Approximation Algorithms for the Maximum Satisfiability Problem

Takao Asano* Takao Ono** Tomio Hirata**
* Department of Information and System Engineering, Chuo University
** School of Engineering, Nagoya University
E-mail: asano@ise.chuo-u.ac.jp {takao,hirata}@hirata.nuee.nagoya-u.ac.jp

The maximum satisfiability problem (MAX SAT) is : given a set of clauses with weights, find a truth assignment that maximizes the sum of the weights of the satisfied clauses. In this paper, we present approximation algorithms for MAX SAT, including a 0.76544-approximation algorithm. The previous best approximation algorithm for MAX SAT was proposed by Goemans-Williamson and has a performance guarantee of 0.7584. Our algorithms are based on semidefinite programming and the 0.75-approximation algorithms of Yannakakis and Goemans-Williamson.

MAX SATに対する近似アルゴリズム

浅野 孝夫* 小野 孝男** 平田 富夫**
* 中央大学 理工学部 情報工学科 ** 名古屋大学 工学部

MAX SAT(充足性最大化問題) とは, 節の集合と各節の重みが与えられたとき, 充足する節の重みの総和を最大にするような真偽割り当てを求める問題である. MAX SAT に対する従来の最善の近似アルゴリズムは, Goemans-Williamson による 0.7584-近似アルゴリズムであるが, 本論文では, 0.76544-近似アルゴリズムを含むいくつかの近似アルゴリズムを与える。これらのアルゴリズムは, 半正定値計画法および Yannakakis と Goemans-Williamson の 0.75-近似アルゴリズムに基づいている。

1 Introduction

We consider the maximum satisfiability problem (MAX SAT): given a set of clauses with weights, find a truth assignment that maximizes the sum of the weights of the satisfied clauses. MAX 2SAT, the restricted version of MAX SAT where each clause has at most 2 literals, is well known to be NP-complete even if the weights of the clauses are identical, and thus MAX SAT is also NP-complete. Thus, many researchers have proposed approximation algorithms and the best approximation algorithms for MAX SAT had been 0.75-approximation algorithms proposed by Goemans-Williamson [3] and Yannakakis [8]. On the other hand, Goemans-Williamson recently obtained a 0.878-approximation algorithm for MAX 2SAT based on semidefinite programming, a new and very useful technique in the approximation algorithms and noted that it leads to a 0.755-approximation algorithm for MAX SAT [2]. Then they obtained another improvement and the 0.7584-approximation algorithm is the best known algorithm [4].

In this paper, we first present a 0.75899-approximation algorithm for MAX SAT based on semidefinite programming and 0.75-approximation algorithm of Goemans-Williamson [3]. Although this algorithm may be quite similar to the above 0.7584-approximation algorithm, in that both algorithms are based on semidefinite programming and the 0.75-approximation algorithm of Goemans-Williamson, our algorithm will not explicitly transform a MAX SAT instance into a MAX 2SAT instance. We next give a 0.76544-approximation algorithm by combining the above 0.75899-approximation algorithm with the 0.75-approximation algorithm of Yannakakis based on the probabilistic method. Yannakakis's algorithm divides the variables of a given instance into three groups P', $(P-P')\cup Q$ and Z and sets the variables in P', $(P-P')\cup Q$ and Z to be true with probabilities 3/4, 5/9 and 1/2, respectively. In our algorithm, we set the variables in P', $(P-P')\cup Q$ and Z to be true with probabilities 3/4, 0.568729 and 1/2, respectively. Combined with the 0.75899 approximation algorithm, this leads to the approximation algorithm with performance guarantee 0.76544.

2 Preliminaries

An instance of the maximum satisfiability problem (MAX SAT) is defined by a collection of boolean clauses $\mathcal{C}$ where each clause $C_j \in \mathcal{C}$ is a disjunction of literals and has a nonnegative weight w_j (a *literal* is either a variable x_i or its negation $\bar{x}_i$). Let $X = \{x_1, \dots, x_n\}$ be the set of variables in the clauses in $\mathcal{C}$. We assume that no variable appears more than once in a clause in $\mathcal{C}$, that is, we do not allow a clause like $x_1 \vee \bar{x}_1 \vee x_2$. For each variable $x_i \in X$, we consider $x_i = 1$ ($x_i = 0$, resp.) if x_i is true (false, resp.). Then, $\bar{x}_i = 1 - x_i$ and a clause $C_j \in \mathcal{C}$ can be considered to be a function of $x = (x_1, \dots, x_n)$ as follows:

$$C_j = C_j(x) = 1 - \prod_{x_i \in X_j^+} (1 - x_i) \prod_{x_i \in X_j^-} x_i \qquad (1)$$

where X_j^+ (X_j^-, resp.) denotes the set of variables appearing unnegated (negated, resp.) in C_j. Thus, $C_j = C_j(x) = 0$ or 1 for any *truth assignment* $x \in \{0,1\}^n$ (i.e., an assignment of 0 or 1 to each $x_i \in X$), and C_j is *satisified* (*not satisfied*, resp.) if $C_j(x) = 1$ ($C_j(x) = 0$, resp.). The *value* of an assignment x is defined to be

$$F(x) = \sum_{C_j \in \mathcal{C}} w_j C_j(x). \qquad (2)$$

That is, the value of x is the sum of the weights of the clauses in $\mathcal{C}$ satisfied by x. Thus, MAX SAT is to find a truth assignment of maximum value.

