Technical Reports on Mathematical and Computing Sciences: TR-C191
title: On Generating Instances for MAX2SAT with Optimal Solutions
author: Masaki Yamamoto
affiliation:
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
Meguro-ku Ookayama, Tokyo 152-8552, Japan
(masaki.yamamoto@is.titech.ac.jp)

Abstract: A *test instance generator* (an *instance generator* for short) for MAX2SAT is a procedure that produces, given a number n of variables, a 2-CNF formula F of n variables (randomly chosen from some reasonably large domain), and simultaneously provides one of the optimal solutions for F. We propose an outline to design an instance generator using an expanding graph of a certain type, called here an "exact 1/2-enlarger". We first show a simple algorithm for constructing an exact 1/2-enlarger, thereby deriving one concrete polynomial-time instance generator `GEN`. We also show that an exact 1/2-enlarger can be obtained with high probability from graphs *randomly* constructed. From this fact, we propose another type of instance generator `RGEN`; it produces a 2-CNF formula with a solution which is optimal for the formula with high probability. However, `RGEN` produces less structured formulas and much larger class of formulas than `GEN`'s. In fact, we prove the NP-hardness of MAX2SAT over the set of 2-CNF formulas produced by `RGEN`.

1 Introduction

For testing the performance of an algorithm for a given problem, it would be useful if we could generate typical instances of the problem with their solutions. A "test instance generator", an instance generator for short, is a procedure that generates typical instances with their solutions. In this paper, we propose instance generators for MAX2SAT, which is a problem of obtaining a maximum assignment for F, an assignment satisfying the maximum number of clauses of a given a 2-CNF formula.

An *instance generator* for MAX2SAT is a randomized procedure that takes n, the number of variables, and produces a pair of a 2-CNF formula of n variables and its maximum assignment. For a "reasonable" instance generator, we would like to require some hardness, e.g., NP-hardness, for solving MAX2SAT over the formulas produced. Unfortunately, however, the following proposition suggests that there doesn't seem to exist a polynomial-time generator producing such hard instances.

Proposition 1.1 Unless NP = co-NP, there is no polynomial-time instance generator `G` such that MAX2SAT over $\mathcal{F}(\mathtt{G})$ is NP-hard, where $\mathcal{F}(\mathtt{G})$ is a set of 2-CNF formulas produced by `G`.

Thus, instead of "computational hardness", we below consider "structural hardness" for formulas produced. Let F be a 2-CNF formula and T be an assignment. We denote by $\mathrm{unsat}_T(F)$ a set of unsatisfied clauses of F under T. If T is a maximum assignment, $\mathrm{unsat}_T(F)$ is called *hole* of F, and denoted by $\mathrm{hole}(F)$. Since 2SAT is solvable in polynomial time, we should produce formulas with large hole; that is, if a 2-CNF formula F is of size m and has a hole of size a constant c, a maximum satisfying assignment for F can be found by running an algorithm solving 2SAT in at most $\sum_{i=0}^{c} \binom{m}{i}$ times, which is a polynomial of m. Thus, we require our

generator to produce formulas with hole of size more than any constant. On the other hand, we would like to produce formulas with small size, that is, those with few clauses because it is considered relatively easy to solve MAX2SAT on 2-CNF formulas with large size. Thus, the number of clauses of formulas produced should be kept linear to the number of variables. More specifically to say above all, for some constant $\alpha > 0$ and $\beta > 0$, and for every number n of variables, we require the following conditions for all formulas F produced:

$$(1)\ |F| \leq \alpha n \quad \text{and} \quad (2)\ |\mathrm{hole}(F)| \geq \beta |F|,$$

Of course, there is a trivial way to produce formulas satisfying these conditions; just producing a set of four clauses $(x \vee y), (x \vee \overline{y}), (\overline{x} \vee y)$, and $(\overline{x} \vee \overline{y})$ for pairs x and y of variables. This case bears $\beta = 0.25$. In this paper, while it is hard to formally specify "nontriviality", we aim at producing nontrivial formulas satisfying these two conditions.

Our idea is to use directed graphs with a certain "expansion" property. Here we explain an outline of producing 2-CNF formulas briefly. Let X be a variable set $\{x_1, \cdots, x_n\}$, and V be a vertex set $\{v_1, \cdots, v_n\}$. We first construct a directed graph G (that has some nice property) over V, and then use it to produce a 2-CNF formula over X as follows: First, we label every vertex of V some literal of X by a (random) permutation π over $\{1, \cdots, n\}$. Then, regarding each edge (v_i, v_j) of G as a logical expression $x_{\pi(i)} \Rightarrow x_{\pi(j)}$, we produce a clause $(\overline{x_{\pi(i)}} \vee x_{\pi(j)})$ equivalent to the logical expression. The 2-CNF formula produced is a collection of such clauses.

We *intuitively* explain how the expansion property is used. Suppose that a 2-CNF formula F over X is obtained from a directed graph $G = (V, E)$ over V. Let us further suppose that all vertices $v_i \in V$ are simply labeled with a positive literal $x_i \in X$. Now consider any assignment T to the variables of X. The vertex set is divided into two sets; namely P and N, which are sets of vertices assigned respectively true and false under the assignment T. Then under T, a clause produced from an edge (v_i, v_j) is unsatisfied if and only if $v_i \in P$ and $v_j \in N$. Hence, the number of unsatisfied clauses under T is the number of edges from P to N, i.e., the number of cut edges.

This observation suggests that in order to have a 2-CNF formula F with large hole, we need a graph with many cut edges for all cuts. More specifically, we would like to have a graph with the following expansion property

$$(*)\ |S \times (V - S) \cap E| \ \geq\ \delta |S| \quad \text{and} \quad (**)\ |(V - S) \times S \cap E| \ \geq\ \delta |S|.$$

Furthermore, in order to keep the number of clauses linear to n, we require that the number of edges is linearly bounded by n, say cn for some constant c. We call a graph with such properties an (n, c, δ)-*enlarger*. Recall that we also provide a maximum assignment at the same time. For this end, we need to provide a cut satisfying $(*)$ and $(**)$ both at equality. (See Definition **??**.) An (n, c, δ)-enlarger having such a cut is called an *exact* (n, c, δ)-*enlarger*. Once an exact (n, c, δ)-enlarger is given, by following the outline explained above, we can produce a 2-CNF formula and its maximum assignment; we actually use two exact (n, c, δ)-enlargers of $\delta = 1/2$. (Below, an exact $(n, c, 1/2)$-enlarger will be simply called an exact 1/2-enlarger.)

For presenting a concrete generator following the outline above, we first show an algorithm that, for a given n, constructs an exact 1/2-enlarger within polynomial time of n. The algorithm is based on the explicit construction of expanders [3]. Then, we give our polynomial-time instance generator `GEN` that produces 2-CNF formulas satisfying the conditions (1) and (2). We also investigate the probability that an exact 1/2-enlarger is constructed from random graphs, that

is equivalent to the probability that every random graph $G = (V, E)$ has some expansion property (see Lemma 3.1) so that those random graphs derive an exact 1/2-enlarge. We show that under a certain edge degree, this probability is high. Using this graph G constructed from random graphs, and following the outline above, we can again obtain a 2-CNF formula and its maximum assignment if the graph G is indeed an exact 1/2-enlarger. This gives us a slightly weaker instance generator `RGEN`; while `RGEN` produces a 2-CNF formula and its maximum assignment with high probability, there is some chance that the assignment provided with a formula is not optimal, which happens rarely. On the other hand, compared with formulas produced by `GEN`, those produced by `RGEN` have the following advantages: (i) they have less structure, and (ii) they have a better hole ratio β. We also show that `RGEN` produces formulas with reasonable hardness, by demonstrating that MAX2SAT is NP-hard on the set $\mathcal{F}$(`RGEN`), the set of formulas produced by `RGEN`. From Proposition 1.1, this result about hardness would be impossible for $\mathcal{F}$(`G`) such that `G` *always* produces a pair of formulas and its maximum assignment.

