ISSN 1342-2812

# Research Reports on Mathematical and Computing Sciences

Message Passing Algorithms for MLS-3LIN Problem

Osamu Watanabe

October 2011, C–276

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

# Message Passing Algorithms for MLS-3LIN Problem

Osamu Watanabe
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
E-mail: watanabe@is.titech.ac.jp

MLS-3LIN problem is a problem of finding a most likely solution for a given system of perturbed 3LIN-equations under a certain planted solution model. This problem is essentially the same as MAX-3XORSAT problem. We investigate the average-case performance of message passing algorithms for this problem, where input instances are generated under the planted solution model with equation probability $p$ and perturbation probability $q$. For some variant of a typical message passing algorithm, we prove that $p = \Theta(1/(n \ln n))$ is the threshold for the algorithm to work w.h.p. for any fixed constant $q < 1/2$.

## 1 Introduction

In this paper we investigate the average-case performance of message passing algorithms to find a "most likely solution" for a given system of "perturbed" linear equations on GF(2). In particular, we consider the case that equations are restricted to having three variables in each equation, which we call 3LIN-equations (see, e.g., Figure 1). For our average-case scenario, we consider a "planted solution model" that is the following way to generate a set of linear equations on $n$ variables for a given $n$: for a randomly chosen *planted solution*, generate a set of linear equations by putting each 3LIN-equation that is consistent with the planted solution into the set with probability $p$ independently. Then *perturb* some of those equations by flipping their right hand side values independently at random with some small probability $q < 1/2$. Under this scenario, a *most likely solution* is essentially the planted solution (see Section 2.2 for the explanation). Thus, our problem is to find the planted solution for a given system of perturbed 3LIN-equations generated by the above scenario. We call this problem **M**ost-**L**ikely **S**olution finding problem for perturbed **3LIN**-equations, MLS-3LIN for short.

$$\begin{aligned} x_1 \oplus x_2 \oplus x_4 &= +1 \\ x_2 \oplus x_3 \oplus x_4 &= -1 \\ x_3 \oplus x_5 \oplus x_6 &= +1 \\ \vdots & \quad \vdots \end{aligned}$$

An example of 3LIN-equations. In this paper we use $-1$ for `true` (or 1 in GF(2)) and $+1$ for `false` (or 0 in GF(2)). Then the exclusive-or operation $\oplus$ (or the mod 2 addition) is simply the integer multiplication.

**Figure 1.** 3LIN-equations

This problem is essentially the same as MAX-3LIN problem (resp., MAX-3XORSAT problem), a problem of finding an optimal assignment, i.e., an assignment to variables satisfying the largest number of 3LIN-equations (resp., XOR-clauses). It is well known that MAX-3LIN (even its restricted version MAX-2LIN) is NP-hard; furthermore, it has been conjectured [KV05] that MAX-2LIN is hard to approximate well in polynomial-time. On the other hand, some heuristics seem to work well for solving these problems *on average*. As an example of such heuristics, we consider simple message passing algorithms that have been studied for

similar problems [OW06, WY10, WY06], and we investigate their average-case performance on MLS-3LIN instances generated by the above planted solution model.

We show that, for some variant of message passing algorithm, it works w.h.p. if the equation probability $p$ is greater than $c/(n \ln n)$ for sufficiently large $c$ (for any constant perturbation probability $q < 1/2$) and it fails w.h.p. if $p$ is smaller than $c/(n \ln n)$ for sufficiently small $c$ (even if $q = 0$). The positive result is proved by a reduction to MLS-2LIN problem and a variation of the known spectral analysis given in, e.g., [FO04]. The negative result is proved by an extension of the analysis of [COW10].

# 2 Preliminaries

We explain notations that may require some remarks. For any set $S$, we use $\#S$ to denote the number of elements in $S$. Vectors are often specified as, e.g., $(a_1, ..., a_n)$, and we use bold italic letters for denoting vectors. By $\mathbf{1}$ we denote a vector $(+1, ..., +1)$ consisting of all $+1$; unless specified otherwise, its size should be determined by context. We use Greek letters, e.g., $\boldsymbol{\xi}$ for unit vectors, and for any vector $\boldsymbol{u}$, let $\overline{\boldsymbol{u}}$ to denote its normalized one, i.e., $\boldsymbol{u}/\|\boldsymbol{u}\|$. The inner product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is denoted as $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$.

Let $\ln n$ denote the natural logarithm, while the binary logarithm (i.e., the base 2 logarithm) is denoted by log. Let $o(1)$ denote any function that goes 0 when $n$ goes $\infty$. By "w.h.p." (with high probability) we mean the probability $1 - o(1)$.

## 2.1 MLS-3LIN Problem and its Planted Solution Model

When discussing linear equations over GF(2), we use $+1$ for 0 (or false) and $-1$ for 1 (true). Then the GF(2) addition or the XOR operation $\oplus$ is simply the integer multiplication; that is, we have $+1 \oplus +1 = +1$, $-1 \oplus -1 = +1$, etc. This notation is somewhat convenient because we can now use 0 (for undefined) and assume that $x \oplus 0 = 0 \oplus x = 0$.

In the following discussion, we consider MLS-2LIN and MLS-3LIN problems. But since notions necessary for our discussion are defined in the same way for both problems, we explain them by using MLS-2LIN as an example. Throughout this paper, we use $n$ to denote the number of variables and $m$ to denote the number of equations. We use $x_1, ..., x_n$ for denoting variables, and a *2LIN-equation* is an equation like $x_2 \oplus x_3 = -1$ that has two variables (added by $\oplus$) on the left-hand side of the equation and either $+1$ or $-1$ on the right-hand side of the equation. Each equation is denoted by, e.g., $e$, $e_i$, etc, and we use $E = \{e_1, ..., e_m\}$ to denote a system of 2LIN-equations. We often use a vector $\boldsymbol{a} = (a_1, ..., a_n) \in \{-1, +1\}^n$ as an assignment to variables $x_1, ..., x_n$

**Remark.** *We assume that the left-hand side of a kLIN-equation is of the form $x_{i_1} \oplus \cdots \oplus x_{i_k}$ for some $i_1 < i_2 < \cdots < i_k$. Hence, there are $\binom{n}{k}$ possible patterns for the left-hand side of kLIN equations over $n$ variables.*

