1}$  in  $\alpha$ . The first line of the above formula assures that the assignment  $\alpha'$  assigns 0 to at least one variable and 1 to at least one variable in each clause  $C_i \in C_\phi$ . Thus,  $\alpha'$  is an NAE-satisfying assignment of  $\phi$ .

From the above argument, this lemma holds.  $\blacktriangleleft$

We are ready to prove the following Theorem.

► **Theorem 24.** DISCONN 3-HORN-4 is NP-complete.

**Proof.** Let  $\phi$  be an instance of MONOTONE NAE E3-SAT-E4. Let  $\varphi_\phi$  be a 3-Horn formula constructed from  $\phi$ , and  $\Phi_{\varphi_\phi}$  be the formula constructed from  $\varphi_\phi$ . From Lemma 10, if  $\Phi_{\varphi_\phi}$  has a non-zero satisfying assignment, the solution graph  $G_{\varphi_\phi}$  is disconnected. Thus, DISCONN 3-HORN-4 belongs to NP. Lemma 22 and Lemma 23 imply that DISCONN 3-HORN-4 is NP-hard. Therefore, DISCONN 3-HORN-4 is NP-complete.  $\blacktriangleleft$

The coNP-completeness of CONN 3-HORN-4 immediately follows from Theorem 24. We then show the following Theorem.

► **Theorem 4.** CONN 3-HORN is coNP-complete even when each variable appears exactly four times.

**Proof.** We prove the coNP-hardness of CONN 3-HORN-E4 by a polynomial-time reduction from CONN 3-HORN-4. We first show the construction of a 3-Horn formula with each variable appearing exactly four times from an instance of CONN 3-HORN-4. Let  $\phi$  be an instance of CONN 3-HORN-4. Then, for every variable  $v$  which appears three times, we introduce new variables  $v_1, v_2$  and add four clauses  $(\overline{v_1} \vee \overline{v_2})(\overline{v_1} \vee v_2)(v_1 \vee \overline{v_2})(v_1 \vee v_2 \vee \overline{v})$ . For every

variable that appears at most twice, we repeat the same operation until each variable appears four times. Thus, we obtain an instance  $\varphi$  of CONN 3-HORN-E4. Then, we transform  $\phi$  and  $\varphi$  to  $\Phi_\phi$  and  $\Phi_\varphi$ , respectively, for simplicity of proving the correctness of our reduction. From the above construction,  $\Phi_\varphi$  has unit clauses with negative literals of all added variables. Therefore,  $\Phi_\varphi$  is equivalent to  $\Phi_\phi$  by assigning 0 to all added variables of  $\varphi$ . It means that  $\Phi_\phi$  has a non-zero satisfying assignment if and only if  $\Phi_\varphi$  has a non-zero satisfying assignment, and by Lemma 10 this completes the proof.  $\blacktriangleleft$

By duality, coNP-completeness of the Boolean connectivity for dual-Horn formulas with the same restrictions follows from the same discussion.

## 5 Conclusion

We proved that CONN 3-HORN remains coNP-complete even when the length of each clause is at most three and each variable appears exactly four times. Furthermore, we provided a polynomial-time algorithm for CONN 3-HORN with each variable appearing at most three times and the length of each clause exactly three. These results seem to give a borderline between P and coNP-complete. However, it remains the complexity of two cases of CONN 3-HORN: (a) the length of each clause is exactly three and each variable appears exactly or at most four times; (b) the length of each clause is at most three and each variable appears exactly or at most three times. The complexity of MONOTONE NAE 3-SAT might be closely related to that of CONN 3-HORN for the following reasons. MONOTONE NAE 3-SAT is polynomial-time solvable when the length of each clause is exactly three and each variable appears at most three times in [11]. MONOTONE NAE 3-SAT remains NP-complete even when the length of each clause is at most three and each variable appears exactly four times. These complexities are equivalent to those of DISCONN 3-HORN with the same restrictions. Furthermore, MONOTONE NAE 3-SAT is NP-complete even when the length of each clause is at most three and each variable appears exactly three times [4]. It also remains NP-complete even when the length of each clause is exactly three and each variable appears exactly four times [3]. Thus, we conjecture that cases (a) and (b) are coNP-complete.