2 Preliminaries

In this section, we first recall about MAX2SAT. In what follows, we use $x_1, \cdots, x_n$ to denote Boolean *variables*, and use X to denote the set of given Boolean variables. A *literal* over X is a variable $x_i \in X$ or a negated variable $\overline{x_i}$ of $x_i \in X$, and a *clause* is a disjunction of literals, e.g., $(x_1 \vee \overline{x_2} \vee x_3)$. The size of a clause is the number of literals in the clause. A *CNF* (abbrev. for *Conjunctive Normal Form*) formula is a conjunction of clauses, e.g., $C_1 \wedge C_2 \wedge \cdots$, where each C_i, $i = 1, 2, \cdots$, is a clause. A *k-CNF* formula is a CNF formula such that the size of each clause is at most k. In this paper, we will focus on 2-CNF formulas. In particular, we will deal with 2-CNF formulas such that every clause has exactly two literals.

A *truth assignment* (or simply, an *assignment*) over X is a map from X to $\{0, 1\}$. We say that an assignment satisfies a clause if it is evaluated 1 under the assignment. If an assignment simultaneously satisfies all clauses in a formula, the formula is called *satisfiable*. Let F be a 2-CNF formula. An assignment for F is called a *maximum assignment* if the number of satisfied clauses by the assignment is maximum of all assignments.

We next review an *expander* used for producing instances, and obtain a precise value about the edge degree of desired graphs. Let $G = (U, V, E)$ be an undirected bipartite graph such that $U = \{u_1, \cdots, u_n\}$, and $V = \{v_1, \cdots, v_n\}$. We denote by $\Delta(S)$ for $S \subset U$ the number of edges from S to vertices of V having the different index numbers from S; that is,

$$\Delta(S) \;=\; \{(u_i, v_j) \in E \mid u_i \in S \text{ and } v_j \in V \text{ and } i \neq j\}.$$

Definition 2.1 An (n, c, δ)-*expander* is an undirected bipartite graph $G = (U, V, E)$ such that $|U| = |V| = n$, $|E| \leq cn$, and

$$\forall S \subset U \left[|\Delta(S)| \geq \delta \cdot \left(1 - \frac{|S|}{n}\right) |S| \right].$$

An expander may have multiple edges.

Our definition of an expander is a little different from [3]'s; The definition above is w.r.t. $|\Delta(S)|$, i.e., the number of *edges* emanated from S while an expander of [3] is defined w.r.t. the number of *neighbors*, which are defined as *vertices* of V connected to vertices of S. But for our definition, we can also obtain the same theorem as [3]'s.

According to [3], Margulis [2] first found a way to explicitly construct a family of expanders for $c = 5$ although the constant value δ was not presented. After that, Gabber and Galil [3] proposed similar constructions for some specified value of δ. Here we explain one of their constructions, which we will use for our instance generators.

Let $n = h^2$ for any integer $h \geq 1$, and A be $\{0, 1, \cdots, h-1\} \times \{0, 1, \cdots, h-1\}$. Consider the following seven permutations $\sigma_0, \cdots, \sigma_6$ on A:

$$\begin{aligned}
&\sigma_0(x, y) = (x, y), \\
&\sigma_1(x, y) = (x, y + 2x), \quad \sigma_2(x, y) = (x, y + 2x + 1), \quad \sigma_3(x, y) = (x, y + 2x + 2), \\
&\sigma_4(x, y) = (x + 2y, y), \quad \sigma_5(x, y) = (x + 2y + 1, y), \quad \sigma_6(x, y) = (x + 2y + 2, y),
\end{aligned}$$

where the $+$ is modulo h. Then, we define a bipartite graph $G = (U, V, E)$ as follows: $U = V = A$, and $E = \{((x, y), (x', y')) | (x', y') = \sigma_i(x, y) \text{ for some } i,\ 0 \leq i \leq 6\}$. We call here this construction *GG method*. Note that G may have multiple edges between some pairs of vertices $(x, y) \in U$ and $(x', y') \in V$. It is clear that we can easily construct such a graph G from a given n. Gabber and Galil showed that G is indeed an expander of the following values:

Theorem 2.1 (Gabber and Galil [3]) For any n, the graph G defined above is an $(n, 7, \delta)$-expander where $\delta = (2 - \sqrt{3})/2$.

From now on, we simply call an (n, c, δ)-expander a δ-expander in the case that the values n and c are given in the context.

As is shown in the next section, our generators need δ-expanders such that $\delta \geq 2/3$. The graph G given by GG method is not such one because the ratio $\delta' = (2 - \sqrt{3})/2$ is less than $2/3$. But, note that our definition of an expander is w.r.t. the number of edges (*not* vertices). Thus, by merging the number l of δ'-expanders where $l = \lceil (2/3) \div \delta' \rceil = 5$, we can easily obtain a $2/3$-expander. More specifically, by using random permutations on vertices, we construct l isomorphic δ'-expanders and merge them to a single graph, thereby yielding a $2/3$-expander. Note that this resulting expander has $l \cdot 7n$ $(= 35n)$ edges; that is, it is an $(n, 35, 2/3)$-expander. In the following sections, we denote by (n, c_0, δ_0)-expander such an expander, where $c_0 = 35$ and $\delta_0 = 2/3$.

3 Explicit Construction of Exact Enlarger

In this section, we introduce our new notions called an "enlarger" and an "exact enlarger", and show their explicit constructions.

As is defined in the previous section, an expander $G = (U, V, E)$ is an *undirected bipartite* graph with a property that every subset S of U has many edges between S and V. Intuitively, an "enlarger" is a *directed graph* having such a property.

Definition 3.1 An (n, c, δ)-*enlarger* $G = (V, E)$ is a directed graph such that $|V| = n$, $|E| \leq cn$, and

$$\forall S \subset V :\ |S| \leq \frac{n}{2} \quad \Big[\ |S \times (V - S) \cap E| \geq \delta |S|, \ \text{ and } \ |(V - S) \times S \cap E| \geq \delta |S| \ \Big].$$

An enlarger may have multiple edges, but doesn't have any self-loops.

Again, we simply call an (n, c, δ)-enlarger a δ-enlarger in the case that the values n and c are given in the context.

Here, we show the construction of an $(n, 2c_0, \delta_0)$-enlarger from two isomorphic (n, c_0, δ_0)-expanders. Recall that for constructing δ-expanders by `GG` method, the number n of vertices must be h^2 for some integer h. In the following construction as well, we assume such an input number n.

Algorithm 1 Construction of Enlarger (input n: a number of vertices)

step 1: By `GG` method, construct an (n, c_0, δ_0)-expander $G_{out} = (U, W, E_{out})$, where $U = \{u_1, \cdots, u_n\}$ and $W = \{w_1, \cdots, w_n\}$.

step 2: Choose a random permutation π over W, and construct an (n, c_0, δ_0)-expander $G_{in} = (U, W_\pi, E_{in})$, where $U = \{u_1, \cdots, u_n\}$ and $W_\pi = \{w_{\pi(1)}, \cdots, w_{\pi(n)}\}$.

step 3: Construct a directed graph $G = (V, E)$ such that $V = \{v_1, \cdots, v_n\}$ and $E = E' \cup E''$, where

$$\begin{aligned} E' &= \{(v_i, v_j) \ : \ (u_i, w_j) \in E_{out} \text{ and } i \neq j\}, \quad \text{and} \\ E'' &= \left\{(v_{\pi(j)}, v_i) \ : \ (u_i, w_{\pi(j)}) \in E_{in} \text{ and } i \neq \pi(j)\right\}. \end{aligned}$$

In this algorithm, we use a permutated W_π in order to prevent graphs constructed (thus, formulas produced) from being trivial. We could use a graph G' identical to G_{out} as G_{in}. But an enlarger G constructed from G_{out} and G' has a very simple structure; that is, the graph G has an edge (u, v) if and only if it has an edge (v, u). An instance generator which use such an simple graph would result in producing trivial instances.