For a distribution model, we use the planted solution model explained in Introduction. For simplicity, we fix our planted solution to $\mathbf{1} = (+1, ..., +1)$; hence, the "correct" right-hand side value of each equation is always $+1$. Throughout this paper, we use $p$ to denote *equation*

*density*, a parameter that determines the probability of selecting each equation, and $q$ to denote *perturbation probability*, a parameter that determines the probability of flipping each value to $-1$, the value that is inconsistent to our assumed planted solution $\mathbf{1}$. We use $E$ to denote a generated set of 2LIN-equations, which is indeed an instance of MLS-2LIN problem. Note that $E$ is a random variable, and throughout this paper, all the probabilities are essentially on this random variable. For example, the number $m$ of equations of a random MLS-2LIN instance is a random variable following the binomial distribution $\mathrm{Bin}(n(n-1)/2, p)$. Similarly, the number of equations consistent (resp., inconsistent) with the planted solution follows the binomial distribution $\mathrm{Bin}(m, 1-q)$ (resp., $\mathrm{Bin}(m, q)$).

Consider any set $E$ of 2LIN-equations. Note first that under the planted solution model (with parameters $p > 0$ and $0 < q < 1/2$), $E$ can be generated from any planted solution $\boldsymbol{a} \in \{-1, +1\}^n$; also note that the generation probability $\Pr[E|\boldsymbol{a}]$ varies depending on $\boldsymbol{a}$. Now *MLS-2LIN problem* (**M**ost-**L**ikely **S**olution finding problem for **2LIN**-equations) is to find a solution $\boldsymbol{a}$ that maximizes $\Pr[E|\boldsymbol{a}]$. It should be noticed here that the notion of "solution" for MLS-2LIN depends on a way to generate instances. But as shown below, under our planted solution model, the notion of "solution" is robust for any $p > 0$ and $q < 1/2$.

## 2.2 Basic Properties of Solutions of MLS-3LIN Problem

Here we consider MLS-3LIN problem for our explanation, but the corresponding properties hold for MLS-2LIN problem.

Fix parameters $p > 0$ and $q < 1/2$, and consider any instance $E$ for MLS-3LIN problem. Since $q < 1/2$, it is clear that a solution of $E$ (w.r.t. MLS-3LIN problem) is nothing but an *optimal assignment*, i.e., an assignment $\boldsymbol{a} \in \{-1, +1\}^n$ that maximizes the number of satisfiable equations of $E$, in other words, a solution of $E$ w.r.t. MAX-3LIN problem. Thus, (under our planted solution model) both MLS-3LIN and MAX-3LIN problems ask for essentially the same solutions.

**Proposition 1** *For any $p > 0$ and $q < 1/2$, and for any set $E$ of 3LIN-equations, $\boldsymbol{a}$ maximizes $\Pr[E|\boldsymbol{a}]$ (under the planted solution model w.r.t. $p$ and $q$) if and only if it is an optimal assignment of $E$.*

Consider an instance $E$ generated from some planted solution $\boldsymbol{a}$. If $\boldsymbol{a}$ is used as an assignment, then *on average* $m(1-q)$ equations are satisfied while $mq$ equations are not. It has been shown [BO09] that this is indeed best possible with high probability; furthremore, there is any other comparable solution if $p$ is large enough.

**Proposition 2** *[BO09] For any sufficiently large constants $c$ and $c'$, let $p$ and $q$ be any parameters satisfying both (1) $p > c \log n/n^2$, and (2) $q < 1/2 - c'\sqrt{\log n/(n^2 p)}$. Consider a set $E$ of 3LIN-equations generated from some planted solution $\boldsymbol{a}$ under the planted solution model w.r.t. $p$ and $q$. Then w.h.p. the planted solution $\boldsymbol{a}$ is the unique solution of $E$ for MLS-3LIN problem; in other words, $\boldsymbol{a}$ is the unique optimal assignment for $E$.*

**Remark.** *The same property holds for MLS-2LIN if both (1) $p > c \log n/n$, and (2) $q < 1/2 - c'\sqrt{\log n/(np)}$ hold for sufficiently large $c$ and $c'$.*

**procedure Alg_2LIN** for MLS-2LIN problem
**input** An instance $E = \{e_1, ..., e_m\}$ for MLS-2LIN problem;
assign $+1$ or $-1$ to $\mathtt{x}_1$ and assign 0 to all the other variables; $(*1)$
**repeat** MAXSTEP times **do** {
**for** all $\mathtt{x}_i$, $1 \le i \le n$, **in parallel do** {

$$\mathtt{x}_i := \mathrm{sgn}\left(\sum_{e \in E_i} m_{e \to i}\right); \quad (*2)$$

}
**if** no change is made on $\vec{\mathtt{x}} = (\mathtt{x}_1, ..., \mathtt{x}_n)$ **then exit**; $(*3)$
}
output $(\mathrm{sgn}(\mathtt{x}_1), ..., \mathrm{sgn}(\mathtt{x}_n))$;
**end-procedure**
*Note (see also explanation in the text):*

- $E_i$ = the set of equations containing $\mathtt{x}_i$.
- $m_{e \to i}$ = a "message" from $e$ to $\mathtt{x}_i$ (either $-1$, $+1$, or 0), and
- $\mathrm{sgn}(z) = +1$ (resp., $= -1$) if $z > 0$ (resp., $z < 0$), and it is 0 if $z = 0$.

**Figure 2.** A simple message passing algorithm for MLS-2LIN problem

# 3 Message Passing Algorithms

We consider simple message passing algorithms such as algorithm **Alg_2LIN** for MLS-2LIN stated in Figure 2. Following this description, let us see the outline of the algorithm. Algorithm **Alg_2LIN** uses $n$ integer variables $\vec{\mathtt{x}} = (\mathtt{x}_1, ..., \mathtt{x}_n)$ to keep the current candidate for an optimal assignment. Initially at $(*1)$ only $\mathtt{x}_1$ is given either $+1$ or $-1$, and the others are set 0. Then at $(*2)$ of the main loop, values of all $\mathtt{x}_i$'s are updated in parallel, which is regarded as one "updating process". The algorithm executes this updating process for at most MAXSTEP times, or until values of $\vec{\mathtt{x}}$ are "stabilized", meaning that no change is made after the updating process. Both experimentally and theoretically it is enough to set MAXSTEP to $10 \log n$.

