Simpler 3/4-Approximation Algorithms for MAX SAT

Anke van Zuylen

Max Planck Institute for Informatics,
66123, Saarbücken, Germany
anke@mpi-inf.mpg.de

Abstract. We consider the recent randomized $\frac{3}{4}$-algorithm for MAX SAT of Poloczek and Schnitger. We give a much simpler set of probabilities for setting the variables to true or false, which achieve the same expected performance guarantee. Our algorithm suggests a conceptually simple way to get a deterministic algorithm: rather than comparing to an unknown optimal solution, we instead compare the algorithm's output to the optimal solution of an LP relaxation. This gives rise to a new LP rounding algorithm, which also achieves a performance guarantee of $\frac{3}{4}$.

1 Introduction

The maximum satisfiability problem (MAX SAT) is a fundamental NP-hard problem. Given a set of variables, $x_1, \ldots, x_n$, and a set of weighted disjunctive clauses $C_1, \ldots, C_m$ of literals, where a literal is either a variable x_i or its negation $\bar{x}_i$, we want to find a truth assignment to the variables that maximizes the weight of the satisfied clauses.

Let W be the weight of all clauses. A simple approximation algorithm for MAX SAT sets each variable to true with probability $\frac{1}{2}$; by linearity of expectation, the expected weight of the satisfied clauses is at least $\frac{1}{2}W$, and, hence, this is a randomized $\frac{1}{2}$-approximation algorithm. This algorithm can be derandomized using the method of conditional expectation, which gives rise to the following algorithm: Consider the variables one at a time. For a clause C_j with weight w_j that is not yet satisfied by the assignment of the variables considered so far, let c_j be the number of variables occurring in C_j for which the truth assignment has not yet determined. Define the modified weight of C_j as $\mu(C_j) = w_j \left(\frac{1}{2}\right)^{c_j}$. Note that this is the expected weight of clause C_j that is *not* satisfied, if the remaining variables are set to true with probability $\frac{1}{2}$. We now set the next variable x_i to true if the modified weight of the clauses containing x_i is greater than or equal to the modified weight of the clauses containing $\bar{x}_i$, and to false otherwise. This deterministic algorithm is due to Johnson [6] and is known as Johnson's algorithm. The fact that it can be interpreted as the derandomization of the randomized algorithm that sets each variable to true with probability $\frac{1}{2}$ was noted by Yannakakis [9]. Chen, Friesen and Zhang [2] showed that the approximation ratio of the derandomized algorithm is in fact $\frac{2}{3}$; see also Engebretsen [4] for a simplified analysis.

Better approximation algorithms are known, both for the general case and for certain special cases, but until recently, all of these used the optimal solution to a linear program or semidefinite program. See for example Yannakakis [9] and Goemans and Williamson [5]. The best known approximation algorithm is due to Avidor, Berkovitch and Zwick [1] and achieves a guarantee of 0.7968.

Very recently, Poloczek and Schnitger [8] gave the first approximation algorithm with performance guarantee $\frac{3}{4}$ that is purely combinatorial. They define a randomized variant of Johnson's algorithm, which sets variable x_i to true or false with probability proportional to the modified weight of the clauses containing x_i and $\bar{x}_i$ respectively. They then show how to slightly modify these probabilities so that the expected weight of the clauses satisfied by the algorithm is at least $\frac{3}{4}$ of the weight of the optimal solution.

The probabilities determined by the algorithm are rather complicated, and they depend on previous decisions by the algorithm. Derandomization of this algorithm seems therefore highly non-trivial. In fact, Poloczek [7] shows that, under certain assumptions, no deterministic variant of the algorithm of Poloczek and Schnitger [8] can achieve the same guarantee: Poloczek shows that no deterministic *adaptive priority algorithm* can achieve an approximation ratio of $\frac{3}{4}$. Priority algorithms are a formalization of greedy algorithms, and need to make an irrevocable decision when a data item is revealed. In the setting considered by Poloczek, a data item is the name of a variable, say x; the set of clauses that contain the variable x; and for each such clause, the data item contains the sign of x in the clause, the weight, and the other variables appearing in the clause (but not whether these appear negated or not). Based on this information, the algorithm has to decide whether to set x to true or false. In an adaptive priority algorithm, the algorithm may adaptively change the order in which it considers the data items, but when the data item corresponding to variable x is revealed, it still needs to irrevocably determine the value of x.