---

## References

- 1 Paul S. Bonsma, Amer E. Mouawad, Naomi Nishimura, and Venkatesh Raman. The Complexity of Bounded Length Graph Recoloring and CSP Reconfiguration. In *Proceedings of the 9th International Symposium on Parameterized and Exact Computation (IPEC 2014)*, volume 8894 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 2014. doi:10.1007/978-3-319-13524-3\_10.
- 2 Jean Cardinal, Erik D. Demaine, David Eppstein, Robert A. Hearn, and Andrew Winslow. Reconfiguration of Satisfying Assignments and Subset Sums: Easy to Find, Hard to Connect. In *Proceedings of 24th International Computing and Combinatorics Conference (COCOON 2018)*, volume 10976 of *Lecture Notes in Computer Science*, pages 365–377. Springer, 2018. doi:10.1007/978-3-319-94776-1\_31.
- 3 Andreas Darmann and Janosch Döcker. On a simple hard variant of Not-All-Equal 3-Sat. *Theoretical Computer Science*, 815:147–152, 2020. doi:10.1016/J.TCS.2020.02.010.
- 4 Ali Dehghan, Mohammad-Reza Sadeghi, and Arash Ahadi. On the Complexity of Deciding Whether the Regular Number is at Most Two. *Graphs and Combinatorics*, 31(5):1359–1365, 2015. doi:10.1007/S00373-014-1446-9.
- 5 Oya Ekin, Peter L. Hammer, and Alexander Kogan. On Connected Boolean Functions. *Discrete Applied Mathematics*, 96-97:337–362, 1999. doi:10.1016/S0166-218X(99)00098-0.

- 6 Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. The Connectivity of Boolean Satisfiability: Computational and Structural Dichotomies. *SIAM Journal on Computing*, 38(6):2330–2355, 2009. doi:10.1137/07070440X.
- 7 Kazuhisa Makino, Suguru Tamaki, and Masaki Yamamoto. On the Boolean connectivity problem for Horn relations. *Discrete Applied Mathematics*, 158(18):2024–2030, 2010. doi:10.1016/J.DAM.2010.08.019.
- 8 Kazuhisa Makino, Suguru Tamaki, and Masaki Yamamoto. An exact algorithm for the Boolean connectivity problem for  $k$ -CNF. *Theoretical Computer Science*, 412(35):4613–4618, 2011. doi:10.1016/j.tcs.2011.04.041.
- 9 Amer E. Mouawad, Naomi Nishimura, Vinayak Pathak, and Venkatesh Raman. Shortest Reconfiguration Paths in the Solution Space of Boolean Formulas. In *Proceedings of the 42nd EATCS International Colloquium on Automata, Languages, and Programming (ICALP 2015)*, volume 9134 of *Lecture Notes in Computer Science*, pages 985–996. Springer, 2015. doi:10.1007/978-3-662-47672-7\_80.
- 10 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability Coding Lemma. *Chicago Journal of Theoretical Computer Science*, 1999. URL: <http://cjtc.cs.uchicago.edu/articles/1999/11/contents.html>.
- 11 Stefan Porschen, Bert Randerath, and Ewald Speckenmeyer. Linear Time Algorithms for Some Not-All-Equal Satisfiability Problems. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2003. doi:10.1007/978-3-540-24605-3\_14.
- 12 Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC 1978)*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.
- 13 Patrick Scharpfenecker. On the Structure of Solution-Graphs for Boolean Formulas. In *Proceedings of the 20th International Symposium (FCT 2015)*, volume 9210 of *Lecture Notes in Computer Science*, pages 118–130. Springer, 2015. doi:10.1007/978-3-319-22177-9\_10.
- 14 Patrick Scharpfenecker and Jacobo Torán. Solution-Graphs of Boolean Formulas and Isomorphism. *Journal on Satisfiability, Boolean Modelling and Computation*, 10(1):37–58, 2016. doi:10.3233/SAT190113.
- 15 Konrad W. Schwerdtfeger. A Computational Trichotomy for Connectivity of Boolean Satisfiability. *Journal on Satisfiability, Boolean Modelling and Computation*, 8(3/4):173–195, 2014. doi:10.3233/sat190097.