The core of the algorithm is the updating process at $(*2)$. Here the new value of each $\mathtt{x}_i$ is determined by taking the majority vote of all messages from equations containing $\mathtt{x}_i$. Let $e$ be any 2LIN-equation containing $\mathtt{x}_i$, and assume that it is of the form "$\mathtt{x}_i \oplus \mathtt{x}_j = b$". Then a "message" $m_{e \to i}$ from $e$ to variable $x_i$ is computed as $m_{e \to i} := \mathtt{x}_j \oplus b$. For example, if $b = -1$ and $\mathtt{x}_j = +1$, then $m_{e \to i} = -1$. This is the value of $\mathtt{x}_i$ in order to satisfy the equation $e$ with the current value of $\mathtt{x}_j$. Then the new value for $\mathtt{x}_i$ is determined by taking the majority vote of these messages.

Algorithm for MLS-3LIN

Algorithm **Alg_3LIN** we consider for MLS-3LIN problem is almost the same as **Alg_2LIN**.

A message is computed similarly for 3LIN-equations. For example, for $e =$ "$\mathtt{x}_i \oplus \mathtt{x}_j \oplus \mathtt{x}_k = b$",

a message $m_{e\to i}$ from $e$ to $\mathtt{x}_i$ is computed as $m_{e\to i} := \mathtt{x}_j \oplus \mathtt{x}_k \oplus b$, which has the same meaning as the 2LIN case. Note here that no message or 0 message is computed as $m_{e\to i}$ if either $\mathtt{x}_j$ or $\mathtt{x}_k$ is 0, i.e., undetermined.

Relatively big difference is the number of variables that are assigned $+1$ or $-1$ at $(*1)$. This number should be more than one for MLS-3LIN problem, because otherwise all messages become 0 and no update occurs at $(*2)$. In our **Alg_3LIN** we assign $+1$ or $-1$ to $\mathtt{x}_1, ..., \mathtt{x}_{\log n}$ variables. Note that the number of all possible assignments to $\mathtt{x}_1, ..., \mathtt{x}_{\log n}$ is $2^{\log n} = n$; hence, by iterating just $n$ times, we can try all possible assignments and one of them must be consistent with the planted solution for generating the input instance $E$. Thus, we may assume that the correct values are assigned to $\mathtt{x}_1, ..., \mathtt{x}_{\log n}$ at $(*1)$.

# 4 Analysis of an Message Passing Algorithm for MLS-2LIN

In order to analyze the average-case performance of some message passing algorithm for MLS-3LIN, we first show that some variant of **Alg_2LIN** performs well for relatively small $p$. Our analysis is an application of the analysis of spectral algorithms that have been studied extensively [AK97, CGLS03, FO04]. In particular, we follow the outline given in [FO04] and use a key technical lemma from [CGLS03].

## 4.1 Modification of the Algorithm ⇒ Spectral Algorithm

For our analysis, we modify algorithm **Alg_2LIN** on three points.

First, we introduce some preprocessing on a given input set $E$ of 2LIN-equations. Fix any sufficiently large constant $c_{\mathrm{alg}} > 0$. We remove variables that appear more than $c_{\mathrm{alg}} np$ times and equations containing such variables. Note that each variable appears $(n-1)p$ times on average; hence, the number of such variables with much larger occurrence should be very small, which is shown formally in the next section. Thus, even if we use some fixed assignment for those removed variables, introduced error is negligible. Let $n'$ be the number of variables remained after this preprocessing; we can assume[1] that all variables appear at most $c_{\mathrm{alg}} n' p$ times.

Secondly, we simplify the updating formula at $(*2)$ by omitting the application of the sign function. That is, a new updating process is

$$\left.\begin{array}{l}\textbf{for all } \mathtt{x}_i,\ 1 \le i \le n, \textbf{ in parallel do } \{ \\ \quad \mathtt{x}_i \ := \ \displaystyle\sum_{e \in E_i} m_{e\to i}; \\ \} \\ \vec{\mathtt{x}} \ := \ \vec{\mathtt{x}} / \|\vec{\mathtt{x}}\|;\end{array}\right\} \ (*4)$$

Here we introduce normalization, which is necessary in practice (but not so important theoretically) since otherwise the values of $\vec{\mathtt{x}}$ grow very quickly.

[1] Precisely speaking, we may have to remove some more variables due to the change of $n$. But we can show that this removal process converges very quickly; see, e.g., [Coj06] for its detail treatment.

Thirdly, we change the stopping condition at $(*3)$. We determine the "stabilization" by checking whether sign is not changed at all variables $\mathtt{x}_i$, $1 \leq i \leq n$, after the updating process.

Now for a given set $E$ of 2LIN-equations, consider an $n \times n$ matrix $A_E = (a_{ij})$, where

$$a_{ij} \;=\; a_{ji} \;=\; \begin{cases} 1, & \text{if } x_i \oplus x_j = +1 \text{ exists in } E, \\ -1, & \text{if } x_i \oplus x_j = -1 \text{ exists in } E, \\ 0, & \text{otherwise.} \end{cases}$$

Then it is easy to see that the above $(*4)$ is expressed as $\vec{\mathtt{x}} := A_E \vec{\mathtt{x}}^{\mathrm{T}}$; That is, the updating process is nothing but one iteration of the power method for computing *the first eigenvector* for $A_E$, i.e., an eigenvector with the largest eigenvalue. Thus, let us call the modified algorithm **AlgS_2LIN** (where "S" for *spectral*). From this view point and by using well known facts on the power method, the following property can be shown easily.