Let A be an algorithm for MAX SAT and let $w^A(\mathcal{C})$ be the value of a truth assignment $x^A(\mathcal{C})$ produced by A for an instance $\mathcal{C}$. If $w^A(\mathcal{C})$ is at least α times the value $w^*(\mathcal{C})$ of an optimal truth assignment $x^*(\mathcal{C})$ for any instance $\mathcal{C}$, then A is called an approximation algorithm with *performance guarantee* α. A polynomial time algorithm A with performance guarantee α is called an *α-approximation algorithm.*

The 0.75-approximation algorithm of Yannakakis is based on the probabilistic method. Let x^p be a *random* truth assignment obtained by setting independently each variable $x_i \in X$ to be true with probability p_i (i.e., $0 \le x_i^p = p_i \le 1$). Then the probability of the clause $C_j \in \mathcal{C}$ satisified by the assignment x^p is

$$C_j(x^p) = 1 - \prod_{x_i \in X_j^+} (1 - p_i) \prod_{x_i \in X_j^-} p_i\,. \qquad (3)$$

Thus, the expected value of the random truth assignment x^p is

$$F(x^p) = \sum_{C_j \in \mathcal{C}} w_j C_j(x^p). \qquad (4)$$

The probabilistic method assures that there is a truth assignment $x^q \in \{0,1\}^n$ such that its value is at least $F(x^p)$. Such a truth assignment x^q can be obtained by the method of conditional probability ([3],[8]).

3 0.75899-Approximation Algorithm

In this section we present an approximation algorithm A for MAX SAT. The algorithm is based on semidefinite programming used first by Goemans and Williamson for MAX 2SAT [2] and if it is combined with the 0.75-approximation algorithm of Goemans-Williamson then it achieves a performance guarantee of 0.75899. The algorithm A consists of the following four steps.

1. Translate a maximization problem of $\{0,1\}$-variables into a maximization problem of $\{-1,1\}$-variables, that is, we regard MAX SAT as a maximization problem of a polynomial of variables in $\{-1,1\}$.
2. Obtain another maximization problem of a polynomial of low degree.
3. By relaxing the new maximization problem, formulate it as a semidefinite programming problem and solve it.
4. Construct a truth assignment of the original MAX SAT from a solution of the semidefinite programming problem.

We explain each of the above steps in the following subsections.

3.1 From a Maximization of $\{0,1\}$-Variables to a Maximization of $\{-1,1\}$-Variables

We translate an instance of MAX SAT of n $\{0,1\}$-variables into a polynomial of n $\{-1,1\}$-variables in the following way.

We introduce n variables $x'_1,\ldots,x'_n$ and consider

$$x_i \equiv \frac{1+x'_i}{2}. \qquad (5)$$

Thus, $\bar{x}_i = 1-x_i = \frac{1-x'_i}{2}$ and $x_i = 1$ ($x_i = 0$, resp.) if and only if $x'_i = 1$ ($x'_i = -1$, resp.). Let $x' = (x'_1,\ldots,x'_n)$. By this replacement, clause C_j in (1) becomes

$$C_j(x') = 1 - \prod_{x_i\in X_j^+} \frac{1-x'_i}{2} \prod_{x_i\in X_j^-} \frac{1+x'_i}{2}. \qquad (6)$$

Thus, finding a truth assignment $x \in \{0,1\}^n$ that maximizes $F(x)$, the sum of the weights of the satisfied clauses in $\mathcal{C}$, is equivalent to finding an assignment $x' \in \{-1,1\}^n$ (assignment of -1 or 1 to each x'_i) that maximizes

$$F(x') = \sum_{C_j\in\mathcal{C}} w_j C_j(x'). \qquad (7)$$

$F(x')$ is called the value of x'. Thus we can regard MAX SAT as the following problem:

(P): Maximize $F(x')$ subject to $x' \in \{-1,1\}^n$.

3.2 Maximization Problem of a Polynomial of Lower Degee

Instead of maximizing $F(x')$, we will consider another maximization problem. Let $\mathcal{C}_k$ be the set of clauses in $\mathcal{C}$ with k literals and let $C_j = x_1 \vee x_2 \vee \cdots \vee x_k \in \mathcal{C}_k$. Then, by (6), $C_j = C_j(x')$ is

$$\begin{aligned}&\tfrac{1}{2^k}\textstyle\sum_{i=1}^k(1+x'_i) + \tfrac{1}{2^k}\sum_{1\le i_1<i_2\le k}(1-x'_{i_1}x'_{i_2})\\ &\quad+\tfrac{1}{2^k}\textstyle\sum_{1\le i_1<i_2<i_3\le k}(1+x'_{i_1}x'_{i_2}x'_{i_3})\\ &\quad+\cdots+\tfrac{1}{2^k}(1-(-1)^k x'_1x'_2\cdots x'_k).\end{aligned}$$