It may however still be the case that a deterministic variant, which is not an (adaptive) priority algorithm, achieves a guarantee of $\frac{3}{4}$. In this paper, we give a simple expression for the probability with which to set the next variable to true or false, which gives the same performance guarantee as the algorithm of Poloczek and Schnitger [8]. Our probabilities are not necessarily the same as those given by Poloczek and Schnitger [8], but they do satisfy the inequalities that are required for their analysis (and, by extension, our version of the analysis) to hold. Although the expression of the probabilities is simple, the probabilities still depend on the past decisions made by the algorithm, and, hence, the question whether this algorithm can be derandomized remains non-trivial. However, if we allow our algorithm to use linear programming, derandomization becomes relatively straightforward. Our second result is therefore a new deterministic LP rounding algorithm, which achieves an approximation ratio of $\frac{3}{4}$.

The remainder of this paper is structured as follows: we begin in Section 2 by introducing the notion of a potential function, which is implicitly used in the analysis of Poloczek and Schnitger. We summarize some key ideas of their analysis in terms of the potential function. We then give a new randomized algorithm which has very simple probabilities of setting the next variable to true or false, and we prove that it satisfies the conditions derived in Section 2. Our new algorithm suggests a conceptually simple way to get a deterministic algorithm: rather than comparing to an unknown optimal solution, we can instead compare the algorithm's output to the optimal solution of an LP relaxation. This gives rise to the new rounding algorithm described in Section 4.

2 Analysis with a Potential Function

Let the input be a set of variables $x_1, \ldots, x_n$, and a set of disjunctive clauses $C_1, \ldots, C_m$ with weights $w_1, \ldots, w_m \geq 0$, where the literals in the clauses are variables or their negation. Let $W = \sum_{j=1}^m w_j$. The algorithms we consider iteratively determine the value (either 1 (true) or 0 (false)) to which we set variables $x_1, \ldots, x_n$, and our aim is to prove that the expected weight of the satisfied clauses is at least $\frac{3}{4}$ times the weight of the optimal assignment.

For a given index i, let $SAT(i)$ be the weight of the clauses that are satisfied by the algorithm's values for $x_1, \ldots, x_i$, and let $UNSAT(i)$ be the weight of the clauses which contain only $x_1, \ldots, x_i$, or their negations, and that are not satisfied by the chosen values. Suppose we have already determined the assignment for $x_1, \ldots, x_{i-1}$, and the algorithm now fixes the assignment for x_i. Then $SAT(i) - SAT(i-1)$ is the weight of the clauses that become satisfied by the algorithm's assignment for x_i (and that were not already satisfied by the assignment for $x_1, \ldots, x_{i-1}$), and $UNSAT(i) - UNSAT(i-1)$ is the weight of the clauses that become unsatisfiable by the assignment to x_i. If for all i, we could determine an assignment such that

$$(SAT(i) - SAT(i-1)) - 3(UNSAT(i) - UNSAT(i-1)) \geq 0, \tag{1}$$

then this would imply a $\frac{3}{4}$-approximation algorithm: Note that $\sum_{i=1}^m \Big((SAT(i) - SAT(i-1)) - 3(UNSAT(i) - UNSAT(i-1))\Big) = SAT(n) - 3UNSAT(n)$, where $SAT(n)$ is the weight of the clauses satisfied by the algorithm's solution, and $UNSAT(n)$ is the weight of the clauses that the algorithm does not satisfy, i.e. $UNSAT(N) = W - SAT(n)$. So we would get that $SAT(n) - 3(W - SAT(n)) \geq 0$, or $SAT(n) \geq \frac{3}{4}W$.

There does not always exist an assignment to i such that (1) holds, but note that we only need the inequality to hold, summed over all i. We therefore introduce the idea of a potential function $\varPhi$. This idea is implicit in the analysis of Poloczek and Schnitger [8]. One can think of $\varPhi$ as a "bank account" for the algorithm. In the course of the algorithm, we may add or remove some amount to the potential function to allow us to satisfy the inequality $(SAT(i) - SAT(i-1) - 3(UNSAT(i) - UNSAT(i-1)) \geq 0$.

