IEICE TRANS. INF. & SYST., VOL.E83–D, NO.3 MARCH 2000

INVITED SURVEY PAPER *Special Issue on Algorithm Engineering: Surveys*

Approximation Algorithms for MAX SAT

Tomio HIRATA[†], ***Member and*** **Takao ONO**[†], ***Nonmember***

SUMMARY Maximum Satisfiability Problem (MAX SAT) is one of the most natural optimization problems. Since it is known to be NP-hard, approximation algorithms have been considered. The aim of this survey is to show recent developments of approximation algorithms for MAX SAT.

key words: *MAX SAT, approximation algorithms, semidefinite programming*

1. Introduction

Maximum Satisfiability Problem (MAX SAT) is one of the most natural optimization problems. It is known to be NP-hard. Hence, approximation algorithms have been considered. The aim of this survey is to show recent developments of approximation algorithms for MAX SAT. We will confine ourselves to approximation algorithms with theoretical performance guarantees. For other approximation algorithms with no theoretical guarantees, such as local search, tabu search, simulated annealing, etc., which often show high approximation performance in practice, the reader is referred to [9], [11], [17], [25].

As we shall see in this survey, algorithms with better and better approximation ratios have been proposed. Hence, it is natural to ask whether it is possible to further improve approximation properties. Since MAX 3SAT is known to be MAX SNP-complete [23], there is a limit for the approximation ratio under the assumption that P $\neq$ NP. This comes from the theory of probabilistically checkable proofs (PCP) [3]. This is a principal motivation for researchers for challenging approximation algorithms with better and better approximation ratios approaching to this limit. Håstad showed that unless P = NP MAX 3SAT (and hence MAX SAT) cannot be approximated in polynomial time within an approximation ratio greater than 7/8 [16].

In Sect. 2, some definitions are given. Johnson's algorithm and a randomized algorithm using LP are presented in Sect. 3. We also mention about a property 3/4 function and briefly review Yannakakis's method which uses a network flow algorithm. In Sect. 4, Goemans and Williamson's algorithm based on semidefinite programming is presented. Their algorithm is originally for MAX 2SAT but it can be used for MAX SAT to improve the approximation ratio. We show, in Sect. 5, that further improvement is possible by introducing perturbation to the solution of this algorithm. In Sect. 6, more recent results are given and derandomization is discussed.

Manuscript received July 3, 1999.
Manuscript revised November 5, 1999.

[†]The authors are with the School of Engineering, Nagoya University, Nagoya-shi, 464–8603 Japan.

2. Definitions

Definition 1 (Maximum Satisfiability Problem): Given a set $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of clauses with associated positive weights $w_1, w_2, \ldots, w_m$, the Maximum Satisfiability Problem (MAX SAT) is the problem to find the Boolean assignment to the variables $x_1, x_2, \ldots, x_n$ which maximizes the sum of the weights of the satisfied clauses. The size of a clause C, i.e., the number of the literals in C, is denoted by $|C|$. We denote by $\mathcal{C}_k = \{C \in \mathcal{C} \mid |C| = k\}$ the set of the clauses of size k. X_j^+ (X_j^-) denotes the set of indices of the positive (negative, respectively) literals in $C_j \in \mathcal{C}$. The Maximum k-Satisfiability Problem, MAX kSAT, is MAX SAT where every clause has at most k literals.

For $k \geq 2$, MAX kSAT is known to be NP-hard. Moreover, it is known to be MAX SNP-complete.

Definition 2 (Approximation Ratio): Let A be an approximation algorithm for MAX SAT. We denote by $w(\boldsymbol{x})$ the weight of an assignment $\boldsymbol{x}$, i.e., the sum of the weights of the clauses satisfied by $\boldsymbol{x}$. If there exists a constant α such that for any MAX SAT instance $\mathcal{I}$,

$$w(A(\mathcal{I})) \geq \alpha w(\mathrm{opt}(\mathcal{I})),$$

then the approximation ratio of A is α, where $A(\mathcal{I})$ is the approximate solution produced by A for $\mathcal{I}$ and $\mathrm{opt}(\mathcal{I})$ is the optimum solution to $\mathcal{I}$. Polynomial time algorithm with approximation ratio α is called an α-approximation algorithm.

For randomized algorithm A, we define its approximation ratio by

$$\min_{\mathcal{I}} \frac{E[w(A(\mathcal{I}))]}{w(\mathrm{opt}(\mathcal{I}))},$$

where $E[X]$ is the expectation of the random variable X.

