Bounds on Greedy Algorithms for MAX SAT *†

Matthias Poloczek

Institute of Computer Science, University of Frankfurt, Frankfurt, Germany.
Email: matthias@thi.cs.uni-frankfurt.de

Abstract

We study adaptive priority algorithms for MAX SAT and show that no such deterministic algorithm can reach approximation ratio $\frac{3}{4}$, assuming an appropriate model of data items. As a consequence we obtain that the Slack–Algorithm of [13] cannot be derandomized. Moreover, we present a significantly simpler version of the Slack–Algorithm and also simplify its analysis. Additionally, we show that the algorithm achieves a ratio of $\frac{3}{4}$ even if we compare its score with the optimal *fractional* score.

1 Introduction

In the maximum satisfiability problem (MAX SAT) we are given a collection of clauses and their (nonnegative) weights. Our goal is to find an assignment that satisfies clauses of maximum total weight.

The currently best approximation ratio of 0.797 for MAX SAT is achieved by an algorithm due to Avidor, Berkovitch and Zwick [3] that uses semidefinite programming. Moreover, they give an additional algorithm with a conjectured performance of 0.843 (see their paper for further details).

The first greedy algorithm for MAX SAT is due to Johnson [12] and is usually referred to as *Johnson's algorithm*. Chen, Friesen and Zheng [7] showed that Johnson's algorithm guarantees a $\frac{2}{3}$ approximation for any variable ordering and asked whether a random variable ordering gives a better approximation. Recently, Costello, Shapira, and Tetali [8] gave a positive answer by proving that the approximation ratio is actually improved to $\frac{2}{3} + c$ for some constant $c > 0$. However, Poloczek and Schnitger [13] showed that the approximation ratio can be as bad as $2\sqrt{15} - 7 \approx 0.746 < \frac{3}{4}$. Instead they proposed the Slack–Algorithm [13], a greedy algorithm that achieves an expected approximation ratio of $\frac{3}{4}$ without sophisticated techniques such as linear programming. This result even holds if variables are given in a worst case fashion, i.e. the Slack–Algorithm can be applied in an online scenario.

To study the limits of greedy algorithms for MAX SAT, we employ the concept of priority algorithms, a model for *greedy–like* algorithms introduced in a seminal paper by Borodin, Nielsen and Rackoff [6]. The model was extended to algorithms for graph problems in [9, 5]. Angelopoulos and Borodin [2] studied randomized priority algorithms for facility location and makespan scheduling. MAX SAT was first examined by Alekhnovich et al. [1] in the more powerful model of priority branching–trees. Independently of our work, Yung [16] gave a $\frac{5}{6}$ inapproximability result for adaptive priority algorithms.

*partially supported by DFG SCHN 503/5-1

†This paper is to appear in the Proceedings of ESA 2011. The original publication is available at www.springer.com

1.1 Priority Algorithms for MAX SAT. The framework of priority algorithms covers a broad variety of greedy–like algorithms. The concept crucially relies on the notion of *data items* from which input instances are built. Our data items contain the name of a variable, say x, to be processed and a list of clauses x appears in. For each clause c the following information is revealed:

- the sign of variable x in c,
- the weight of c,
- a list of the names of still unfixed variables appearing in c. No sign is revealed.

(Identical clauses are merged by adding their weights.)

Thus, we restrict access to information that, at least at first sight, is helpful only in revealing global structure of the formula.

A *fixed* priority algorithm specifies an ordering of all possible data items in advance and receives in each step the currently smallest data item of the actual input. In our situation, upon receiving the data item of variable x the algorithm is forced to fix x irrevocably. An *adaptive* priority algorithm may submit a new ordering in each step and otherwise behaves identically.

The variable order is an influential parameter for greedy algorithms. A randomized ordering was shown to improve the approximation ratio of Johnson's algorithm by some additive constant [8]. Moreover, for every 2CNF formulae there exists a variable ordering such that Johnson's algorithm determines an optimal solution [13] – and in fact our lower bounds utilize clauses of length at most two. An adaptive priority algorithm may reorder data items at will, using all the knowledge about existing and non–existing data items collected so far.

Observe that priority algorithms are not resource bounded. Hence any lower bound utilizes "the nature and the inherent limitations of the algorithmic paradigm, rather than resource constraints" [2].

