Journal of Computer and System Sciences **58**, 622–640 (1999)
Article ID jcss.1998.1612, available online at http://www.idealibrary.com on IDEAL®

Tight Bound on Johnson's Algorithm for Maximum Satisfiability

Jianer Chen* and Donald K. Friesen

Department of Computer Science, Texas A&M University, College Station, Texas 77843-3112
E-mail: {chen, friesen}@cs.tamu.edu

and

Hao Zheng

Concurrent Technologies Corporation, 100 CTC Drive, Johnstown, Pennsylvania 15904
E-mail: zhengh@ctc.com

Received May 6, 1998

We present new techniques that give a more thorough analysis on Johnson's classical algorithm for the Maximum Satisfiability problem. In contrast to the common belief for two decades that Johnson's Algorithm has performance ratio 1/2, we show that the performance ratio is 2/3 and that this bound is tight. Moreover, we show that simple generalizations of Johnson's algorithm do not improve the performance ratio bound 2/3.

1. INTRODUCTION

The Maximum Satisfiability problem (MAX-SAT) is one of the fundamental optimization problems. MAX-SAT is known to be NP-hard. Hence, it is unlikely that there is a polynomial-time algorithm that solves MAX-SAT optimally.

Approximation algorithms for MAX-SAT have been considered. About two decades ago, Johnson [7] proposed his famous approximation algorithm for MAX-SAT and proved that his algorithm guarantees a performance ratio at least 1/2. Following up investigation on approximation of MAX-SAT has been a very active research area. However, there was no approximation algorithm known before 1992 that has a proven performance ratio better than 1/2. For the studies on the problem before 1990, readers are referred to Hansen and Jaumard's survey [5].

Recent research [1, 9] has shown that MAX-SAT is a generic complete problem under an approximation-preserving reduction for the class APX, which consists of

* Corresponding author. Supported in part by the National Science Foundation under Grants CCR-9110824 and CCR-9613805.

0022-0000/99 $30.00

all NP-optimization problems that can be approximated to a constant performance ratio in polynomial time. This implies that there is a constant $c < 1$ such that no polynomial-time approximation algorithm for MAX-SAT can have a performance ratio larger than c unless $P = NP$. Many current research activities have been concentrating on deriving the precise value for this constant. In particular, Hastad [6] proved that no polynomial time approximation algorithm for MAX-SAT can have performance ratio larger than $7/8 + \varepsilon$ for any constant $\varepsilon > 0$ unless $P = NP$. On the other hand, a variety of techniques have been proposed to develop polynomial time approximation algorithms for MAX-SAT whose performance ratios seem to approach to 7/8 [13, 3, 4, 2, 12, 11, 8].

In the current paper, we present new techniques for analyzing approximation algorithms for MAX-SAT which give a more thorough analysis on Johnson's classical algorithm. In contrast to the common belief that Johnson's Algorithm has a performance ratio 1/2, we prove that the performance ratio is 2/3. Moreover, the bound 2/3 is tight in the sense that there are infinitely many instances for MAX-SAT for which the weight of the assignments constructed by Johnson's Algorithm is exactly 2/3 of the weight of optimal assignments.

There are several reasons why a more thorough and careful analysis for Johnson's Algorithm is needed:

- One major difficulty for designing approximation algorithms is the lack of powerful techniques for deriving tight performance ratios for approximation algorithms. Many approximation algorithms "intuitively" look good. However, they remain suspect because people are not able to derive a provable good performance ratio for them. Therefore, deriving a tight performance ratio using new techniques for a classical algorithm for an important optimization problem may have potential impact on performance analysis for other approximation algorithms.

- MAX-SAT has become central to the study of approximability of NP optimization problems.

- Johnson's Algorithm is an excellent illustration for the probabilistic method [3, 13], which has been playing a more and more important role in design and analysis of approximation algorithms for NP optimization problems. Therefore, a more thorough understanding of Johnson's Algorithm may gain further insight into this powerful method.

- Johnson's Algorithm still plays a fundamental role in many recent improved approximation algorithms for MAX-SAT. Most recent improved approximation algorithms for MAX-SAT combine Johnson's Algorithm with the proposed new algorithms to achieve an improved performance ratio [3, 4, 13].

- Johnson's Algorithm is far more efficient and much simpler than most other approximation algorithms for MAX-SAT. The other approximation algorithms involve solving maximum flow problems [13], solving linear programming problems [3], or solving semidefinite programming problems [4], which in general are much more time-consuming.

One major obstacle in the analysis of approximation algorithms for MAX-SAT is the estimation of the value of an optimal solution to a given instance—this, in fact,

has been a major obstacle in analysis of approximation algorithms for many NP optimization problems. Most early analyses for approximation algorithms for MAX-SAT use the sum of the weights of all clauses in a given instance as an estimation for the value of optimal solutions [7, 10], which is an obvious, but also obviously very loose, upper bound for the value of optimal solutions for the instance of MAX-SAT. More recent analyses for approximation algorithms using linear or semidefinite programming relaxation use the value of optimal solutions for the corresponding relaxation as an estimation for the value of optimal solutions to the instance of MAX-SAT [3, 4]. One contribution of the current paper is that we present a simple technique that shows, in the analysis of Johnson's Algorithm, the performance ratio can be derived based on the *actual* value of optimal solutions for an instance of MAX-SAT.

Our results also reveal interesting facts on the probabilistic method for the MAX-SAT problem. According to Yannakakis [13], Johnson's Algorithm can be interpreted using the probabilistic method, based on uniform distribution, which assigns the value TRUE to each boolean variable with probability 1/2. Based on the analysis of the probabilistic method, one can derive that Johnson's Algorithm has performance ratio 1/2. Yannakakis [13] has suggested further generalizations of the approach of uniform distribution by allowing variables to be assigned the value TRUE with different probability. These generalizations have succeeded in re-interpretation of Lieberherr and Specker's study on the MAX-SAT problem and in achieving better approximation algorithms. Now, observing that Johnson's Algorithm has a performance ratio 2/3, which is better than 1/2, one may suspect that the generalizations of Johnson's Algorithm should have a performance ratio better than 2/3. This is, unfortunately, not true—we are able to construct instances for MAX-SAT for which simple generalizations of Johnson's Algorithm, as suggested by Yannakakis [13], do not have a performance ratio better than 2/3. These examples show that, in the worst case, simple nonuniform distribution does not do better than uniform distribution in the probabilistic method for approximation of the MAX-SAT problem.

Section 2 introduces necessary definitions and reviews Johnson's Algorithm. In Section 3, a modified version of Johnson's Algorithm is presented and a key lemma is proved. Using this lemma in Section 4, we derive the tight bound on the performance ratio for Johnson's Algorithm for MAX-SAT. Remarks on Johnson's Algorithm and its variations are also given.

2. PRELIMINARY

Let $X = \{x_1, ..., x_n\}$ be a set of boolean variables. A *literal* in X is either a boolean variable x_i, or its negation $\bar{x}_i$, for some $1 \leqslant i \leqslant n$. A *clause* on X is a disjunction of literals in X. An instance ϕ of the *Maximum Satisfiability* problem (or MAX-SAT for short) consists of a set of clauses $C_1, ..., C_m$ with corresponding positive weights $w_1, ..., w_m$. The *weight* of a truth assignment τ (i.e., assignment of the value TRUE or FALSE to each variable in X) is defined to be the sum of the weights of the clauses that are satisfied by τ and is denoted by $w(\phi, \tau)$.