3. Johnson's Algorithm and LP-Relaxation Algorithm

3.1 Johnson's Algorithm

Johnson's algorithm for MAX SAT [18] is a simple randomized algorithm: For each variable, choose its assignment with equal probability. It is clear that the assignment $\boldsymbol{x}_J$ produced by Johnson's algorithm satisfies a clause of size k with probability $\alpha_{1,k} = 1 - 1/2^k$, where we use suffix 1 to distinguish several approximation algorithms. Therefore, the expected weight of the solution produced by Johnson's algorithm is

$$\sum_{k\geq 1} \sum_{C_j \in \mathcal{C}_k} \alpha_{1,k} w_j.$$

It is obvious that the weight of the optimal solution is at most

$$\sum_{k\geq 1} \sum_{C_j \in \mathcal{C}_k} w_j.$$

Thus, the approximation ratio of this algorithm is at least

$$\begin{aligned} \min \frac{\sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k} \alpha_{1,k} w_j}{\sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k} w_j} &= \min_{k\geq 1} \frac{\sum_{C_j\in\mathcal{C}_k} \alpha_{1,k} w_j}{\sum_{C_j\in\mathcal{C}_k} w_j} \\ &= \min_{k\geq 1} \alpha_{1,k} = \frac{1}{2}. \end{aligned}$$

We will discuss the derandomization of Johnson's algorithm in Sect. 6. With more sophisticated analysis, it turns out that Johnson's algorithm is 2/3 approximation algorithm [10].

We can generalize Johnson's algorithm as follows: For variable x_i let p_i be the probability that x_i is set to **true**. The probability that a clause C_j of size k is satisfied by this random assignment is

$$1 - \prod_{i\in X_j^+} (1-p_i) \prod_{i\in X_j^-} p_i.$$

Johnson's algorithm is the case that all p_i's are 1/2. Both Yannakakis's algorithm and LP-relaxation algorithm compute these probabilities to achieve better approximation ratio.

3.2 LP-Relaxation Algorithm

It is well known that linear programming problem (LP) can be solved in polynomial time. Goemans and Williamson proposed an approximation algorithm with relaxation from MAX SAT instances into LP instances [13]. Following is their algorithm.

Introduce 0-1 variables y_i's and z_j's for Boolean variables x_i's and clauses C_j's, respectively, where 0 corresponds to **false** and 1 corresponds to **true**. We define

$$c_j(\boldsymbol{y}) = \sum_{i\in X_j^+} y_i + \sum_{i\in X_j^-} (1-y_i)$$

for C_j. It is easy to see that $c_j(\boldsymbol{y}) = 0$ if and only if $C_j =$ **false** and $c_j(\boldsymbol{y}) \geq 1$ if and only if $C_j =$ **true**. Thus, the following optimization problem is equivalent to MAX SAT.

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} w_j z_j \\ \text{s.t.} \quad & z_j \leq c_j(\boldsymbol{y}) \qquad \text{for all } j, \\ & y_i, z_j \in \{0,1\}. \end{aligned} \tag{1}$$

Relaxing the conditions for y_i's and z_j's with $0 \leq y_i,\ z_j \leq 1$, we obtain the following LP instance.

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} w_j z_j \\ \text{s.t.} \quad & z_j \leq c_j(\boldsymbol{y}) \qquad \text{for all } j, \\ & 0 \leq y_i, z_j \leq 1. \end{aligned} \tag{2}$$

We can solve this LP instance in polynomial time to have an optimal solution y_1^*, y_2^*, …, y_n^*, z_1^*, z_2^*, …, z_m^*. If we consider the value y_i^* as the probability that the Boolean variable x_i is set to **true**, the probability that this random assignment $\boldsymbol{x}_L$ satisfies clause $C_j \in \mathcal{C}$ of size k is

$$\begin{aligned} &\Pr\{C_j = 1\} \\ &= 1 - \prod_{i\in X_j^+} (1-x_i^*) \prod_{i\in X_j^-} x_i^* \\ &\geq 1 - \left[\frac{\sum_{i\in X_j^+}(1-x_i^*) + \sum_{i\in X_j^-} x_i^*}{k}\right]^k \\ &= 1 - \left\{\frac{k - [\sum_{i\in X_j^+} x_i^* + \sum_{i\in X_j^-}(1-x_i^*)]}{k}\right\}^k \\ &\geq 1 - \left(1 - \frac{z_j^*}{k}\right)^k \\ &\geq \left[1 - \left(1-\frac{1}{k}\right)^k\right] z_j^* \\ &= \alpha_{2,k} z_j^*. \end{aligned}$$

In the above equation, the first inequality is due to the relation between arithmetic and geometric means, the second inequality comes from the constraint that $z_j \leq c_j(\boldsymbol{y})$, the last inequality comes from the constraint that $z_j \leq 1$. We obtain an approximate solution to the original MAX SAT instance from this random assignment by the method of conditional probability [24]. The expectation of this random assignment $E[w(\boldsymbol{x}_L)]$ satisfies the following inequality:

$$\begin{aligned}
E[w(\boldsymbol{x}_L)] &\geq \sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k}\left[1-\left(1-\frac{1}{k}\right)^k\right]w_j z_j^*\\
&\geq \sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k}\left(1-\frac{1}{e}\right)w_j z_j^*\\
&= \left(1-\frac{1}{e}\right)\sum_{j=1}^{m} w_j z_j^*.
\end{aligned}$$

The sum $\sum_{j=1}^m w_j z_j^*$ is an optimal value of the relaxed linear programming (2) and thus it is no less than the optimal value of the original MAX SAT. Therefore, we conclude that this algorithm is $1-1/e \approx 0.632$-approximation algorithm.

