ISSN 1342-2812

# Research Reports on Mathematical and Computing Sciences

Average-case Analysis for the MAX-2SAT Problem

Osamu Watanabe and Masaki Yamamoto

Aug. 2006, C–225

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

# Average-case Analysis for the MAX-2SAT Problem

Osamu Watanabe*
Dept. of Math. and Comp. Sci.
Tokyo Inst. of Technology
watanabe@is.titech.ac.jp

Masaki Yamamoto
School of Informatics
Kyoto University
masaki.yamamoto@lab2.kuis.kyoto-u.ac.jp

Research Report C-225

**Abstract**

We propose a simple probability model for MAX-2SAT instances for discussing the average-case complexity of the MAX-2SAT problem. Our model is a "planted solution model", where each instance is generated randomly from a target solution. We show that for a large range of parameters, the planted solution (more precisely, one of the planted solution pair) is the optimal solution for the generated instance with high probability. We then give a simple linear time algorithm based on a message passing method, and we prove that it solves the MAX-2SAT problem with high probability for random MAX-2SAT instances under this planted solution model for probability parameters within a certain range.

## 1 Introduction

We discuss the average-case analysis of the difficulty of MAX-SAT problems. In particular, we consider the MAX-2SAT problem, the simplest variation of MAX-SAT problems, where input CNF formulas are restricted to those consisting of only clauses with *two* literals. MAX-SAT problems are well-known as typical NP-type hard optimization problems, and it is known that even the MAX-2SAT problem is NP-hard, though the 2SAT problem is in P. Furthermore, it is also proved [4] that the MAX-2SAT problem is NP-hard to approximate within a certain constant approximation ratio. However, it has been shown that some algorithms/heuristics solve MAX-SAT problems quite well *on average*: [2, 3], and by standard SAT solvers [5, 14]. Such algorithms may solve MAX-2SAT as well on average. On the other hand, not so much theoretical investigation has been made for the average-case performance of such algorithms, in particular, on MAX-2SAT instances, compared with various detail studies on SAT instances.

For discussing the average-case complexity of MAX-2SAT problem, we propose one simple probability model for generating MAX-2SAT instances, thereby giving one instance distribution for the MAX-2SAT problem. Our model is one of the planted solution models. That is, we can guarantee (with high probability) that a planted solution is the optimal solution for a generated instance.

*Supported in part by a Grant-in-Aid for Scientific Research on Priority Areas "New Horizons in Computing" 2004-2006.

We also demonstrate that a simple linear-time algorithm can solve the MAX-2SAT with high probability when input formulas are given under this distribution with probability parameters in a certain range. Our parameter range is for a dense regime; we could prove that our algorithm solves the MAX-2SAT problem with high probability for random formulas with $\Omega(n^{1.5} \ln n)$ clauses. It is an interesting open problem to show some efficient algorithm for sparse formulas. (Somewhat related result has been shown by Scott and Sorkin [13]. They studied random 2CSP instances (including 2SAT) and showed, among other results, some deterministic algorithm solving MAX-2CSP in polynomial-time on average for random sparse instances, i.e., formulas with linear number of clauses. The authors ask for algorithms solving MAX-2SAT on dense instances. But note that our distribution is different the one considered in their paper.)

We begin by introducing some notions and notations for discussing SAT and MAX-SAT problems. Throughout this paper, we will use $n$ and $m$ to denote respectively the number of variables and clauses of a given input Boolean formula. We will use $x_1, \ldots, x_n$ for denoting Boolean variables. A *CNF formula* is a conjunction of clauses, a *clause* is a disjunction literals, and a *literal* is either a Boolean variable or its negation. In particular, a *2CNF formula* is a formula defined as a conjunction of clauses of two literals, where each clause is specified as $(x_i \vee x_j)$, $(x_i \vee \overline{x}_j)$, $(\overline{x}_i \vee x_j)$, or $(\overline{x}_i \vee \overline{x}_j)$, for some $1 \leq i \leq j \leq n$. In this paper, we will assume that clauses are syntactically one of the above four types; e.g., there is no clause like $(x_j \vee \overline{x}_i)$ for some $i < j$. Note that it is possible that a formula has a clause like $(x_i \vee x_i)$, $(x_i \vee \overline{x}_i)$, or $(\overline{x}_i \vee \overline{x}_i)$. (Since $(\overline{x}_i \vee x_i)$ is semantically the same as $(x_i \vee \overline{x}_i)$, we do not allow clauses of this type. Thus, there are altogether $\binom{n}{2} \times 4 + 3n = 2n^2 + n$ clauses.) We will use $\ell_i$ to denote either $x_i$ or $\overline{x}_i$.

An *assignment* is a function $t$ mapping $\{x_1, \ldots, x_n\}$ to $\{-1, +1\}$; $t(x_i) = +1$ (resp., $t(x_i) = -1$) means to assign true (resp., false) to a Boolean variable $x_i$. An assignment is also regarded as a sequence $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ of $\pm 1$'s, where $a_i = t(x_i)$ for each $i$, $1 \leq i \leq n$. For a given CNF formula $F$, its *optimal assignment* is an assignment satisfying the largest number of clauses in $F$. Now our MAX-2SAT problem is to find, for a given 2CNF formula of the above syntax, find an optimal assignment for the formula.

We explain our probability model for generating MAX-2SAT instances. This model is defined as a "planted solution model", a method for generating a problem instance so that a target solution, which is also generated in some way, is the answer to this instance (with high probability). In our model, we generate a sequence $\boldsymbol{a} = (a_1, \ldots, a_n)$ uniformly at random; let $\boldsymbol{a}'$ be its *complement assignment* $(-a_1, \ldots, -a_n)$, i.e., an assignment obtained by flipping the sign of all individual assignments. Then we use a pair of $\boldsymbol{a}$ and $\boldsymbol{a}'$ as a *planted solution pair*. For constructing a formula, we generate each clause satisfied by both assignments with probability $p$, and it is added to the formula. Since there are $n^2$ such clauses, the number of clauses of this type added to the formula is *on average* $pn^2$. In order to make the formula unsatisfiable, we also generate each clause that is unsatisfied by $\boldsymbol{a}$ (resp., $\boldsymbol{a}'$) with probability $r < p$. Again *on average* the formula has $rn(n+1)/2$ clauses that are not satisfied by $\boldsymbol{a}$ (resp., by $\boldsymbol{a}'$). Hence, the generated formula has *on average* $pn^2 + rn(n+1)$ clauses and each assignment of the planted solution pair fails to satisfy $rn(n+1)/2$ clauses *on average*. Note that under this generation model, every literal appears with the same probability. As stated below, we prove that if $p$ is large enough, then one of the planted solution pair is indeed the optimal and no other assignment is as well as this optimal assignment.

**Theorem 1.** For sufficiently large $n$, if $p = \Omega(\ln n/n)$ and $p > 3r$, then for a randomly

generated 2CNF formula $F$ from a random planted solution pair, with high probability, one of the planted solution pair is the optimal assignment of $F$ and no other assignment satisfies as many clauses as this assignment.