The MAX-SAT problem is to find an assignment of maximum weight given an instance ϕ.

An *approximation algorithm* A for MAX-SAT constructs an assignment of the boolean variables given an instance ϕ of MAX-SAT. We say that the algorithm A has *performance ratio* c if for any instance ϕ, the condition $w(\phi, \tau_A)/w(\phi, \tau_{\text{opt}}) \geqslant c$ is true, where τ_A is the assignment constructed by the algorithm A and τ_{opt} is an optimal (i.e., maximum) assignment to ϕ.

Johnson's approximation algorithm [7] for MAX-SAT is described in Fig. 1, where $|C_j|$ denotes the number of literals in the clause C_j.

Initially, each clause C_j is assigned a measure[1] $M(C_j) = w_j/2^{|C_j|}$, and the sum of the measures of the clauses in the set LEFT is equal to

$$\sum_{C_j \in \text{LEFT}} M(C_j) = \sum_{j=1}^{m} M(C_j) = \sum_{j=1}^{m} w_j/2^{|C_j|}.$$

It can be easily verified that Johnson's Algorithm never increases the value $\sum_{C_j \in \text{LEFT}} M(C_j)$. Moreover, at the end of the algorithm, each clause C_j in the set LEFT has measure exactly w_j, and LEFT contains exactly those clauses that are not satisfied by the constructed assignment. Therefore, the assignment constructed by Johnson's Algorithm has weight

$$\sum_{j=1}^{m} w_j - \sum_{C_j \in \text{LEFT}} w_j \geqslant \sum_{j=1}^{m} w_j - \sum_{j=1}^{m} w_j/2^{|C_j|}.$$

If each clause in ϕ contains at least k literals, then $\sum_{j=1}^{m} w_j/2^{|C_j|} \leqslant (\sum_{j=1}^{m} w_j)/2^k$ and the assignment constructed by Johnson's Algorithm has weight at least $(2^k - 1)(\sum_{j=1}^{m} w_j)/2^k$. In other words, the performance ratio for Johnson's Algorithm is at least $(2^k - 1)/2^k$. Note that here we have used the sum $\sum_{j=1}^{m} w_j$ of the weights of all clauses in the instance ϕ as an upper bound for the weight of an optimal assignment to ϕ. In particular, if ϕ contains unit clauses (i.e., clauses that consist of a single literal), then the performance ratio for Johnson's Algorithm is at least 1/2.

Johnson [7] shows that for $k \geqslant 2$, the above bound is tight. That is, there are instances ϕ for MAX-SAT for which Johnson's Algorithm constructs an assignment τ such that the ratio of the weight of τ and the weight of an optimal assignment to ϕ is exactly $(2^k - 1)/2^k$. However, his argument does not apply to general instances of MAX-SAT.

Yannakakis [13] observed that the performance ratio of Johnson's Algorithm cannot be larger than 2/3. To see this, consider the following instance ϕ_h of $3h$

[1] Johnson's original algorithm [7] was presented for unweighted satisfiability problem, in which the measure $M(C_j)$ was called the "weight" of the clause C_j. Since we present Johnson's Algorithm here for weighted satisfiability problem (a more general case), we use the word "measure" instead of "weight" for this value to avoid confusion.

Johnson's Algorithm.

Input: a set of clauses $\phi = \{C_1, \ldots, C_m\}$ on $\{x_1, \ldots, x_n\}$, where the weight of the clause C_j is w_j, $1 \leq j \leq m$
Output: a truth assignment to $\{x_1, \ldots, x_n\}$

1. **for** each clause C_j **do** $M(C_j) = w_j/2^{|C_j|}$
2. LEFT $= \{C_1, \ldots, C_m\}$;
3. **for** $t = 1$ **to** n **do**
   find all clauses $C_1^T, \ldots, C_r^T$ in LEFT that contain x_t;
   find all clauses $C_1^F, \ldots, C_s^F$ in LEFT that contain $\bar{x}_t$;
   if $\sum_{i=1}^{r} M(C_i^T) \geq \sum_{i=1}^{s} M(C_i^F)$
   then $x_t =$ TRUE; delete $C_1^T, \ldots, C_r^T$ from LEFT;
   for $i = 1$ **to** s **do** $M(C_i^F) = 2M(C_i^F)$
   else $x_t =$ FALSE; delete $C_1^F, \ldots, C_s^F$ from LEFT;
   for $i = 1$ **to** r **do** $M(C_i^T) = 2M(C_i^T)$

FIG. 1. Johnson's Algorithm.

clauses for MAX-SAT, where h is any integer larger than 0, and all clauses have weight 1:

$$\phi_h = \{(x_{3k+1} \vee x_{3k+2}), (x_{3k+1} \vee x_{3k+3}), (\bar{x}_{3k+1}) \mid 0 \leq k \leq h-1\}.$$

It is easy to verify that Johnson's Algorithm assigns $x_t =$ TRUE for all $1 \leq t \leq 3h$, and this assignment satisfies exactly $2h$ clauses $(x_{3k+1} \vee x_{3k+2}), (x_{3k+1} \vee x_{3k+3})$ for $0 \leq k \leq h-1$. On the other hand, the assignment $x_{3k+1} =$ FALSE, $x_{3k+2} = x_{3k+3} =$ TRUE for all $0 \leq k \leq h-1$ obviously satisfies all $3h$ clauses in ϕ_h.

3. THE MAIN LEMMA

In order to analyze Johnson's Algorithm, based on the weight of an optimal assignment to a given instance ϕ, we may need to "flip" a boolean variable x_t, i.e., interchange x_t and $\bar{x}_t$, in the instance ϕ. Let ϕ' be the instance obtained from ϕ by this flipping operation. At first sight, applying Johnson's Algorithm to ϕ and ϕ' should produce assignments that satisfy exactly the same set of clauses. However, the following subtle difference should be observed.

Suppose that in the tth iteration of the **for** loop in step 3 of Johnson's Algorithm when we are deciding the truth assignment for the variable x_t, we have

$$\sum_{i=1}^{r} M(C_i^T) = \sum_{i=1}^{s} M(C_i^F), \tag{1}$$

where $C_i^T, \ldots, C_r^T$ are the clauses containing x_t and $C_1^F, \ldots, C_s^F$ are the clauses containing $\bar{x}_t$.

If x_t is not flipped, then both sets $\{C_1^T, \ldots, C_r^T\}$ and $\{C_1^F, \ldots, C_s^F\}$ are the same when Johnson's Algorithm is applied to the instance ϕ and to the instance ϕ'. In

this case, Johnson's Algorithm assigns $x_t = \text{TRUE}$ for both instances. Thus, the sets of clauses satisfied in this iteration for the two instances are the same.

On the other hand, if the variable x_t is flipped, then for the instance ϕ', $\{C_1^F, ..., C_s^F\}$ is exactly the set of clauses in LEFT that contain x_t, while $\{C_1^T, ..., C_r^T\}$ is exactly the set of clauses in LEFT that contain $\bar{x}_t$. According to the algorithm, thus, when condition (1) holds, Johnson's Algorithm would assign $x_t = \text{TRUE}$ and make the clauses $C_1^T, ..., C_r^T$ satisfied for the instance ϕ and would assign $x_t = \text{TRUE}$ and make the clauses $C_1^F, ..., C_r^F$ satisfied for the instance ϕ'.