Johnson's algorithm as well as the Slack–Algorithm can be implemented with our data type, even assuming a worst case ordering. However, no deterministic greedy algorithm with approximation ratio $\frac{3}{4}$ is known. For this data type, is randomization more powerful than determinism? We show in Sect. 2:

THEOREM 1.1. *For any $\varepsilon > 0$, no* adaptive *algorithm can approximate MAX SAT within $\frac{\sqrt{33}+3}{12} + \varepsilon$ (using our data type). Thus, the Slack–Algorithm cannot be derandomized.*

We compare our work to Yung [16]: Utilizing a stronger data type that reveals all signs of neighboring variables, Yung considers the version of MAX SAT where each clause contains exactly two literals and shows that no adaptive algorithm can obtain an approximation ratio better than $\frac{5}{6}$. Note that Johnson's algorithm guarantees a $\frac{3}{4}$ approximation in this case.

The result of Alekhnovich et al. [1] (for priority branching trees, using the stronger data type) implies a lower bound of $\frac{21}{22} + \varepsilon$ for *fixed* priority algorithms that must stick with an a priori ordering of potential data items.

1.2 The Slack–Algorithm The Slack–Algorithm works in an online scenario where variables are revealed one by one together with the clauses they appear in. The algorithm has to decide on the spot: assume that the variable x is to be fixed. Let fanin (fanout) be the weight of all clauses of length at least two that contain the literal x (resp. $\overline{x}$). For the weights $w_x, w_{\overline{x}}$ of the unit clauses $x, \overline{x}$ and $\Delta = \text{fanin} + 2w_x + \text{fanout} + 2w_{\overline{x}}$ we define

$$q_0 := \text{prob}[x = 0] = \frac{2w_{\overline{x}} + \text{fanout}}{\Delta} \quad \text{and} \quad q_1 := \text{prob}[x = 1] = \frac{2w_x + \text{fanin}}{\Delta}$$

as the canonical assignment probabilities. (Notice that we double the weight of unit clauses but keep the weight of longer clauses unchanged.) However these probabilities do not yield an expected $\frac{3}{4}$ approximation if the slack of the decision is small compared to the weights of the unit clauses [13]. In this case the Slack–Algorithm modifies the assignment probabilities suitably.

Since carefully adjusted assignment probabilities are essential for this algorithm, is it possible that the approximation ratio can be improved by some elaborate fine-tuning of its probabilities?

THEOREM 1.2. *For any $\varepsilon > 0$, no randomized priority algorithm achieves a $\frac{3}{4} + \varepsilon$ approximation for Online MAX SAT whp (using our data type). Hence, the Slack–Algorithm is an optimal online algorithm.*

In this proceedings Azar, Gamzu and Roth [4] present a randomized $\frac{2}{3}$ approximation algorithm for submodular MAX SAT that also works in an online fashion. They prove their algorithm optimal by showing that for Online MAX SAT no algorithm can achieve a $\frac{2}{3} + \varepsilon$ approximation for any $\varepsilon > 0$, using a different data type. Note that this result does not contradict Theorem 1.2, since their data items do not reveal the length of a clause and thus the Slack–Algorithm cannot be applied.

In Sect. 3 we present a significantly simpler version of the Slack–Algorithm and also simplify its analysis. A preview of the refined algorithm is given below. Moreover, we show that the Slack–Algorithm

---

procedure SLACK–ALGORITHM(Formula ϕ) ▷ the formula ϕ has variable set V
  for all $x \in V$ **do** ▷ process variables in the given order
    $q_0 \leftarrow \frac{2w_{\overline{x}}+\text{fanout}}{\Delta}$, $q_1 \leftarrow \frac{2w_x+\text{fanin}}{\Delta}$
    if $q_0 \leq q_1$ **then**
      $\varepsilon^* \leftarrow$ the smallest $\varepsilon \geq 0$ with
      $(q_1 - q_0) \cdot \Delta\,((q_1 - q_0) + 2\varepsilon) - (q_1 - q_0)(w_x + w_{\overline{x}}) + \varepsilon(\text{fanin} + \text{fanout}) \geq 0$
      $p_0 \leftarrow q_0 - \varepsilon^*$, $p_1 \leftarrow q_1 + \varepsilon^*$
    else
      $\varepsilon^* \leftarrow$ the smallest $\varepsilon \geq 0$ with
      $(q_0 - q_1) \cdot \Delta\,((q_0 - q_1) + 2\varepsilon) - (q_0 - q_1)(w_x + w_{\overline{x}}) + \varepsilon(\text{fanin} + \text{fanout}) \geq 0$
      $p_0 \leftarrow q_0 + \varepsilon^*$, $p_1 \leftarrow q_1 - \varepsilon^*$
    assign $x = 0$ with probability p_0 and $x = 1$ with probability p_1