**Remark 1.** (Alternative Probability Model)
Under the above condition for $p$, a generated formula has, with high probability, $\Omega(n \log n)$ clauses. This condition is necessary for the above probability model, where each clause is generated independently. On the other hand, for a probability model where the number of occurrences of each literal is fixed, we can relax this condition and discuss random instances with $O(n)$ clauses.

Consider, for example, the following distribution. Here we consider all $4n^2$ possible clauses, including, e.g., $(x_i \vee x_j)$ with $i > j$ and $(x_i \vee \overline{x}_i)$. Consider any $p$ and $r$ so that both $pn/2$ and $rn/2$ are integers. Again we assume that a pair of planted solutions is $(1, 1, ..., 1)$ and $(-1, -1, ..., -1)$. We generate a formula (i.e., a set of clauses) uniformly at random under the constraint that each literal appears exactly $(p + r)n$ times, $pn$ times in clauses consistent with both planted solutions and $rn$ times in clauses consistent with only one of the planted solution pair. More specifically, we use two sequences $S_1$ and $S_2$, where $S_1$ is a sequence of $pn/2$ copies of $x_1, ..., x_n$ and $S_2$ is its random permutation. Then from the first elements in both sequences in order, we make clauses from corresponding pairs of literals in two sequences. For example, if $S_1 = (x_1, x_1, x_2, x_2, x_3, x_3)$ and $S_2 = (x_3, x_2, x_2, x_1, x_3, x_1)$, then we generate clauses $(x_1 \vee x_3)$, $(x_1 \vee x_2)$, ..., $(x_3 \vee x_1)$. Similarly, by using two sequences consisting of $pn/2$ copies of $\overline{x}_1, ..., \overline{x}_n$, we generate clauses of the form $(\overline{x}_i \vee \overline{x}_j)$. On the other hand, by using two sequences consisting of $rn/2$ copies of $x_1, ..., x_n$ and $\overline{x}_1, ..., \overline{x}_n$ respectively, we generate clauses of the form $(x_i \vee \overline{x}_j)$ and $(\overline{x}_i \vee x_j)$. A formula consisting all these clauses contains each literal exactly $(p + r)n$ times.

For this probability model, we can show a property similar to the above theorem, with a condition that $p > (2 + \epsilon)r$ for any $\epsilon > 0$ and $p \geq c_\epsilon/n$ for some constant $c_\epsilon > 0$.

Note that the analysis of our algorithm reported in this paper cannot be used for this probability model. (On the other hand, it is unlikely that our algorithm does not work under this probability model.) □

Next we introduce a simple message passing algorithm, and we show that its one instance, which runs in time $\mathcal{O}(n+m)$, can solve the MAX-2SAT problem correctly with high probability if input formulas are given by the above planted solution model with parameters $p$ and $r$ within a certain range.

The idea of the algorithm is simple and intuitively clear. Let $F$ be any 2CNF formula $F$. Considers the case that $x_1 = -1$ in the optimal assignment of $F$. Suppose that there is a clause $(x_i \vee x_1)$ (resp., $(\overline{x}_i \vee x_1)$) in $F$, which can be restated as $(\overline{x}_i \to x_1)$ (resp., $x_i \to x_1)$). Thus, in order to satisfy this clause, we must assign $-1$ to $\overline{x}_i$ (resp., $x_i$). Such "negative beliefs" are passed to the other literals from $x_1$ following backwards implication edges, i.e., directed edges corresponding to implications to $x_1$. Then for each $x_i$, a "belief for $x_i = +1$" is computed as $b(x_i) - b(\overline{x}_i)$, where $b(\ell_i)$ is the sum of (negative) beliefs that literal $\ell_i$ received. Next from literals with a negative belief, their negative beliefs are sent to the other literals through implication edges backwards, and then beliefs are updated based on received beliefs. This process is iterated until the signs of all beliefs get stabled *or* the number of iterations reaches to a given time bound MAXSTEP. Finally, an assignment to individual $x_i$ is computed

from the sign of the final belief of $x_i$. The other case that $x_1 = +1$ (i.e., $\overline{x}_1 = -1$) is equally considered; we can use one of the outputs satisfying more clauses as a solution.

For our theoretical analysis, we consider a simplified version where MAXSTEP $= 2$, which can be implemented to run in $\mathcal{O}(n + m)$ steps. Even for this simple version, we prove that it solves the MAX-2SAT problem with high probability under the planted solution model with nontrivial parameters $p$ and $r$.

**Theorem 2.** For sufficiently large $n$, if $p \geq 3r$ and $p = \Omega(\sqrt{\ln n/n})$, then for a randomly generated 2CNF formula $F$ from a random planted solution pair, the algorithm yields these two planted solutions (by executing twice with different beliefs for $x_1$) with high probability.
**Remark.** The coefficient 3 of the condition $p \geq 3r$ is not essential, and we can use any constant larger than 1 here. On the other hand, the condtition $p > 3r$ is needed anyway for using Theorem 1 to gurantee the optimality of one of the planted solutions. Thus, we use below the condition $p \geq 3r$ for the sake of simplicity.

*Related Work and Open Problems*

There are some proposals of algorithms generating MAX-SAT instances for testing MAX-SAT algorithms [8, 9, 15]. These algorithms first generate or fix a target assignment and generate an instance so that this assignment becomes an optimal solution. Thus, they are regarded as a planted solution model. In fact, our planted solution model is based on the generation algorithm proposed by Yamamoto [15]. Our improvement here is to introduce a planted solution pair and generate clauses based on two symmetric planted solutions. This makes our model much simpler than the one by Yamamoto's algorithm. Furthermore, we can guarantee that the occurrence of all literals are statistically the same, which prevents solvers to use "majority vote" strategy. The same approach has been proposed [1] for generating hard sat. instances for $k$SAT problems. The important difference here is the point that inconsistent clauses are also added with probability $r < p$. This is for generating unsatisfiable formulas; otherwise, i.e., if only satisfiable formulas were generated, the problem would be trivially easy because the 2SAT problem is in P, which is different from the other $k$SAT problem $k \geq 3$. One open problem here is to extend our approach to MAX-$k$SAT problems for $k \geq 3$.

Our message passing algorithm for MAX-2SAT is motivated by a modified belief propagation algorithm for graph partitioning problems [11]. Recently, Pearl's belief propagation [12] has been used for solving several NP-hard problems, e.g., [7]; but it is also reported that the belief propagation may not be appropriate for solving SAT problems, because the role of literals in each clause is not symmetric. Here we ignore positive literals (i.e., literals assigned true) and use messages from only negative literals. On the other hand, while the belief propagation computes messages by some formula based on the underlying probability model, our algorithm computes messages in a straightforward/naive way. It may be possible to improve our algorithm by using more careful method for computing messages.

Our theoretical analysis of the algorithm is for a special case where MAXSTEP $= 2$, i.e., the case where only two updating iterations is allowed. It is easy to see that $p = \Omega(n^{-1/2})$, which is close to the condition of Theorem 2, is necessary; otherwise, a message from $x_1$ (or $\overline{x}_1$) cannot reach to the majority of literals. On the other hand, computer experiments show that by allowing more iterations, e.g., MAXSTEP $= 20$ (for $n = 5000$), the algorithm works well for much smaller $p$. An important open question is to develop some method for analyzing the algorithm's execution for large number of iterations.