Therefore, if condition (1) happens, the assignments constructed by Johnson's Algorithm in general may satisfy different sets of clauses for the instances ϕ and ϕ'.

In order to take care of this abnormality, we will augment Johnson's Algorithm with a boolean array $b[1..n]$ such that when condition (1) occurs, the assignment to the variable x_t is determined based on the boolean value $b[t]$. The augmented boolean array $b[1..n]$ will be part of the input to the algorithm. We call such an algorithm the *augmented Johnson's Algorithm*. Our tight bound on the performance ratio is proved based on Johnson's Algorithm augmented with *an arbitrary boolean array*. Since the original Johnson's Algorithm on an instance ϕ can be precisely simulated by the augmented Johnson's Algorithm, augmented by a properly chosen boolean array on an instance ϕ' that is obtained from ϕ by possibly a number of flipping operations, the bound on the performance ratio for the augmented Johnson's Algorithm will imply the same bound on the performance ratio for the original Johnson's Algorithm. Moreover, the flipping operations can be applied now on a given instance.

The augmented Johnson's Algorithm is given formally in Fig. 2.

Augmented Johnson's Algorithm.

Input: a set of clauses $\phi = \{C_1, \ldots, C_m\}$ on $\{x_1, \ldots, x_n\}$, where the weight of the clause C_j is w_j, $1 \le j \le m$;
a boolean array $b[1..n]$

Output: a truth assignment to $\{x_1, \ldots, x_n\}$

1. **for** each clause C_j **do** $M(C_j) = w_j/2^{|C_j|}$
2. LEFT $= \{C_1, \ldots, C_m\}$;
3. **for** $t = 1$ **to** n **do**
   find all clauses $C_1^T, \ldots, C_r^T$ in LEFT that contain x_t;
   find all clauses $C_1^F, \ldots, C_s^F$ in LEFT that contain $\bar{x}_t$;
   case 1. $\sum_{i=1}^{r} M(C_i^T) > \sum_{i=1}^{s} M(C_i^F)$ **or**
   $\sum_{i=1}^{r} M(C_i^T) = \sum_{i=1}^{s} M(C_i^F)$ **and** $b[t] = \text{TRUE}$
   $x_t = \text{TRUE}$; delete $C_1^T, \ldots, C_r^T$ from LEFT;
   for $i = 1$ **to** s **do** $M(C_i^F) = 2M(C_i^F)$
   case 2. $\sum_{i=1}^{r} M(C_i^T) < \sum_{i=1}^{s} M(C_i^F)$ **or**
   $\sum_{i=1}^{r} M(C_i^T) = \sum_{i=1}^{s} M(C_i^F)$ **and** $b[t] = \text{FALSE}$
   $x_t = \text{FALSE}$; delete $C_1^F, \ldots, C_s^F$ from LEFT;
   for $i = 1$ **to** r **do** $M(C_i^T) = 2M(C_i^T)$

FIG. 2. The Augmented Johnson's Algorithm.

The original Johnson's Algorithm corresponds to the Augmented Johnson's Algorithm augmented with the boolean array $b[1..n]$ such that $b[t] = \text{TRUE}$ for all t.

In the following, we prove a key lemma for the Augmented Johnson's Algorithm. To do this, we need to introduce some terminologies and notations.

A literal is a *positive literal* if it is a boolean variable x_i for some i, and a *negative literal* if it is the negation $\bar{x}_i$ of a boolean variable.

For the rest of this section, we fix an instance $\phi = \{C_1, ..., C_m\}$ for MAX-SAT and let $b[1..n]$ be any fixed boolean array. Let r_ϕ be the size (i.e., the number of literals) of the longest clause in ϕ. Apply the Augmented Johnson's Algorithm on ϕ and $b[1..n]$. Consider a fixed moment in the execution of the Augmented Johnson's Algorithm. We say that a literal is still *active* if it has not been assigned a truth value yet. A clause C_j in ϕ is *satisfied* if at least one literal in C_j has been assigned value TRUE. A clause C_j is *killed* if all literals in C_j are assigned value FALSE. A clause C_j is *negative* if it is neither satisfied nor killed, and all active literals in C_j are negative literals.

DEFINITION. Fix a $t, 0 \leqslant t \leqslant n$, and suppose that we are at the end of the tth iteration of the **for** loop in step 3 of the Augmented Johnson's Algorithm. Let $S^{(t)}$ be the set of satisfied clauses, let $K^{(t)}$ be the set of killed clauses, and let $N_i^{(t)}$ be the set of negative clauses with exactly i active literals. For a set S of clauses in ϕ, denote by $\text{wt}(S)$ the sum of weights of the clauses in S. That is, $\text{wt}(S) = \sum_{C_j \in S} w_j$. Similarly, let $M(S)$ denote the sum of measures of all clauses in S.

LEMMA 3.1. *Let ϕ be any instance for* MAX-SAT, *and let $b[1..n]$ be any boolean array. Apply the Augmented Johnson's Algorithm on ϕ and $b[1..n]$. For each $t, 0 \leqslant t \leqslant n$, let $S^{(t)}$, $K^{(t)}$, and $N_i^{(t)}$ be the sets as given in the previous definition. Then*

$$\text{wt}(S^{(t)}) \geqslant 2\text{wt}(K^{(t)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \text{wt}(N_i^{(t)}) - A_0,$$

where $A_0 = \sum_{i=1}^{r_\phi} (1/2^{i-1}) \text{wt}(N_i^{(0)})$.

Proof. The proof proceeds by induction on t. For $t = 0$, since $S^{(0)} = K^{(0)} = \varnothing$ and $\sum_{i=1}^{r_\phi} \text{wt}(N_i^{(t)})/2^{i-1} = A_0$, the lemma is true.

Now suppose $t > 0$. We need to introduce two more notations. At the end of the tth iteration for the **for** loop in step 3 of the Augmented Johnson's Algorithm, let $P_{i,j}$ be the set of clauses that contain the positive literal x_{t+1} such that each clause in $P_{i,j}$ contains exactly i active literals, of which exactly j are positive, and let $N_{i,j}$ be the set of clauses that contain the negative literal $\bar{x}_{t+1}$ such that each clause in $N_{i,j}$ contains exactly i active literals, of which exactly j are positive. Note that according to the Augmented Johnson's Algorithm, if at this moment a clause C_h has exactly i active literals, then the value $M(C_h)$ equals exactly $w_h/2^i$.