**Proposition 3** *For any $p > 0$ and $q < 1/2$, consider a random instance $E$ for MLS-2LIN problem generated under the planted model w.r.t. $p$ and $q$. Let $\boldsymbol{\xi}_1 = (\xi_{1,1}, ..., \xi_{1,n})$ be the first eigenvector of $A_E$. Then w.h.p. algorithm* **AlgS_2LIN** *on $E$ terminates in* `MAXSTEP` *steps, yielding $(\mathrm{sgn}(\xi_{1,1}), ..., \mathrm{sgn}(\xi_{1,n}))$ as an output assignment $\boldsymbol{a}_{\mathrm{ans}}$.*

Now we would like to show that the assignment $\boldsymbol{a}_{\mathrm{ans}}$ produced by **AlgS_2LIN** is close to the planted solution $\mathbf{1}$ with high probability. Here in order to measure the difference between two solutions, we use

$$\mathrm{diff}(\mathbf{1}, \boldsymbol{a}_{\mathrm{ans}}) \;\stackrel{\mathrm{def}}{=}\; 1 - \frac{\langle \mathbf{1}, \boldsymbol{a}_{\mathrm{ans}} \rangle}{n} \;=\; 1 - \langle \overline{\mathbf{1}}, \overline{\boldsymbol{a}}_{\mathrm{ans}} \rangle.$$

Then our goal is precisely stated as follows. (Although we fix the form of $p$ to $c/n$ for some constant, the result holds for larger $p$. Note that we sometimes require $p \leq \log n^{O(1)}/n$ in the later analysis; but a simpler argument is applicable for larger $p$.)

**Theorem 1** *Let $p = c/n$ and $q = 1/2 - \delta$, where $c > 0$ and $\delta$, $0 < \delta \leq 1/2$, are any constants w.r.t. $n$. Let $\boldsymbol{a}_{\mathrm{ans}}$ denote the output of* **AlgS_2LIN** *on a random instance $E$ for MLS-2LIN problem generated under the planted solution model w.r.t. $p$ and $q$. Then for some constant $d_0 > 0$, if $c\delta$ is sufficiently large, then w.h.p. we have $\mathrm{diff}(\mathbf{1}, \boldsymbol{a}_{\mathrm{ans}}) \leq d_0 (c\delta)^{-1}$.*

From this theorem, for any fixed $\delta > 0$, if $c$ is large enough, then we can guarantee that the difference $\mathrm{diff}(\mathbf{1}, \boldsymbol{a}_{\mathrm{ans}})$ between the planted solution and the obtained solution is small w.h.p.

### 4.2 Spectral Analysis of AlgS_2LIN

We explain the outline of the proof of Theorem 1. In this abstract, all proofs of technical lemmas are omitted, which can be found in Appendix.

Technically we base the following lemma reported as Lemma 39 in [CGLS03]. (The lemma is stated in a slightly generalized form; see Appendix for the explanation.)

**Lemma 1** *Let $\widetilde{c}_0 > 0$ be some sufficiently large constant. For any $\widetilde{p}'$ such that $\widetilde{c}_0/n \leq \widetilde{p}' \leq (\ln n)^{O(1)}/n$, let $D = (d_{ij})_{1 \leq i,j \leq n}$ be an $n \times n$ random symmetric matrix such that $d_{ii} = 0$ for each $1 \leq i \leq n$, and $d_{ij} = 1$ (resp., $d_{ij} = 0$) with probability $\widetilde{p}'$ (resp., $1 - \widetilde{p}'$) independently for each $1 \leq i < j \leq n$. Then for any $\widetilde{p} \geq \widetilde{p}'$ and any $\widetilde{c}_{\mathrm{alg}} > 1$, there exists some $\widetilde{c}_1 > 0$ such that the following three statements hold w.h.p.*

*(1) Let $V'' = \{i \in \{1, ..., n\} | \sum_{j=1}^{n} d_{ij} > \widetilde{c}_{\mathrm{alg}} n\widetilde{p}\}$, and $V' = \{1, ..., n\} - V''$. Let $n' = \#V'$ and $D' = (d_{ij})_{i,j \in V'}$ be the induced $n' \times n'$ matrix. Then we have $n' \geq (1 - \exp(-n\widetilde{p}/\widetilde{c}_1))n$.*

*(2) For any unit vector $\boldsymbol{\xi} \perp \mathbf{1}$, we have $\|D'\boldsymbol{\xi}\| \leq \widetilde{c}_1\sqrt{n'\widetilde{p}}$.*

*(3) $\|D'\overline{\mathbf{1}} - n'\widetilde{p}'\overline{\mathbf{1}}\| \leq \widetilde{c}_1\sqrt{n'\widetilde{p}}$.*

For our discussion, we introduce new notations. Let $c_0$ denote the constant $\widetilde{c}_0$ of the above lemma. Fix $p = c/n$ and $q = 1/2 - \delta$ with some $c$ and $\delta$ satisfying two conditions (C1) and (C2), where (C1) $\iff$ $c \geq c_0$ and $0 < \delta \leq 1/2$, while (C2) is specified later. For sufficiently large $n$, let $E$ be 2LIN-equations over $n$ variables generated randomly following the planted solution model w.r.t. $p$ and $q$. Let $A_E = (a_{ij})$ denote the matrix defined from $E$ as before. We fix $n$, $E$, and $A_E$ throughout this subsection.

First we show that not so many variables are removed by the preprocessing of **AlgS_2LIN**. The following lemma is easily obtained by applying the above lemma to matrix $(|a_{ij}|)$ with $\widetilde{c}_1$ slightly smaller than $c_{\mathrm{alg}}$.

**Lemma 2** *The number of variables removed by the preprocessing of* **AlgS_2LIN** *is bounded by $n \exp(-c/d)$ for some $d > 0$.*

Thus, even if we use some fixed assignment to those removed variables (and they are all incorrect w.r.t. the planted solution), its effect in $\mathrm{diff}(\mathbf{1}, \boldsymbol{a}_{\mathrm{ans}})$ is bounded by $\exp(-c/d)$.

In the following, we redefine $n$, $E$, and $A_E$ to the number of variables, the set of 2LIN-equations, and its corresponding matrix after the preprocessing. Below we simply state $A$ for this $A_E$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be $A$'s eigenvalues, and for each $i$, $1 \leq i \leq n$, let $\boldsymbol{\xi}_i$ be the *unit* eigenvector corresponding to $\lambda_i$.

Now let $B = (b_{ij})$ and $C = (c_{ij})$ denote matrices that have 1 at each $ij$-entry such that $a_{ij} = 1$ and $a_{ij} = -1$ respectively. Then we have $A = B - C$; also roughly, we have $\Pr[b_{ij} = 1] = p(1 - q)$ and $\Pr[c_{ij} = 1] = pq$. Furthermore, though $B$ and $C$ are not independent, we may assume that $b_{ij}$, $1 \leq i < j \leq n$ (resp., $c_{ij}$, $1 \leq i < j \leq n$) are mutually independent. Then we can apply Lemma 1 to $B$ and $C$ from which the following lemma follows. (Here and throughout this subsection, we use $p'$ to denote $p(1 - 2q) = c\delta/n$.)