More precisely, let $\varPhi(i)$ be the value of the potential function after determining the truth assignment of variable x_i (where $\varPhi(0)$ is the potential function at the start of the algorithm). Let OPT be the weight of the satisfied clauses in an optimal solution. The potential function Φ, combined with the algorithm, must satisfy the following three properties:

(i) $\Phi(0) \leq 3(W - OPT)$;
(ii) $\Phi(n) \geq 0$;
(iii) For each variable x_i, the algorithm (randomly) determines a truth assignment to x_i such that

$$E\big[SAT(i) - SAT(i-1) - 3(UNSAT(i) - UNSAT(i-1))\big] \\ \geq E\big[\Phi(i) - \Phi(i-1)\big].$$

If we have a potential function Φ with an algorithm that together satisfy these three properties, then $E\big[SAT(n) - 3(W - SAT(n))\big] \geq \Phi(n) - \Phi(0) \geq -\Phi(0) \geq 3(OPT - W)$, which gives $E\big[SAT(n)\big] \geq \frac{3}{4}OPT$.

We remark that the potential functions in this paper will in fact have $\Phi(0) = 2(W - OPT)$, which is less than what is allowed by (i), but that increasing it to $3(W - OPT)$ does not help in our analysis.

2.1 Poloczek and Schnitger's Potential Function

Poloczek and Schnitger [8] do not explicitly define the idea of a potential function, but their analysis implicitly uses the following potential function. Let $x_i = x_i^*$ for $i = 1, \dots, n$ be an optimal solution, where each x_i^* is either 1 (true) or 0 (false). Let x_i^a be the truth assignment to x_i by the algorithm's solution, if x_i has already been determined. Let "time i" be the time when the algorithm has determined the truth assignment to $x_1, \dots, x_i$. We'll say a clause is alive at time i if it contains some literal from $\{x_{i+1}, \dots, x_n\}$, and it is not (yet) satisfied by setting $x_1 = x_1^a, \dots, x_i = x_i^a$. We'll say a live clause is contradictory at time i if it is not satisfied by setting $x_1 = x_1^a, \dots, x_i = x_i^a$ according to the algorithm's solution, and $x_{i+1} = x_{i+1}^*, \dots, x_n^*$. We will make sure that at any point in time $\Phi(i)$ is (at least) twice the weight of the clauses that are alive and contradictory at time i. Note that we thus have the $\Phi(0) = 2(W - OPT)$.

Let $W_i, \overline{W}_i$ be the weight of the clauses that are alive at time $i-1$ and contain x_i and $\bar{x}_i$ respectively, but do not contain $x_{i+1}, \dots, x_n$. Let $F_i, \overline{F}_i$ be the weight of the remaining clauses that are alive at time $i-1$ and that contain x_i and $\bar{x}_i$ respectively. We note that $W_i, \overline{W}_i, F_i, \overline{F}_i$ are random variables that are determined by the algorithm's decisions for $x_1, \dots, x_{i-1}$. Let 1_A be the indicator function that is 1 if A holds and 0 otherwise. A contradictory clause at time $i-1$ is not contradictory at time i when it is no longer alive at time i because either it becomes satisfied or it has no literals in $x_{i+1}, \dots, x_n$. We can thus lower bound the weight of the contradictory clauses that are alive at time $i-1$ and not alive at time i by $W_i \mathbf{1}_{\{x_i^*=0\}} + \overline{W}_i \mathbf{1}_{\{x_i^*=1\}}$.

On the other hand, the only clauses that can become contradictory when going from time $i-1$ to time i are clauses that are alive at time $i-1$ and at time i, that contain either x_i or $\bar{x}_i$, and for which the algorithm's setting for x_i is not the same as the setting in the optimal solution. Hence we can upper bound the weight of the clauses that become contradictory by $\mathbf{1}_{\{x_i^*=0\}}\mathbf{1}_{\{x_i=1\}}\overline{F}_i + \mathbf{1}_{\{x_i^*=1\}}\mathbf{1}_{\{x_i=0\}}F_i$.

We thus have that

$$\Phi(i) - \Phi(i-1) \leq 2\left(-W_i + \mathbf{1}_{\{x_i=1\}}\overline{F}_i\right)\mathbf{1}_{\{x_i^*=0\}} + 2\left(-\overline{W}_i + \mathbf{1}_{\{x_i=0\}}F_i\right)\mathbf{1}_{\{x_i^*=1\}}.$$

We note that the expression $E\left[c' - c\right]$ in the analysis of Poloczek and Schnitger [8] is equal to $E\left[\Phi(i) - \Phi(i-1)\right]$, and that a similar inequality is given in their Lemma 2.2.