Case 1. Suppose that the Augmented Johnson's Algorithm assigns $x_{t+1} = \text{TRUE}$. Then according to the algorithm, regardless of the value $b[t]$ we must have

$$\sum_{i=1}^{r_\phi} \sum_{j=1}^{i} M(P_{i,j}) \geqslant \sum_{i=1}^{r_\phi} \sum_{j=0}^{i-1} M(N_{i,j}).$$

This is equivalent to

$$\sum_{i=1}^{r_\phi} \frac{\sum_{j=1}^{i} \mathrm{wt}(P_{i,j})}{2^i} \geqslant \sum_{i=1}^{r_\phi} \frac{\sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j})}{2^i}. \tag{2}$$

Now we have

$$\begin{aligned}
N_1^{(t+1)} &= (N_1^{(t)} - N_{1,0}) \cup N_{2,0} \\
N_2^{(t+1)} &= (N_2^{(t)} - N_{2,0}) \cup N_{3,0} \\
&\cdots \\
N_{r_\phi - 1}^{(t+1)} &= (N_{r_\phi - 1}^{(t)} - N_{r_\phi - 1,0}) \cup N_{r_\phi,0} \\
N_{r_\phi}^{(t+1)} &= (N_{r_\phi}^{(t)} - N_{r_\phi,0}).
\end{aligned}$$

This gives us

$$\begin{aligned}
&\mathrm{wt}(N_1^{(t+1)}) + \frac{1}{2}\mathrm{wt}(N_2^{(t+1)}) + \cdots + \frac{1}{2^{r_\phi - 1}}\mathrm{wt}(N_{r_\phi}^{(t+1)}) \\
&\quad = \mathrm{wt}(N_1^{(t)}) + \frac{1}{2}\mathrm{wt}(N_2^{(t)}) + \cdots + \frac{1}{2^{r_\phi - 1}}\mathrm{wt}(N_{r_\phi}^{(t)}) \\
&\qquad - \mathrm{wt}(N_{1,0}) + \frac{1}{2}\mathrm{wt}(N_{2,0}) + \frac{1}{2^2}\mathrm{wt}(N_{3,0}) + \cdots + \frac{1}{2^{r_\phi - 1}}\mathrm{wt}(N_{r_\phi,0}) \\
&\quad = \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}}\mathrm{wt}(N_i^{(t)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}}\mathrm{wt}(N_{i,0}) - 2\mathrm{wt}(N_{1,0}).
\end{aligned} \tag{3}$$

On the other hand, we have

$$S^{(t+1)} = S^{(t)} \cup \bigcup_{i=1}^{r_\phi} \bigcup_{j=1}^{i} P_{i,j} \tag{4}$$

and

$$K^{(t+1)} = K^{(t)} \cup N_{1,0}. \tag{5}$$

Combining relations (2)–(5) and using the inductive hypothesis, we get

$$\begin{aligned}
\mathrm{wt}(S^{(t+1)}) &= \mathrm{wt}(S^{(t)}) + \sum_{i=1}^{r_\phi} \sum_{j=1}^{i} \mathrm{wt}(P_{i,j}) \\
&\geqslant 2\mathrm{wt}(K^{(t)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}}\mathrm{wt}(N_i^{(t)}) - A_0 + \sum_{i=1}^{r_\phi} \frac{\sum_{j=1}^{i} \mathrm{wt}(P_{i,j})}{2^{i-1}} \\
&\geqslant 2\mathrm{wt}(K^{(t)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}}\mathrm{wt}(N_i^{(t)}) - A_0 + \sum_{i=1}^{r_\phi} \frac{\sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j})}{2^{i-1}}
\end{aligned}$$

$$\geqslant 2[\mathrm{wt}(K^{(t)}) + \mathrm{wt}(N_{1,0})]$$

$$+ \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_i^{(t)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_{i,0}) - 2\mathrm{wt}(N_{1,0}) - A_0$$

$$= 2\mathrm{wt}(K^{(t+1)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_i^{(t+1)}) - A_0.$$

Therefore, the induction goes through in this case.

Case 2. Suppose that the Augmented Johnson's Algorithm assigns $x_{t+1} =$ FALSE. The proof for this case is similar but slightly more complicated. We will concentrate on describing the differences.

According to the Augmented Johnson's Algorithm, we have

$$\sum_{i=1}^{r_\phi} \frac{\sum_{j=1}^{i} \mathrm{wt}(P_{i,j})}{2^i} \leqslant \sum_{i=1}^{r_\phi} \frac{\sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j})}{2^i}. \tag{6}$$

Remark. This is the only place that differs in the original Johnson's Algorithm and the Augmented Johnson's Algorithm. In the original Johnson's Algorithm if the variable x_{t+1} is assigned FALSE, then we must have

$$\sum_{i=1}^{r_\phi} \frac{\sum_{j=1}^{i} \mathrm{wt}(P_{i,j})}{2^i} < \sum_{i=1}^{r_\phi} \frac{\sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j})}{2^i}.$$

Based on the relations

$$N_1^{(t+1)} = (N_1^{(t)} - N_{1,0}) \cup P_{2,1}$$
$$N_2^{(t+1)} = (N_2^{(t)} - N_{2,0}) \cup P_{3,1}$$
$$\cdots$$
$$N_{r_\phi - 1}^{(t+1)} = (N_{r_\phi - 1}^{(t)} - N_{r_\phi - 1,0}) \cup P_{r_\phi,1}$$
$$N_{r_\phi}^{(t+1)} = (N_{r_\phi}^{(t)} - N_{r_\phi,0}),$$

we get

$$\mathrm{wt}(N_1^{(t+1)}) + \frac{1}{2}\mathrm{wt}(N_2^{(t+1)}) + \cdots + \frac{1}{2^{r_\phi - 1}}\mathrm{wt}(N_{r_\phi}^{(t+1)})$$

$$= \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_i^{(t)}) + \sum_{i=2}^{r_\phi} \frac{1}{2^{i-2}} \mathrm{wt}(P_{i,1}) - \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_{i,0}). \tag{7}$$

Moreover, we have

$$S^{(t+1)} = S^{(t)} \cup \bigcup_{i=1}^{r_\phi} \bigcup_{j=0}^{i-1} N_{i,j} \tag{8}$$

and

$$K^{(t+1)} = K^{(t)} \cup P_{1,1}. \tag{9}$$

Combining relations (7) and (9) and using the inductive hypothesis,

$$\begin{aligned}
&2\mathrm{wt}(K^{(t+1)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_i^{(t+1)}) - A_0 \\
&\quad = 2\mathrm{wt}(K^{(t)}) + 2w(P_{1,1}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_i^{(t)}) \\
&\qquad + \sum_{i=2}^{r_\phi} \frac{1}{2^{i-2}} \mathrm{wt}(P_{i,1}) - \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_{i,0}) - A_0 \\
&\quad \leqslant \mathrm{wt}(S^{(t)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-2}} \mathrm{wt}(P_{i,1}) - \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_{i,0}) \\
&\quad = \mathrm{wt}(S^{(t)}) + \sum_{i=1}^{r_\phi} \sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j}) \\
&\qquad + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-2}} \mathrm{wt}(P_{i,1}) - \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_{i,0}) - \sum_{i=1}^{r_\phi} \sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j}).
\end{aligned}$$

Now according to Eq. (8),

$$\mathrm{wt}(S^{(t+1)}) = \mathrm{wt}(S^{(t)}) + \sum_{i=1}^{r_\phi} \sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j}).$$

Moreover, since

$$\begin{aligned}
&\sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_{i,0}) + \sum_{i=1}^{r_\phi} \sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j}) \\
&\qquad \geqslant \mathrm{wt}(N_{1,0}) + \mathrm{wt}(N_{1,0}) + \sum_{i=2}^{r_\phi} \sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j}) \\
&\qquad \geqslant 2\mathrm{wt}(N_{1,0}) + \sum_{i=2}^{r_\phi} \frac{\sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j})}{2^{i-2}} \\
&\qquad = \sum_{i=1}^{r_\phi} \frac{\sum_{j=0}^{i-1} \mathrm{wt}(N_{i,j})}{2^{i-2}} \\
&\qquad \geqslant \sum_{i=1}^{r_\phi} \frac{\sum_{j=1}^{i} \mathrm{wt}(P_{i,j})}{2^{i-2}} \\
&\qquad \geqslant \sum_{i=1}^{r_\phi} \frac{1}{2^{i-2}} \mathrm{wt}(P_{i,1}),
\end{aligned}$$

the third inequality above follows from relation (6), we conclude

$$2\mathrm{wt}(K^{(t+1)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \mathrm{wt}(N_i^{(t+1)}) - A_0 \leqslant \mathrm{wt}(S^{(t+1)}).$$

Thus, the lemma also holds for this case.