Lemma 3.1 For any graph $G = (V, E)$ constructed by Algorithm 1, we have that for any subset S of V

$$|S \times (V - S) \cap E| \ \geq \ 2\delta_0 \left(1 - \frac{|S|}{n}\right) |S|, \quad \text{and} \quad |(V - S) \times S \cap E| \ \geq \ 2\delta_0 \left(1 - \frac{|S|}{n}\right) |S|.$$

Proof. Since $G_{out} = (U, W, E_{out})$ is a δ_0-expander, for any subset S of V, we have that

$$|S \times (V - S) \cap E_{out}| \ \geq \ \delta_0 \left(1 - \frac{|S|}{n}\right) |S|.$$

Similarly, since $G_{in} = (U, W_\pi, E_{in})$ is also a δ_0-expander, by the construction, for any subset S of V, we have that

$$|S \times (V - S) \cap E_{in}| \ \geq \ \delta_0 \left(1 - \frac{|V - S|}{n}\right) |V - S| \ = \ \delta_0 \left(1 - \frac{|S|}{n}\right) |S|.$$

Therefore, by combining two inequalities, we have

$$\begin{aligned} |S \times (V - S) \cap E| &= |S \times (V - S) \cap E_{out}| \ + \ |S \times (V - S) \cap E_{in}| \\ &\geq 2\delta_0 \left(1 - \frac{|S|}{n}\right) |S|. \end{aligned}$$

We can obtain the same result for $|(V - S) \times S \cap E|$. ■

Note that $2\delta_0(1-|S|/n)\cdot|S| \geq \delta_0|S|$ for all S of size up to $n/2$. Thus, by this lemma, we can see that any graph $G=(V,E)$ constructed by Algorithm 1 is an $(n, 2c_0, \delta_0)$-enlarger.

By definition of enlarger, for any subsets $S,T \subset V$ such that $S \cup T = V$, $S \cap T = \phi$, and $|S| = |T|$, the following inequalities hold: $|S \times T \cap E| \geq \lceil \delta n/2 \rceil$ and $|T \times S \cap E| \geq \lceil \delta n/2 \rceil$. For our generators, among all those partitions of (S,T), we need a δ-enlarger having an "optimal" partition for which the equalities hold in the inequalities above, which we call an "exact enlarger".

Definition 3.2 An *exact* (n,c,δ)*-enlarger* $G=(V,E)$ is an (n,c,δ)-enlarger which has a partition (U,W) (i.e., $U \cup W = V$ and $U \cap W = \phi$) with $|U| = |W|$ such that

$$|U \times W \cap E| = \left\lceil \delta \cdot \frac{n}{2} \right\rceil, \quad \text{and} \quad |W \times U \cap E| = \left\lceil \delta \cdot \frac{n}{2} \right\rceil.$$

Here, we show the construction of an exact $(n, 2c_0{+}1/2, 1/2)$-enlarger from two $(n/2, 2c_0, \delta_0)$-enlargers. Recall that for constructing any $(n/2, c, \delta)$-enlarger by Algorithm 1, the number $n/2$ of vertices must be h^2 for some integer h. Thus, in the following construction we assume that an input number n satisfies $n = 2h^2$ for some *even* h.

Algorithm 2 Construction of Exact Enlarger (input n: a number of vertices)

step 1: By Algorithm 1, construct two $(n/2, 2c_0, \delta_0)$-enlargers $L = (U, E_l)$ and $R = (W, E_r)$, where $U = \{u_1, \cdots, u_{n/2}\}$ and $W = \{w_1, \cdots, w_{n/2}\}$.

step 2: Construct a graph $G=(V,E)$ defined as $V = U \cup W$ and $E = E_l \cup E_r \cup A \cup B$, where

$$A \;=\; \{(u_i, w_i) \;:\; 1 \leq i \leq n/4\}, \quad \text{and} \quad B \;=\; \{(w_i, u_i) \;:\; n/4+1 \leq i \leq n/2\}.$$

In what follows, we call a correspondence between U and W defined by A and B above a *one-to-one balanced correspondence*.

Lemma 3.2 Any graph $G=(V,E)$ constructed by Algorithm 2 is an $(n, 2c_0{+}1/2, 1/2)$-enlarger.

Proof. Consider a graph $G=(V,E)$ constructed by the algorithm with $|V| = n$. We first count the number of edges of G. Since G is composed of two $(n/2, 2c_0, \delta_0)$-enlargers and additional $n/2$ edges, the number of edges is $2 \cdot (2c_0 \cdot n/2) + n/2$, which is $(2c_0 + 1/2)|V|$, where $|V| = n$.

Next, we show that G is a $1/2$-enlarger. Consider an arbitrary $S \subset V$ of size at most $n/2$. Let $S = S_U \cup S_W$, where $S_U \subset U$ and $S_W \subset W$. We count the number of edges from S to $V - S$. (A similar argument holds for $|(V-S) \times S \cap E|$.) Since $|S| \leq n/2$, and the construction of G is symmetric, it suffices to prove for the following two cases:

$$(I):\ |S_U|, |S_W| \leq \frac{n}{4}, \quad \text{and} \quad (II):\ |S_U| \geq \frac{n}{4} \ \text{ and } \ |S_W| \leq \frac{n}{4}.$$

In the first case, it follows from the definition of δ_0-expander that

$$|S_U \times (U - S_U) \cap E_l| \;\geq\; \delta_0|S_U|, \quad \text{and} \quad |S_W \times (W - S_W) \cap E_r| \;\geq\; \delta_0|S_W|.$$

By combining the two inequalities, we have that

$$|S \times (V-S) \cap (E_l \cup E_r)| \;\geq\; \delta_0(|S_U| + |S_W|).$$

Therefore, the total number of edges from S to $V - S$ is,

$$\begin{aligned} |S \times (V-S) \cap E| &= |S \times (V-S) \cap (E_l \cup E_r \cup A \cup B)| \\ &\geq |S \times (V-S) \cap (E_l \cup E_r)| \\ &\geq \delta_0(|S_U| + |S_W|) \geq |S|/2 \quad (\because \delta_0 = 2/3)\,. \end{aligned}$$

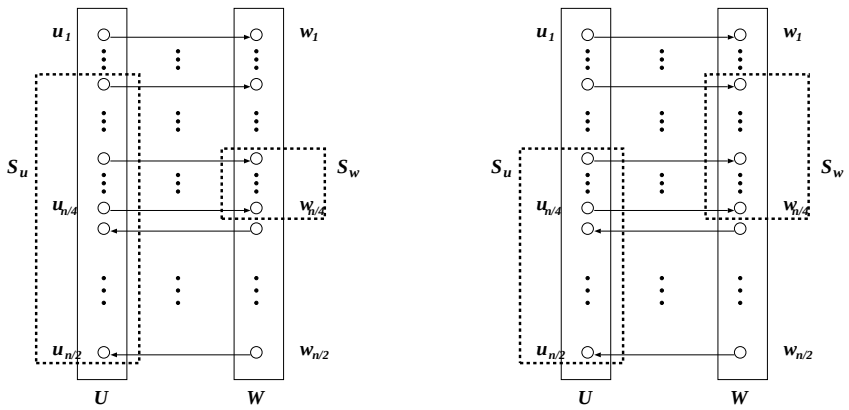
In the second case, by Lemma 3.1, we have that

$$\begin{aligned} |S_U \times (U - S_U) \cap E_l| &\geq 2\delta_0 \left(|S_U| - \frac{|S_U|^2}{n/2} \right), \quad \text{and} \\ |S_W \times (W - S_W) \cap E_r| &\geq 2\delta_0 \left(|S_W| - \frac{|S_W|^2}{n/2} \right). \end{aligned}$$

By combining the two inequalities, we have that

$$|S \times (V-S) \cap (E_l \cup E_r)| \geq 2\delta_0 \left(|S_U| + |S_W| - \frac{|S_U|^2}{n/2} - \frac{|S_W|^2}{n/2} \right). \tag{1}$$

This bound is the number of edges from S to $V - S$ *within* the subgraphs L and R; There are some more edges, that is, those of $A \cup B$. For obtaining the lower bound for the number of those additional edges, we let $|S_U| = n/4 + nx/4$ and $|S_W| = ny/4$ by parameters x and y. Then, for fixed values of x and y such that $0 \leq x, y \leq 1$, we estimate the minimum number of those edges. From the construction of A and B, it is easy to see that the following claim holds. (See the figure below.)