It is easy to see that Johnson's algorithm has better performance ratio when a clause gets larger. On the other hand, LP-relaxation algorithm is good for small clauses. Thus, if we combine these algorithms, performance ratio will be better. In the following, we show that the algorithm, which solves a given MAX SAT instance by Johnson's algorithm and LP-relaxation algorithm and outputs a better one, is 0.75-approximation algorithm.

Let us denote by $\boldsymbol{x}_J$ and $\boldsymbol{x}_L$ the solutions of Johnson's and LP-relaxation algorithms, respectively. Their weights satisfy the following inequalities:

$$w(\boldsymbol{x}_J) \geq \sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k}\left(1-\frac{1}{2^k}\right)w_j,$$

$$w(\boldsymbol{x}_L) \geq \sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k}\left[1-\left(1-\frac{1}{k}\right)^k\right]w_j z_j^*,$$

where z_j^*'s come from the optimal solution of the relaxed LP instance. The sum of these weights is thus:

$$\begin{aligned}
&w(\boldsymbol{x}_J)+w(\boldsymbol{x}_L)\\
&\quad\geq \sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k}\left(1-\frac{1}{2^k}\right)w_j\\
&\qquad+\sum_{k\geq 1}\sum_{C_j\in\mathcal{C}_k}\left[1-\left(1-\frac{1}{k}\right)^k\right]w_j z_j^*\\
&\quad\geq \sum_{k\geq 1}\left\{\left(1-\frac{1}{2^k}\right)+\left[1-\left(1-\frac{1}{k}\right)^k\right]\right\}\\
&\qquad\times\sum_{C_j\in\mathcal{C}_k} w_j z_j^*,
\end{aligned}$$

where we used the constraint that $z_j^* \leq 1$. We can see that for any integer $k \geq 1$,

$$\left(1-\frac{1}{2^k}\right)+\left[1-\left(1-\frac{1}{k}\right)^k\right] \geq \frac{3}{2},$$

thus

$$\frac{E[w(\boldsymbol{x}_j)]+E[w(\boldsymbol{x}_L)]}{2} \geq \frac{3}{4}\sum_j w_j z_j^*.$$

This implies that the algorithm which chooses a better solution of $\boldsymbol{x}_J$ and $\boldsymbol{x}_L$ is 3/4-approximation.

In this algorithm we considered that x_i^* is the probability that the Boolean variable x_i is set to **true**. However, this is not the only interpretation of x_i^*, that is, tuning the probability with a bit care leads to the better approximation ratio. Goemans and Williamson defined *property* 3/4[13] and proved that the general algorithm described in Sect. 3.1 is 3/4-approximation algorithm if we set $p_i = f(y_i^*)$ for $f(t)$ with property 3/4. Examples of $f(t)$ with property 3/4 shown in [13] are:

$$1-4^{-t} \leq f_1(t) \leq 4^{t-1},$$

$$f_2(t) = \alpha + (1-2\alpha)t \left(2-\frac{3}{\sqrt[3]{4}} \leq \alpha \leq \frac{1}{4}\right),$$

$$f_3(t) = \begin{cases} 3t/4+1/4 & \text{if } 0 \leq t \leq 1/3,\\ 1/2 & \text{if } 1/3 < t \leq 2/3,\\ 3t/4 & \text{if } 2/3 < t \leq 1 \end{cases}$$

Note that using $f_2(t)$ for $\alpha = 1/4$, the probability that x_i is set to **true** is the average of Johnson's algorithm and the plain LP-relaxation algorithm.

3.3 Another 3/4-Approximation Algorithm by Yannakakis

Yannakakis proposed another 3/4-approximation algorithm [26]. This algorithm uses a very different approach, that is, it exploits maximum flow on a graph to determine p_i, the probability for variable x_i to be **true**. Since this algorithm is complicated, we restrict our attention on the case where the input instance is MAX 2SAT instance. For general MAX SAT instances, the reader is referred to [4], [26].

We consider MAX 2SAT instance $\mathcal{I}$. We construct a directed graph $G = (V, E)$ as follows: all the literals $x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ are the vertices of G. G also includes a source vertex s and a sink vertex t, corresponding to Boolean **true** and **false**, respectively. For each clause $C = x_i$ of size 1, add two edges $s \to x_i$, $\bar{x}_i \to t$ into E, whose weights are half of the weight of C. For each clause $C = x_i \vee x_j$ of size 2, we add two edges $\bar{x}_i \to x_j$, $\bar{x}_j \to x_i$ into E, whose weights are half of the weight of C.

We say that two edges $a \to b$ and $\bar{b} \to \bar{a}$ are corresponding to each other, where we consider that $\bar{\bar{a}} = a$ and $\bar{s} = t$. If each pair of corresponding edges has the same weight, we say that the graph is symmetric. Also, if a flow f satisfies $f(e) = f(e')$ for any corresponding edges e and e', f is called a symmetric flow. From the above construction, graph G is clearly symmetric. Also we can invert this construction, i.e., we can construct MAX 2SAT instance from any symmetric graph.