In general, it would be interesting to see whether there is some efficient algorithm that solves the MAX-2SAT problem on average for much smaller $p$. It has been known that SAT problems (under the standard planted solution model) are relatively easy if there are enough number of clauses, which may be also true for MAX-SAT problems. Under the parameter range of Theorem 2 (i.e., $p = \Omega(\sqrt{\ln n/n})$, the number of clauses is *on average* $\Theta(n\sqrt{n \ln n})$. On the other hand, our planted solution model can be used as long as $p = \Omega(\ln n/n)$, in which case generated instances have $\Theta(n \ln n)$ clauses on average. This probability distribution may be an interesting target for MAX-2SAT algorithms.

*Preliminaries: the Chernoff bound*

In this paper, we will use the following version of the Chernoff bound, modified from the one in [6].

**Proposition 1.1.** Let $X_1, \ldots, X_n$ be independent random variables such that $0 \leq X_i \leq c$ for $1 \leq i \leq n$. Let $S = \sum_{1 \leq i \leq n} X_i$ and $\mu = \mathrm{E}[S]$. Then for any $\varepsilon > 0$, we have

$$\Pr[\, S \geq (1+\varepsilon)\mu \,] \;\leq\; \exp\left(-\frac{\varepsilon^2 \mu}{3c}\right).$$

## 2 A Planted Solution Model for MAX-2SAT

In this section, we define a planted solution model for MAX-2SAT, our probability distribution on instances of MAX-2SAT. More specifically, for a given $n \geq 1$, we describe a way of generating a 2-CNF formula over $n$ variables $x_1, \ldots, x_n$.

Consider any assignment $(a_1, \ldots, a_n)$, where $a_i \in \{-1, +1\}$ for $1 \leq i \leq n$, to variables $(x_1, \ldots, x_n)$, and its compliment assignment $(-a_1, \ldots, -a_n)$, i.e., an assignment obtained by flipping all values of $(a_1, \ldots, a_n)$. Such a pair of assignments is used as a planted solution pair. For any planted solution pair, a clause is called *consistent* with the planted solution if it is satisfied by both of two assignments of the planted solution, and it is called *partially inconsistent* with the planted solution if it is not satisfied by one of them. (Note that any clause is satisfied by at least one of the planted solution pair.) Now we generate a 2-CNF formula as follows: First generate a planted solution pair uniformly at random. Then each clause of the form $(\ell_i \vee \ell_j)$, where $i \leq j$, is added to the formula, with probability $p$ if it is consistent with the planted solution and with probability $r$ if it is partially inconsistent with the planted solution; see below for the case $i = j$.

**Remark 2.** Recall that $\ell_i$ denotes a literal either $x_i$ or $\overline{x}_i$. As explained in Introduction, we consider only clauses of the form $(\ell_i \vee \ell_j)$ for some $1 \leq i \leq j \leq n$. For simplifying our analysis, we define our planted solution model so that clauses $(x_i \vee x_i)$ and $(\overline{x}_i \vee \overline{x}_i)$ are respectively added to a formula with probability $r$, whereas a clause $(x_i \vee \overline{x}_i)$ is generated with probability $p$. Note that clauses like $(x_i \vee \overline{x}_i)$ are satisfied by any solution. Thus, they can be ignored when discussing the optimality of solutions. □

Here in order to simplify our discussion, we will explain with one fixed planted solution, a pair of all $+1$ assignment and all $-1$ assignment, which we call *our planted solution pair*; we denote them by $\boldsymbol{a}^+$ and $\boldsymbol{a}^-$ respectively. For this solution pair, clauses $(\overline{x}_i \vee x_j)$ and $(x_i \vee \overline{x}_j)$, where $i \leq j$, are consistent and generated with probability $p$; on the other hand, clauses $(x_i \vee x_j)$ and $(\overline{x}_i \vee \overline{x}_j)$ are partially inconsistent and generated with probability $r$. □

We show below that if $p > 3r$ and $p = \Omega(\ln n/n)$, then for a randomly generated formula $F$ under this planted solution model, one of the planted solution pair is the optimal assignment (and no others) with high probability.

**Theorem 2.1.** There exists some constant $c_{\mathrm{dist}}$ such that for any $n \geq 1$ and for any $\delta > 0$, if probability parameters $p$ and $r$ satisfy $p > 3r$ and $p \geq c_{\mathrm{dist}} \ln(n/\delta)/n$, then for a randomly generated formula $F$ under our planted solution model with parameters $p$ and $r$, with probability $\geq 1 - \delta$, one of the two planted solution pair is an optimal assignment for $F$; furthermore, there is no optimal assignment other than the planted solution pair.

**Proof.** Consider $p$, $r$, and $n$ satisfying the condition of the theorem. In particular, we consider below the case where $r = p/4$, which is intuitively the hardest case; in fact, the case where $r < p/4$ can be proved by a similar argument.

Here we explain with our planted solution pair, i.e., a pair of all $+1$ assignment $\boldsymbol{a}^+$ and all $-1$ assignment $\boldsymbol{a}^-$ to $n$ variables $x_1, \ldots, x_n$. Let $F$ be a randomly generated formula for this planted solution pair. Our goal is to show that, with high probability, either $\boldsymbol{a}^+$ or $\boldsymbol{a}^-$ satisfies the most number of clauses in $F$, which cannot be achieved by any other assignment.

For our discussion, we consider a directed graph $G = (V, E)$ naturally defined as follows: $V = V_+ \cup V_-$, where $V_+ = \{v_{+1}, \ldots, v_{+n}\}$ and $V_- = \{v_{-n}, \ldots, v_{-1}\}$. $E$ consists of two directed edges *corresponding to* a clause $(\ell_i \vee \ell_j)$, where $i \leq j$, in $F$. For example, for a clause $(x_i \vee x_j)$, $E$ has two directed edges $(v_{-i}, v_{+j})$ and $(v_{-j}, v_{+i})$, each of which corresponds to $(\overline{x}_i \to x_j)$ and $(\overline{x}_j \to x_i)$; clauses of the other type define two corresponding directed edges in $E$ similarly. Note that the obtained graph $G$ may have multiple edges. Recall that we do not consider clauses like $(\overline{x}_i \vee x_i)$; also as mentioned in the above remark, we ignore clauses like $(x_i \vee \overline{x}_i)$. Thus, the obtained graph $G$ has no loop edge.

Consider any assignment $t$ to $x_1, \ldots, x_n$. We regard this also as an assignment to $V$; specifically, for each $i \in \{1, \ldots, n\}$, define $t(v_{+i}) = t(x_i)$ and $t(v_{-i}) = -t(v_{+i})$. In general, an assignment $t$ to $V$ satisfying $t(v_{-i}) = -t(v_{+i})$ for all $i$ is called a *legal assignment*. It is easy to see that a clause $(\ell \vee \ell')$ is unsatisfied by $t$ if and only if its two corresponding directed edges are from a vertex assigned $+1$ to a vertex assigned $-1$, which we call *unsatisfied edges*. That is, the number of unsatisfied clauses is half of that of the unsatisfied edges. Thus, in order to prove the theorem, we estimate the number of unsatisfied edges under an arbitrary legal assignment to $V$.