On the other hand,

$$\begin{aligned} &SAT(i) - SAT(i-1) - 3(UNSAT(i) - UNSAT(i-1)) \\ &\quad = \mathbf{1}_{\{x_i=1\}}(W_i + F_i - 3\overline{W}_i) + \mathbf{1}_{\{x_i=0\}}(\overline{W}_i + \overline{F}_i - 3W_i) \end{aligned}$$

Let p be the probability that the algorithm set x_i to 1. Then, in order to satisfy property (iii), we need:

$$\begin{aligned} &p(W_i + F_i - 3\overline{W}_i) + (1-p)(\overline{W}_i + \overline{F}_i - 3W_i) \\ &\quad - 2\left(-W_i + p\overline{F}_i\right)\mathbf{1}_{\{x_i^*=0\}} - 2\left(-\overline{W}_i + (1-p)F_i\right)\mathbf{1}_{\{x_i^*=1\}} \geq 0. \end{aligned} \tag{2}$$

3 A New Combinatorial Randomized Algorithm

In the following lemma and its proof, we will define $\frac{c}{0} = \infty$ if $c \geq 0$ and $\frac{c}{0} = -\infty$ if $c < 0$.

Lemma 1. *Consider the randomized algorithm that iteratively determines the assignment to $x_1, \ldots, x_n$ as follows: Given the assignment of $x_1, \ldots, x_{i-1}$, let $W_i, \overline{W}_i$ be the weight of the clauses that are not yet satisfied and contain x_i and $\bar{x}_i$ respectively, but do not contain $x_{i+1}, \ldots, x_n$. Let $F_i, \overline{F}_i$ be the weight of the remaining clauses that are not yet satisfied and that contain x_i and $\bar{x}_i$ respectively. Let $\alpha = \frac{W_i + F_i - \overline{W}_i}{F_i + \overline{F}_i}$, and let x_i be set to 1 with probability*

$$p = \begin{cases} 0 & \text{if } \alpha \leq 0, \\ \alpha & \text{if } \alpha \in (0,1), \\ 1 & \text{if } \alpha \geq 1. \end{cases}$$

Then the expected weight of the clauses satisfied by the algorithm is at least $\frac{3}{4}OPT$.

Proof. We will show that inequality (2) holds, by giving a lower bound B on

$$2\left(W_i - p\overline{F}_i\right)\mathbf{1}_{\{x_i^*=0\}} + 2\left(\overline{W}_i - (1-p)F_i\right)\mathbf{1}_{\{x_i^*=1\}}, \tag{3}$$

in the case when $\alpha \leq 0, \alpha \geq 1$ and $\alpha \in (0,1)$, and showing that for each of these cases, $p(W_i + F_i - 3\overline{W}_i) + (1-p)(\overline{W}_i + \overline{F}_i - 3W_i) + B \geq 0$.

We first consider the case when $\alpha \leq 0$, i.e., when $W_i + F_i \leq \overline{W}_i$. Then $p = 0$ and $W_i - p\overline{F}_i = W_i \leq \overline{W}_i - F_i = \overline{W}_i - (1-p)F_i$, so (3) is at least $2W_i - p\overline{F}_i = 2W_i$. Therefore, the lefthand side of (2) is at least $\overline{W}_i + \overline{F}_i - 3W_i + 2W_i = \overline{W}_i + \overline{F}_i - W_i$.

Note that this cannot be negative, since combined with $W_i + F_i - \overline{W}_i \leq 0$ this would give $F_i + \overline{F}_i < 0$.

If $\alpha \geq 1$, then $W_i + F_i - \overline{W}_i \geq F_i + \overline{F}_i$, i.e., $W_i - \overline{F}_i \geq \overline{W}_i$. Since $p = 1$, (3) is at least $2\overline{W}_i$. So the lefthand side of (2) is at least $W_i + F_i - \overline{W}_i$ and this cannot be negative, as this would imply $F_i + \overline{F}_i < 0$ by the fact that $\overline{W}_i + \overline{F}_i - W_i \leq 0$.