In a symmetric graph G, for any flow f we can construct a symmetric flow $\tilde{f}$ with the same value: Just assign $\tilde{f}(e) = \tilde{f}(\bar{e}) = (f(e)+f(\bar{e}))/2$ for every edge e in G, where $\bar{e}$ is a corresponding edge of e. Let f be a symmetric flow on G whose value is $v(f)$. Because the graph G and flow f are symmetric, the residual graph [26] N of G with respect to the flow f is also symmetric, so that there is a MAX 2SAT instance $\mathcal{I}'$ corresponding to N. We have an important lemma [26]:

Lemma 1: For any Boolean assignment $\boldsymbol{x}$ and for any symmetric flow f,

$$w_{\mathcal{I}}(\boldsymbol{x}) = w_{\mathcal{I}'}(\boldsymbol{x}) + v(f),$$

where $w_{\mathcal{I}}(\boldsymbol{x})$ denotes the value of $\boldsymbol{x}$ on $\mathcal{I}$.

Let f be a maximum symmetric flow. Because f is a maximum flow, there are no paths from s to t with positive capacity in N. Thus we divide the set of vertices V into three sets as follows: P, a set of any vertices reachable from s in N; P', a set of any vertices from which t is reachable in N; and Q, a set of vertices neither reachable from s nor from which t is reachable in N. Note that there is no complementing pair of vertices in both P and P'. This is because if both x and $\bar{x} \in P$, then there is a path from s to t, namely, $s \to x \to t$ by the symmetry of N. This contradicts the maximality of f. We note also that $P' = \{\bar{x} \mid x \in P\}$. Therefore, a MAX 2SAT instance $\mathcal{I}'$ constructed from the residual graph N includes no complementing pair of unit clauses. We consider the assignment that each variable in P is set to **true** with probability 3/4 and each variable in Q is set to **true** with probability 1/2. Any clause containing a literal in P is satisfied with the probability at least 3/4. Any clause $a \vee b$, where both a and $b \in Q$, is satisfied with the probability $1-(1/2)\cdot(1/2) = 3/4$. Note that if there exists a clause $\bar{a} \vee b$ in $\mathcal{I}'$ for $\bar{a} \in P'$, then b must be in P. Thus, this random assignment satisfies every clause in $\mathcal{I}'$ with the probability at least 3/4, which implies that it is 3/4-approximate solution to $\mathcal{I}'$. By Lemma 1, it is also 3/4-approximate solution to $\mathcal{I}$.

As we have seen in Sect. 3.1, Johnson's algorithm achieves 3/4-approximation if the input instance has no unit clauses. Thus instead of assigning **true** to the literals in P with probability 3/4, we could have set every literal in P to **true**, because such an assignment leaves no unit clauses.

4. SDP-Relaxation Algorithm

Goemans and Williamson proposed an approximation algorithm for MAX 2SAT which relaxes MAX 2SAT instances into semidefinite programming (SDP) instances [14]. SDP is the problem asking an $n \times n$ matrix $X = (x_{ij})$ optimizing a linear combination of x_{ij} subject to the linear constraints of x_{ij} and the constraint that X is symmetric and positive semidefinite. It is known that SDP can be solved within additional error ϵ in polynomial time [1], that is, we can compute a solution in polynomial time to within any desired precision. Goemans-Williamson's SDP-relaxation algorithm for MAX 2 SAT is generalized for the case of MAX SAT [7], [14], [21], [22]. In the following we describe an SDP-relaxation algorithm for MAX SAT and analyze its approximation ratio.

First we arithmetize a MAX SAT instance by introducing new variables, $y_1, y_2, \ldots, y_n$ for Boolean variables $x_1, x_2, \ldots, x_n$ and $z_1, z_2, \ldots, z_m$ for clauses $C_1, C_2, \ldots, C_m \in \mathcal{C}$. Further more, we introduce a special variable y_0. The value of each y_i $(i = 0, 1, \ldots, n)$ is -1 or 1, while $z_j \in \{0,1\}$. The special variable y_0 stands for "which value means Boolean **true**," that is, for $i = 1, 2, \ldots, n$, $y_i = y_0$ if and only if $x_i =$ **true**. For a clause $C_j = x_1 \vee x_2 \vee \ldots \vee x_k$, we construct a quadratic function $c_j(\boldsymbol{y})$ of y_i's as follows:

$$c_j(\boldsymbol{y}) = \frac{1}{k}\left[\sum_{i=1}^{k}\frac{1+y_0y_i}{2} + \sum_{1\le i<j\le k}\frac{1-y_iy_j}{2}\right],$$

where if x_i appears as a negative literal in C_j, then we replace every occurrence of y_i with $-y_i$. It is not hard to see that $c_j(\boldsymbol{y}) = 0$ if and only if $C_j =$ **false** and that $c_j(\boldsymbol{y}) \ge 1$ if and only if $C_j =$ **true**. The following optimization problem is thus equivalent to the original MAX SAT.

$$\begin{array}{ll} \max & \displaystyle\sum_{j=1}^{m} w_j z_j \\ \text{s.t.} & z_j \le c_j(\boldsymbol{y}) \quad \text{for all } j, \\ & z_j \in \{0,1\}, \\ & y_i \in \{-1,1\}. \end{array} \tag{3}$$