First, we estimate the number of unsatisfied edges within $G[V_+]$ and $G[V_-]$, which are subgraphs of $G$ induced respectively by $V_+$ and $V_-$, by the well-known fact that a random graph is almost surely an expander.

A directed graph $G' = (V', E')$ is said to be a *$d$-expander* if for every $S \subset V'$ with $\|S\| \leq \|V'\|/2$, the following holds: $\|E'(S, \overline{S})\| \geq d\|S\|$, and $\|E'(\overline{S}, S)\| \geq d\|S\|$, where $E'(S, \overline{S})$ is the set of edges in $E'$ from vertices in $S$ to vertices in $\overline{S}$. We denote by $\mathcal{G}_{n,q}$ a distribution of graphs $G' = (V', E')$ over $n$ vertices that are generated as follows: for every ordered pair $(v'_i, v'_j)$ of distinct vertices, generate a directed edge $(v'_i, v'_j)$ with probability $q$ as an edge of $E'$. Recall that when generating a formula $F$, $G[V_+]$ is generated in this way, and so is $G[V_-]$. We here show the following expansion property.

**Claim 1.** For any $n$, any $\delta' > 0$, and any $\epsilon' > 0$, if $q \geq (2/\epsilon'^2) \ln(4en/\delta')/n$, then for a random graph $G' \in \mathcal{G}_{n,q}$, with probability $\geq 1 - \delta'$, $G' = (V', E')$ is a $(1/2 - \epsilon')qn$-expander.

**Proof of the Claim.** For any $n$, $\delta' > 0$, and $\epsilon' > 0$, consider any $q$ satisfying the condition of the claim. Let $S$ be a subset of $V'$ with size at most $n/2$. Let $\mathrm{Bad}(S)$ be an event that $S$ does not meet the condition of a $d$-expander with $d = (1/2 - \epsilon')qn$, i.e., either $\|E'(S, \overline{S})\| < d\|S\|$ or $\|E'(\overline{S}, S)\| < d\|S\|$ holds. We estimate the upper bound of $\Pr[\,|E'(S, \overline{S})| < d\|S\|\,]$. Note that the value of $\Pr[\mathrm{Bad}(S)]$ is at most two times of this value. This is done by using the standard Chernoff bound in the following way: Fix $S \subset V'$ such that $\|S\| \leq n/2$. For all pairs of $u \in S$ and $v \in \overline{S}$, we introduce independent random variables $Y_{u,v}$ such that $\Pr[Y_{u,v} = 1] = q$ and $\Pr[Y_{u,v} = 0] = 1 - q$. Let $Y = \sum_{u \in S, v \in \overline{S}} Y_{u,v}$; that is, $Y = \|E'(S, \overline{S})\|$. Note that $\mathrm{E}[Y] = q(n - \|S\|)\|S\|$; hence, we have

$$\begin{aligned} \mathrm{E}[Y] - d\|S\| &= q(n - \|S\|)\|S\| - d\|S\| \\ &\geq (q(n/2) - d) \cdot \|S\| = \epsilon' qn\|S\| > 0. \end{aligned}$$

Then from the standard Chernoff bound (see, e.g., [6]), it follows

$$\begin{aligned} \Pr[\,\mathrm{Bad}(S)\,] &\leq 2\Pr\left[\,\left\|E'\left(S, \overline{S}\right)\right\| < d\|S\|\,\right] \\ &= 2\Pr[\,\mathrm{E}[Y] - Y > \mathrm{E}[Y] - d\|S\|\,] \\ &< 2\exp\left(-\frac{(\mathrm{E}[Y] - d\|S\|)^2}{2\mathrm{E}[Y]}\right) \\ &\leq 2\exp\left(-\frac{(q(n/2) - d)\|S\|)^2}{2q(n - \|S\|)\|S\|}\right) \\ &\leq 2\exp\left(-\frac{\epsilon'^2 qn}{2} \cdot \|S\|\right). \end{aligned}$$

Since the probability that $G'$ is not $d$-expander is the probability that $\mathrm{Bad}(S)$ holds for some $S \subset V'$, $\|S\| \leq n/2$, we have

$$\begin{aligned} \Pr\left[\bigcup_S \mathrm{Bad}(S)\right] &\leq \sum_S \Pr[\,\mathrm{Bad}(S)\,] \\ &\leq \sum_{s=1}^{n/2} \binom{n}{s} 2\exp\left(-\frac{\epsilon'^2 qn}{2} \cdot s\right) \\ &\leq \sum_{s=1}^{n/2} 2\left(\frac{en}{s} \cdot \exp\left(-\frac{\epsilon'^2 qn}{2}\right)\right)^s \\ &\leq \sum_{s=1}^{n/2} 2\left(en \cdot \exp\left(-\frac{\epsilon'^2 qn}{2}\right)\right)^s. \end{aligned}$$

Then the claim holds because the last one is bounded by $\delta'$, which is argued as follows: From the assumption that $q \geq (2/\epsilon'^2)\ln(4en/\delta')/n$, we have $en \cdot \exp(-\epsilon'^2 qn/2) \leq \delta'/4$; hence, we have $\sum_s (en \cdot \exp(-\epsilon'^2 qn/2))^s \leq \delta'/2$. □ (Proof of the Claim)

Since $G[V_+]$ (resp., $G[V_-]$) can be regarded as a random graph from $\mathcal{G}_{n,p}$, and $p \geq c\ln(n/\delta)/n$ for some sufficiently large constant $c$ by our assumption, we may assume from the above claim that $G[V_+]$ and $G[V_-]$ are $(1/2 - \epsilon')pn$-expanders for some $\epsilon'$, $0 < \epsilon' < 1/2$,

which will be fixed at the end. That is, for each $U \in \{V_+, V_-\}$ and for every $S \subset U$, we have

$$\begin{aligned} &\|E(S, U-S)\| \geq (1/2-\epsilon')pn\|S\| \\ &\quad \text{and} \\ &\|E(U-S, S)\| \geq (1/2-\epsilon')pn\|S\|. \end{aligned}$$

Now consider any legal assignment $t$ to $V$ that is different from our two planted solutions. By $h$ we denote the number of unsatisfied edges of $G$ under $t$. On the other hand, let $h_0 = \min\{|E \cap (V_+ \times V_-)|, |E \cap (V_- \times V_+)|\}$; that is, $h_0$ is the number of unsatisfied edges by a better assignment among our two planted solutions. From now on, we estimate $h$ and show that $h > h_0$ with high probability.

Let $A_+$ and $B_+$ be subsets of $V_+$ assigned $+1$ and $-1$ respectively under $t$; on the other hand, let $A_-$ and $B_-$ be subsets of $V_-$ assigned $-1$ and $+1$ respectively. Note that $|A_+| = |A_-|$ and $|B_+| = |B_-|$, and let $a$ and $b$ be the number of $|A_+|$ and $|B_+|$ respectively; we may assume that $a, b \geq 1$. In the case of $a \leq b$, we show that

$$h \ > \ \|E \cap (V_- \times V_+)\| \tag{1}$$

holds with high probability. In the other case, i.e., $a \geq b$, we show that $h > \|E \cap (V_+ \times V_-)\|$ holds with high probability. Then we can conclude $h > h_0$. Below we will consider only the former case, i.e., the case $a \leq b$.