Case (i) Case (ii)

Figure 1: Two Cases

Claim 1 For fixed values of x and y, $|S \times (V-S) \cap (A \cup B)|$ is minimized in the case that $S_U \supset \{u_{n/4+1}, \cdots, u_{n/2}\}$ and $S_W \subset \{w_1, \cdots, w_{n/4}\}$, and one of the followings holds:

$$\begin{aligned} &\text{Case (i)}: \quad \forall (u,w) \in A\,[w \in S_W \Rightarrow u \in S_U]\,, \\ &\text{Case (ii)}: \quad \forall (u,w) \in A\,[u \in S_U \Rightarrow w \in S_W]\,. \end{aligned}$$

Note that Case (i) is for $x \geq y$ and Case (ii) is for $x \leq y$. We show that G is a 1/2-enlarger for any x, y, $0 \leq x, y \leq 1$ in both cases. In Case (i), it is easy to see from Fig.1 that

$|S \times (V-S) \cap A| = (x-y) \cdot n/4$. By combining this equality and inequality (1), we have that

$$\begin{aligned} |S \times (V-S) \cap E| &\geq 2\delta_0 \left(|S_U| + |S_W| - \frac{|S_U|^2}{n/2} - \frac{|S_W|^2}{n/2} \right) + \frac{x-y}{4} n \\ &= \frac{n}{4} \left(\delta_0(1 - x^2 - y^2 + 2y) + x - y \right). \end{aligned}$$

Recall $|S| = n/4 + (n/4)x + (n/4)y$; thus, for proving that G is a 1/2-enlarger, it suffices to show

$$\frac{n}{4} \cdot \left(\delta_0(1 - x^2 - y^2 + 2y) + x - y \right) \geq \frac{1}{2} \cdot \left(\frac{n}{4} + \frac{n}{4}x + \frac{n}{4}y \right).$$

Applying $\delta_0 = 2/3$, the inequality above is equivalent to the following:

$$\left(x - \frac{3}{8} \right)^2 + \left(y + \frac{1}{8} \right)^2 \leq \frac{13}{32}. \tag{2}$$

That means G is a 1/2-enlarger if (x, y) is within the circle defined by the above. On the other hand, in Case (i), x and y must satisfy the following bounds: $y \geq 0$, $y \leq x$, and $y \leq -x + 1$. (The last bound follows from $|S| = |S_U| + |S_W| \leq n/2$.) In other words, the domain of (x, y) is within the triangle defined by these three inequalities. Note that this triangle is within the circle defined by (2) since its three vertices $(0, 0)$, $(1, 0)$, $(1/2, 1/2)$ all satisfy (2). Thus, any pair of x and y in the domain satisfies (2).

In Case (ii), it is easy to see from Fig.1 that $|S \times (V-S) \cap A| = 0$. By combining this equality and inequality (1), we have that

$$|S \times (V-S) \cap E| = \frac{n}{4} \cdot \delta_0(1 - x^2 - y^2 + 2y).$$

Likewise to Case (i), for proving that G is a 1/2-enlarger, it suffices to show

$$\frac{n}{4} \cdot \delta_0(1 - x^2 - y^2 + 2y) \geq \frac{1}{2} \cdot \left(\frac{n}{4} + \frac{n}{4}x + \frac{n}{4}y \right),$$

which is equivalent to

$$\left(x + \frac{3}{8} \right)^2 + \left(y - \frac{5}{8} \right)^2 \leq \frac{25}{32}. \tag{3}$$

On the other hand, in Case (ii) x and y must satisfy the following bounds: $x \geq 0$, $y \geq x$, and $y \leq -x + 1$. Again, any pair of x and y in the domain defined by these inequality satisfies (3).

Thus, $G = (V, E)$ is a 1/2-enlarger with n vertices and $(2c + 1/2) \cdot n$ edges, that is, an $(n, 2c + 1/2, 1/2)$-enlarger. ■

From the construction, it is clear that (U, W) is a partition for which the number of edges between U and W is exactly $1/2 \cdot n/2$ in both directions. This proves the following main theorem of this section.

Theorem 3.3 For any n such that $n = 2h^2$ for some even h, the graph $G = (V, E)$ constructed by Algorithm 2 with input n, is an exact $(n, 2c_0 + 1/2, 1/2)$-enlarger.

4 Generating Algorithms

In this section, we first describe an outline of generators for MAX2SAT by showing a generator which uses exact 1/2-enlargers shown in the previous section. Then, we discuss the possibility of using random graphs; that is, the maximum assignment provided with a formula is optimal with high probability.

4.1 An outline of instance generators

For producing a 2-CNF formula of n variables, we use exact 1/2-enlargers having n vertices. Recall our construction of an exact 1/2-enlarger; Since we require the number n to be h^2 for some even h, we also assume this condition on n below.

Algorithm 3 Constructive Generator – `GEN` – (input n: a number of variables)

step 1: Divide a vertex set $V = \{v_1, \cdots, v_n\}$ into two sets $U = \{v_1, \cdots, v_{n/2}\}$ and $W = \{v_{n/2+1}, \cdots, v_n\}$.

step 2: By Algorithm 2, construct independently two exact $(n, 2c_0 + 1/2, 1/2)$-enlargers $G_k = (V, E_k)$ for $k = 1, 2$.

step 3: Let X be a set of variables of size n. Choose a random assignment $t : t_1, \cdots, t_n \in \{0, 1\}$ to X and a random permutation π over X. Then, label each vertex of the two graphs as follows:

$$l_1(v_i) = x_{\pi(i)}^{t_i} \quad \text{for } v_i \in V \text{ of } G_1, \quad \text{and} \quad \begin{cases} l_2(v_i) = x_{\pi(i)}^{t_i} & \text{for } v_i \in U \text{ of } G_2, \\ l_2(v_i) = x_{\pi(i)}^{t_i \oplus 1} & \text{for } v_i \in W \text{ of } G_2, \end{cases}$$

where $x_{\pi(i)}^{t_i}$ is $x_{\pi(i)}$ if $t_i = 1$, and otherwise $\overline{x_{\pi(i)}}$.

step 4: Define a set F of clauses as follows:

$$F \;=\; \left\{ \left(\overline{l_k(u)} \vee l_k(v) \right) \;:\; (u, v) \in E_k, k = 1, 2 \right\}.$$

Set an assignment T_0 to be $T_0(x_1) = t_1, \cdots$, and $T_0(x_n) = t_n$.

step 5: Output F as a 2-CNF formula and T_0 as a maximum assignment.

Theorem 4.1 For any pair of a 2-CNF formula F and an assignment T_0 produced by `GEN`, we have $\mathrm{hole}(F) = n/4$, and T_0 is a maximum assignment for F.

Proof. We may fix w.l.o.g. the random assignment t and the random permutation π used by `GEN` to, e.g., $t_1 = 1, \cdots, t_n = 1$ and $\pi(1) = 1, \cdots, \pi(n) = n$, respectively. Let T be an arbitrary assignment over X. By the assignment T, we assume that U is partitioned into sets A, B of vertices, and W is partitioned into sets C, D of vertices as follows:

$$\begin{cases} A = \{v_i \in U \;:\; T(x_i) = \text{true}\}, \\ B = \{v_i \in U \;:\; T(x_i) = \text{false}\}, \end{cases} \quad \text{and} \quad \begin{cases} C = \{v_i \in W \;:\; T(x_i) = \text{true}\}, \\ D = \{v_i \in W \;:\; T(x_i) = \text{false}\}. \end{cases}$$

We denote by a, b, c, d the size of A, B, C, D, respectively.