We relax y_i's by $(n+1)$-dimensional unit vectors $\boldsymbol{v}_i$'s and replace the product y_iy_j with inner product $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$ of the corresponding vectors. We also relax the constraints for z_j with $z_j \le 1$. With this relaxation we obtain the following problem:

$$\begin{array}{ll} \max & \displaystyle\sum_{j=1}^{m} w_j z_j \\ \text{s.t.} & z_j \le c'_j(\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \quad \text{for all } j, \\ & z_j \le 1, \\ & \|\boldsymbol{v}_i\| = 1, \end{array} \tag{4}$$

where $c'_j(\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ is a relaxed version of $c_j(\boldsymbol{y})$, that is,

$$c'_j(\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) = \frac{1}{k}\left[\sum_{i=1}^{k}\frac{1+\boldsymbol{v}_0\cdot\boldsymbol{v}_i}{2} + \sum_{1\le i<j\le k}\frac{1-\boldsymbol{v}_i\cdot\boldsymbol{v}_j}{2}\right].$$

Let us denote by y_{ij} the inner product $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$, this problem is rewritten as:

$$\begin{array}{ll} \max & \sum_{j=1}^{m} w_j z_j \\ \text{s.t.} & z_j \le c_j(Y) \text{ for all } j, \\ & z_j \le 1, \\ & y_{ii} = 1, \\ & Y = (y_{ij}) \text{ is symmetric} \\ & \quad \text{and positive semidefinite,} \end{array} \qquad (5)$$

where

$$c_j(Y) = \frac{1}{k}\left[\sum_{i=1}^{k} \frac{1+y_{0i}}{2} + \sum_{1\le i<j\le k} \frac{1-y_{ij}}{2}\right].$$

This problem is an SDP problem. Therefore we can find a semi-optimal solution $\tilde{Y}$, $\tilde{z}_1$, $\tilde{z}_2$, …, $\tilde{z}_m$ within additional error ϵ in polynomial time in n, m and $\log 1/\epsilon$ [1]. We calculate $\tilde{\boldsymbol{v}}_i$'s by the Cholesky-decomposition of $\tilde{Y}$. Now we have to "round" $\tilde{\boldsymbol{v}}_i$'s to obtain an approximate solution to original MAX SAT. To do this, we choose a random unit vector $\boldsymbol{r}$ and set $\tilde{y}_i = 1$ if $\boldsymbol{r} \cdot \tilde{\boldsymbol{v}}_i > 0$, $\tilde{y}_i = -1$ otherwise. Finally $\tilde{x}_i = \textbf{true}$ if and only if $\tilde{y}_i = \tilde{y}_0$. We denote by $\tilde{\boldsymbol{x}}$ the approximate solution obtained by the algorithm.

In the analysis we need two quantities:

$$\alpha = \min_{0<\theta\le\pi} \frac{\theta/\pi}{(1-\cos\theta)/2},$$

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

The value of α is the approximation ratio of Goemans-Williamson's algorithm for MAX 2SAT and is estimated at 0.878 [14], and the value of α_k is chosen to ensure that $\alpha_k c_j(\boldsymbol{y}) \le 1$ for any clause C_j of size k. Let us define a function

$$C_j(\boldsymbol{x}) = \begin{cases} 1 & \text{if } C_j \text{ is satisfied by } \boldsymbol{x} \\ 0 & \text{otherwise.} \end{cases}$$

By the definition of α_k, the following inequality holds for arbitrary $\boldsymbol{y} = (y_0, y_1, \ldots, y_n)$:

$$\alpha_k c_j(\boldsymbol{y}) \le C_j(\boldsymbol{x}),$$

where $x_i = \textbf{true}$ if and only if $y_i = y_0$.

Thus we can bound the probability that $C_j = x_1 \vee x_2 \ldots \vee x_k$ is satisfied from below as follows:

$$\begin{aligned} &\Pr\{C_j \text{ is satisfied}\} \\ &= E[C_j(\tilde{\boldsymbol{x}})] \\ &\ge E[\alpha_k c_j(\tilde{\boldsymbol{y}})] \\ &= \alpha_k \frac{1}{k}\left\{\sum_{i=1}^{k} E\left[\frac{1+\tilde{y}_0\tilde{y}_i}{2}\right] \right. \\ &\quad \left. + \sum_{1\le i<j\le k} E\left[\frac{1-\tilde{y}_i\tilde{y}_j}{2}\right]\right\} \\ &\ge \alpha_k \frac{1}{k}\left\{\sum_{i=1}^{k} \alpha\frac{1+\tilde{y}_{0i}}{2} + \sum_{1\le i<j\le k} \alpha\frac{1-\tilde{y}_{ij}}{2}\right\} \\ &= \alpha\alpha_k c_j(\tilde{Y}) \\ &\ge \alpha\alpha_k \tilde{z}_j. \end{aligned}$$

Thus, for any clause C_j the probability that C_j is satisfied by $\tilde{x}_i$'s is at least $\alpha_{3,k}\tilde{z}_j = \alpha\alpha_k\tilde{z}_j$. Therefore the expected weight of this approximate solution $\boldsymbol{x}_S = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ is at least

$$\sum_{j=1}^{m} w_j \Pr\{C_j = 1\} \ge \sum_{k\ge1} \alpha_{3,k} \sum_{C_j\in\mathcal{C}_k} w_j\tilde{z}_j.$$

Although this algorithm achieves good approximation ratio for the clauses of size 1 and 2, its approximation ratio for MAX SAT is 0 because the quantity $\alpha_{3,k}$ tends to 0 as k tends to infinity.