This completes the proof for the lemma. ∎

4. TIGHT BOUND FOR JOHNSON'S ALGORITHM

Now we are ready to prove our main theorem. Let us come back to the original Johnson's Algorithm. Lemma 3.1 on the Augmented Johnson's Algorithm enables us to analyze the original Johnson's Algorithm using the weight of an optimal assignment to a given instance of MAX-SAT in the evaluation of the performance ratio.

THEOREM 4.1. *Let ϕ be an instance to the* MAX-SAT *problem. Let* $\mathrm{wt}_{\mathrm{opt}}(\phi)$ *be the weight of an optimal solution to ϕ and let* $\mathrm{wt}_{\mathrm{tot}}(\phi)$ *be the sum of weights of all clauses in ϕ. Then the assignment to ϕ constructed by the (original) Johnson's Algorithm is at least*

$$\tfrac{1}{3}(\mathrm{wt}_{\mathrm{opt}}(\phi) + \mathrm{wt}_{\mathrm{tot}}(\phi)).$$

Proof. Suppose that τ_o is an arbitrary optimal assignment to ϕ. Now we construct another instance ϕ' for MAX-SAT as follows. Starting with ϕ, if for a boolean variable x_t, we have $\tau_o(x_t) = \mathrm{FALSE}$, then we "flip" x_t (i.e., interchange x_t and $\bar{x}_t$) in ϕ. No clause weight is changed in this process. Thus, there is a one-to-one correspondence between the set of clauses in ϕ and the set of clauses in ϕ', and the corresponding clauses have the same weight.

We also construct a boolean array $b[1..n]$ such that $b[t] = \mathrm{FALSE}$ if and only if $\tau_o(x_t) = \mathrm{FALSE}$.

It is easy to see that the weight of an optimal assignment to ϕ' is equal to the weight of an optimal assignment to ϕ. In the following, we show that the assignment constructed by the original Johnson's Algorithm on the instance ϕ and the assignment constructed by the Augmented Johnson's Algorithm on the instance ϕ' augmented by the boolean array $b[1..n]$ have the same weight.

Inductively, suppose that for the first $(t-1)$th iterations of the **for** loop in step 3, both algorithms satisfy exactly the same set of clauses. Now consider the tth iteration of the algorithms.

If x_t in ϕ is not flipped, then $b[t] = \mathrm{TRUE}$. Thus, the Augmented Johnson's Algorithm assigns $x_t = \mathrm{TRUE}$ and makes the clauses $C_1^T, ..., C_r^T$ satisfied during the tth iteration if and only if $\sum_{i=1}^{r} M(C_i^T) \geqslant \sum_{i=1}^{s} M(C_i^F)$, where $C_1^T, ..., C_r^T$ are the clauses in LEFT containing x_t and $C_1^F, ..., C_s^F$ are the clauses in LEFT containing $\bar{x}_t$. On the other hand, if x_t in ϕ is flipped, then $b[t] = \mathrm{FALSE}$, and the Augmented Johnson's Algorithm assigns $x_t = \mathrm{FALSE}$ and makes the clauses $C_1^F, ..., C_s^F$ satisfied if and only if $\sum_{i=1}^{r} M(C_i^T) \leqslant \sum_{i=1}^{s} M(C_i^F)$. Note that if x_t is not flipped, then

$\{C_1^T, ..., C_r^T\}$ is exactly the set of clauses containing x_t in the tth iteration of the original Johnson's Algorithm for the instance ϕ, while if x_t is flipped, then $\{C_1^F, ..., C_s^F\}$ is exactly the set of clauses containing x_t in the tth iteration of the original Johnson's Algorithm for the instance ϕ. Therefore, in the tth iteration, the set of the clauses satisfied by the Augmented Johnson's Algorithm on ϕ' and $b[1..n]$ corresponds exactly to the set of clauses satisfied by the original Johnson's Algorithm on ϕ. In conclusion, the assignment constructed by the original Johnson's Algorithm on the instance ϕ and the assignment constructed by the Augmented Johnson's Algorithm on the instance ϕ' and the boolean array $b[1..n]$ have the same weight.

Therefore, we only need to analyze the performance ratio of the Augmented Johnson's Algorithm on the instance ϕ' and the boolean array $b[1..n]$. Moreover, note that the assignment τ'_o for the instance ϕ' that assigns $x_t = \text{TRUE}$ for all $1 \leqslant t \leqslant n$ corresponds to the optimal assignment τ_o for the instance ϕ; thus it is an optimal assignment for the instance ϕ'.

Let $K^{(t)}$, $S^{(t)}$, and $N_i^{(t)}$ be the sets as defined in Section 3 for the Augmented Johnson's Algorithm on the instance ϕ' and the boolean array $b[1..n]$. According to Lemma 3.1, we have

$$\text{wt}(S^{(t)}) \geqslant 2\text{wt}(K^{(t)}) + \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \text{wt}(N_i^{(t)}) - A_0. \tag{10}$$

for all $0 \leqslant t \leqslant n$, where $A_0 = \sum_{i=1}^{r_\phi} (1/2^{i-1}) \text{wt}(N_i^{(0)})$.

At the end of the Augmented Johnson's Algorithm, i.e., $t = n$, $S^{(n)}$ is exactly the set of clauses satisfied by the constructed assignment and $K^{(n)}$ is exactly the set of clauses not satisfied by the constructed assignment. Moreover, $N_i^{(n)} = \varnothing$ for all i.

According to (10), we have

$$\text{wt}(S^{(n)}) \geqslant 2\text{wt}(K^{(n)}) - A_0. \tag{11}$$

Note that

$$A_0 = \sum_{i=1}^{r_\phi} \frac{1}{2^{i-1}} \text{wt}(N_i^{(0)}) \leqslant \sum_{i=1}^{r_\phi} \text{wt}(N_i^{(0)}). \tag{12}$$

Combining relations (11) and (12), we get

$$\tfrac{3}{2} \text{wt}(S^{(n)}) \geqslant \text{wt}(S^{(n)}) + \text{wt}(K^{(n)}) - \tfrac{1}{2} \sum_{i=1}^{r_\phi} \text{wt}(N_i^{(0)}). \tag{13}$$

Since $S^{(n)} \cup K^{(n)}$ is the whole set of clauses in ϕ' thus

$$\text{wt}_{\text{tot}}(\phi) = \text{wt}(S^{(n)}) + \text{wt}(K^{(n)}) \tag{14}$$

and the assignment $x_t = \text{TRUE}$ for all $1 \leqslant t \leqslant n$ is an optimal assignment to the instance ϕ', which satisfies all clauses in ϕ' except those in $N_i^{(0)}$ for $1 \leqslant i \leqslant r_\phi$; thus

$$\text{wt}_{\text{opt}}(\phi) = \text{wt}(S^{(n)}) + \text{wt}(K^{(n)}) - \sum_{i=1}^{r_\phi} \text{wt}(N_i^{(0)}). \tag{15}$$

Now combining Eq. (14) and (15) with Eq. (13), we get

$$\text{wt}(S^{(n)}) \geqslant \tfrac{1}{3}(\text{wt}_{\text{opt}}(\phi) + \text{wt}_{\text{tot}}(\phi)).$$

This proves the theorem since $\text{wt}(S^{(n)})$ is exactly the weight of the assignment constructed by the Augmented Johnson's Algorithm. ▌

COROLLARY 4.2. *The performance ratio for Johnson's Algorithm is* 2/3.