Figure 1: The Slack–Algorithm

satisfies clauses with an expected weight of at least $\frac{3}{4}$ of the score of an optimal *fractional* assignment. The score of the fractional optimum can be larger by a factor of $\frac{4}{3}$ than the integral optimum – as for the clauses $(x \vee y), (\overline{x} \vee y), (x \vee \overline{y}), (\overline{x} \vee \overline{y})$ – and this bound is tight [11].

Independently of our work, van Zuylen [14] gave an alternative randomized approximation algorithm that also achieves $\frac{3}{4}$ of the objective value of the LP relaxation. The new algorithm can be implemented using our data type and processes variables in an arbitrary order, hence the optimality of its assignment probabilities is implied by Theorem 1.2. Moreover, van Zuylen proposed a novel deterministic rounding scheme for the LP relaxation that gives a performance guarantee of $\frac{3}{4}$.

We also give an improved bound on Johnson's algorithm that relates to the optimal *fractional* score and thereby demonstrate that our technique is also useful for the analysis of other greedy algorithms. Johnson's algorithm satisfies clauses with a total weight of at least $\frac{\text{Opt}+W}{3} \geq \frac{2}{3}\text{Opt}$. The proof appears in Sect. C in the appendix.

2 Limits of Priority Algorithms for MAX SAT

In the next section we investigate the optimality of the Slack Algorithm for Online MAX SAT, before we give a bound on adaptive priority algorithms in Sect. 2.2.

2.1 Optimal Bounds for Online MAX SAT In Online MAX SAT the adversary reveals data items successively. The algorithm must fix the corresponding variable instantly and irrevocably. We do not impose restrictions on the decision process: the algorithm may use all information deducible from previous data items and may decide probabilistically.

In [13] the Slack–Algorithm was shown $\frac{3}{4}$ approximative under these conditions. Can one come up with a better algorithm?

Proof of Theorem 1.2. Let X_b and Y_b for $b \in \{0,1\}$ denote sets of n variables each. The clause set is constructed as follows: For each variable pair in $X_b \times Y_b$ there is an equivalence and for each pair in $X_b \times Y_{1-b}$ an inequivalence. It is crucial that the algorithm cannot distinguish both constraint types.

The bound follows by an application of Yao's principle [15]. Since we are dealing with an online setting, the adversary selects a *random* permutation π on the data items of $X_0 \cup X_1$. When given the data items according to π, the (deterministic) algorithm cannot decide whether the current variable belongs to X_0 or X_1, since the sets of clauses look identical for all variables in $X_0 \cup X_1$ according to our type of data items. Hence the algorithm ends up with at least $2n^2 - o(n^2)$ violated (in-)equivalences whp. The bound on the approximation ratio follows, since $2n^2 - o(n^2)$ clauses are not satisfied out of $8n^2$ obtained by the optimum. □

Remark. We point out that all variables in $X_0 \cup X_1$ must be fixed before the first y–variable is decided. If the algorithm was allowed to fix a single y–variable in advance, it could easily come up with an optimal assignment.

Note that randomized online algorithms are strictly more powerful than deterministic ones: assume that the adversary presents the clauses $(\overline{x} \vee y), (x \vee \overline{y})$ for variable x. If the algorithm fixes $x = 1$, the formula is completed by adding unit clause $(\overline{y})$ and (y) otherwise. Thus, no deterministic online algorithm is better than $\frac{2}{3}$ approximative – and Johnson's algorithm is optimal in its class. Observe that the (randomized) Slack–Algorithm achieves approximation ratio $\frac{3}{4}$.