**Lemma 3** *The following holds w.h.p. for some constant $c_1 > 0$.*

*(1) For any unit vector $\boldsymbol{\eta} \perp \mathbf{1}$ and for any unit vector $\boldsymbol{\xi}$, we have $|\langle A\boldsymbol{\eta}, \boldsymbol{\xi}\rangle| \leq 2c_1\sqrt{np}$.*

*(2) For any $\lambda_i$, $2 \leq i \leq n$, we have $|\lambda_i| \leq 3c_1\sqrt{np}$.*

*(3) $\|A\overline{\mathbf{1}} - np'\overline{\mathbf{1}}\| \leq 2c_1\sqrt{np}$.*

Now we define the condition (C2) $\iff$ $6c_1\sqrt{np} \leq np'$ ( $\iff$ $(6c_1\delta^{-1})^2 \leq c$). By using (2) of the above lemma, we can show the following lower bound for $\langle \overline{\mathbf{1}}, \boldsymbol{\xi}_1 \rangle$.

**Lemma 4** *Assume that (C2) holds for $c$ and $\delta$. Then w.h.p. we have*

$$\langle \overline{\mathbf{1}}, \boldsymbol{\xi}_1 \rangle \ \geq \ \sqrt{1 - \frac{16c_1^2}{np'}} \ = \ \sqrt{1 - \frac{16c_1^2}{c\delta}}.$$

Now we are ready to prove Theorem 1.

**Proof of Theorem 1.** Let $\alpha n$ be the number of nonpositive element of $\boldsymbol{\xi}_1$. Here we have the following constraints:

$$\sum_{i=1}^{n} \xi_{1,i}^2 \ = \ 1, \ \text{ and } \ \sum_{i=1}^{n} \xi_{1,i} \cdot \frac{1}{\sqrt{n}} \ \geq \ \sqrt{1 - \frac{16c_1^2}{c\delta}}.$$

Then $\alpha n$ becomes the largest if $\alpha n$ elements of $\boldsymbol{\xi}_1$ is 0 and the other $(1-\alpha)n$ elements of $\boldsymbol{\xi}_1$ is $1/\sqrt{(1-\alpha)n}$. In this case, we have

$$\frac{\sqrt{(1-\alpha)n}}{\sqrt{n}} \ \geq \ \sqrt{1 - \frac{16c_1^2}{c\delta}},$$

which is equivalent to $\alpha \leq 16c_1^2(c\delta)^{-1}$.

Note that $\mathrm{diff}(\mathbf{1}, \boldsymbol{a}_{\mathrm{ans}})$ is the sum of $\alpha$ and the error from the removed variables at the preprocessing. Since the latter is bounded by $\exp(-c/d)$ that is bounded by $d'(c\delta)^{-1}$ for some $d' > 0$ (for sufficiently large $c$), the bound of the theorem is shown with $d_0 = 4c_1^2 + d'$. $\square$

# 5 Analysis of an Message Passing Algorithm for MLS-3LIN

Armed with the analysis of the previous section, we now analyze some message passing type algorithm for MLS-3LIN problem. Again though we base the algorithm **Alg_3LIN**, we consider some modifications to make it easier to analyze. Our idea is simple; we regard the step ($*1$) of the algorithm as a reduction from MLS-3LIN to MLS-2LIN. Recall that in the execution of **Alg_3LIN**, variables $\mathtt{x}_1, ..., \mathtt{x}_{\log n}$ are assigned values (where we may assume that this partial assignment is consistent with the planted solution because all possible combinations of values are examined), and then the main iteration of the updating process is executed. Thus we may expect that some number of 2LIN-equations are generated by this partial assignment and that the main iteration is somewhat similar to executing **Alg_2LIN** on those 2LIN-equations. We analyze an algorithm that executes exactly in this way. That is, we consider an algorithm that executes as follows: (1) assign values to $\mathtt{x}_1, ..., \mathtt{x}_{\log n}$; (2) collect 2LIN-equations[2] obtained by this partial assignment to some set $E'$, and then (3) execute **AlgS_2LIN** on $E'$. The algorithm runs (1) $-$ (3) with all possible partial assignments. We call this algorithm as **AlgR_3LIN** (where "R" is for *reduction*).

For this algorithm, it is not so hard to obtain some positive result similar to MLS-2LIN problem.

[2]Precisely speaking, if more than two equations with the same left-hand side are obtained, then its right-hand side is determined by taking the majority vote.

**Theorem 2** *Let $p = c/(n \log n)$ and $q = 1/2 - \delta$, where $c > 0$ and $\delta$, $0 < \delta \leq 1/2$, are any constants w.r.t. $n$. Let $\boldsymbol{a}_{\rm ans}$ denote the output of* **AlgR_3LIN** *on a random instance $E$ for MLS-3LIN problem generated following the planted solution model w.r.t. $p$ and $q$. Then for some constant $d_0 > 0$, if $c\delta$ is sufficiently large, then w.h.p. we have* $\mathrm{diff}(\mathbf{1}, \boldsymbol{a}_{\rm ans}) \leq d_0(c\delta)^{-1}$.

**Proof.** Consider the point in the execution of **AlgR_3LIN** where the partial assignment consistent with the planted solution is chosen. Let $E'$ be the set of 2LIN-equations obtained from $E$ by this partial assignment. It suffices to show that the distribution of $E'$ is equivalent to the one under the planted solution model for MLS-2LIN w.r.t. $\widehat{p}$ and $\widehat{q}$ with $\widehat{p} \geq c/(2n)$ and $\widehat{q} \leq q$.