We estimate $|\mathrm{unsat}_T(F)|$, the number of F's clauses unsatisfied by T. Recall that each clause of F is yielded from an edge (u, v) of G_k. Consider the case, for example, that (u, v) is an edge of G_2. In this case, we have that a clause yielded from (u, v) is unsatisfied if and only if $l_2(u) = 1$ and $l_2(v) = 0$ under T, and this is further iff $u \in U_1 \cup W_0$ and $v \in U_0 \cup W_1$, where U_0 and U_1 is a subset of U containing all vertices which are labelled literals assigned under T false and true, respectively. (W_0 and W_1 are similarly defined.) That is, (u, v) is a cut edge from $P = U_1 \cup W_0$ to $N = U_0 \cup W_1$. Since G_2 is an exact 1/2-enlarger, the total number of such edges of G_2 is at least

$$\frac{1}{2} \min\{|U_1| + |W_0|, |U_0| + |W_1|\} \;=\; \frac{1}{2} \min\{a + d, b + c\}.$$

In other words, $\mathrm{unsat}_T(F_{G_2}) \geq \min\{a+b, c+d\}/2$, where F_{G_k} is a set of clauses yielded from edges of G_k. By similar arguments, we also have $\mathrm{unsat}_T(F_{G_1}) \geq \min\{a+b, c+d\}/2$. Thus, for the total number of unsatisfied clauses under T, we have that

$$\begin{aligned}
|\mathrm{unsat}_T(F)| &= |\mathrm{unsat}_T(F_{G_1})| + |\mathrm{unsat}_T(F_{G_2})| \\
&\geq \frac{1}{2}\min\{a+b, c+d\} + \frac{1}{2}\min\{a+d, b+c\} \\
&= \frac{1}{2}\min\{2a+b+d, 2d+a+c, 2b+a+c, 2c+b+d\} \\
&\geq \frac{1}{2}\min\{2a+n/2, 2d+n/2, 2b+n/2, 2c+n/2\} \;\geq\; \frac{n}{4}.
\end{aligned}$$

The derivation of the inequalities above comes from $a+c = b+d = n/2$.

Since each G_k is an exact 1/2-enlarger, that is, the number of edges between U and W is exactly $1/2 \cdot n/2$ in each direction, the assignment T_0 produced by `GEN`, that in this proof assigns true to all variables of X, leads to the number exactly n of unsatisfied clauses. (Those unsatisfied clauses all comes from edges of G_2.) Note that the number of satisfied clauses by T_0 attains the lower bound of $\mathrm{unsat}_T(F)$. Hence, we conclude that $\mathrm{hole}(F) = n/4$, and T_0 is a maximum assignment. ■

As is mentioned in Introduction, we require the ratio of hole to the size of F, i.e., $|\mathrm{unsat}_{T_0}|/|F|$ under the maximum assignment T_0 of F, as large as possible. We calculate the precise value of the ratio. Recall that by Lemma 3.2, an exact 1/2-enlarger has $(2c_0+1/2)n$ edges for n vertices, where $c_0 = 35$. Since the generator uses two exact 1/2-enlargers, and produces one clause from one edge of those graphs, the number m of clauses produced is equal to the total number of edges of two graphs, that is, $m = 2 \cdot (2 \cdot 35 + 1/2)n = 141n$. Thus, the ratio of hole is

$$\frac{\mathrm{hole}(F)}{m} \;=\; \frac{n/4}{141n} \;\approx\; 0.0017730496.$$

Remark 1 We have shown a generator using two exact 1/2-enlargers here. Note that the generator presented here can never produce a clause of the form $(\overline{x_1} \to x_2)$, for example, supposing $t_i = 1$ and $\pi(i) = i$ for all i. However, we can actually present a generator producing every kind of clauses by using six exact 1/2-enlargers. Moreover, it has a little better ratio of hole to m, that is, approximately 0.00236. But, instances produced by the generator using two is sufficient for the argument about hardness discussed in the next section.

4.2 Another generator based on random graphs

The generator `GEN` uses exact 1/2-enlargers based on expanders given by the explicit construction of `GG` method. Even though we use randomness several times in `GEN`, it might be still possible that the formulas produced by `GEN` have a certain structure. On the other hand, it has been known that random graphs, which have no structures, mostly have the expanding property. Here, we'll discuss the possibility of producing instances from random graphs. We use the same outline as the generator `GEN`, called *`RGEN`*; 2-CNF formulas and their maximum assignments are produced from two exact 1/2-enlargers. The only difference from `GEN` is that those exact 1/2-enlargers are constructed from random graphs; That means, for some random graphs used for the construction those are not actually exact 1/2-enlargers so that we can't guarantee the solution provided by `RGEN` is not an optimal. But we show that happens rarely.

Recall the explicit construction of an exact 1/2-enlarger; By Lemma 3.2, we can conclude that the procedure presented in Algorithm 2 outputs an exact 1/2-enlarger *if* for each graph $G = (V, E)$ of the two graphs composing the exact 1/2-enlarger, it satisfies the following conditions for any subset S of V:

$$|S \times (V - S) \cap E| \;\geq\; 2\delta \left(1 - \frac{|S|}{|V|}\right) |S|, \quad \text{and} \tag{4}$$

$$|(V - S) \times S \cap E| \;\geq\; 2\delta \left(1 - \frac{|S|}{|V|}\right) |S|, \tag{5}$$

where $\delta = 2/3$. Note that due to the symmetry, if G satisfies (4) for all subsets of V, so does G (5) for all subsets of V. Thus, in what follows, we will consider only the first condition.

We first estimate the probability that the random graph satisfies (4) where $\delta = 1/2$ (*not* $\delta = 2/3$, for an easier calculation) for any subset of V of size up to $n/2$ (*not* n). After that, using the value, we obtain the probability for $\delta = 2/3$ and size up to n.

Here, we define the probability space for a random graph $G = (V, E)$ of $|V| = n$; For each $u \in V$, we choose d edges emanating from u by sampling *with replacement* d vertices from V independently and uniformly. The sampling with replacement means that there can be multi-edges emanating from a vertex, or even exist a vertex having no edges from itself.

Lemma 4.2 The random graph $G = (V, E)$ described above satisfies (4) where $\delta = 1/2$ for any subset $S \subset V$ of size $\leq n/2$, with probability at least p, where

$$p \;=\; \frac{1 - 2r}{1 - r}, \quad \text{and} \quad r \;=\; de^2 \cdot \left(\frac{1}{2}\right)^{d-5/2}.$$

Proof. Consider an arbitrary subset S of V such that $|S| = s \leq n/2$. We say here that S is *bad* if $|S \times (V - S) \cap E| < \lceil c(s) \rceil$, where $c(s) = 2\delta(1 - s/n) \cdot s$. Since there are d edges emanating from each vertex of S, S is bad if and only if at least $ds - (\lceil c(s) \rceil - 1)$ edges are introverted; that is, there are at least $ds - \lceil c(s) \rceil + 1$ edges whose sources are in S. Thus, we have that

$$\begin{aligned} \Pr\{S \text{ is bad}\} \;&\leq\; \binom{ds}{ds - \lceil c(s) \rceil + 1} \left(\frac{s}{n}\right)^{ds - \lceil c(s) \rceil + 1} \\ &=\; \binom{ds}{\lceil c(s) \rceil - 1} \left(\frac{s}{n}\right)^{ds - \lceil c(s) \rceil + 1}. \end{aligned}$$