Proof. Obviously we have $\text{wt}_{\text{opt}}(\phi) \leqslant \text{wt}_{\text{tot}}(\phi)$ for any instance ϕ for MAX-SAT. Therefore, Theorem 4.1 says that the weight of the assignment constructed by Johnson's Algorithm is at least 2/3 of that of an optimal assignment. The corollary follows. ▌

We point out that in order to obtain the bound 2/3, it is necessary to use the weight of an actual optimal assignment in the above analysis. If instead we use the sum $\text{wt}_{\text{tot}}(\phi)$ of weights of all clauses in the given instance ϕ of MAX-SAT, then the weight of an optimal assignment to ϕ, and thus, the weight of an assignment to ϕ constructed by any approximation algorithm, can be as small as $\text{wt}_{\text{tot}}(\phi)/2$. For example, ϕ consists of contradicting pairs of unit clauses $\phi = \{(x_t), (\bar{x}_t) \mid 1 \leqslant t \leqslant n\}$, where all clauses have weight 1. Even under the assumption that ϕ contains no contradicting pairs of unit clauses, Lieberherr and Specker [10] have been able to demonstrate instances ϕ such that the weight of optimal assignments is bounded by $\alpha \cdot \text{wt}_{\text{tot}}(\phi)$, where α is any constant larger than $\alpha_0 = (\sqrt{5} - 1)/2 \approx 0.618$ (the reciprocal of the "golden ratio"). We have also constructed the instance without contradicting pairs of unit clauses for which Johnson's Algorithm constructs an assignment of weight bounded by $3\text{wt}_{\text{tot}}(\phi)/5$:

$$\phi_r = \{(x_{3t+1}), (x_{3t+2}), (x_{3t+3}), (\bar{x}_{3t+1} \vee \bar{x}_{3t+2}), (\bar{x}_{3t+1} \vee \bar{x}_{3t+3}) \mid 0 \leqslant t \leqslant r-1\}, \tag{16}$$

where r is an arbitrary integer and all clauses have weight 1. Moreover, we can prove (see the Appendix) that the weight of the assignment constructed by Johnson's Algorithm for instances without contradicting pairs of unit clauses is at least $3\text{wt}_{\text{tot}}(\phi)/5$. The example in (16) shows that this bound is tight.

Another remark we would like to make concerns the probabilistic method for approximation of MAX-SAT. As described by Yannakakis [13], Johnson's Algorithm can be interpreted using the probabilistic method based on uniform distribution, in which each variable is assigned value TRUE with probability 1/2. Based on the analysis of the probabilistic method, one can derive that Johnson's Algorithm has performance ratio 1/2. Yannakakis [13] has suggested further generalizations

of the approach of uniform distribution by allowing variables to be assigned value TRUE with different probability. For example, if an instance ϕ of MAX-SAT is 2-*satisfiable* (i.e., any two clauses in ϕ are simultaneously satisfiable, or equivalently, ϕ does not contain contradicting pairs of unit clauses), then we can assign TRUE value to the literals in unit clauses in ϕ with probability α_0, where $\alpha_0 = (\sqrt{5}-1)/2 \approx 0.618$, and assign TRUE value to literals not in any unit clauses in ϕ with probability 1/2. The probabilistic argument shows that, using this probability distribution, an assignment of weight at least $\alpha_0 \cdot \mathrm{wt}_{\mathrm{tot}}(\phi)$ can be constructed. Similarly, if an instance ϕ of MAX-SAT is 3-*satisfiable* (i.e., any three clauses in ϕ are simultaneously satisfiable), then we can assign the TRUE value to the literals in unit clauses in ϕ with probability 2/3, which results in an assignment of weight at least $2\mathrm{wt}_{\mathrm{tot}}(\phi)/3$.

Thus, using the probabilistic method based on nonuniform distribution seems to improve the performance ratio for approximation of MAX-SAT. In particular, by Corollary 4.2, Johnson's Algorithm has performance ratio 2/3. One may suspect that based on the nonuniform distributions suggested by Yannakakis [13], a generalization of Johnson's Algorithm should have a performance ratio better than 2/3. This is, unfortunately, not true as we will demonstrate below.

We say that a probabilistic method is *based on a simple nonuniform distribution* if it assigns the TRUE value to literals in unit clauses with probability p and it assigns the TRUE value to literals not in any unit clauses with probability 1/2, where $0 < p < 1$ is a fixed constant. Of course, the probabilistic method based on a simple nonuniform distribution is applicable only to 2-satisfiable instances of MAX-SAT. This loses no generality since the problem of approximating MAX-SAT can always be reduced to the problem of approximating 2-satisfiable instances of MAX-SAT (see the Appendix).

We have the following theorem.

THEOREM 4.3. *No probabilistic method based on a simple nonuniform distribution has a performance ratio larger than* 2/3.

Proof. Let A be a probabilistic method, based on a simple nonuniform distribution, that assigns literals in unit clauses the value TRUE with probability p and assigns literals not in any unit clauses the value TRUE with probability 1/2. Then, according to Yannakakis [13], the probabilistic method A can be implemented by Johnson's Algorithm with changes in initialization, comparison, and modifications of the measure values $M(C_j)$ for each clause C_j. We indicate these changes in Fig. 3, in which the omitted parts are identical to Johnson's Algorithm in Fig. 1. Without loss of generality, we may assume that all unit clauses in the input instance ϕ contain a positive literal (otherwise, we flip some variables). For each t, $p_t = p$ if x_t is contained in a unit clause of ϕ and $p_t = 1/2$ if x_t is not contained in any unit clause of ϕ.

Consider the following instance ϕ_h for MAX-SAT, where h is an arbitrary positive integer and each clause has weight 1. Note that the instance ϕ_h is 2-satisfiable:

$$\phi_h = \{(x_{3k+1} \vee x_{3k+2}), (x_{3k+1} \vee x_{3k+3}), (\bar{x}_{3k+1}) \mid 0 \leqslant k \leqslant h-1\}.$$

Probabilistic Method A.

.....

1. **for** each clause C_j **do** $M(C_j) = w_j(\prod_{\bar{x}_t \in C_j} p_t)(\prod_{x_t \in C_j}(1-p_t))$
2. .....
3. **for** .....

if $p_t \sum_{i=1}^{r} M(C_i^T) \geq (1-p_t) \sum_{i=1}^{s} M(C_i^F)$
then .....
for $i = 1$ **to** s **do** $M(C_i^F) = M(C_i^F)/p_t$
else .....
for $i = 1$ **to** r **do** $M(C_i^T) = M(C_i^T)/(1-p_t)$

FIG. 3. The probabilistic Method A.