For edges in $V_+$ and $V_-$, we see from the above expansion property that the number of unsatisfied edges in each of $V_+$ and $V_-$ is respectively at least $(1/2-\epsilon')pn \cdot a$; that is, $\|E(B_+, V_+ - B_+)\|$, $\|E(A_-, V_- - A_-)\| \geq (1/2-\epsilon')pna$. Consider then edges between $V_+$ and $V_-$; here we estimate only the number of unsatisfied edges from $V_-$ to $V_+$. Since any unsatisfied edge is from a $+1$ vertex to a $-1$ vertex, unsatisfied edges are those from $B_-$ to $B_+$. Thus, we have $h \geq (1-2\epsilon')pna + |E \cap (B_- \times B_+)|$, where we decompose the last term as follows.

$$\begin{aligned} \|E \cap (B_- \times B_+)\| &= \|E \cap (V_- \times V_+)\| \\ &\quad -\|E \cap (A_- \times V_+)\| - \|E \cap (B_- \times A_+)\| \\ &\geq \|E \cap (V_- \times V_+)\| \\ &\quad -\|E \cap (A_- \times V_+)\| - \|E \cap (V_- \times A_+)\|. \end{aligned}$$

Hence, for our goal (1), it suffices to show that $(1-2\epsilon')pna - \|E \cap (A_- \times V_+)\| - \|E \cap (V_- \times A_+)\|$ is positive. By using a variation of the Chernoff bound (i.e., Proposition 1.1), we can show (Claim 2) that, with high probability, both $\|E \cap (A_- \times V_+)\|$ and $\|E \cap (V_- \times A_+)\|$ are close to their expectations; more precisely, they are respectively less than $(1+\epsilon'')rna$. Hence, we have

$$\begin{aligned} &(1-2\epsilon')pna - \|E \cap (A_- \times V_+)\| - \|E \cap (V_- \times A_+)\| \\ &\quad > ((1-2\epsilon')p - 2(1+\epsilon'')r)na. \end{aligned}$$

Here by letting $\epsilon' = \epsilon'' = 1/8$, we can show that the right-hand side is positive if $p > 3r$.

Finally we check the probability that the above inequality holds for all assignments. We know from Claim 1 that $\|E(B_+, V_+ - B_+)\| + \|E(A_-, V_- - A_-)\| < (1-2\epsilon')pna$ occurs for some assignment such that $a \leq b$ is at most $2\delta'$. On the other hand, from the claim below, the probability that $\|E \cap (A_- \times V_+)\| + \|E \cap (V_- \times A_+)\| \geq 2(1+\epsilon'')rna$ occurs for some assignment such that $a \leq b$ is at most $2\delta''$. Thus, by choosing $\delta' = \delta'' = \delta/8$, we can show that the probability that the above event fails to hold is at most $\delta/2$. Considering the other case as well, we can conclude that with probability $1-\delta$, the desired inequality holds for all assignments. $\square$

**Claim 2.** We use notations in the above proof. For any $n$, any $\delta'' > 0$, and any $\epsilon'' > 0$, let $G = (V, E)$ be a random graph constructed from $F$ generated by our planted solution model with parameters $p$ and $r$. If $r \geq (8/\epsilon''^2)\ln(2en/\delta'')/n$, then the probability that $\|E \cap (A_- \times V_+)\| \geq (1+\epsilon')rna$ occurs for some assignment such that $a \leq b$ is bounded by $\delta''$. The same statement also holds for $\|E \cap (V_- \times A_+)$. (The proof is almost the same as Claim 1, and it is omitted here.)

**Remark 3.** (Proof for the Alternative Probability Model)
As stated in Remark 1, we can prove somewhat stronger statement for more balanced probability models. Consider, for example, the one defined in Remark 1. In this case, since we may assume that the corresponding directed graphs $G[V_+]$ and $G[V_-]$ are both $pn$-regular, by following a standard argument [10] we can show that the desired expander property for these graphs if $p \geq c/n$ for sufficiently large $c > 0$. On the other hand, we know from the assumption that the number of crossing edges (i.e., $\|E \cap (A_- \times V_+)\|$ and $\|E \cap (V_- \times A_+)\|$ respectively) is exactly $2rna$ (for which we do not need any proof like Claim 2). Then by an argument similar to the above, we can show the corresponding statement if $p \geq (2+\epsilon)r$ and $p \geq c_\epsilon/n$, for any $\epsilon > 0$ and for some constant $c_\epsilon > 0$. □

# 3 A Simple Algorithm

For our probability model for the average-case analysis of MAX-2SAT, we show in this section that a simple algorithm can solve MAX-2SAT on average when parameters $p$ and $r$ are in a certain but nontrivial range. The algorithm is a message passing algorithm stated in Figure 1; this algorithm is motivated by the modified belief propagation algorithm for graph partitioning problems [11].

We explain the outline of the algorithm[1]. First define the meaning of symbols used in the algorithm. The algorithm is executed on a directed graph $G = (V, E)$ that is constructed from a given formula $F$ in essentially the same way as in the proof of Theorem 2.1. $V$ is a set of $2n$ vertices $v_s$, $s \in S = \{-n, -(n-1), \ldots, -1, +1, \ldots, +(n-1), +n\}$, and $E$ consists of two directed edges corresponding to each clause $(\ell_i \vee \ell_j)$ of $F$, where $i < j$; on the other hand, only one edge is added to $E$ for each clause of type $(\overline{x}_i \vee x_i)$. Note that graph $G$ has no multiple edge, while it may have some self-loops. (*Cf.* In the proof of Theorem 2.1, multiple edges were allowed; on the other hand, we ignored self-loops.) The algorithm computes a "belief" $b(v_s)$ at each vertex $v_s$, an integral value indicating whether the Boolean variable $x_{|s|}$ should be assigned true (i.e., $+1$) or false (i.e., $-1$). More specifically, for an optimal assignment, the algorithm suggests, for each $x_i$, to assign $x_i = +1$ if the final value of $b(v_{+i})$ is positive and $x_i = -1$ if it is negative. Note that $b(v_{-i}) = -b(v_{+i})$; we may regard $b(v_{-i})$ as a belief for $\overline{x}_i$.

These belief values are initially set to 0 except for one pair of vertices, e.g., $v_{+1}$ and $v_{-1}$ that are assigned $+1$ or $-1$ initially. In the algorithm of Figure 1, $b(v_{+1})$ (resp., $b(v_{-1})$) is set to $+1$ (resp., $-1$), which considers the case that $x_1$ is true in the optimal assignment. Clearly we need to consider the other case; that is, the algorithm is executed again with the initial assignment $b(v_{+1}) = -1$ and $b(v_{-1}) = +1$, and one of the obtained assignments satisfying more clauses is used as an answer. Now consider the execution of the algorithm. The algorithm updates beliefs based on messages from the other vertices. At each iteration,

[1]In this section, we will use $i$ and $j$ to denote unsigned (i.e., positive) indices in $\{1, \ldots, n\}$, whereas $s$ and $t$ will be used for signed indices in $S$.