Thus, we have the probability that there exists a bad subset of size s,

$$\begin{aligned} \Pr\left\{ \bigcup_{S:|S|=s} \{S \text{ is bad}\} \right\} \;&\leq\; \sum_{S:|S|=s} \Pr\{S \text{ is bad}\} \\ &\leq\; \binom{n}{s} \binom{ds}{\lceil c(s) \rceil - 1} \left(\frac{s}{n}\right)^{ds - \lceil c(s) \rceil + 1} \\ &\leq\; \left(\frac{ne}{s}\right)^{s} \left(\frac{dse}{\lceil c(s) \rceil - 1}\right)^{\lceil c(s) \rceil - 1} \left(\frac{s}{n}\right)^{ds - (\lceil c(s) \rceil - 1)}. \end{aligned}$$

Here, applying $\delta = 1/2$, we have $\lceil c(s) \rceil - 1 \leq (1 - s/n)s$. Since a function $f(x) = (dse/x)^x$

increases for $0 < x < ds$, and $(1 - s/n)s < ds$, we have that

$$\begin{aligned}
\Pr\left\{\bigcup_{S:|S|=s}\{S \text{ is bad}\}\right\} &\leq \left(\frac{ne}{s}\right)^s \left(\frac{dse}{(1-s/n)s}\right)^{(1-s/n)s} \left(\frac{s}{n}\right)^{ds-(1-s/n)s} \\
&= \left(\frac{ne}{s} \cdot \left(\frac{de}{1-s/n}\right)^{1-s/n} \left(\frac{s}{n}\right)^{d-(1-s/n)}\right)^s \\
&\leq \left(de^2 \cdot \left(\frac{s}{n}\right)^{d-2} \left(\frac{1}{1-s/n}\right)^{1-s/n} \left(\frac{s}{n}\right)^{s/n}\right)^s \qquad (6) \\
&\leq \left(de^2 \cdot \left(\frac{1}{2}\right)^{d-2} \left(\frac{1}{1-s/n}\right)^{1-s/n} \left(\frac{s}{n}\right)^{s/n}\right)^s \qquad (7) \\
&\leq \left(de^2 \cdot \left(\frac{1}{2}\right)^{d-5/2}\right)^s. \qquad (8)
\end{aligned}$$

The derivations (6) and (7) come from $de \geq (de)^{1-s/n}$ and $s/n \leq 1/2$, respectively. We can obtain the last derivation (8) from the following fact: $x^x \leq 1$ for $0 \leq x \leq 1/2$, and $x^{-x} \leq (1/2)^{(-1/2)}$ for $1/2 \leq x \leq 1$. Thus, we have that

$$\Pr\{S \subset V \text{ of size } s \text{ is bad}\} \;\leq\; r^s, \quad \text{where } r \;=\; de^2 \cdot \left(\frac{1}{2}\right)^{d-5/2}.$$

Finally, summing up this probability over $1 \leq s \leq n/2$ to obtain the probability that some subset of size $\leq n/2$ is bad, we have that

$$\Pr\left\{\bigcup_{s=1}^{n/2}\{S \subset V \text{ of size } s \text{ is bad}\}\right\} \;\leq\; \sum_{s=1}^{n/2} r^s \;\leq\; \frac{r - r^{n/2+1}}{1-r} \;\leq\; \frac{r}{1-r}.$$

Thus, we have the probability that every subset of size $\leq n/2$ is not bad, that is, the probability that G satisfies (4) where $\delta = 1/2$ for any $S \subset V$ of size up to $n/2$, at least:

$$p \;=\; 1 - \frac{r}{1-r} \;=\; \frac{1-2r}{1-r}.$$

■

Next, we extend the upper bound on the size of S to n; it is easily done by generating another random graph and merge them. More specifically, we generate two random graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, and construct $G = (V, E)$ defined as;

$$E \;=\; \{(u,v) \;:\; (u,v) \in E_1 \text{ or } (v,u) \in E_2\}.$$

We claim that with probability at least p^2, the graph G satisfies (4) for every subset of V. For subsets of size up to $n/2$, G satisfies (4) since G_1 covers it. On the other hand, for subsets of size $> n/2$, G also satisfies (4) since G_2 covers it; That is, $|S \times (V-S) \cap E| \geq |(V-S) \times S \cap E_2|$, and for any subset S of size $\geq n/2$,

$$|(V-S) \times S \cap E_2| \;\geq\; 2\delta\left(1 - \frac{|V-S|}{|V|}\right)|V-S| \;=\; 2\delta\left(1 - \frac{|S|}{|V|}\right)|S|,$$

since G_2 satisfies (5) for subsets of size $\leq n/2$. Hence, $G = (V, E)$ merged in this way satisfies (4) for every subset of V with probability at least p^2.

Finally, we extend the value of δ to 2/3; it is also done in the same way. We generate another random graph and simply merge them. Then, we have that with probability at least p^4, the random graph satisfies (4) with $\delta = 2/3$ for every subset of V. Therefore, we conclude that the probability we obtain one exact 1/2-enlarger from random graphs is at least p^8.

Now, we estimate the specific degree of edges such that the probability that two exact 1/2-enlargers are obtained from random graphs, which is at least $(p^8)^2$, is high. For this end, by taking $d = 15$, for example, we have $p^{16} > 0.992911$; that is, the success probability, the probability of obtaining two exact 1/2-enlargers, is more than 99%. Thus, under this choice of parameter d, we can conclude that, for more than 99% of 2-CNF formulas produced by `RGEN`, the assignments provided with formulas are indeed maximum assignments.

Notice also that the number m of edges (or clauses) is $m = 2 \cdot (15 + 0.5) \cdot n = 31n$ when $d = 15$, which is about one fifth of the number of clauses produced by `GEN`. Thus, the ratio of unsatisfied clauses to m becomes five times in compensation for small chance of providing non-maximum assignments.

5 Hardness of Instances by `RGEN`

We have proposed a constructive generator and a randomized generator; the generators produce formulas and also provide their maximum assignments simultaneously although it is with high probability for the randomized one. This is because the randomized generator uses random graphs to produce formulas. On the other hand, it can be considered that the randomness makes the generator to produce formulas with no structures; that is, hard formulas to solve. In this section, we discuss about the hardness to solve such formulas. We define the problem of those instances as follows:

Balanced MIN-ASSIGNCUT

Instance: Two vertex sets U and W with $|U| = |W|$, and two directed graphs $G_i = (V_i, E_i)$ for $i = 1, 2$ as follows: We define V_i as $U \cup W$. Moreover, for each graph G_i, there exists a one-to-one balanced correspondence between U and W for $i = 1, 2$.

Question: For any subset T of V, let $\mathrm{cost}(T)$ be

$$\mathrm{cost}(T) \;=\; \sum_{i=1,2} c_i, \quad \text{where} \left\{ \begin{array}{l} c_1 = \left|T \times \overline{T} \cap E_1\right| \\ c_2 = \left|S \times \overline{S} \cap E_2\right| \end{array} \right. ,$$

where $S = (T \cap U) \cup (\overline{T} \cap W)$. Then, what is a subset T of V such that $\mathrm{cost}(T)$ is minimized?

As in the sections before, in the problem above, we assume that any directed graph may have multi-edges for any pair of vertices, but doesn't have any self-loops. Furthermore, we assume that for each G_i the number of edges is lower bounded by linear to n. Note that graphs as input may not be bipartite graphs.

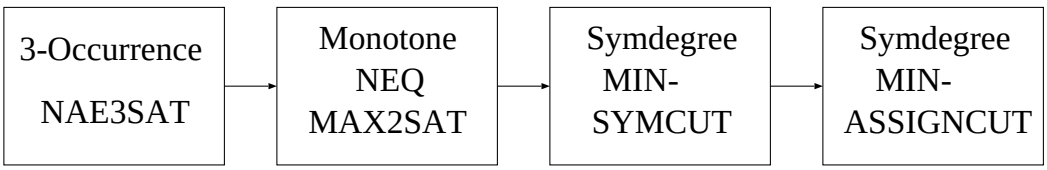
It is easy to see that the problem above is MAX2SAT search problem where instances for the problem is restricted to a subset of 2-CNF formulas. Moreover, all of those instances can be

produced by our randomized generator RGEN. This follows from the following observation. Let X be a set of variables and t be an assignment to X. In the problem above, we consider X as V, and correspond t to cut T_t of V such that for each edge e and its corresponding clause C_e, $\mathrm{cost}(T_t)$ is increased by e if and only if C_e is not satisfied under the assignment t.