First it is clear that each pair $x_i$ and $x_j$ appears in the left-hand side of an equation in $E'$ independently. Furthermore, the probability $\widehat{p}$ that each pair appears can be bounded by

$$\widehat{p} \;=\; 1 - (1-p)^{\log n} \;\geq\; 1 - 1 + p \log n - \cdots . \;\geq\; \frac{c}{2n}.$$

On the other hand, since no new error is introduced, it is easy to show that $\widehat{q} \geq q'$. $\square$

Next we discuss the limitation of **Alg_3LIN**[3]. For this, we consider the extreme case where $q = 0$, that is, the case where no perturbation is introduced[4]. We show below even in this case the algorithm terminates without determining values of many variables if $p$ is small. Note that when we start from the updating process from the partial assignment to $\mathtt{x}_1, ..., \mathtt{x}_{\log n}$ that is consistent with the planted solution, the value of each $\mathtt{x}_i$, $\log n + 1 \leq i \leq n$, is determined correctly once some equation $e$ containing $\mathtt{x}_i$ sends a message $m_{e \to i}$ to $\mathtt{x}_i$. This occurs if and only if the other two variables in $e$ are assigned some value. On the other hand, the value of $\mathtt{x}_i$ is not determined if no message is sent to $\mathtt{x}_i$ from all equations that $\mathtt{x}_i$ appears. If this happens on many variables in an early stage of the execution, the algorithm stops without giving any value to many variables.

A process of passing such messages is called "propagation", and it has been investigated in depth in [BO09, COW10]. By extending the argument in [COW10], we can show the following limitation of **Alg_3LIN**.

**Theorem 3** *For any positive $c$ less that 1 (say, $c = 0.99$), consider the execution of* **Alg_3LIN** *on a random instance $E$ for MLS-3LIN problem generated under the planted solution model w.r.t. $p = c/(n \log n)$ and $q = 0$. Then w.h.p. the algorithm terminates by assigning nonzero values to less than $\log n$ variables of $\mathtt{x}_{\log n+1}, ..., \mathtt{x}_n$.*

**Proof.** Following [COW10] we introduce a *propagation process* for investigating how many variables get assigned some nonzero value during the execution of **Alg_3LIN**. (Note that the propagation process defined below is not used for simulating the execution of **Alg_3LIN**; rather it is used to estimate situation/staistics at the end of the execution.)

[3]Though we state our result for **Alg_3LIN**, a similar result is possible for **AlgR_3LIN**.

[4]Note that in this case one can use the standard method in linear algebra to obtain the optimal solution (which is the solution satisfying all given equations). Here we consider this extreme case in order to show the limit of message passing algorithms.

We classify $n$ variables into three types: dead, alive and neutral. Let $\mathcal{D}_t$, $\mathcal{A}_t$, and $\mathcal{N}_t$ denote respectively the set of dead, alive, and neutral variables at time $t$ of the process. We also use $D_t$, $A_t$, and $N_t$ to denote their size. Initially, $\mathcal{D}_0 = \{\mathtt{x}_1\}$, $\mathcal{A}_0 = \{\mathtt{x}_2, ..., \mathtt{x}_{\log n}\}$, and the rest is put into $\mathcal{N}_0$. At each time $t = 1, 2, ...$, we select one variable $\mathtt{x}_j$ from $\mathcal{A}_{t-1}$, and for every $\mathtt{x}_i \in \mathcal{N}_{t-1}$, check whether there is an equation $e$ in $E$ containing $\mathtt{x}_i$, $\mathtt{x}_j$, and some $\mathtt{x}_k \in \mathcal{D}_{t-1}$. If so, we move $\mathtt{x}_i$ from $\mathcal{N}_{t-1}$ to $\mathcal{A}_t$. After all possibilities are examined, $\mathtt{x}_j$ is moved from $\mathcal{A}_{t-1}$ to $\mathcal{D}_t$. Thus, we have $D_t = t$ and $N_t = n - t - A_t$. Also note that the process terminates at time $t$ if and only if either $A_t = 0$ or $N_t = 0$. We use $T$ to denote this stopping time. We say that the process *succeeds* if $\mathcal{N}_t$ gets empty first and *fails* if $\mathcal{A}_t$ gets empty first. It is easy to see that the process succeeds if and only if the algorithm **Alg_3LIN** assigns values to all variables.

Let $X_t$ denote the number of variables that are newly added to $\mathcal{A}_t$. Then $\sum_{1 \le t \le T} X_t$ is the number of variables that are assigned $+1$ or $-1$ value by **Alg_3LIN**. Here we show that $T$ as well as $\sum_{1 \le t \le T} X_t$ is small. Clearly, the process does not terminates until time $\log n - 1$ because $\mathcal{A}_0$ contains initially $\log n - 1$ variables. Here we consider $t_0 = 2 \log n - 1$, and discuss the probability that $T \ge t_0$. Suppose that $T \ge t_0$, i.e., the process does not terminate before time $t_0$. Then we have $A_t \ge 1$ for all $t \in [t_0]$, and since one variable is moved from $\mathcal{A}$ to $\mathcal{D}$ at each time, at least $t_0 - (\log n - 1)$ $(= \log n)$ new alive variables must be created; that is, we have $\sum_{1 \le t \le t_0} X_t \ge \log n$. On the other hand, the following lemma states that this is unlikely to occur. Thus, w.h.p. the process terminates before time $t_0$ and at most $t_0$ variables are assigned some nonzero value. This proves the theorem. $\square$

**Lemma 5** *W.h.p. we have*

$$\sum_{1 \le t \le t_0} X_t \ < \ \log n.$$

**Proof.** Note that the process is deterministic, and the randomness is due to the set $E$ of equations that are randomly generated. Here instead of executing the process for a given and fixed $E$, for each equation $e$, the decision whether it is put into $E$ or not is made randomly when $e$ is examined during the propagation process, which occurs *at most once* during the whole process. Under this interpretation, we may consider that $X_t$ is a random variable binomially distributed $\mathrm{Bin}(n - t - A_{t-1}, 1 - (1-p)^t)$.

Here we note that $X_t = \mathrm{Bin}(n - t - A_{t-1}, 1 - (1-p)^t)$ is dominated by a much simpler random variable $\widetilde{X}_t = \mathrm{Bin}(nt, p)$. Using $\widetilde{X}_t$, we can bound by

$$\begin{aligned} \Pr\left[ \sum_{1 \le t \le t_0} X_t \ge t_0 \right] \ &\le \ \Pr\left[ \sum_{1 \le t \le t_0} \widetilde{X}_t \ge t_0 \right] \\ = \ \Pr\left[ \mathrm{Bin}\left( n \cdot \sum_{1 \le t \le t_0} t, \ p \right) \ge t_0 \right] \ &= \ \Pr\left[ \mathrm{Bin}\left( \frac{nt_0(t_0+1)}{2}, \ p \right) \ge t_0 \right] \end{aligned} \tag{1}$$

Let $\mu_0$ denote the expectation of the last binomial distribution. Then since we assume that $c = 1.99$, we have

$$\mu_0 \ = \ \frac{nt_0(t_0+1)}{2} \cdot p \ = \ c(2 \log n - 1) \ < \ 1.98 \log n.$$

Thus, $t_0 = 2\log n + 1$ is much larger than the expectation. Then by using Chernoff bound, the last probability of (1) can be bounded by $o(1)$. $\square$

## Acknowledgement

The author would like to thank Dr. Amin Coja-Oghlan, Dr. Mikael Onsjö, and Dr. Masaki Yamamoto for their collaborations on related topics, which lead to this work.