We divide C_j into two parts $c_j^{(1)}$ and $c_j^{(2)}$: $c_j^{(1)}$ is the sum of the terms in C_j of forms $1+x'_i$ and $1-x'_{i_1}x'_{i_2}$ and $c_j^{(2)}$ is the rest, i.e.,$c_j^{(1)} = c_j^{(1)}(x')$ is

$$\frac{1}{2^k}\sum_{i=1}^{k}(1+x'_i) + \frac{1}{2^k}\sum_{1\le i_1<i_2\le k}(1-x'_{i_1}x'_{i_2}) \qquad (8)$$

and $c_j^{(2)} = c_j^{(2)}(x') = C_j(x') - c_j^{(1)}(x')$. Similarly, for any $C_{j'} \in \mathcal{C}$, we can define $c_{j'}^{(1)}$ and $c_{j'}^{(2)}$ in the same way even if some literal in $C_{j'}$ appears as a negation $\bar{x}_i$ (in this case we have only to replace x'_i with $-x'_i$). Note $c_j^{(1)}(x') = C_j(x')$ for any $C_j \in \mathcal{C}_k$ with $k \le 2$, and that $1 \pm x'_i \ge 0$ and $1 \pm x'_{i_1}x'_{i_2} \ge 0$ since $x' \in \{-1,1\}^n$.

Let

$$\alpha_k = \begin{cases}\frac{4k}{(k+1)^2} & (k \text{ is odd})\\ \frac{4k}{(k+1)^2-1} & (k \text{ is even})\end{cases} \qquad (9)$$

and define

$$G(x') = \sum_{k\ge1}\sum_{C_j\in\mathcal{C}_k} \frac{2^{k+1}}{4k}\alpha_k w_j c_j^{(1)}(x'). \qquad (10)$$

Thus, $G(x')$ is a polynomial with nonconstant terms of degree at most 2 and we use a solution of the following problem as an approximate solution of (P).

(Q): Maximize $G(x')$ subject to $x' \in \{-1,1\}^n$.

Now we introduce new variables $y = (y_0, y_1, \ldots, y_n)$ and consider

$$x'_i \equiv y_0 y_i \text{ with } |y_0| = |y_i| = 1 \tag{11}$$

for later use in semidefinite programming. Then $1+x'_i$ ($1-x'_i$, resp.) becomes $1+y_0y_i$ ($1-y_0y_i$, resp.) and $1+x'_{i_1}x'_{i_2}$ ($1-x'_{i_1}x'_{i_2}$, resp.) becomes $1+y_{i_1}y_{i_2}$ ($1-y_{i_1}y_{i_2}$, resp.). Now they are of the same form and $c_j^{(1)}(y) = c_j^{(1)}(x')$ is a linear combination of the terms $1+y_{i_1}y_{i_2}$, $1-y_{i_1}y_{i_2}$. Thus, $G(y) = G(x')$ can be expressed by

$$\sum_{0\le i_1<i_2\le n} a^+_{i_1i_2}(1+y_{i_1}y_{i_2}) + \sum_{0\le i_1<i_2\le n} a^-_{i_1i_2}(1-y_{i_1}y_{i_2}) \tag{12}$$

with some nonnegative numbers $a^+_{i_1i_2}$, $a^-_{i_1i_2}$. Note that $G(y)$ is a polynomial with nonconstant terms of degree 2. Thus, (Q) is reduced to the following problem (Q') and we use its solution as an approximate solution of (P) based on (11).

(Q'): Maximize $G(y)$ subject to $y \in \{-1,1\}^{n+1}$.

3.3 Relaxation and Semidefinite Programming Problem

Since (Q') is difficult to solve, we consider a relaxation of (Q') as described in [2]. Let v_i be an $(n+1)$-dimensional vector with norm $\|v_i\| = 1$ corresponding to y_i with $|y_i| = 1$. We replace $y_{i_1}y_{i_2}$ with an inner product $v_{i_1}\cdot v_{i_2}$. With this relaxation, (Q') is reduced to the following problem (R).

(R): Maximize $G(v) = \sum_{0\le i_1<i_2\le n} a^+_{i_1i_2}(1+v_{i_1}\cdot v_{i_2}) + \sum_{0\le i_1<i_2\le n} a^-_{i_1i_2}(1 - v_{i_1}\cdot v_{i_2})$ subject to $\|v_i\| = 1$ for all $i = 0, \ldots, n$.

Now let $y_{i_1i_2} = v_{i_1}\cdot v_{i_2}$. Then, a matrix $Y = (y_{i_1i_2})$ is symmetric and positive semidefinite. Thus, the following problem is equivalent to (R).

(R'): Maximize $G(Y) = \sum_{0\le i_1<i_2\le n} a^+_{i_1i_2}(1 + y_{i_1i_2}) + \sum_{0\le i_1<i_2\le n} a^-_{i_1i_2}(1 - y_{i_1i_2})$ subject to the conditions that the matrix $Y = (y_{i_1i_2})$ is symmetric and positive semidefinite, and that $y_{ii} = 1$ for all $i = 0, \ldots, n$.

Since (R') is a semidefinite programming problem as in [2], we can find an approximate solution $\bar{Y} = (\bar{y}_{i_1i_2})$ within a constant error ϵ in polynomial time. An approximate solution $\bar{v} = (\bar{v}_0, \bar{v}_1, \ldots, \bar{v}_n)$ of (R) can be obtained by Cholesky decomposition of $\bar{Y} = (\bar{y}_{i_1i_2})$. At this point we have an approximate solution of (R) in polynomial time.