Thereby, if we prove the hardness of the problem above, we can conclude that the randomized generator surely produces hard formulas although those hard formulas might be a small fraction of all the formulas produced. Thus, the theorem to prove here is as follows:

Theorem 5.1 Balanced MIN-ASSIGNCUT is NP-hard.

We will present the proof of this theorem by a sequence of reductions as follows. Starting reductions with a variant of NAE3SAT, where the number of occurrences of variables is a constant, we'll show the hardness of the target problem, Balanced MIN-ASSIGNCUT, in the bottom-up manner. We show the sequence of reductions in the figure below. The leftmost



problem, which we start the reductions with, is a variant of NAE3SAT, which is a well-known NP-complete problem [1]. The variant is also NP-complete as shown below. The restriction of variables' occurrences is related to the constant edge degree of graphs as instances for the target problem. The rightmost problem is directly reduced to the target problem. Thus, this sequence of reductions proves Theorem 5.1. The variant of NAE3SAT is the following problem:

3-Occurrence NAE3SAT

Instance: A 3-CNF formula F over X such that each variable of X appears in F at most three times. Moreover, for each clause $C \in F$, $|C| = 2$ or 3.

Question: Is there an assignment to X such that every clause of F has at least one true literal and at least one false literal?

This problem is easily proven to be NP-complete by reduction from NAE3SAT where for each formula F and for each clause $C \in F$, $|C| = 3$. Let F be an arbitrary instance for NAE3SAT. For each variable x which appears $k > 3$ times, we rename the ith occurrence's to y_i, and then add the following chain clauses:

$$F_x \;=\; \{(y_1 \vee \overline{y_2}), (y_2 \vee \overline{y_3}), \cdots, (y_k \vee \overline{y_1})\}\,.$$

We define F' as the sum of F and $\cup_{x \in X'} F_x$, where X' is the set of variables appearing more than three times in F. Note that F_x plays the role of enforcing the consistency on $y_1, \cdots, y_k$, that is, to satisfy F_x in the not-all-equal sense, the NAE sense for short, we need to assign to $y_1 \cdots, y_k$ either all 0s or all 1s. It is clear that the satisfiability (in the NAE sense) of F and F' is consistent, and the number of occurrences of variables of F' is at most three.

We next move to a variant of MAX2SAT called "Monotone NEQ MAX2SAT". A 2-CNF formula F is said to be *monotone* if each clause of F contains either only positive literals or only negative literals. A 2-CNF formula F over X is said to be *NEQ* 2-CNF if we can make pairs of $(l \vee l')$ and $(\overline{l} \vee \overline{l'})$ for all clauses of F such that no pairs have overlap of the clauses.

Monotone NEQ MAX2SAT

Instance: A monotone NEQ 2-CNF formula F over X such that for each clause $C \in F$, $|C| = 2$, and an integer k.

Question: Is the number of satisfiable clauses of F at least k?

Lemma 5.2 12-Occurrence Monotone NEQ MAX2SAT is NP-complete.

Proof. We show by reducing from 3-Occurrence NAE3SAT. Let F be an arbitrary 3-CNF formula for 3-Occurrence NAE3SAT. Note that in 3-clauses of F, there can be four types: (1) all positive literals, namely, $(x \vee y \vee z)$, (2) one negative literal, namely, $(\overline{x} \vee y \vee z)$, (3) two negative literals, namely, $(\overline{x} \vee \overline{y} \vee z)$, and (4) all negative literals, namely, $(\overline{x} \vee \overline{y} \vee \overline{z})$. We transform each type of those clauses as follows:

$$
\begin{aligned}
(1) \ &: \ (x \vee y \vee z) \longrightarrow (x \vee y), (\overline{x} \vee \overline{y}), (y \vee z), (\overline{y} \vee \overline{z}), (z \vee x), (\overline{z} \vee \overline{x}), \\
(2) \ &: \ (\overline{x} \vee y \vee z) \longrightarrow (x' \vee y), (\overline{x'} \vee \overline{y}), (y \vee z), (\overline{y} \vee \overline{z}), (z \vee x), (\overline{z} \vee \overline{x}), (x' \vee x), (\overline{x'} \vee \overline{x}), \\
(3) \ &: \ (\overline{x} \vee \overline{y} \vee z) \longrightarrow (x' \vee y), (\overline{x'} \vee \overline{y}), (y' \vee z), (\overline{y'} \vee \overline{z}), \\
&\qquad\qquad (z \vee x), (\overline{z} \vee \overline{x}), (x' \vee x), (\overline{x'} \vee \overline{x}), (y' \vee y), (\overline{y'} \vee \overline{y}), \\
(4) \ &: \ (\overline{x} \vee \overline{y} \vee \overline{z}) \longrightarrow (x \vee y), (\overline{x} \vee \overline{y}), (y \vee z), (\overline{y} \vee \overline{z}), (z \vee x), (\overline{z} \vee \overline{x}),
\end{aligned}
$$

where x' and y' are additional variables. In 2-clauses of F, there can be three types, which are transformed as follows:

$$
\begin{aligned}
(1') \ &: \ (x \vee y) \longrightarrow (x \vee y), (\overline{x} \vee \overline{y}), \\
(2') \ &: \ (\overline{x} \vee y) \longrightarrow (x \vee y'), (\overline{x} \vee \overline{y'}), (y \vee y'), (\overline{y} \vee \overline{y'}), \\
(3') \ &: \ (\overline{x} \vee \overline{y}) \longrightarrow (x \vee y), (\overline{x} \vee \overline{y}).
\end{aligned}
$$

We define F' as the conjunction of all those clauses transformed as above. It is clear that F' is a monotone NEQ 2-CNF formula, and the maximum number of occurrences of variables of F' is twelve.

Next, we show the equivalence of F and F'. Note that the satisfiability of F is taken w.r.t. in the NAE sense while the satisfiability of F' is taken w.r.t. in the ordinary sense. Let m_1, m_2, m_3, m_4 be the number of type 1, 2, 3, 4 clauses of F, respectively, and m'_1, m'_2, m'_3 be the number of type 1', 2', 3' clauses of F, respectively, Let us take a look carefully at type 1 clause $C = (x \vee y \vee z)$, for example. Consider any assignment that satisfies C (in the NAE sense). Then, the four clauses out of $C' = (x \vee y), (\overline{x} \vee \overline{y}), (y \vee z), (\overline{y} \vee \overline{z}), (z \vee x), (\overline{z} \vee \overline{x})$, transformed from C are satisfied (in the ordinary sense). On the other hand, in the case that C is not satisfied, three clauses out of C' are satisfied. We can easily check the other types in the same way. Thus we have that if F is satisfiable in the NAE sense, the maximum number of satisfiable clauses of F' is exactly

$$k = 4m_1 + 6m_2 + 8m_3 + 4m_4 + 2m'_1 + 4m'_2 + 2m'_3,$$

otherwise $k' = 3m_1 + 4m_2 + 6m_3 + 3m_4 + m'_1 + 2m'_2 + m'_3$. The values k and k' are strictly different for any instance of 3-Occurrence NAE3SAT. ■

At this point, we convert 2-CNF formulas to graphs. By regarding a pair of 2-clauses $(x \vee y)$ and $(\overline{x} \vee \overline{y})$ as a pair of edges $(\overline{x}, y)$ and $(\overline{y}, x)$ of a directed graph, it is easy to see that Monotone NEQ MAX2SAT problem can be considered as the following *minimization* problem:

MIN-SYMCUT

Instance: A directed bipartite graph $G = (U, W, E)$ over the vertex sets U and W, where $U = \{u_1, \cdots, u_n\}, W = \{w_1, \cdots, w_n\}$ such that if $(u_i, w_j) \in E$, then there exists its inverse edge (w_j, u_i) in E, and vice versa, i.e., if $(w_j, u_i) \in E$, then there exists its inverse edge (u_i, w_j) in E. And an integer k.