**procedure** `MPalgo_for_MAX2-SAT` $(F)$;
// An input $F = C_1 \wedge \cdots C_m$ is a 2CNF formula over variables $x_1, \ldots, x_n$.
// Let $S = S_+ \cup S_-$, where $S_+ = \{+1, \ldots, +n\}$ and $S_- = \{-n, \ldots, -1\}$.
// This is an execution under the assumption that $x_1 = +1$ (i.e., true).
**begin**
construct $G = (V, E)$,
where $V = \{\, v_s : s \in S \,\}$, and $E = \bigcup_{1 \le k \le m} E(C_k)$;
set $b(v_s)$ to 0 for all $s \in S$;
$b(v_{+1}) \leftarrow +1$; $b(v_{-1}) \leftarrow -1$;
**repeat** MAXSTEP times **do** {
**for each** $i \in \{2, \ldots, n\}$ **in parallel do** {

$$b(v_{+i}) \leftarrow \sum_{v_s \in N^{-1}(v_{+i})} \min(0, b(v_s));$$

$$b(v_{-i}) \leftarrow \sum_{v_s \in N^{-1}(v_{-i})} \min(0, b(v_s)); \qquad \text{— (1)}$$

$b(v_{+i}) \leftarrow b(v_{+i}) - b(v_{-i})$; $b(v_{-i}) \leftarrow -b(v_{+i})$; — (2)
}
**if** $\mathrm{sign}(b(v_i))$ is stabilized for all $i \in \{2, \ldots, n\}$
**then break**;
// $b(v_{+1}) \leftarrow 0$; $b(v_{-1}) \leftarrow 0$; — (3)
}
output( $+1, \mathrm{sign}(b(v_{+2})), \ldots, \mathrm{sign}(b(v_{+n}))$ );
**end-procedure**

Figure 1: A message passing algorithm for the MAX-2SAT problem

the belief of each vertex $v_{+i}$ (resp., $v_{-i}$) is recomputed based on the last belief values of its neighbor vertices. More specifically, if there is an edge from $v_{+i}$ to $v_s$, and $b(v_s)$ is negative, then this negative belief is sent to $v_{+i}$ (from $v_s$) and used for computing the next belief of $v_{+i}$. The edge $v_{+1} \to v_s$ corresponds to a clause $(x_i \to \ell_{|s|})$ (where $\ell_{|s|}$ is the literal corresponding to $v_s$), and the condition that $b(v_s) < 0$ means that the literal $\ell_{|s|}$ is assigned false (under the current belief). Thus, in order to satisfy the clause $(x_i \to \ell_{|s|})$, we need to assign false to $x_i$. This is the reason for the message from $v_s$. Belief $b(v_{+i})$ at this iteration is defined as the sum of these messages. It should be remarked here that all belief values are updated *in parallel*; that is, updated beliefs are not used when updating the other beliefs in the same iteration, but those computed at the previous iteration are used. This update is repeated until no belief value is changed its sign after one updating iteration (in which case we say that sings are all stabilized) *or* the number of iterations reaches to a bound MAXSTEP.

This is the outline of our algorithm. There are some remarks on its detail implementation. First note that it is possible to implement the algorithm on the standard unit cost RAM model so that each iteration can be done in time $\mathcal{O}(n + m)$. For our theoretical analysis, we use MAXSTEP = 2; that is, beliefs are updated only twice. Furthermore, in order to simplify our analysis, we assume that statement (2) is executed only after the second iteration, and that statement (3) is executed. On the other hand, in more practical implementation, we would run the algorithm with, e.g., MAXSTEP = 20 for better performance. In this case, it is better (at least from our preliminary experiments) to execute statement (2) at each iteration

and remove statement (3) as stated in Figure 1.

Now we show that if $n$ is large enough (more precisely, $n = \Omega(\ln n/p^2)$, or equivalently, $p = \Omega(\sqrt{\ln n/n})$ and $p \geq 3r$, then the algorithm yields a planted solution with high probability. More precisely, the planted solution that the algorithm gives is the one consistent with the initial value of $b(v_{+1})$. Thus, by running the algorithm twice with two different initial values (i.e., $+1$ and $-1$), we can get two planted solutions. As argued in the previous section, if an input formula $F$ is generated under our planted solution model with parameters $p \geq 3r$, then one of the two planted solutions is an optimal assignment for $F$ with high probability; thus, by running the algorithm twice with two different initial values, we can obtain the optimal assignment with high probability.

In the following discussion, we fix $n$, the number of variables, and fix our planted solution pair to all $+1$ (i.e., true) assignment and all $-1$ (i.e., false) assignment. We consider the situation where the algorithm is executed with initial values $b(v_{+1}) = +1$ and $b(v_{-1}) = -1$; thus, our goal is to show that the algorithm outputs all $+1$ assignment. We assume that an input formula $F$ is randomly generated following our planted solution model for this planted solution pair with parameters $p$ and $r$ such that $p \geq 3r$. Thus, $F$ is a random variable in our following discussion. Below we introduce some more random variables, all of which depends on the random variable $F$.

Let $G$ be a graph constructed from $F$ in the algorithm. For any $s$ and $t$ in $S$, let $E_{s,t}$ be a random variable indicating whether there is an edge from $v_s$ to $v_t$ in $G$; i.e., $E_{s,t} = 1$ if the edge $(v_s, v_t)$ exists and $E_{s,t} = 0$ otherwise. From the choice of our planted solution pair and the definition of the graph $G$, it is clear that random variables $E_{s,t}$ and $E_{s',t'}$ are mutually independent except that it holds $E_{s,t} = E_{-t,-s}$, and that $E_{s,t}$ takes either 0 or 1 value as follows:

$$E_{s,t} \;=\; \begin{cases} 1, & \text{with prob. } p, \\ 0, & \text{with prob. } 1-p. \end{cases} \qquad \text{s.t. } \mathrm{sign}(s) = \mathrm{sign}(t) \tag{2}$$

and

$$E_{s,t} \;=\; \begin{cases} 1, & \text{with prob. } r, \\ 0, & \text{with prob. } 1-r. \end{cases} \qquad \text{s.t. } \mathrm{sign}(s) \neq \mathrm{sign}(t) \tag{3}$$

For discussing the status of the algorithm after the first iteration, we consider the following random variables. (We sometimes need to consider vertices other than $v_{+1}$ and $v_{-1}$; for simplifying our discussion, we will use the following notations: $S'_+ = \{+2, +3, \ldots, +n\}$, $S'_- = \{-n, \ldots, -3, -2\}$.)

$$W_+ \;=\; \{\; v_s \;:\; s \in S'_+ \;\wedge\; b(v_s) = -1 \;\}, \quad W_- \;=\; \{\; v_s \;:\; s \in S'_- \;\wedge\; b(v_s) = -1 \;\},$$
$$Y_+ \;=\; \|W_+\|, \;\text{ and }\; Y_- \;=\; \|W_-\|.$$