To improve the approximation ratio let us consider the combination of three algorithms again: Johnson's, LP-relaxation, and SDP-relaxation [14]. The actual algorithm runs these three algorithms on a given instance and outputs the best solution, but we analyze the algorithm which randomly chooses one of three algorithms and runs it on a given instance. In order to determine the probabilities for each algorithm to be chosen which maximize the approximation ratio of this algorithm, it is necessary to solve the following linear programming problem:

$$\begin{array}{ll} \max & a \\ \text{s.t.} & a \le p_1\alpha_{1,k} + p_2\alpha_{2,k} + p_3\alpha_{3,k} \\ & \qquad\qquad \text{for } k = 1, 2, \ldots, \\ & p_1, p_2, p_3 \ge 0, \\ & p_1 + p_2 + p_3 = 1, \end{array}$$

where $\alpha_{1,k} = 1-1/2^k$, $\alpha_{2,k} = 1-(1-1/k)^k$, and $\alpha_{3,k} = \alpha\alpha_k$. The optimal value of a is the approximation ratio, and in this case it is $0.75899\ldots$.

We note that for MAX 2SAT, the result of Goemans and Williamson was improved by Feige and Goemans [12], in which they used a stronger semidefinite programming than the one considered in [14]. The achieved approximation ratio α is 0.931. But we cannot substitute our α with this ratio, since they used a non-uniform rounding scheme.

5. SDP-Relaxation Algorithm with Perturbation

In the previous section we described an SDP-relaxation algorithm for MAX SAT. In this section we introduce perturbation to make more improvement. Perturbation to improve approximation ratio is first introduced by Andersson and Engebretsen [2]. They improved the approximation ratio for Set Splitting Problem and Maximum Not-All-Equal SAT Problem. We use this technique to the approximation algorithm for MAX SAT and improve the approximation ratio [22].

Perturbation is a technique that for a probabilistic assignment $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, we alter the value of each variable with suitable small probability p. Intuitively, perturbation is effective for algorithms which achieve good approximation for small clauses but are poor for large clauses, e.g., SDP-relaxation algorithm, because large clauses can be satisfied easily by random assignment.

We shall estimate the effect of perturbation. Fix a clause C of size k, which is assumed, without loss of generality, to be of the form $x_1 \vee x_2 \vee \ldots \vee x_k$, a random assignment $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, and probability of perturbation $p < 1/2$. Suppose that $\boldsymbol{x}$ satisfies C with probability β. We consider two cases: If C is satisfied, without loss of generality we can assume that $x_1 = x_2 = \ldots = x_i =$ **true** and that $x_{i+1} = x_{i+2} = \ldots = x_k =$ **false**. In this case, perturbed random assignment $\boldsymbol{x}'$ does not satisfy C when $x_1, x_2, \ldots, x_i$ is perturbed to be 0 and the other variables are not perturbed. Thus the probability that C is not satisfied by $\boldsymbol{x}'$ is $p^i(1-p)^{k-i}$. Therefore, C is satisfied by $\boldsymbol{x}'$ assuming C is satisfied by $\boldsymbol{x}$ is at least

$$\min_{i \geq 1}\left[1 - p^i(1-p)^{k-i}\right] = 1 - p(1-p)^{k-1}.$$

Let us consider another case where C is not satisfied by $\boldsymbol{x}$. This implies that $x_1 = x_2 = \ldots = x_k =$ **false**. Thus C is satisfied by $\boldsymbol{x}'$ if at least one variable is altered by perturbation. Therefore the probability that C is satisfied by $\boldsymbol{x}'$ assuming C is not satisfied by $\boldsymbol{x}$ is

$$1 - (1-p)^k.$$

Because these two cases occur with probability β and $1-\beta$, respectively, the overall probability $\mathit{perturb}_k(\beta, p)$ that C is satisfied by $\boldsymbol{x}'$ is

$$\begin{aligned} &\mathit{perturb}_k(\beta, p) \\ &\geq \beta\left[1 - p(1-p)^{k-1}\right] + (1-\beta)\left[1 - (1-p)^k\right] \\ &= \left[1 - (1-p)^k\right] + \beta(1-2p)(1-p)^{k-1}. \end{aligned}$$

Note that if $\alpha < \beta$ then $\mathit{perturb}_k(\alpha, p) \leq \mathit{perturb}_k(\beta, p)$ for any $p < 1/2$. Also note that assuming $\alpha < 1$ and $\beta < 1$, $\mathit{perturb}_k(\alpha\beta, p) \geq \mathit{perturb}_k(\alpha, p)\beta$.