Question: Let $V = U \cup W$, and T be $T_u \cup T_w$, where T_u be a subset of U and T_w be a subset of W such that

$$\{j : w_j \in T_w\} = \{1, 2, \cdots, n\} \backslash \{i : u_i \in T_u\}.$$

For the symmetric cut T, we define $\mathrm{cost}(T) = |T \times (V - T) \cap E|$. Then, is there a subset T of V such that $\mathrm{cost}(T) \leq k$?

Note that since formulas converted to directed graphs of this problem are NEQ 2-CNF, for each $u_i \in U$, out-degree and in-degree of u_i are equal, denoted by $d_u(i)$. It is similar for W, and also denoted by $d_w(i)$. Furthermore, by doubling each clause of a monotone NEQ 2-CNF formula, we can obtain a symmetric graph from such a formula; that is, for every $1 \leq i \leq n$, $d_u(i) = d_w(i)$, denoted by $d(i)$. If a bipartite graph G has such a property, we say that G has *symmetric degree*, and define the edge degree function for G as d for $i = 1, \cdots, n$. Here, we call MIN-SYMCUT problem of bipartite graphs with symmetric degree "Symdegree MIN-SYMCUT". Thus, we have the following lemma:

Lemma 5.3 12-Symdegree MIN-SYMCUT is NP-complete.

In MIN-SYMCUT problem, we have to choose a cut in the symmetric way. On the other hand, we are allowed to choose an arbitrary cut in the target problem. Now, we introduce a problem to eliminate the constraint of symmetry on the cut of MIN-SYMCUT.

Symdegree MIN-ASSIGNCUT

Instance: Two directed bipartite graphs over the same vertex set $U \cup W$ where $U = \{u_1, \cdots, u_n\}$, and $W = \{w_1, \cdots, w_n\}$, namely, $G_1 = (U, W, E_1)$ and $G_2 = (U, W, E_2)$ such that G_1 and G_2 have the symmetric degree. Moreover, the edge degree functions d_1 and d_2 for G_1 and G_2 are identical. And an integer k.

Question: Let T_u be a subset of U and T_w be a subset of W, and T be $T_u \cup T_w$. We let $S = T_u \cup (W - T_w)$, and

$$\mathrm{cost}(T) \ = \ |T \times (V - T) \cap E_1| \ + \ |S \times (V - S) \cap E_2|.$$

Then, is there a subset T of V such that $\mathrm{cost}(T) \leq k$?

Lemma 5.4 12-Symdegree MIN-ASSIGNCUT is NP-complete.

Proof. We show by reducing from Symdegree MIN-SYMCUT. We don't care the precise value of edge degree since it will be clear.

Let $G = (U, W, E)$ be an arbitrary graph for Symdegree MIN-SYMCUT such that $U = \{u_i, \cdots, u_n\}$ and $W = \{w_1, \cdots, w_n\}$. Recall the form of G; that is, G is a directed bipartite graph over U and W such that for every $1 \leq i \leq n$, out-degree and in-degree of $u_i \in U$ and

$w_i \in W$ are all equal to $d(i)$. We transform G into an instance G' of $G'_1 = (U', W', E'_1)$ and $G'_2 = (U', W', E'_2)$ as follows: $U' = U$, $W' = W$, and

$$E'_1 \;=\; \bigcup_{1\le i}^{n} \bigcup_{1\le j}^{d(i)} \{(u'_i, w'_i), (w'_i, u'_i)\} \quad \text{and} \quad E'_2 \;=\; E,$$

where E'_1 is a multi-set. It is clear that G'_1 and G'_2 both have the symmetric degree and that the edge degree functions are identical.

We next show that the optimal cuts for G and G' are identical. It is proven by understanding that E'_1 plays the role of enforcing the symmetry on the cut, i.e., for any *asymmetric* cut T of $U' \cup W'$, there exists a symmetric cut T_0 of $U' \cup W'$ such that $\text{cost}(T_0) \le \text{cost}(T)$. Consider an asymmetric cut $T = T_u \cup T_w$ such that for some $1 \le i \le n$, $u'_i \in T_u$ and $w'_i \notin T_w$. By transferring w'_i into $W' \backslash T_w$, we decrease $\text{cost}(T)$ by $d(i)$ w.r.t. E'_1. On the other hand, by the transfer, we increase $\text{cost}(T)$ by at most $d(i)$ w.r.t. E'_2 because those are only edges *into* w'_i. It is the same argument for an asymmetric cut $T = T_u \cup T_w$ such that for some i, $u'_i \notin T_u$ and $w'_i \in T_w$. Thus, we can make the optimal cuts for G and G' identical. ■

At this end, we'll show the final reduction from Symdegree MIN-ASSIGNCUT. Likewise to the previous proof, we don't care the precise value of edge degree.

Let $G_1 = (U, W, E_1)$ and $G_2 = (U, W, E_2)$ be graphs of an arbitrary instance for Symdegree MIN-ASSIGNCUT, and d be the edge degree function for G_1 and G_2. We define $G'_1 = (U', W', E'_1)$ and $G'_2 = (U', W', E'_2)$ for Balanced MIN-ASSIGNCUT as follows: For each $1 \le i \le n$, we make $2d(i)$ copies of u_i and w_i for U' and W', respectively. For each edge $(u_i, w_j) \in E_l$ for $l = 1, 2$, we put an edge of E'_l from a copy of u_i to a copy of w_j so that for every vertex of $U' \cup W'$, either out-degree or in-degree of the vertex is exactly one w.r.t. E'_l. It is the same transformation for $(w_j, u_i) \in E_l$. Finally, for each $u_i \in U$, we form a clique for E'_l among all the copy vertices yielded from u_i. It is also the same transformation for vertices $w_j \in W$.

It is clear that this transformation is valid; that is, for each graph of G'_1 and G'_2, there exists a one-to-one balanced correspondence between U' and W'. Moreover, for every vertex, the edge degree of the vertex is a constant; that is, $2(d(i)-1)+1$ for $u'_i \in U'$ and $w'_i \in W'$. It is obtained from the following lemma that the optimality of G and G' holds:

Lemma 5.5 There exists an optimal cut T_0 of $U' \cup W'$ such that for each vertex $v \in U \cup W$, all the copy vertices yielded from v is in the same partition divided by T_0.

Proof. Let $U'_i = \{u'^1_i, u'^2_i, \cdots, u'^{d(i)}_i\}$ be the set of copy vertices yielded from a vertex u_i of U by the transformation. Suppose that a cut T divides U'_i into two parts A and B such that $|A| \ge |B|$. Consider that we move an arbitrary vertex b from B to A. The transfer of the vertex b doesn't increase $\text{cost}(T)$ because we can decrease it by at least $|A|$ while we increase it by at most $|B \backslash \{b\}| + 1 = |B|$. Thus, we can find out an optimal cut where all copy vertices of $u'^1_i, u'^2_i, \cdots, u'^{d(i)}_i$ are in the same partition. ■

Therefore, we have finally shown the hardness of Balanced MIN-ASSIGNCUT.

Acknowledgement

I thank Professor Osamu Watanabe for all his supports: giving valuable comments, discussing, and improving this report. I also thank Umeno-kun for discussing about generating instances.

References

[1] M. R. Garey and D. S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness", W. H. Freeman, 1979.

[2] G. A. Margulis, "Explicit constructions of concentrators", Problemy Peredachi Informatsii, 71-80, 1973. English translation in Problems of Information Transmission 9, 325-332, 1975.

[3] O. Gabber and Z. Galil, "Explicit constructions of linear-sized superconcentrators", Journal of Computer and System Sciences 22, 407-420, 1981.