For the second iteration, we consider random variables $X_s$'s, where each $X_s$ is the value of $b(v_s)$ after executing statement (1) and before executing statement (2). Each $X_s$ is then expressed as follows with random variables $X_s^+$ and $X_s^-$. Here $b(v_t)$ denotes the belief of $v_t$ computed at the first iteration. (Recall that we are considering a slightly modified algorithm, which does not execute statement (2) in the first iteration.)

$$X_s^+ \;=\; \sum_{+j \in N^{-1}(v_s)} b(v_{+j}), \qquad X_s^- \;=\; \sum_{-j \in N^{-1}(v_s)} b(v_{-j}), \;\text{ and}$$
$$X_s \;=\; \sum_{t \in N^{-1}(v_s)} b(v_t) \;=\; X_s^+ + X_s^-.$$

Finally, for each $i$, define $Z_i = X_{+i} - X_{-i}$; this is the final belief $b(v_{+i})$ computed by the algorithm. Recall that the algorithm determines an output assignment to $x_i$ by the sign of $b(v_{+i})$; hence, our goal is to show that every $Z_i$ is positive with high probability.

Since all random variables $Z_i$'s follow the same distribution, in the following, we will fix some $i$ and consider $Z_i$; let $s$ to denote either $+i$ or $-i$ only. Below we state some basic relations on these random variables, which are clear from our choice and definition.

$$\begin{array}{rclcrcl} W_+ & = & S'_+ \cap N^{-1}(v_{-1}), & & W_- & = & S'_- \cap N^{-1}(v_{-1}), \\ X_s^+ & = & \displaystyle\sum_{+j \in W_+} -E_{s,+j}, & \text{and} & X_s^- & = & \displaystyle\sum_{-j \in W_-} -E_{s,-j}. \end{array} \tag{4}$$

We first estimate the expectation of $Z_i$. (Below let $n'$ denotes $n-1$.)

**Lemma 3.1.** $\mathrm{E}[\, Z_i \,] \;=\; (p-r)^2 n'$.

**Proof.** Consider $X_{+i}^+$, for example. From (2), (3) and (4), we can easily derive the following.

$$\mathrm{E}[X_{+i}^+] \;=\; \sum_{+j \in W_+} -\mathrm{E}[E_{+i,+j}] \;=\; \sum_{+j \in W_+} -p \;=\; -prn'.$$

The first equation holds because the value of each $E_{+i,+j}$ (where $j \neq 1$) is independent from $W_+$, connections from $v_{+j'}$ to $v_{-1}$. Similarly, we have

$$\mathrm{E}[X_{+i}^-] = -prn', \;\; \mathrm{E}[X_{-i}^+] = -r^2 n', \;\; \text{and} \;\; \mathrm{E}[X_{-i}^-] = -p^2 n'.$$

Hence, we have

$$\mathrm{E}[Z_i] \;=\; \Big( \mathrm{E}[X_{+i}^+] + \mathrm{E}[X_{+i}^-] \Big) - \Big( \mathrm{E}[X_{-i}^+] + \mathrm{E}[X_{-i}^-] \Big) \;=\; (p-r)^2 (n-1). \tag{5}$$

□

It follows from this analysis that if $p-r$ is large enough (i.e., $p - r > (n-1)^{-1/2}$), then $Z_i$ is positive *on average*. Then for proving that $Z_i$ is positive *with high probability* it now suffices to show that their deviation from the average is small with high probability, which is our goal in the following analysis.

For estimating the probability that $Z_i$ deviates from its expectation, we introduce positive parameters $\sigma_1$, $\sigma_2$, $\gamma_1$, ..., $\gamma_4$, and consider, for example, the following situation. (The other cases, which can be analyzed similarly, are omitted here.)

$$\begin{array}{rcl} Y_+ & = & \mathrm{E}[Y_+] + \sigma_1 n' = (r+\sigma_1)n', \quad Y_- \;=\; \mathrm{E}[Y_-] - \sigma_2 n' = (p-\sigma_2)n', \\ X_{+i}^+ & = & \mathrm{E}[X_{+i}^+ | Y_+ = (r+\sigma_1)n'] - \gamma_1 (r+\sigma_1)n' \;=\; -(pr - p\sigma_1 - \gamma_1(r+\sigma_1))n', \\ X_{+i}^- & = & \mathrm{E}[X_{+i}^- | Y_- = (p-\sigma_2)n'] - \gamma_2 (p-\sigma_2)n' \;=\; -(rp + r\sigma_2 - \gamma_2(p-\sigma_2))n', \\ X_{-i}^+ & = & \mathrm{E}[X_{-i}^+ | Y_- = (p-\sigma_2)n'] + \gamma_3 (p-\sigma_2)n' \;=\; -(p^2 + p\sigma_2 + \gamma_3(p-\sigma_2))n', \\ X_{-i}^- & = & \mathrm{E}[X_{-i}^- | Y_+ = (r+\sigma_1)n'] + \gamma_4 (r+\sigma_1)n' \;=\; -(r^2 - r\sigma_1 + \gamma_4(r+\sigma_1))n'. \end{array} \tag{6}$$

By ignoring the positive deviation, we can bound $Z_i$ as follows under this situation.

$$\begin{array}{rcl} Z_i & \geq & (p-r)^2 n' - ((\sigma_1+\sigma_2)p - \sigma_2 r \\ & & -\gamma_1(r+\sigma_1) - \gamma_2 p - \gamma_3 p - \gamma_4(r+\sigma_1))n'. \end{array} \tag{7}$$

Also for simplifying our argument we consider the case[2] that $r = p/3$. Then in order to show that $Z_i > 0$ with high probability, it suffices to show that

$$\sigma_1 \ < \ \frac{r}{2}, \qquad \sigma_2 \ < \ \frac{p}{2}, \tag{8}$$

and

$$\max\left( (\sigma_1 + \sigma_2)p,\, \sigma_2 r,\, \gamma_1(r+\sigma_1),\, \gamma_2 p,\, \gamma_3 p,\, \gamma_4(r+\sigma_1) \right) \ < \ \frac{(p-r)^2}{6}. \tag{9}$$

hold with high probability, which is analyzed by the next lemma.

**Lemma 3.2.** For any $\delta' > 0$, the probability that some of the bounds of (8) and (9) does not hold is at most $\delta'$ if

$$n' \ \geq \ \frac{(c'_{\mathrm{algo}} \ln(8/\delta'))}{p^2}, \tag{10}$$

where $c'_{\mathrm{algo}}$ is some sufficiently large constant independent from $n'$ and $p$.

**Proof.** A crucial point here is that random variables $E_{s',t'}$ and $E_{s'',t''}$ are mutually independent except that $E_{s',t'} = E_{-t',-s'}$ for all $s'$ and $t'$. Thus, probability analysis for (8) is a straightforward application of the standard Chernoff bound. Also it is easy to see that, for example, variables $E_{+i,+j}$, $+j \in S'_+$, considered for analyzing $X^+_{+i}$ under the condition $Y_+ = n'(r+\sigma_1)$ are mutually independent and independent from this condition (more precisely, independent from the value of $W_+$). Thus, analysis for (9) is again easy. As one example, we prove here that the probability that $\gamma_1(r+\sigma_1) \geq (p-r)^2/6$ is bounded by $\delta'/8$ for $n'$ and $p$ satisfying the condition (10) of the lemma.