Finally, if $\alpha \in (0, 1)$, then we have that $p = \alpha$ and, by definition of α, $-W_i + p\overline{F}_i = -\overline{W}_i + (1-p)F_i$. Hence, the quantity in (3) does not depend on whether x_i^* is zero or one, since it is either $2W_i - 2p\overline{F}_i$ or $2\overline{W}_i - 2(1-p)F_i$ which are equal. Thus (3) is also equal to $p(2W_i - 2p\overline{F}_i) + (1-p)(2\overline{W}_i - 2(1-p)F_i)$. Plugging this into (2) gives

$$\begin{aligned}
&p(W_i + F_i - 3\overline{W}_i) + (1-p)(\overline{W}_i + \overline{F}_i - 3W_i) + \\
&2p(W_i - p\overline{F}_i) + 2(1-p)(\overline{W}_i - (1-p)F_i) \\
=&(6p-3)W_i - (6p-3)\overline{W}_i + (5p - 2p^2 - 2)F_i - (2p^2 + p - 1)\overline{F}_i \\
=&(2p-1)(3W_i + (1-p)F_i - 3\overline{W}_i - p\overline{F}_i + F_i - \overline{F}_i) \\
=&(2p-1)(2W_i + F_i - 2\overline{W}_i - \overline{F}_i),
\end{aligned}$$

where the first two equalities follow by rearranging terms, and the last equality uses the fact that $W_i + (1-p)F_i = \overline{W}_i + p\overline{F}_i$. Now, either $p \geq \frac{1}{2}$ in which case $2p - 1 \geq 0$ and $2W_i + F_i \geq 2W_i + 2(1-p)F_i = 2\overline{W}_i + 2p\overline{F}_i \geq 2\overline{W}_i + \overline{F}_i$, so $2W_i + F_i - 2\overline{W}_i - \overline{F}_i \geq 0$. Otherwise, $p < \frac{1}{2}$, in which case $2p - 1 < 0$ and also $2\overline{W}_i + \overline{F}_i > 2W_i + F_i$. Hence in either case the inequality (2) holds. □

Remark 2. Let α be defined as in Lemma 1. If we let

$$p = \begin{cases} 0 & \text{if } \alpha \leq \frac{1}{3}, \\ \alpha & \text{if } \alpha \in (\frac{1}{3}, \frac{2}{3}), \\ 1 & \text{if } \alpha \geq \frac{2}{3}, \end{cases}$$

then the expected weight of the clauses satisfied by the algorithm is also at least $\frac{3}{4}OPT$.

Proof. We only need to verify that inequality (2) holds for this choice of p, if $\alpha \in (0, \frac{1}{3}]$ or if $\alpha \in [\frac{2}{3}, 1)$. If $\alpha \in (0, \frac{1}{3}]$ then $p = 0$, and we note that $W_i - p\overline{F}_i \geq W_i - \alpha\overline{F}_i = \overline{W}_i - (1-\alpha)F_i \geq \overline{W}_i - (1-p)F_i$. Hence (3) is at least $2\overline{W}_i - 2(1-p)F_i$, and therefore the lefthand side of (2) is at least

$$\overline{W}_i + \overline{F}_i - 3W_i + 2\overline{W}_i - 2F_i.$$

Now, note that $3\overline{W}_i + \overline{F}_i \geq 3\overline{W}_i + 3\alpha\overline{F}_i = 3W_i + 3(1-\alpha)F_i \geq 3W_i + 2F_i$ hence (2) holds.

Similarly, if $\frac{2}{3} \leq \alpha < 1$, then setting $p = 1$ will give that (3) is at least $2W_i - 2\overline{F}_i$, and hence the lefthand side of (2) is at least $3W_i + F_i - 3\overline{W}_i - 2\overline{F}_i$ and this is nonnegative by the fact that $\alpha \geq \frac{2}{3}$. □