3.4 Finding an Approximate Solution of MAX SAT

As described in [2], we find a randomized approximate solution $\bar{y} = (\bar{y}_0, \bar{y}_1, \ldots, \bar{y}_n)$ of (Q') and the corresonding solution of (Q) and (P) from an approximate solution $\bar{v} = (\bar{v}_0, \bar{v}_1, \ldots, \bar{v}_n)$ of (R). Taking a random vector r, let $\bar{y}_i = +1$ if $\bar{v}_i \cdot r > 0$, or $\bar{y}_i = -1$ otherwise. $\bar{x}' = (\bar{x}'_1, \ldots, \bar{x}'_n)$ is also obtained by $\bar{x}'_i = \bar{y}_0\bar{y}_i$ based on (11).

We show in Section 4 that the expected value $F(\bar{x}')$ of $\bar{x}'$ is at least 0.75899 times the value $F(x'^*)$ of the optimal solution x'^* of (P) under some conditions. We can derandomize this method in the same way as in [2],[4],[6] and obtain an approximate solution $y^A = (y_0^A, y_1^A, \ldots, y_n^A)$ of (Q') and the corresonding approximate solution $x'^A = (x_1'^A, \ldots, x_n'^A)$ of (P) by $x_i'^A = y_0^A y_i^A$. Since $x_i = \frac{1+x'_i}{2}$ by (5) (i.e., $x_i = 1$ (x_i =true) if and only if $x'_i = 1$ and $x_i = 0$ (x_i =false) if and only if $x'_i = -1$), we have an truth assignment $x^A = (x_1^A, \ldots, x_n^A)$ which is an approximate solution of MAX SAT.

4 Analysis of the Performance Guarantee

Based on the argument in the previous section, solutions x of MAX SAT, x' of (P) and y of (Q') can be obtained from one another based on (5) and (11). Thus, we can assume, from now on, that any one solution of x, x' and y implies any other two solutions in hand even if we do not say explicitly. In this section we analyze the performance guarantee of the above algorithm. Let $x^A = (x_1, \ldots, x_n)$ be an approximate solution of MAX SAT obtained by the algorithm A described in the previous section. Let x^G be an assignment obtained by using 0.75-approximation algorithm of Goemans-Williamson. Let B be an algorithm choosing the better solution x^B between x^A and x^G. Then, roughly speaking, the performance guarantee of x^B will be 0.75899.

The following lemmas play critical roles throughout the paper.

Lemma 1 *Let C_j be any clause in $\mathcal{C}$. Then, for any $x' = (x'_1, \ldots, x'_n) \in \{-1,1\}^n$, $C_j(x') \neq 0$ (C_j is true) if and only if $c_j^{(1)}(x') \neq 0$. Furthermore, for any $x' \in \{-1,1\}^n$ and any $C_j \in \mathcal{C}_k$, if k is odd*

$$\frac{4k}{2^{k+1}} C_j(x') \leq c_j^{(1)}(x') \leq \frac{(k+1)^2}{2^{k+1}} C_j(x')$$

and if k is even

$$\frac{4k}{2^{k+1}} C_j(x') \leq c_j^{(1)}(x') \leq \frac{(k+1)^2-1}{2^{k+1}} C_j(x').$$

Proof. By symmetry we can assume $C_j = x_1 \vee x_2 \vee \cdots \vee x_k \in \mathcal{C}$. By equations (1), (5) and (6), if $C_j(x') \neq 0$ then $C_j(x') = 1$ since $C_j = C_j(x')$ is boolean and $x'_i = 1$ for some x'_i and thus $c_j^{(1)}(x') \neq 0$ by (8). On the other hand, if $c_j^{(1)}(x') \neq 0$, then $x'_i = 1$ for some x'_i by (8) and thus $C_j(x') = 1$ by (6). Thus, we have $C_j(x') \neq 0$ if and only if $c_j^{(1)}(x') \neq 0$. Let ℓ be the number of x'_i's with $x'_i = 1$ ($i = 1, 2, ..., k$). If $c_j^{(1)}(x') \neq 0$ then $1 \leq \ell \leq k$ and $c_j^{(1)}(x') = \frac{\ell+\ell(k-\ell)}{2^{k-1}}$ by (8). Since

$$\ell + \ell(k-\ell) = -(\ell - \frac{k+1}{2})^2 + \frac{(k+1)^2}{4},$$

we have the lemma. □

Lemma 2 *Let x'^A be a solution of (Q) obtained by the alogrithm A in Section 3 for an instance $\mathcal{C}$. Let x'^* be any solution in $\{-1,1\}^n$ that maximizes $F(x')$ and let $W_k^* = \sum_{C_j \in \mathcal{C}_k} w_j C_j(x'^*)$. Then the value $w^A(\mathcal{C}) = F(x'^A)$ of x'^A satisfies*

$$w^A(\mathcal{C}) = F(x'^A) \geq G(x'^A) \geq \alpha \sum_{k \geq 1} \alpha_k W_k^*,$$

where $\alpha = 0.87856$ and α_k is defined in (9), i.e.,

$$\alpha_k = \begin{cases} \frac{4k}{(k+1)^2} & (k \text{ is odd }) \\ \frac{4k}{(k+1)^2-1} & (k \text{ is even}). \end{cases}$$

Proof. For any $C_j \in \mathcal{C}_k$ and any $x' \in \{-1,1\}^n$, we have

$$\frac{2^{k+1}}{4k} \alpha_k c_j^{(1)}(x') \leq C_j(x') \qquad (13)$$

since $c_j^{(1)}(x') \leq \frac{(k+1)^2}{2^{k+1}} C_j(x')$ (k is odd) and $c_j^{(1)}(x') \leq \frac{(k+1)^2-1}{2^{k+1}} C_j(x')$ (k is even) by Lemma 1 and by the definition of α_k, and thus, we have $F(x'^A) \geq G(x'^A)$.