Precisely speaking, our task is to bound the following probability, for each value $W$ of $W_+$ such that $\|W\| = (r+\sigma_1)n'$.

$$\Pr\left[ \gamma_1(r+\sigma_1) \geq \frac{(p-r)^2}{6} \ \middle|\ W_+ = W \right].$$

Recall that $\gamma_1$ is defined to express $X^+_{+i}$ by $X^+_{+i} = -\mu - \gamma_1(r+\sigma_1)n'$, where $\mu = -\mathrm{E}[X^+_{+i} | Y_+ = (r+\sigma_1)n']$. Also note that $\mu = -\mathrm{E}[X^+_{+i} | W_+ = W]$ $(= p(r+\sigma_1)n')$. Thus, we can restate the above by

$$\begin{aligned}
&\Pr\left[ X^+_{+i} \leq -\mu - \frac{(p-r)^2 n'}{6} \ \middle|\ W_+ = W \right] \\
&= \ \Pr\left[ \sum_{+j \in W_+} -E_{+i,+j} \leq -\left(1 + \frac{(p-r)^2 n'}{6\mu}\right)\mu \ \middle|\ (*) \right] \\
&= \ \Pr\left[ \sum_{+j \in W} E_{+i,+j} \geq \left(1 + \frac{(p-r)^2 n'}{6\mu}\right)\mu \ \middle|\ (*) \right],
\end{aligned}$$

[2] That is, we consider the case that $r$ is largest under our assumption $p \geq 3r$, which is intuitively the hardest case. Technically speaking, though, a bit more careful argument is required when $r$ is much smaller than $p$ because then $Y_+$ may be too small for using the Chernoff bound. But this case can be handled by using a bound such as $\sigma_1 < p/4$ instead of $\sigma_1 < r/2$ in (8). We leave this analysis to the interest reader.

where $(*)$ is event $W_+ = W$. Then since variables $E_{+i,+j}$ are independent (mutually and from the condition $W_+ = W$), we can simply use the standard Chernoff bound to get the following desired bound.

$$\begin{aligned}
&\Pr\left[\sum_{+j\in W} E_{+i,+j} \geq \left(1+\frac{(p-r)^2n'}{6\mu}\right)\mu \;\middle|\; W_+ = W\right] \\
&\quad\leq \exp\left(-\left(\frac{(p-r)^2n'}{6\mu}\right)^2\cdot\frac{\mu}{3}\right) \;=\; \exp\left(-\frac{(p-r)^4n'}{3\cdot 36p(r+\sigma_1)}\right) \\
&\quad\leq \exp\left(-\frac{(p-r)^4n'}{2\cdot 3^7pr}\right) \;\leq\; \exp\left(-\frac{2^3p^2n'}{3^7}\right) \\
&\quad\leq \exp\left(-\frac{2^3p^2}{3^7}\cdot\frac{c'_{\mathrm{algo}}\ln(8/\delta')}{p^2}\right) \;\leq\; \delta'/8.
\end{aligned}$$

Here we use the assumption that $r = p/3$; the last inequality holds if $c'_{\mathrm{algo}}$ is sufficiently large, i.e., if $c'_{\mathrm{algo}} \geq 2^3\cdot 3^7$. Similar arguments can be used for the other seven bounds of (8) and (9); hence, the probability that some of them does not hold is bounded by $(\delta'/8)\cdot 8 = \delta'$. □

In summary, if all bounds of (8) and (9) hold, then we have $Z_i > 0$, which means that the algorithm outputs $+1$ for $x_i$; thus, if the same situation holds for every $Z_{i'}$, $2 \leq i' \leq n$, then the algorithm yields all $+1$ planted solution. Therefore, we have the following theorem.

**Theorem 3.3.** There exists some constant $c_{\mathrm{algo}}$ such that for any $\delta > 0$, if $n \geq c_{\mathrm{algo}}\ln(16n/\delta)/p^2$, then the algorithm, executed with two different initial values for $b(v_{+1})$ (resp., $b(v_{-1})$), yields a pair of all $+1$ and all $-1$ planted solution with probability $1-\delta$.

## References


[1] D. Achlioptas, H. Jia, and C. Moore, Hiding satisfying assignments: two are better than one, *J. AI Research* 24, pp.623–639, 2005. (Preliminary version in *Proc. 19th Natl. Conf. on Artificial Intelligence* (AAAI '04), 131–136.)

[2] D. Boughaci and H. Drias, Efficient and experimental meta-heuristics for MAX-SAT problems, in Proc. *Workshop on Experimental/Efficient Algorithms*, Lecture Notes in Computer Science 3503, pp.501–512, 2005.

[3] H. Drias, Scatter search with walk strategy for solving hard Max-W-Sat problems, in Proc. of *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Lecture Notes in Compute Science 2070, pp.35–44, 2001.

[4] J. Håstad, Some optimal inapproximability results, *Journal of the ACM* 48, pp.798–859, 2001.

[5] B. Mazure, L. Sais, and E. Greroire, A tabu search for sat, in Proc. of *the 15th National Conf. on Artificial Intelligence* (AAAI97), pp.281–285, 1997.

[6] C. McDiarmid, Concentration, in *Probabilistic Methods for Algorithmic Discrete Mathematics* (M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, eds.), Springer, 1998.

[7] R. McEliece, D. MacKay, and J. Cheng, Turbo decoding as an instance of Pearl's "Belief Propagation" algorithm, in *IEEE J. on Selected Areas in Comm.* 16(2), 1998.

[8] M. Motoki, Random instance generation for MAX 3SAT, in Proc. of *7th Annual Int'l Computing and Combinatorics Conference* (COCOON'01), Lecture Notes in Computer Science 2108, pp.502–508, 2001.

[9] M. Motoki, Test instance generation for MAX 2SAT, in Proc. of *11th Int'l Conf. on Principles and Practice of Constraint Programming* (CP'05), Lecture Notes in Computer Science 3709, pp.787–791, 2005.

[10] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge Univ. Press, 1995.

[11] M. Onsjö and O. Watanabge, Simple algorithms for graph partition problems, Research Report C-212, Dept. of Math. and Comput. Sci., Tokyo Inst. of Tech., 2005.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., 1988.

[13] A.D. Scott and G.B. Sorkin, Faster algorithms for MAX CUT and MAX CSP, with polynomial expected time for sparse instances, in *Proc. APPROX and RANDOM 2003*, LNCS 2764, pp.382–395, 2003.

[14] B. Selman, A. Henry, Z. Kautz, and B. Cohen, A new method for solving hard satisfiability problems, in Proc. of *the 10th National Conf. on Artificial Intelligence* (AAAI92), pp.440-446, 1992.

[15] M. Yamamoto, Generating instances for MAX2SAT with optimal solutions, *Theory of Computing Systems*, to appear.