Note that one way to view an iteration of the algorithm is as a 2-player-zero-sum game. We get to choose p, our probability of playing $x_i = 1$, and the opponent gets to choose q, which is the optimum's probability of playing $x_i = 1$. We are trying to maximize

$$p(W_i+F_i-3\overline{W}_i)+(1-p)(\overline{W}_i+\overline{F}_i-3W_i)+2(1-q)(W_i-p\overline{F}_i)+2q(\overline{W}_i-(1-p)F_i)$$

and the opponent is trying to minimize this quantity. We show that the value of this game is nonnegative by showing that there exists a randomized strategy p such that for any strategy q the outcome is nonnegative. When $W_i + F_i < \overline{W}_i$ then $\overline{W}_i - (1-p)F_i \geq W_i - p\overline{F}_i$ for any $p \geq 0$, and hence $q = 0$ is an optimal strategy for the opponent. It is easily verified that, given $q = 0$, $p = 0$ is an optimal strategy for the algorithm. Similarly, when $\overline{W}_i + \overline{F}_i < W_i$, then $q = 1, p = 1$ are a pair of optimal strategies. In all other cases, the proof of Lemma 1 shows that $q = (1-p)$ is an optimal strategy for the opponent, given our strategy.

Note that we thus achieve an expected non-negative value even if we allow fractional values $q \in [0,1]$. Hence, our algorithm achieves at least $\frac{3}{4}$ of the weight of any fractional assignment as well; something that was recently shown by Poloczek [7] for the algorithm in [8].

In fact, allowing the opponent to use fractional assignments makes it easy to derandomize the algorithm: we can compute the optimum's probability q of playing $x_i = 1$ by solving a linear program. Given this information, there exists a pure strategy p that achieves a nonnegative value. This gives rise to the deterministic algorithm in the next section.

4 A New Deterministic LP Rounding Algorithm

Let q_i be the variable in the linear program corresponding to the decision $x_i = 1$, and let z_j be a variable corresponding to the j-th clause, and let w_j be the weight of the j-th clause. We let P_j be the indices of the literals i such that x_i appears in the clause, and N_j the indices of the literals such that $\bar{x}_i$ appears in the clause. Then the linear programming relaxation is:

$$\begin{aligned}
&\min \sum_j w_j z_j \\
&\text{s.t.} \sum_{i\in P_j} q_i + \sum_{i \in N_j} (1-q_i) \geq z_j && \text{for } j = 1,\ldots,m \\
&0 \leq z_j \leq 1 && \text{for } j = 1,\ldots,m \\
&0 \leq q_i \leq 1 && \text{for } i = 1,\ldots,n
\end{aligned}$$

For ease of notation, we again define $\frac{c}{0} = \infty$ if $c \geq 0$ and $\frac{c}{0} = -\infty$ if $c < 0$.

Lemma 3. *Let q^* be an optimal LP solution, with objective value OPT_{LP}. Using the parameters defined in Lemma 1, let α again be defined as $\frac{W_i+F_i-\overline{W}_i}{F_i+\overline{F}_i}$, and let x_i be set to 1 with probability*

$$p = \begin{cases} 0 & \textit{if } \alpha \leq 0, \textit{ or if } \alpha \in (0,1) \textit{ and } q_i^* < (1-\alpha)/2\alpha \\ 1 & \textit{if } \alpha \geq 1, \textit{ or if } \alpha \in (0,1) \textit{ and } q_i^* \geq (1-\alpha)/2\alpha. \end{cases}$$

Then the weight of the clauses satisfied by the algorithm is at least $\frac{3}{4}OPT_{LP}$.

Proof. We'll again say a clause is alive at time i if it contains some literal from $\{x_{i+1}, \ldots, x_n\}$, and it is not satisfied yet by the algorithm's solution on $x_1, \ldots, x_i$. We will say the contradictory weight of a live clause j at time i is $w_j(1 - \min\left\{1, \sum_{i' \in P_j : i' \geq i} q^*_{i'} + \sum_{i' \in N_j : i' \geq i}(1 - q^*_{i'})\right\})$.

We define the potential function $\Phi(i)$ to be twice the contradictory weight of the live clauses. Initially, $\Phi(0) = 2(W - OPT_{LP}) \leq 2(W - OPT)$, since all clauses are alive at time 0, and the contradictory weight of clause j at time 0 is $w_j(1 - z_j)$.

We now consider $\Phi(i) - \Phi(i-1)$. Note that $\Phi(i)$ does not contain any contradictory weight for clauses that are alive at time $i - 1$ that are not alive at time i. Hence Φ drops by at least $2W_i(1 - q^*_i) + 2\overline{W}_i q^*_i$. On the other hand, the contradictory weight for any clause that is still alive at time i will increase only if the clause contains x_i or $\bar{x}_i$ (i.e. the clause is contained in F_i or $\overline{F}_i$ respectively) and it is not satisfied by the algorithm's setting (i.e. if we set $x_i = 0$ or $x_i = 1$ respectively). The increase in the contradictory weight is thus at most $2q^*_i F_i$ if we set $x_i = 0$, and $2(1 - q^*_i)\overline{F}_i$ if we set $x_i = 1$.