We have $p_{3k+1} = p$ and $p_{3k+2} = p_{3k+3} = 1/2$ for all k. During the $(3k+1)$st iteration of the **for** loop in step 3 in Probabilistic Method A, the clauses containing x_{3k+1} are $(x_{3k+1} \vee x_{3k+2})$ and $(x_{3k+1} \vee x_{3k+3})$, each of which has measure $(1-p)/2$, and the clause containing $\bar{x}_{3k+1}$ is $(\bar{x}_{3k+1})$, which has measure p. Thus, the algorithm will assign $x_{3k+1} = \text{TRUE}$, which satisfies two but kills one of the above three clauses. In conclusion, the probabilistic method A constructs an assignment that satisfies exactly $2h$ clauses in ϕ_h. On the other hand, an optimal assignment $x_{3k+1} = \text{FALSE}$, $x_{3k+2} = x_{3k+3} = \text{TRUE}$ satisfies all $3h$ clauses in ϕ_h. ∎

In conclusion, using the probabilistic method based on simple nonuniform distribution may improve the ratio of the weight of the constructed assignment versus the value $\text{wt}_{\text{tot}}(\phi)$, but it does not improve the ratio of the weight of the constructed assignment versus the weight $\text{wt}_{\text{opt}}(\phi)$ of an optimal assignment.

We close the paper with two remarks:

The instance ϕ_h constructed in the proof of Theorem 4.3 seems to include the unique kind of obstacles for Johnson's Algorithm to overcome the 2/3 ratio bound. One reason that Johnson's Algorithm constructs an assignment of weight $2\text{wt}_{\text{opt}}(\phi_h)/3$ is that Johnson's Algorithm follows a strict order in determining the values for the boolean variables. If Johnson's Algorithm changes the order of the assignment, for example, by considering x_{3k+2} or x_{3k+3} before x_{3k+1}, then it will produce an optimal assignment for the instance ϕ_h. In particular, what is the performance ratio of the variant of Johnson's Algorithm that considers the variables in a *random* order? Is there a simple modification of Johnson's Algorithm that has a performance ratio better than 2/3?

Johnson's Algorithm has some very interesting properties that supplement other approximation algorithms for the MAX-SAT problem. For example, Johnson's Algorithm does better on long clauses than short clauses (see the discussion in Section 2) while most other approximation algorithms for MAX-SAT do better on short clauses than on long clauses (see, for example, [2–4, 12]). Theorem 4.1 shows another such interesting property for Johnson's Algorithm, in the following sense. Some recent research (e.g., [8, 11]) shows that it seems easier to deal with instances that are satisfiable than instances that are highly unsatisfiable (i.e., instances whose

wt_{opt} value is much smaller than the wt_{tot} value). On the other hand, Theorem 4.1 shows that Johnson's Algorithm does better on highly unsatisfiable instances. For example, for some satisfiable instances (such as the instances given in Theorem 4.3), Johnson's Algorithm has a performance ratio 2/3, while for instances with ratio wt_{opt}/wt_{tot} arbitrarily close to $\alpha_0 = (\sqrt{5}-1)/2$,[2] Johnson's Algorithm has a performance ratio arbitrarily close to 7/8. This property may be useful in the further study of approximation algorithms for the MAX-SAT problem.

APPENDIX

Let ϕ be an instance of MAX-SAT. The *total weight* of ϕ, denoted by $wt_{tot}(\phi)$, is the sum of the weights of all clauses in ϕ. A pair of unit clauses (x_t) and $(\bar{x}_t)$ is called a *contradicting pair of unit clauses*. If an instance ϕ of MAX-SAT purely consists of contradicting pairs of unit clauses of the same weight, then the weight of an optimal assignment to ϕ is exactly half of the total weight $wt_{tot}(\phi)$ of ϕ.

It is interesting to know what is the ratio of the weight of an optimal assignment to the total weight $wt_{tot}(\phi)$ of ϕ. In fact, most early analysis on approximation algorithms for MAX-SAT is on derivations of this ratio [7, 10]. Johnson's original analysis on his algorithm [7] shows that this ratio is at least 1/2. The example of contradicting pairs of unit clauses above shows that this bound is tight.

However, it seems in most practice that the case of contradicting pairs of unit clauses can be eliminated. According to Lieberherr and Specker [10], an instance of MAX-SAT is 2-*satisfiable* if it contains no contradicting pairs of unit clauses. Suppose that ϕ is a general instance for MAX-SAT in which there is a contradicting pair of unit clauses (x_t) and $(\bar{x}_t)$ of weight w_t and $\bar{w}_t$, respectively. Without loss of generality, assume $w_t \geq \bar{w}_t$. Then we can delete $(\bar{x}_t)$ from ϕ and decrease the weight of (x_t) to $w_t - \bar{w}_t$. Repeatedly doing this will eliminate all contradicting pairs of unit clauses. Let the resulting 2-satisfiable instance be ϕ'. It is clear that for any assignment τ to $\{x_1, ..., x_n\}$, the ratio $w(\phi, \tau)/w(\phi, \tau_{opt})$ is at least as large as the ratio $w(\phi', \tau)/w(\phi', \tau'_{opt})$, where τ_{opt} is an optimal assignment to ϕ and τ'_{opt} is an optimal assignment to ϕ'. Therefore, an approximation algorithm for 2-satisfiable instances for MAX-SAT implies an approximation algorithm of at least equally good performance ratio for general MAX-SAT problem.

Lieberherr and Specker [10] proved that for a 2-satisfiable instance ϕ of MAX-SAT, the weight of an optimal assignment to ϕ is at least $\alpha_0 = (\sqrt{5}-1)/2 \approx 0.618$ of the total weight $wt_{tot}(\phi)$ of ϕ. Moreover, they show that there are 2-satisfiable instances of MAX-SAT for which this ratio is arbitrarily close to α_0. They have also given a polynomial time algorithm that constructs an assignment whose weight is at least $\alpha_0 \cdot wt_{tot}(\phi)$. Their algorithm has been greatly simplified by Yannakakis [13], using the probabilistic method based on a nonuniform distribution.

In the following, we prove that Johnson's Algorithm, which corresponds to the approach of the probabilistic method based on uniform distribution, constructs an

[2] According to Lieberherr and Specker [10], such instances without containing contradicting pairs of unit clauses always exist.

assignment whose weight is at least $3\text{wt}_{\text{tot}}(\phi)/5$ on 2-satisfiable instances ϕ of MAX-SAT. This is an improvement on the original analysis given by Johnson [7].

THEOREM 5.1. *For any 2-satisfiable instance ϕ of* MAX-SAT, *the assignment constructed by Johnson's Algorithm has weight at least* $3\text{wt}_{\text{tot}}(\phi)/5$.

Proof. Suppose $\phi = \{C_j, ..., C_m\}$. Let U be the set of unit clauses in ϕ, and let $R = \phi - U$. Since ϕ is 2-satisfiable, the set U contains no contradicting pairs of unit clauses.

There are two cases. As before, for a set S of clauses, we denote by $\text{wt}(S)$ the sum of weights of the clauses in S.