Let x'^*_G be any solution in $\{-1,1\}^n$ that maximizes $G(x')$. Then, by the same analysis as in [2],[4] for $\alpha = 0.87856$, we have $G(x'^A) \geq \alpha G(x'^*_G)$ and $G(x'^*_G) \geq G(x'^*)$. On the other hand, we have $c_j^{(1)}(x') \geq \frac{4k}{2^{k+1}} C_j(x')$ for any $C_j \in \mathcal{C}_k$ and any $x' \in \{-1,1\}^n$ by Lemma 1 and thus

$$\frac{2^{k+1}}{4k} \alpha_k c_j^{(1)}(x') \geq \alpha_k C_j(x'). \qquad (14)$$

This implies

$$G(x'^*) = \sum_{k \geq 1} \sum_{C_j \in \mathcal{C}_k} \frac{2^{k+1}}{4k} \alpha_k w_j c_j^{(1)}(x'^*)$$
$$\geq \sum_{k \geq 1} (\alpha_k \sum_{C_j \in \mathcal{C}_k} w_j C_j(x'^*)) = \sum_{k \geq 1} \alpha_k W_k^*. \qquad (15)$$

Thus, we have the lemma by

$$F(x'^A) \geq G(x'^A) \geq \alpha G(x'^*_G) \geq \alpha G(x'^*)$$

obtained above. □

The 0.75-approximation algorithm of Goemans-Williamson is obtained by choosing the better solution x^G between their original linear programming relaxation solution x^L and a solution x^J by Johnson's algorithm [5].

Let x'^* be any solution in $\{-1,1\}^n$ that maximizes $F(x')$ in (P) and let x^* be obtained from x'^* by setting $x_i^* = \frac{1+x_i'^*}{2}$. For $k \geq 1$, let $W_k = \sum_{C_j \in \mathcal{C}_k} w_j$ and $W_k^* = \sum_{C_j \in \mathcal{C}_k} w_j C_j(x'^*)$. Then, the value w^J of x^J satisfies $w^J \geq \sum_{k \geq 1}(1 - \frac{1}{2^k}) W_k$. On the other hand, the value w^L of x^L satisfies $w^L \geq \sum_{k \geq 1}(1 - (1-\frac{1}{k})^k) W_k^*$. Strictly speaking, $w^L \geq \sum_{k \geq 1}(1 - (1 - \frac{1}{k})^k) W_k^{*''}$ for another optimal solution $x'^{*''}$ in $\{-1,1\}^n$ that maximizes $F(x')$ and $W_k^{*''} = \sum_{C_j \in \mathcal{C}_k} w_j C_j(x'^{*''})$. We will later show that we can assume $W_k^{*''} = W_k^*$ for all $k \geq 1$. Thus, if we let

$$\beta_k = \frac{(1 - \frac{1}{2^k}) + 1 - (1 - \frac{1}{k})^k}{2}, \qquad (16)$$

then the value w^G of the better solution x^G satisfies

$$w^G \geq \sum_{k \geq 1} \beta_k W_k^*. \qquad (17)$$

Now we are ready to analyze the performance guarantee of the algorithm B choosing the better one x^B between the solution x^A obtained by the algorithm A and the Goemans-Williamson's solution x^G. Thus, $w^B = \max\{w^A, w^G\}$. Let $b = \frac{\alpha(2-\frac{1}{e})}{\frac{1}{2}-\frac{1}{e}+2\alpha} = 0.758990...$ where e is the number e (the base of the natural logarithm). If

$$W_1^* + W_2^* \geq \frac{1}{\alpha - b} \sum_{k \geq 3} (b - \alpha\alpha_k) W_k^*, \qquad (18)$$

then $w^A \geq \alpha \sum_{k\geq 1} \alpha_k W_k^* \geq b \sum_{k\geq 1} W_k^*$ by Lemma 2 and w^A (as well as w^B) has a required performance b (note that $\alpha_1 = \alpha_2 = 1$). Thus, we can assume

$$W_1^* + W_2^* < \frac{1}{\alpha - b} \sum_{k\geq 3} (b - \alpha\alpha_k) W_k^*. \qquad (19)$$

On the other hand, if

$$W_1^* + W_2^* \leq \frac{1}{b - 0.75} \sum_{k\geq 3} (\beta_k - b) W_k^*,$$

then $w^G \geq \sum_{k\geq 1} \beta_k W_k^* \geq b \sum_{k\geq 1} W_k^*$ by (16) and (17) (note that $\beta_1 = \beta_2 = 0.75$) and w^G has a required performance b. Thus, we can assume

$$W_1^* + W_2^* > \frac{1}{b - 0.75} \sum_{k\geq 3} (\beta_k - b) W_k^*.$$

However, for $b = 0.75899$, it is not difficult to see

$$\frac{\beta_k - b}{b - 0.75} \geq \frac{b - \alpha\alpha_k}{\alpha - b} \qquad (k \geq 3) \qquad (20)$$

and we have

$$W_1^* + W_2^* > \frac{1}{b - 0.75} \sum_{k\geq 3} (\beta_k - b) W_k^*$$

$$\geq \frac{1}{\alpha - b} \sum_{k\geq 3} (b - \alpha\alpha_k) W_k^* > W_1^* + W_2^*,$$

a contradiction.