Hence we get that

$$\Phi(i) - \Phi(i-1) \leq 2\left(-W_i + \mathbf{1}_{\{x_i=1\}}\overline{F}_i\right)(1 - q^*_i) + 2\left(-\overline{W}_i + \mathbf{1}_{\{x_i=0\}}F_i\right)q^*_i.$$

At time n, there are no live clauses, and hence the contradictory weight of the live clauses is zero, or, $\Phi(n) \geq 0$.

As before,

$$\begin{aligned} &SAT(i) - SAT(i-1) - 3(UNSAT(i) - UNSAT(i-1)) \\ &\quad = \mathbf{1}_{\{x_i=1\}}(W_i + F_i - 3\overline{W}_i) + \mathbf{1}_{\{x_i=0\}}(\overline{W}_i + \overline{F}_i - 3W_i) \end{aligned}$$

Let p be the probability with which we set x_i to 1 (which is 1 if $\alpha \geq 1$ or if $\alpha \in (0, 1)$ and $q^*_i \geq (1 - \alpha)/(2\alpha)$ and 0 otherwise). Then, we need to show that p satisfies

$$\begin{aligned} &p(W_i + F_i - 3\overline{W}_i) + (1-p)(\overline{W}_i + \overline{F}_i - 3W_i) \\ &\quad - 2\left(-W_i + p\overline{F}_i\right)(1 - q^*_i) - 2\left(-\overline{W}_i + (1-p)F_i\right)q^*_i \geq 0. \qquad (4) \end{aligned}$$

This is the same as (2), except that we replaced $\mathbf{1}_{\{x^*_i=1\}}$ by q^*_i and $\mathbf{1}_{\{x^*_i=0\}}$ by $(1 - q^*_i)$. Note that the proof of Lemma 1 shows that if $\alpha \leq 0$ or $\alpha \geq 1$, then (4) holds for our choice of p, for any $q^*_i \in [0, 1]$. Hence, we only need to check the case when $\alpha \in (0, 1)$.

If we set $p = 0$ then the lefthand side of (4) becomes

$$\begin{aligned} &\overline{W}_i + \overline{F}_i - 3W_i + 2W_i(1 - q^*_i) + 2\overline{W}_i q^*_i - 2F_i q^*_i \\ &\quad = (1 + 2q^*_i)\left(\overline{W}_i + \frac{1}{1 + 2q^*_i}\overline{F}_i - W_i - \frac{2q^*_i}{1 + 2q^*_i}F_i\right). \end{aligned}$$

To see that this is non-negative, note that, since $p = 0$, and $\alpha \in (0,1)$, we have that $q_i^* < \frac{1-\alpha}{2\alpha}$. Therefore, $\frac{1}{1+2q_i^*} > \alpha$, and $\frac{2q_i^*}{1+2q_i^*} < 1-\alpha$. So, (4) is at least $(1+2q_i^*)\left(\overline{W}_i + \alpha\overline{F}_i - W_i - \alpha F_i\right)$. Finally, note that $\overline{W}_i + \alpha\overline{F}_i - W_i - (1-\alpha)F_i = 0$, by the definition of α.

Similary, if we set $p = 1$ then the lefthand side of (4) becomes

$$\begin{aligned} &W_i + F_i - 3\overline{W}_i + 2W_i(1-q_i^*) + 2\overline{W}_i q_i^* - 2\overline{F}_i(1-q_i^*) \\ &= (3-2q_i^*)\left(W_i + \frac{1}{3-2q_i^*}F_i - \overline{W}_i - \frac{2-2q_i^*}{3-2q_i^*}\overline{F}_i\right). \end{aligned}$$