Let us consider an SDP-relaxation algorithm with perturbation. The random assignment given by this algorithm satisfies a clause C of size k with probability at least $\mathit{perturb}_k(a_{3,k}\tilde{z}_j, p) \geq \mathit{perturb}_k(\alpha_{3,k}, p)\tilde{z}_j$. Thus the approximation ratio of this algorithm is obtained from the following optimization problem:

$$\begin{array}{ll} \max & a \\ \text{s.t.} & a \leq \mathit{perturb}_k(\alpha_{3,k}, p) \quad \text{for all } k. \end{array}$$

The optimum value is 0.7359 at $p = 0.1884$, implying that this algorithm is 0.7359-approximation.

As in the previous section, we consider the combination of three algorithms, Johnson's, LP-relaxation, and SDP-relaxation, but this time with perturbation. We denote by A_i ($i = 1, 2, 3$) these algorithms in this order. In the actual algorithm, we first run these algorithms and then perturb each solution, and finally output the solution which shows the best performance. We analyze, however, the following approximation algorithm for the technical reason. It probabilistically chooses A_i and the probability of perturbation, then runs the chosen algorithm A_i on a given instance, and perturbs the obtained solution with the chosen perturbation probability. We denote by $p_{i,q}$ the probability to choose the solution of A_i and perturbation probability q. To obtain the probabilities $p_{i,q}$ which gives the best approximation ratio of this algorithm, we need to solve the following linear programming problem.

$$\begin{array}{ll} \max & a \\ \text{s.t.} & a \leq \displaystyle\sum_{0 \leq q \leq 1/2} \sum_{i=1}^{3} p_{i,q}\mathit{perturb}_k(\alpha_{i,k}, q), \\ & \displaystyle\sum_{i,p} p_{i,q} = 1, \\ & p_{i,q} \geq 0. \end{array}$$

The optimal solution to this programming is $a \approx 0.7685$, $p_{1,0} = 0.4104$, $p_{2,0} = 0.4143$, $p_{3,0.037} = 0.1753$. Thus, the algorithm which outputs the best of the three solutions, one produced by Johnson's algorithm, one by LP-relaxation algorithm, and one by SDP-relaxation algorithm perturbed with probability 0.037 is 0.7685-approximation algorithm.

6. Further Discussion

We will discuss derandomization of the algorithms. Because all the presented algorithms are probabilistic ones, derandomization is necessary to obtain a deterministic algorithm, or to establish the theoretical bound.

Johnson's algorithm, LP-relaxation algorithm and Yannakakis's algorithm are all considered as the special cases of the random assignment algorithm. Therefore these algorithms are derandomized by the same way. Derandomized algorithms sequentially determine the assignment of the variables based on the conditional probability [24]: Assume that the assignments to the variables $x_1, x_2, \ldots, x_{i-1}$ have been determined. Now we need to determine the assignment of x_i. Let E_1 (E_0) denotes the expected weight of the assignment in which $x_1, x_2, \ldots, x_{i-1}$ have been given and x_i is set to **true** (**false**, respectively). We assign **true** to x_i if $E_1 > E_0$, otherwise we assign **false** to x_i. It is easy to see that the conditional expectations in this course can not decrease. Thus we have the deterministic approximation algorithms with desired approximation ratios.

There is a complicated way of derandomization of SDP-relaxation algorithm [20]. This derandomization requires a sequence of numerical integration and thus the obtained deterministic algorithm takes very long time to run on small instance. We note that this derandomization can be applied to the SDP-relaxation algorithm with perturbation.

We conclude this paper by mentioning the recent results about approximability of MAX SAT. As a negative result, Håstad showed that if $P \neq NP$, no polynomial time algorithm can achieve approximation ratio of $7/8+\epsilon$. That is, there are no $(7/8+\epsilon)$-approximation algorithm for MAX SAT if $P \neq NP$.

As positive results, Karloff and Zwick proposed a 7/8-approximation algorithm for MAX 3SAT [19]. Zwick extended this result to MAX 3 CSP [27], a problem to find, for a given set of arbitrary Boolean functions at most three literals, the assignment to the variables which maximizes the sum of the weights of the satisfied functions. For general MAX SAT, 0.770-approximation algorithm is known [5]. Recently two algorithms were reported: 0.7846-approximation algorithm for MAX SAT given by Asano and Williamson [8], and 0.8721-approximation algorithm for MAX 4SAT given by Helperin and Zwick [15].

For recent results including above ones, the reader is referred to [6].

References

[1] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," SIAM J. Optimization, vol.1, no.5, pp.13–51, 1995.

[2] G. Andersson and L. Engebretsen, "Better approximation algorithms and tighter analysis for set splitting and not-all-equal SAT," Inf. Process. Lett., vol.6, no.65, pp.305–311, 1998.

[3] S. Arora, C. Lund, R. Motowani, M.Sudan, and M. Szegedy, "Proof verification and hardness of approximation problems," Proc. 33rd IEEE Symposium on the Foundation of Computer Science, pp.14–23, 1992.