Thus, we have the following theorem.

Theorem 1 *A* 0.75899-*approximation algorithm can be obtained based on the algorithm A in Section 3 and the* 0.75-*approximation algorithm of Goemans-Williamson.*

5 Formal Formulation

In this section, we formulate MAX SAT based on the the formulation by Goemans and Williamson [3],[4] as follows.

$$(S): \text{Maximize} \sum_{C_j \in \mathcal{C}} w_j z_j \quad \text{subject to:}$$

$$\sum_{x_i \in X_j^+} \frac{1 + y_{0i}}{2} + \sum_{x_i \in X_j^-} \frac{1 - y_{0i}}{2} \geq z_j \qquad \forall C_j \in \mathcal{C}$$

$$\frac{2^{k+1}}{4k} c_j^{(1)}(Y) \geq z_j \qquad \forall C_j \in \mathcal{C}_k$$

$$y_{ii} = 1 \qquad \forall 0 \leq i \leq n$$

$$0 \leq z_j \leq 1 \qquad \forall C_j \in \mathcal{C}$$

$$Y = (y_{i_1 i_2}) \text{ is symmetric positive semidefinite.} \qquad (21)$$

Recall that, by (11) and $y_{i_1 i_2} = v_{i_1} \cdot v_{i_2} = y_{i_1} y_{i_2}$, the matrix $Y = (y_{i_1 i_2})$ is symmetric and positive semidefinite and $c_j^{(1)}(Y) = c_j^{(1)}(y) = c_j^{(1)}(x')$. Let (Y, z^*) is an optimal solution to (S). By fixing z^* and changing Y in the optimal solution (Y, z^*) to (S), we also solve (R') under the same constraints in (S). Let (Y^*, z^*) be an optimal solution to (S) and (R').

To achieve the bound described in the previous section, we consider Algoorithm C consisting of the following three algorithms:

(1) set each variable x_i true independently with probability 0.5;

(2) set x_i true independently with probability $\frac{1+y_{0i}}{2}$ using the optimal solution (Y, z^*) to (S);

(3) take a random unit vector r and set x_i true if and only if $\text{sgn}(\bar{v}_i^* \cdot r) = \text{sgn}(\bar{v}_0^* \cdot r)$ using the optimal solution (Y^*, z^*) to (S) and (R') $(y_{i_1 i_2}^* = v_{i_1}^* \cdot v_{i_2}^*)$.

Suppose we use algorithm (i) with probability p_i, where $p_1 + p_2 + p_3 = 1$. From the arguments in [2],[4] and the previous section, the probability that a clause $C_j \in \mathcal{C}_k$ being satisfied by algorithm (3) is at least $\alpha\alpha_k c_j^{(1)}(Y) \geq \alpha\alpha_k z_j^*$ by Lemmas 1 and 2. Thus, the expected value W^C of the solution is at least

$$\sum_{k\geq 1} \Big(\sum_{C_j \in \mathcal{C}_k} w_j \left[(1 - \frac{1}{2^k}) p_1 + (\gamma_k p_2 + \alpha\alpha_k p_3) z_j^* \right] \Big), \qquad (22)$$

where $\gamma_k = 1 - (1 - \frac{1}{k})^k$. Let $W_k^* = \sum_{C_j \in \mathcal{C}_k} w_j z_j^*$. Then, by $z_j^* \leq 1$ in the constraints of (S), $W_k^* \leq \sum_{C_j \in \mathcal{C}_k} w_j$ and thus,

$$W^C \geq \sum_{k\geq 1} \left[(1 - \frac{1}{2^k}) p_1 + \gamma_k p_2 + \alpha\alpha_k p_3) \right] W_k^*. \qquad (23)$$

This equation also assures the arguments in the preceeding section. If we set $p_1 = p_2 = 0.4650334 = p$ and $p_3 = 0.0699332 = 1 - 2p$, then

$$W^C \geq \sum_{k\geq 1} (2\beta_k p + \alpha\alpha_k (1 - 2p)) W_k^*$$

$(2\beta_k = 1 - \frac{1}{2^k} + 1 - (1 - \frac{1}{k})^k = 1 - \frac{1}{2^k} + \gamma_k$ by (16)). Thus, we obtain Algorithm C is a 0.75899-approximation algorithm, which can be verified by checking

$$2\beta_k p + \alpha\alpha_k (1 - 2p) \geq 0.75899$$

for $k \leq 8$ and noticing $2\beta_k p + \alpha\alpha_k (1 - 2p)$ decreases as k increases, and that, for $k = \infty$, $\beta_k = 1 - \frac{1}{2e}$ and $\alpha_k = 0$ and $2\beta_k p + \alpha\alpha_k (1-2p) = 0.4650334(2 - \frac{1}{e}) \geq 0.75899$.

Thus Theorem 2 in the preceeding section is also obtained. For MAX kSAT where each clause has at most k literals, we can similarly obtain a 0.769 approximation algorithm for $k \leq 5$ by maximizing b in the preceeding section and appropriately fixing p_1, p_2 and p_3.