## References

[AK97] N. Alon and N. Kahale, A spectral technique for coloring random 3-colorable graphs, *SIAM J. Comput.* 26(6), 1733–1748, 1997.

[BO09] R. Berke and M. Onsjö, Propagation connectivity of random hyptergraphs, in *Proc. 5th Symposium on Stochastic Algorithms, Foundations and Applications* (SAGA'09), Lecture Notes in Computer Science 5792, 117–126, 2009.

[Coj06] A. Coja-Oghlan, A spectral heuristic for bisecting random graphs, *Random Struct. and Algorithms* 29(3), 351–398, 2006.

[CGLS03] A. Coja-Oghlan, A. Goerdt, A. Lanka, and F. Schädlich, Techniques from combinatorial approximation algorithms yield efficient algorithms for random $2k$-SAT, *Theoret. Comput. Sci.* 329, 1–45, 2004.

[CMV07] A. Coja-Oghlan, E. Mossel, and D. Vilenchnik, A spectral approach to analyzing belief propagation for 3-coloring, *CoRR* abs/0712.0171, 2007.

[COW10] A. Coja-Oghlan, M. Onjsö, and O. Watanabe, Propagation connectivity of random hypergraphs, in *Proc. 14th Intl. Workshop on Randomization and Computation* (RANDOM'10), Lecture Notes in Computer Science 6302, 490–503, 2010.

[FO04] U. Feige and E. Ofek, Spectral techniques applied to sparse random graphs, *Random Structures and Algorithms* 27, 251–275, 2005.

[FKS81] J. Friedman, J. Kahn, and E. Szemeredi, On the second eigenvalue in random regular graphs, in *Proc 21st Annu ACM Sympos. Theory of Computing* (STOC'89), 587–598, 1989.

[KV05] S.A. Khot and N.K. Vishnoi, The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into L1, in *Proc. of the 46th IEEE Sympos. on Foundations of Comp. Sci.* (FOCS'05), 53–62, 2005.

[OW06] M. Onsjö and O. Watanabe, A simple message passing algorithm for graph partition problem, in *Proc. 17th Int'l Sympos. on Algorithms and Computation* (ISAAC'06), LNCS 4288, 507–516, 2006.

[WY10] O. Watanabe and M. Yamamoto, Average-case analysis for the MAX-2SAT problem, Theoret. Comput. Sci., 411(16-18): 1685–1697, 2010.

[WY06] O. Watanabe and M. Yamamoto, Belief propagation and spectral methods, Report C-248, Dept. of Math. and Comp. Sci., Tokyo Inst. of Tech., Nov. 2007.

## Appendix A: Proof of Lemmas

Here we prove Lemma 3 and Lemma 4. Symbols are used in the same way as in the body without explanation.

We begin with explaining how to apply Lemma 1 to matrices $B$ and $C$. For this we need to consider the original set $E$ of equations before the preprocessing. In order to distinguish symbols from the body, we use here $\widehat{A} = (\widehat{a}_{ij})$ to denote the matrix $A_E$ defined for the original $E$. Similarly, let $\widehat{B} = (\widehat{b}_{ij})$ and $\widehat{C} = (\widehat{c}_{ij})$ denote matrices that have 1 at each $ij$-entry such that $\widehat{a}_{ij} = 1$ and $\widehat{a}_{ij} = -1$ in $\widehat{A}$ respectively. Then we have $\Pr(\widehat{b}_{ij} = 1) = p(1-q)$ and $\Pr(\widehat{c}_{ij} = 1) = pq$. Furthermore, $(\widehat{b}_{ij})_{i<j}$ (resp., $(\widehat{c}_{ij})_{i<j}$) are mutually independent. Thus, we can apply Lemma 1 to them.

Here we remark some technical points for proving Lemma 1. The parameter $\widetilde{p}$ is not used in the original lemma in [CGLS03], but it is easy to modify their proof to show the lemma stated in this paper. Also from their proof, we can see that item (3) of the lemma can be proved even for $D$; that is, removing entries with too many 1's is only needed for proving the bound of item (2).

Now we consider our matrices $\widehat{B}$ and $\widehat{C}$. By the preprocessing of **AlgS_2LIN** all $i$th entries of $\widehat{B}$ (resp., $\widehat{C}$) such that $\sum_j \widehat{b}_{ij} \geq c_{\rm alg}p$ (resp., $\sum_j \widehat{c}_{ij} \geq c_{\rm alg}p$) are removed from $\widehat{B}$ (resp., $\widehat{C}$), which is enough to show the corresponding bounds of item (2) of Lemma 1. Thus, we can apply Lemma 1 to show the following bounds.

**Lemma A.1** There exists some constant $c_1 > 0$ with which the following holds w.h.p.
(1) For any unit vector $\boldsymbol{\xi} \perp \mathbf{1}$, we have $\|B\boldsymbol{\xi}\| \leq c_1\sqrt{np}$ and $\|C\boldsymbol{\xi}\| \leq c_1\sqrt{np}$.
(2) $\|B\overline{\mathbf{1}} - np(1-q)\overline{\mathbf{1}}\| \leq c_1\sqrt{np}$ and $\|C\overline{\mathbf{1}} - npq\overline{\mathbf{1}}\| \leq c_1\sqrt{np}$.

Now we prove Lemma 3 with the constant $c_1$ above.

**Lemma 3** The following holds w.h.p.
(1) For any unit vector $\boldsymbol{\eta} \perp \mathbf{1}$ and for any unit vector $\boldsymbol{\xi}$, we have $|\langle A\boldsymbol{\eta}, \boldsymbol{\xi}\rangle| \leq 2c_1\sqrt{np}$.
(2) For any $\lambda_i$, $2 \leq i \leq n$, we have $|\lambda_i| \leq 3c_1\sqrt{np}$.
(3) $\|A\overline{\mathbf{1}} - np'\overline{\mathbf{1}}\| \leq 2c_1\sqrt{np}$.