[4] T. Asano, K. Hori, T. Ono, and T. Hirata, "A refinement of Yannakakis's algorithm for MAX SAT," IPSJ, SIGAL-TR-54-11, 1996.

[5] T. Asano, K. Hori, T. Ono, and T. Hirata, "A theoretical framework of hybrid approaches to MAX SAT," Proc. 8th International Symposium on Algorithms and Computation, pp.153–162, 1997.

[6] T. Asano, K. Iwama, H. Takada, and Y. Yamashita, "Designing high-quality approximation algorithms for combinatorial optimization problems," IEICE Trans. Inf. Syst., vol.E83-D, no.3, pp.462–479, March, 2000.

[7] T. Asano, T. Ono, and T. Hirata, "Approximation algorithms for the maximum satisfiability problem," Nordic J. Comput., vol.3, pp.388–404, 1996.

[8] T. Asano and D. Williamson, "Simpler approximation algorithms for MAX SAT," Technical Report of IEICE, COMP99-14, pp.33–40, May 1999.

[9] R. Battiti and M. Protasi, "Approximate algorithms and heuristics for MAX-SAT," Handbook of Combinatorial Optimization vol.1, eds. D. -Z. Du and P.M. Pardolos, Kluwer Academic Publishers, pp.77–148, 1998.

[10] J. Chen, D. Friesen, and H. Zheng, "Tight bound on Johnson's algorithm for MAX-SAT," Proc. Twelfth Annual IEEE Conference on Computational Complexity, pp.274–281, 1997.

[11] D. -Z.Du, J. Gu, and P.M. Pardolos, eds., "Satisfiability problem: theory and applications," DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, no.35, 1997.

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

[13] M.X. Goemans and D.P. Williamson, "New 3/4-approximation algorithms for the maximum satisfiability problem," SIAM J. Discrete Mathematics, vol.7, no.4, pp.656–666, Nov. 1994.

[14] M.X. Goemans and D.P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," J. ACM, vol.42, pp.1115–1145, 1995.

[15] E. Halperin and U. Zwick, "Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs," Proc. 7th International IPCO Conference, pp.202–217, 1999.

[16] J. Håstad, "Some optimal inapproximability result," Proc. 28th Annual ACM Symposium on Theory of Computing, El Paso, pp.1–10, 1997.

[17] P. Hensen and B. Jaumard, "Algorithms for the maximum satisfiability problem," Computing, vol.44, pp.279–303, 1990.

[18] D.S. Johnson, "Approximation algorithms for combinatorial problems," J. Comput. Syst. Sci., vol.9, pp.256–278, 1974.

[19] H. Karloff and U. Zwick, "A 7/8-approximation algorithm for MAX 3SAT?," Proc. 38th Symposium on Foundations of Computer Science, pp.406–415, 1997.

[20] S. Mahajan and H. Ramesh, "Derandomizing semidefinite programming based approximation algorithms," Proc. 36th Symposium on Foundations of Computer Science, pp.162–169, 1995.

[21] T. Ono, T. Hirata, and T. Asano, "An approximation algorithm for MAX 3SAT," J. IPSJ, pp.1760–1764, 1996.

[22] T. Ono, T. Hirata, and T. Asano, "Improvement of MAX SAT approximation algorithm with perturbation," IEICE, Trans., vol.J81-D-I, no.9, pp.1107–1111, Sept. 1998.

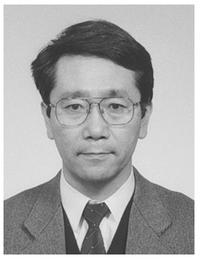
[23] C. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," Proc. 20th ACM Symposium on Theory of Computing, pp.229–234, 1988.

[24] P. Raghavan and C.D. Thompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," Combinatorica, vol.7, no.4, pp.365–374, 1987.

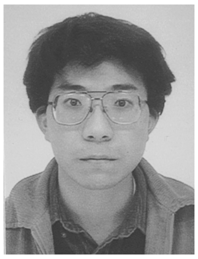
[25] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos, "Approximate solution of weighted MAX-SAT problems using GRASP," Satisfiability Problem: Theory and Applications, D.-Z. Du, J. Gu and P.M. Pardalos, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp.393–405. American Mathematical Society, 1997.

[26] M. Yannakakis, "On the approximation of maximum satisfiability," J. Algorithms, vol.3, no.17, pp.475–502, 1994.

[27] U. Zwick, "Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint," Proc. 9th Symposium on Discrete Algorithms, pp.201–210, 1998.



Tomio Hirata was born in 1949. He received B.S., M.S. and Ph.D. in Computer Science, all from Tohoku University in 1976, 1978, and 1981, respectively. He is currently a Professor in Department of Electronics, Nagoya University. From 1981 to 1986 he was an Assistant Professor of the Computer Science Department, Toyohashi University of Technology. His research interests include graph algorithms and approximation algorithms.



Takao Ono was born in 1970. He received M.E. and Ph.D. from Nagoya University in 1995 and 1999, respectively. He is currently a Research Associate in Department of Electronics, Nagoya University. His research interests include approximation algorithms.