6 0.76544-Approximation Algorithm

We can improve the bound 0.75899 by using the 0.75-approximation algorithm of Yannakakis. It is based on the probabilistic method proposed by Johnson [5] and divides the variables X of a given instance $\mathcal{C}$ into three groups P', $(P-P')\cup Q$ and Z based on maximum network flows. Simultaneously, the set $\mathcal{C}$ of weighted clauses is transformed to an equivalent set of weighted clauses $\mathcal{E}\cup\mathcal{F}\cup\mathcal{G}\cup\mathcal{G}'\cup\mathcal{H}$ (two sets $\mathcal{D},\mathcal{D}'$ of weighted clauses over the same set of variables X are *equivalent* if every truth assignment of X, $\mathcal{D}$ and $\mathcal{D}'$ have the same value). For simlicity, we assume as follows (in fact we can assume without loss of generality):

$\mathcal{E} = \{\bar{x}_1, x_1\}$ with $\bar{x}_1, x_1$ of weight K_E,

$\mathcal{G} = \{x_1, x_2, x_3, \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3\}$ with $x_1, x_2, x_3 \in X \cup \bar{X} - \bar{P}$ of weight K_G and $\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$ of weight $2K_G$,

$\mathcal{G}' = \{x_{g_1}, x_{g_2}, x_{g_3}, x_{g_4}, \bar{x}_{g_1} \vee \bar{x}_{g_2} \vee \bar{x}_{g_3} \vee \bar{x}_{g_4}\}$ with $x_{g_k} \in X \cup \bar{X} - \bar{P}$ of weight $K_{G'}$ $(k = 1, 2, 3, 4)$ and $\bar{x}_{i_1} \vee \bar{x}_{i_2} \vee \bar{x}_{i_3} \vee \bar{x}_{i_4}$ of weight $3K_{G'}$,

$\mathcal{H} = \{x_{h_1}, x_{h_2}, \bar{x}_{h_3}, \bar{x}_{h_1} \vee \bar{x}_{h_2} \vee x_{h_3}, x_1, \bar{x}_1\}$ with $x_{h_k} \in X \cup \bar{X} - \bar{P}$ $(k = 1, 2, 3)$ of weight $2K_H$, $\bar{x}_{h_1} \vee \bar{x}_{h_2} \vee x_{h_3}$ of weight $2K_H$ and $x_1, \bar{x}_1$ of weight $-K_H$,

$\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \mathcal{F}_4 \cup \cdots$ with $\sum_{C_j \in \mathcal{F}_1} w_j = \sum_{C_j \in \mathcal{C}_1} w_j - 2K_E - 3K_G - 4K_{G'} - 4K_H$

$\sum_{C_j \in \mathcal{F}_2} w_j = \sum_{C_j \in \mathcal{C}_2} w_j$

$\sum_{C_j \in \mathcal{F}_3} w_j = \sum_{C_j \in \mathcal{C}_3} w_j - 2K_G - 2K_H$

$\sum_{C_j \in \mathcal{F}_4} w_j = \sum_{C_j \in \mathcal{C}_4} w_j - 3K_{G'}$

$\mathcal{F}_k = \mathcal{C}_k$ for all $k \geq 5$ (weight of a clause in this class is not changed).

Furthermore, we can assume

each literal in $\mathcal{F}_1$ is contained in P',

for each clause $x \vee y \in \mathcal{F}_2$, if one of x, y is contained in $\bar{P}'$ then the other is in P', and if one of x, y is contained in $\bar{Q} \cup (\bar{P} - \bar{P}')$ then the other is in $P \vee Q$,

for each clause $x \vee y \vee z \in \mathcal{F}_3$, all x, y, z are not contained in $\bar{P}'$, and if two of x, y, z are contained in $\bar{P}'$ then the remaining one is in $P \vee Q$,

for each clause $x \vee y \vee z \vee v \in \mathcal{F}_4$, all x, y, z, v are not contained in $\bar{P}'$.

Since the set $\mathcal{C}$ and the set $\mathcal{E}\cup\mathcal{F}\cup\mathcal{G}\cup\mathcal{G}'\cup\mathcal{H}$ described above are equivalent, from now on we assume $\mathcal{C} = \mathcal{E}\cup\mathcal{F}\cup\mathcal{G}\cup\mathcal{G}'\cup\mathcal{H}$.

Then Yannakakis algorithm sets the variables in P', $(P-P')\cup Q$ and Z to be true with probabilities 3/4, 5/9 and 1/2, respectively. In our algorithm we change to set the variables in $(P - P') \cup Q$ to be true with probability $p = \frac{-3+\sqrt{57}}{8} \approx 0.568729$. Let W^*_D be the value of an optimal truth assignment for $\mathcal{D}$, the sum of the weights of the satisifed clauses in $\mathcal{D}$ $(\mathcal{D} = \mathcal{E}, \mathcal{G}, \mathcal{G}', \mathcal{H})$. Then $W^*_E \leq K_E$, $W^*_G \leq 4K_G$, $W^*_{G'} \leq 6K_{G'}$ and $W^*_H \leq 5K_H$. Let $W'_k = \sum_{C_j \in \mathcal{F}_k} w_j$. Thus the probabilistic method assures that we can find a truth assignment $x^Y \in \{0,1\}^n$ of value w^Y which is at least

$$\begin{aligned} W^*_E \quad & +0.770(W^*_G + W^*_{G'} + W^*_H) + \tfrac{3}{4}(W'_1 + W'_2) \\ & +(1 - (\tfrac{3}{4})^2(1-p))W'_3 + (1 - (\tfrac{3}{4})^3 p)W'_4 \\ & + \textstyle\sum_{k\geq 5}(1 - (\tfrac{3}{4})^k)W'_k. \end{aligned} \tag{24}$$