**Proof.** (1) Note first the following holds for any real symmetric matrix $D$.

$$\|D\boldsymbol{\zeta}\| \leq \alpha \iff |\langle D\boldsymbol{\zeta}, \boldsymbol{\nu}\rangle| \leq \alpha \text{ for all unit vector } \boldsymbol{\nu}.$$

Hence, for any unit vector $\boldsymbol{\eta} \perp \mathbf{1}$ and for any unit vector $\boldsymbol{\xi}$, we have

$$\begin{aligned} |\langle A\boldsymbol{\eta}, \boldsymbol{\xi}\rangle| &= |\langle B\boldsymbol{\eta}, \boldsymbol{\xi}\rangle - \langle C\boldsymbol{\eta}, \boldsymbol{\xi}\rangle| \\ &\leq |\langle B\boldsymbol{\eta}, \boldsymbol{\xi}\rangle| + |\langle C\boldsymbol{\eta}, \boldsymbol{\xi}\rangle| \leq 2c_1\sqrt{np} \end{aligned}$$

(2) Let $S_0$ be the set of vectors perpendicular to $\mathbf{1}$. Then since $\dim(S_0) = n-1$, by Theorem of Courant-Fischer, we have

$$\lambda_2 \;\le\; \max_{\boldsymbol{\eta}\in S_0}\langle A\boldsymbol{\eta},\boldsymbol{\eta}\rangle \;\le\; 2c_1\sqrt{np}.$$

On the other hand, to bound $\lambda_n$, we express the $n$th eigenvector as $\boldsymbol{\xi}_n = \alpha\overline{\mathbf{1}} + \beta\boldsymbol{\eta}$, where $\alpha,\beta \ge 0$, $\alpha^2+\beta^2=1$, and $\boldsymbol{\eta}$ is a unit vector perpendicular to $\overline{\mathbf{1}}$. Then we have

$$\begin{aligned}\lambda_n &= \langle A\boldsymbol{\xi}_n,\boldsymbol{\xi}_n\rangle \;=\; \langle A(\alpha\overline{\mathbf{1}}+\beta\boldsymbol{\eta}),\alpha\overline{\mathbf{1}}+\beta\boldsymbol{\eta}\rangle\\ &= \alpha^2\langle A\overline{\mathbf{1}},\overline{\mathbf{1}}\rangle + 2\alpha\beta\langle A\boldsymbol{\eta},\overline{\mathbf{1}}\rangle + \beta^2\langle A\boldsymbol{\eta},\boldsymbol{\eta}\rangle\\ &\ge (2\alpha\beta+\beta^2)(-2c_1\sqrt{np}) \;\ge\; -3c_1\sqrt{np}.\end{aligned}$$

Here we used the facts that $\langle A\boldsymbol{x},\boldsymbol{y}\rangle = \langle \boldsymbol{x}, A\boldsymbol{y}\rangle$ due to the symmetry of $A$, and that $\langle A\overline{\mathbf{1}},\overline{\mathbf{1}}\rangle \ge 0$ w.h.p.

(3) The claim is immediate from the following.

$$\begin{aligned}\|A\overline{\mathbf{1}} - np'\overline{\mathbf{1}}\| &= \|A\overline{\mathbf{1}} - np(1-2q)\overline{\mathbf{1}}\| \;=\; \|(B-C)\overline{\mathbf{1}} - (np(1-q)-npq)\overline{\mathbf{1}}\|\\ &= \|B\overline{\mathbf{1}} - np(1-q)\overline{\mathbf{1}}\| + \|C\overline{\mathbf{1}} - npq\overline{\mathbf{1}}\| \;\le\; 2c_1\sqrt{np}.\end{aligned}$$

□

Next we prove Lemma 4.

**Lemma 4** Assume that (C2) holds for $c$ and $\delta$. Then w.h.p. we have

$$\langle\overline{\mathbf{1}},\boldsymbol{\xi}_1\rangle \;\ge\; \sqrt{1-\frac{16c_1^2}{np'}} \;=\; \sqrt{1-\frac{16c_1^2}{c\delta}}$$

**Proof.** Let $\alpha = \langle\overline{\mathbf{1}},\boldsymbol{\xi}_1\rangle$. Then we have $\overline{\mathbf{1}} = \alpha\boldsymbol{\xi}_1 + \beta\boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is a unit vector perpendicular to $\boldsymbol{\xi}_1$. Thus,

$$\langle\overline{\mathbf{1}},\boldsymbol{\xi}_1\rangle \;=\; \alpha \;=\; \sqrt{1-\beta^2}.$$

On the other hand, letting $\overline{\mathbf{1}} = \alpha_1\boldsymbol{\xi}_1 + \cdots + \alpha_n\boldsymbol{\xi}_n$, we have $\alpha = \alpha_1$ and $\beta^2 = \sum_{i=2}^n \alpha_i^2$. Thus, the lemma is proved by bounding $\sum_{i=2}^n \alpha_i^2$ as follows.

$$\begin{aligned}4c_1^2np' &\ge \left\|A\overline{\mathbf{1}} - np'\overline{\mathbf{1}}\right\|^2 \;=\; \left\|A\left(\sum_{i=1}^n\alpha_i\boldsymbol{\xi}_i\right) - np'\left(\sum_{i=1}^n\alpha_i\boldsymbol{\xi}_i\right)\right\|^2\\ &= \left\|\left(\sum_{i=1}^n\alpha_i\lambda_i\boldsymbol{\xi}_i\right) - np'\left(\sum_{i=1}^n\alpha_i\boldsymbol{\xi}_i\right)\right\|^2 \;=\; \sum_{i=1}^n\left((\lambda_i-np')\alpha_i\boldsymbol{\xi}_i\right)^2\\ &\ge \sum_{i=2}^n(np'-\lambda_i)^2\alpha_i^2 \;\ge\; (np'-3c_1\sqrt{np})^2\sum_{i=2}^n\alpha_i^2 \;\ge\; \left(\frac{np'}{2}\right)^2\sum_{i=2}^n\alpha_i^2.\end{aligned}$$

The last bound is from the condition (C2). □