We claim that for any $\alpha \in (0,1)$

$$\frac{2-3\alpha}{2-2\alpha} \le \frac{1-\alpha}{2\alpha}.$$

This can be seen by noting that $\frac{(2\alpha-1)^2}{\alpha(1-\alpha)} \ge 0$, and

$$\frac{(2\alpha-1)^2}{\alpha(1-\alpha)} = \frac{4\alpha^2-4\alpha+1}{\alpha(1-\alpha)} = -\frac{2\alpha-3\alpha^2}{\alpha(1-\alpha)} + \frac{\alpha^2-2\alpha+1}{\alpha(1-\alpha)} = -\frac{2-3\alpha}{1-\alpha} + \frac{1-\alpha}{\alpha}.$$

Hence, since $p = 1$ implies that $q_i^* \ge \frac{1-\alpha}{2\alpha}$, we also have $q_i^* \ge \frac{2-3\alpha}{2-2\alpha}$. Therefore, $\frac{1}{3-2q_i^*} \ge 1-\alpha$. So, we get that

$$\begin{aligned} &(3-2q_i^*)\left(W_i + \frac{1}{3-2q_i^*}F_i - \overline{W}_i - \frac{2-2q_i^*}{3-2q_i^*}\overline{F}_i\right) \\ &\ge (3-2q_i^*)\left(W_i + \alpha F_i - \overline{W}_i - (1-\alpha)\overline{F}_i\right) \\ &\ge 0. \end{aligned}$$

where the final inequality follows from the fact that $3 - 2q_i^* \ge 1$ and $W_i + \alpha F_i - \overline{W}_i - (1-\alpha)\overline{F}_i = 0$. □

5 Conclusion and Future Directions

The question remains whether there exists a deterministic algorithm that achieves an approximation ratio of $\frac{3}{4}$, which does not use sophisticated techniques such as linear programming. Poloczek and Schnitger [8] gave the first randomized algorithm that achieves this, and our simplified analysis makes it easier to see the need for randomization in their algorithm to "foil an adversarial optimum". We also show that it is possible to derandomize (our version of) their algorithm if one has an optimal solution to a linear programming relaxation. The upcoming paper of Poloczek [7] shows that no adaptive priority algorithm can achieve a guarantee of $\frac{3}{4}$, but this does not completely exclude the existence of a deterministic combinatorial $\frac{3}{4}$-approximation algorithm. For instance, an algorithm that looks at all data items and then chooses the next variable to be determined is not an adaptive priority algorithm, and the upper bound of Poloczek [7] does not apply. Moreover, there seems to be some evidence that carefully choosing the next variable to be determined could lead to improved results by a recent result of Costello, Shapira and Tetali [3]: They showed that Johnson's algorithm has a guarantee strictly better than $\frac{2}{3}$ if the variables are considered in a random order, whereas the best possible guarantee is $\frac{2}{3}$ if the variables are considered in a fixed order.

Acknowledgements. The author thanks David Williamson for pointing her to the question whether a deterministic variant of the algorithm of Poloczek and Schnitger exists.

References

1. Avidor, A., Berkovitch, I., Zwick, U.: Improved Approximation Algorithms for MAX NAE-SAT and MAX SAT. In: Erlebach, T., Persinao, G. (eds.) WAOA 2005. LNCS, vol. 3879, pp. 27–40. Springer, Heidelberg (2006)
2. Chen, J., Friesen, D.K., Zheng, H.: Tight bound on Johnson's algorithm for maximum satisfiability. J. Comput. Syst. Sci. 58, 622–640 (1999)
3. Costello, K.P., Shapira, A., Tetali, P.: Randomized greedy: new variants of some classic approximation algorithms. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 647–655. SIAM (2011)
4. Engebretsen, L.: Simplified tight analysis of Johnson's algorithm. Inf. Process. Lett. 92(4), 207–210 (2004)
5. Goemans, M.X., Williamson, D.P.: New $\frac{3}{4}$ -approximation algorithms for the maximum satisfiability problem. SIAM J. Discrete Math. 7(4), 656–666 (1994)
6. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. System Sci. 9, 256–278 (1974); Fifth Annual ACM Symposium on the Theory of Computing, Austin, Tex. (1973)
7. Poloczek, M.: Bounds on Greedy Algorithms for MAX SAT. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 37–48. Springer, Heidelberg (2011)
8. Poloczek, M., Schnitger, G.: Randomized variants of Johnson's algorithm for MAX SAT. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 656–663. SIAM (2011)
9. Yannakakis, M.: On the approximation of maximum satisfiability. Journal of Algorithms 17, 475–502 (1994)