Now we consider an algorithm D consisting of the following four algorithms:

(1) set each variable x_i true independently with probability 0.5;

(2) set x_i true independently with probability $\frac{1+y_{0i}}{2}$ using the optimal solution $(Y, \boldsymbol{z}^*)$ to (S);

(3) take a random unit vector $\boldsymbol{r}$ and set x_i true if and only if $\mathrm{sgn}(\bar{\boldsymbol{v}}^*_i \cdot \boldsymbol{r})$=$\mathrm{sgn}(\bar{\boldsymbol{v}}^*_0 \cdot \boldsymbol{r})$ using the optimal solution $(Y^*, \boldsymbol{z}^*)$ to (S) and (R') $(y^*_{i_1 i_2} = \boldsymbol{v}^*_{i_1} \cdot \boldsymbol{v}^*_{i_2})$.

(4) set each variable x_i in P', $(P - P') \cup Q$ or Z true independently with probability 3/4, 0.568729 or 1/2, respectively based on the modified Yannakakis's algorithm above.

Suppose we use algorithm (i) with probability p_i, where $p_1 + p_2 + p_3 + p_4 = 1$. If we set $p_1 = p_2 = 0.3110053$, $p_3 = 0.1201425$ and $p_4 = 0.2578469$, then the expected value of $\mathcal{D}$ can be shown to be at least $0.76544 \sum_{C_j \in D} w_j z^*_j$ $(\mathcal{D} = \mathcal{E}, \mathcal{G}, \mathcal{G}', \mathcal{H})$. Furthermore, the expected value of $\mathcal{F}_k$ can also be shown to be at least

$\sum_{C_j \in \mathcal{F}_k} w_j((1 - \frac{1}{2^k})p_1 + \gamma_k p_2 + \alpha\alpha_k p_3 + \delta_k)z^*_j$
$\geq 0.76544 \sum_{C_j \in \mathcal{F}_k} w_j z^*_j$,

where $\gamma_k = 1 - (1 - \frac{1}{k})^k$, $\delta_1 = \delta_2 = 0.75$, $\delta_3 = 1 - 0.75^2(1-p)$, $\delta_4 = 1-0.75^3 p$, $\delta_k = 1-0.75^k$ $(k \geq 5)$.

Thus, we have the following theorem.

Theorem 2 *A* 0.76544-*approximation algorithm can be obtained based on the algorithm A in Section 3, the Yannakakis's algorithm modified as above, and the* 0.75-*approximation algorithm of Goemans-Williamson.*

We have presented two approximation algorithms for MAX SAT, including a 0.76544-approximation algorithm. We believe this approach can be used to further improve the performance guarantee for MAX SAT. Note that Goemans-Williamson's 0.7584-approximation algorithm can be shown to have the same performance guantee 0.75899 by choosing $p_1 = p_2 = 0.4650334$ and $p_3 = 0.0699332$ as in this paper. Although our 0.75899-approximation algorithm is of the same performance as Geomans-Williamson's algorithm, our algorithm has better performances when applied to MAX kSAT (for example 0.769 for $k \leq 5$, while Geomans-Williamson is at most 0.765). On the other hand, Feige and Goemans recently obtained 0.931-approximation algorithm for MAX 2SAT [1] and if it is used in the Goemans-Williamson's 0.7584-approximation algorithm in place of 0.878-approximation algorithm, then the performance guarantees 0.76199 for MAX SAT and 0.770 for MAX 3SAT can be obtained. The techniques in 0.931-approximation algorithm for MAX 2SAT may also be used in our algorithms.

Acknowledgments. The first author was supported in part by Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan.

References

[1] U. Feige and Michel X. Goemans, Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT, In *Proc. 3rd Israel Symposium on Theory of Computing and Systems*, 1995, pp.182–189.

[2] Michel X. Goemans and David P. Williamson, .878-approximation algorithms for MAX CUT and MAX 2SAT, In *Proc. 26th STOC*, 1994, pp.422–431.

[3] Michel X. Goemans and David P. Williamson, New 3/4-approximation algorithms for the maximum satisfiability problem, *SIAM Journal of Disc. Math.*, 7 (1994), pp.656–666.

[4] Michel X. Goemans and David P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, In *Journal of the ACM*, 42 (1995), pp.1115–1145.

[5] David S. Johnson, Approximation algorithms for combinatorial problems, *Journal of Comput. and Sys. Sci.*, 9 (1974), pp.256–278.

[6] S. Mahajan and H. Ramesh, Correctly derandomizing Geomans and Williamson's MAX CUT algorithm, Unpublished manuscript, 1995.

[7] E. Tardos, A strongly polynomial algorithm for solving combinatoral linear program, *Operations Research*, 11 (1986), pp.250-256.

[8] Mihalis Yannakakis, On the approximation of maximum satisfiability, in *Proc. 3rd SODA*, 1992, pp.1–9 (and also in *J. Algorithms*, 17 (1994), pp.475-502).