Case 1. $\text{wt}(U) \leqslant 3\text{wt}(R)/2$. Then initially, the sum of the measures of all clauses in the set LEFT is equal to

$$\begin{aligned}
\sum_{i=1}^{m} M(C_i) &= \sum_{i=1}^{m} \frac{w_i}{2^{|C_i|}} \\
&= \sum_{C_i \in U} \frac{w_i}{2^{|C_i|}} + \sum_{C_i \in R} \frac{w_i}{2^{|C_i|}} \\
&\leqslant \frac{\text{wt}(U)}{2} + \frac{\text{wt}(R)}{4} \\
&= \frac{2\text{wt}(U)}{5} + \frac{\text{wt}(U)}{10} + \frac{\text{wt}(R)}{4} \\
&\leqslant \frac{2\text{wt}(U)}{5} + \frac{1}{10}\left(\frac{3\text{wt}(R)}{2}\right) + \frac{\text{wt}(R)}{4} \\
&= \frac{2\text{wt}(U)}{5} + \frac{2\text{wt}(R)}{5} \\
&= \frac{2}{5}(\text{wt}(U) + \text{wt}(R)) \\
&= \frac{2\text{wt}_{\text{tot}}(\phi)}{5}.
\end{aligned}$$

According to Johnson's Algorithm, the measure $M(\text{LEFT})$ is never increased and at the end of the algorithm, each clause C_i in LEFT has measure exactly $M(C_i) = w_i$. Since at the end of the algorithm the set LEFT contains exactly those clauses that are not satisfied by the constructed assignment τ, we conclude that the assignment τ constructed by Johnson's Algorithm has weight at least $\text{wt}_{\text{tot}}(\phi) - 2\text{wt}_{\text{tot}}(\phi)/5 = 3\text{wt}_{\text{tot}}(\phi)/5$.

Case 2. $\text{wt}(U) > 3\text{wt}(R)/2$. For each t, let U_t be the subset of unit clauses in U that contains the variable x_t or its negation $\bar{x}_t$ (of course, since ϕ is 2-satisfiable, U_t either consists of clauses of form (x_t) or it consists of clauses of form $(\bar{x}_t)$, but not both).

CLAIM. *The weight of the clauses deleted from the set* LEFT *in the tth iteration for the* **for** *loop in step* 3 *of Johnson's Algorithm is at least* $\mathrm{wt}(U_t)$.

Proof of the Claim. By the algorithm, no steps before the tth iteration for the **for** loop in step 3 of Johnson's Algorithm would delete a unit clause in U_t from the set LEFT. Therefore, at the beginning of the tth iteration, all unit clauses in U_t are contained in the set LEFT. Without loss of generality, suppose that the set U_t consists of unit clauses of form (x_t) (the case that U_t consists of unit clauses of form $(\bar{x}_t)$ can be proved similarly). As described in the algorithm, let $C_1^T, ..., C_r^T$ be the clauses containing x_t and let $C_1^F, ..., C_s^F$ be the clauses containing $\bar{x}_t$. Note that the set U_t is a subset of the set $\{C_1^T, ..., C_r^T\}$.

If Johnson's Algorithm assigns $x_t = \text{TRUE}$, then all clauses $C_1^T, ..., C_r^T$ are deleted from LEFT in this iteration. In particular, all unit clauses in U_t are deleted from LEFT. Thus, the weight of the clauses deleted from LEFT in this iteration is at least $\mathrm{wt}(U_t)$.

If Johnson's Algorithm assigns $x_t = \text{FALSE}$, then we must have $\sum_{i=1}^{r} M(C_i^T) < \sum_{i=1}^{s} M(C_i^F)$. Now since for each clause C_i in U_t we have $M(C_i) = \mathrm{wt}(C_i)/2$ and $U_t \subseteq \{C_1^T, ..., C_r^T\}$, we have

$$\tfrac{1}{2}\,\mathrm{wt}(U_t) \leqslant \sum_{i=1}^{r} M(C_i^T) < \sum_{i=1}^{s} M(C_i^F).$$

Moreover, since $M(C_i^F) \leqslant \mathrm{wt}(C_i^F)/2$ for all $i = 1, ..., s$,

$$\mathrm{wt}(U_t) \leqslant \sum_{i=1}^{s} \mathrm{wt}(C_i^F).$$

This proves the claim in this case since $\sum_{i=1}^{s} \mathrm{wt}(C_i^F)$ is the sum of the weights of the clauses deleted from LEFT in this iteration.

This completes the proof for the claim. ▮

Since at the end of the algorithm, the clauses deleted from the set LEFT are exactly those clauses that are satisfied by the constructed assignment, the above claim shows that the weight of the assignment constructed by Johnson's Algorithm is at least $\sum_{i=1}^{n} \mathrm{wt}(U_t) = \mathrm{wt}(U)$. Now the theorem is proved for Case 2 because

$$\begin{aligned}
\mathrm{wt}(U) &= \tfrac{3}{5}\mathrm{wt}(U) + \tfrac{2}{5}\mathrm{wt}(U) \\
&> \tfrac{3}{5}\mathrm{wt}(U) + \tfrac{2}{5}(\tfrac{3}{2}\mathrm{wt}(R)) \\
&= \tfrac{3}{5}[\,\mathrm{wt}(U) + \mathrm{wt}(R)\,] \\
&= \tfrac{3}{5}\mathrm{wt}_{\mathrm{tot}}(\phi)
\end{aligned}$$

This completes the proof of the theorem. ▮

ACKNOWLEDGMENTS

We thank Judy Goldsmith and Peter van Emde Boas who read an early version of this paper and provided a number of helpful suggestions. We would also like to thank David Johnson, Richard Beigel, Jose Balcazar, and Osamu Watanabe for their valuable discussions and comments.

REFERENCES

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and hardness of approximation problems, *in* "Proc. 33rd IEEE Symposium on the Foundation of Computer Science, 1992," pp. 14–23.
2. U. Feige and M. Goemans, Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT, *in* "Proc. 3rd Israel Symposium of Theory of Computing and Systems, 1995," pp. 182–189.
3. M. X. Goemans and D. P. Williamson, New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem, *SIAM J. Discrete Math.* **7** (1994), 656–666.
4. M. X. Goemans and D. P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, *J. ACM* **42** (1995), 1115–1145.
5. P. Hansen and B. Jaumard, Algorithms for the maximum satisfiability problem, *Computing* **44** (1990), 279–303.
6. J. Hastad, Some optimal inapproximability results, *in* "Proc. 28th Annual ACM Symposium on Theory of Computing, 1997," pp. 1–10.
7. D. S. Johnson, Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences* **9** (1974), 256–278.
8. H. Karloff and U. Zwick, A 7/8-approximation algorithm for MAX-SAT?, *in* "Proc. 38th IEEE Symposium on the Foundation of Computer Science, 1997," pp. 406–415.
9. S. Khanna, R. Motwani, M. Sudan, and U. V. Vazirani, On syntactic versus computational views of approximability, *in* "Proc. 35th IEEE Symposium on the Foundation of Computer Science, 1994," pp. 819–830.
10. K. J. Lieberherr and E. Specker, Complexity of partial satisfaction, *Journal of ACM* **28** (1981), 411–421.
11. L. Trevisan, Approximating satisfiable satisfiability problems, *in* "Proc. 5th Annual European Symposium on Algorithms," Lecture Notes in Computer Science, Vol. 1284, pp. 472–485, 1997.
12. L. Trevisian, G. B. Sorkin, M. Sudan, and D. P. Williamson, Gadgets, approximation, and linear programming, *in* "Proc. 37th IEEE Symposium on the Foundation of Computer Science, 1996," pp. 617–626.
13. M. Yannakakis, On the approximation of maximum satisfiability, *J. Algorithms* **17** (1994), 475–502.