2.2 Adaptive Priority Algorithms for MAX SAT In this section we study adaptive priority algorithms that decide deterministically. In particular, an adaptive priority algorithm may choose the order in which variables are to be processed by utilizing the previously obtained information on the structure of the formula.

In the lower bound we study the Adaptive Priority Game which is played between an adversary and the (adaptive priority) algorithm.

1. The game starts with the adversary announcing the number n of variables as well as their names.
2. In each round the algorithm submits an ordering on the data items and the adversary picks a smallest data item d according to this ordering. In particular, no unseen data item less than d appears in the input.

Using our data type, we show that the adversary is able to force any algorithm into an approximation ratio of at most $\frac{\sqrt{33}+3}{12} + o(1) \approx 0.729 < \frac{3}{4}$.

Outline of the Proof of Theorem 1.1. The adversary constructs a 2CNF formula ϕ with n variables. At any time there is an equivalence or an inequivalence of weight 1 for every pair of still unfixed variables. The adversary utilizes that our data items do not allow the algorithm to distinguish equivalences from inequivalences. Thus, data items differ only in the name of the respective variable and in the weights of their two unit clauses; the initial weights of unit clauses will always belong to the set $\{0, 1, 2, \ldots, G\}$.

The game is divided into two phases. Phase (I) lasts for $G < n$ steps, where G is determined later. We set a trap for the algorithm by introducing contradictions for the variables of phase (II) in each round.

In round $k \in \{1, 2, \ldots, G\}$ the adversary allows data items for the remaining variables only if the weights of both unit clause are at least $k-1$ and at most G. In particular, the combinations $(G, 0)$, $(0, G)$ are feasible at the beginning of round 1. A fact that will be used for the variables of phase (II).

Phase (II) consists of $(n-G)$ rounds. Here we make sure that the algorithm does not recover. At the beginning of phase (II), in round $G+1$, the adversary allows only variables with weight combination (G, G) for unit clauses. (It turns out that the algorithm can be forced to "increase" the initial weight combination $(G, 0)$ or $(0, G)$ to (G, G).) From now on, the adversary observes two consecutive steps of the algorithm and then adapts the formula ϕ appropriately. W.l.o.g. assume that $(n-G)$ is even. In round $G+2m$ the combinations $(G+m, G+m)$ and in round $G+2m+1$ the combinations $(G+m+1, G+m), (G+m, G+m+1)$ are allowed, where $m \in \{0, 1, \ldots, \frac{n-G}{2} - 1\}$ holds. Observe that, if the adversary is actually able to achieve this goal, the algorithm has absolutely no advantage whatsoever in determining the "optimal" value of the current variable.

Without knowing ϕ exactly, we can give an upper bound on the total weight satisfied by the algorithm. The algorithm may always satisfy the unit clause of larger weight and exactly one clause of each equivalence, resp. inequivalence. Thus, the weight of satisfied clauses of length two equals

$$\sum_{k=1}^{n} n - k = \sum_{k=1}^{n-1} k = \frac{n^2 - n}{2}. \tag{2.1}$$

The second clause of an (in-)equivalence loses one literal and hence appears again as a unit clause. It will be counted only if the algorithm satisfies the unit clause at some later time.

The weight of unit clauses satisfied in phase (I) is at most G^2, since G is the maximal weight for a unit clause. In phase (II) units of weight at most

$$\begin{aligned} \sum_{m=0}^{\frac{n-G}{2}-1} (G+m) + (G+m+1) &= (2G+1) \cdot \frac{n-G}{2} + \frac{2}{2} \cdot \frac{n-G}{2} \cdot \left(\frac{n-G}{2} - 1 \right) \\ &= \frac{n^2}{4} + \frac{1}{2} nG - \frac{3}{4} G^2 \end{aligned}$$

are satisfied. Thus, the total weight of satisfied clauses is at most

$$\begin{aligned} \text{Sat} &= \frac{n^2 - n}{2} + G^2 + \frac{n^2}{4} + \frac{1}{2} nG - \frac{3}{4} G^2 \\ &= \frac{3}{4} n^2 + \frac{1}{2} nG + \frac{1}{4} G^2 - \frac{n}{2}. \end{aligned} \tag{2.2}$$

We have to show that the adversary is able to construct an almost satisfiable formula.

Proof. We begin by describing the adversary in phase (I). Assume that the algorithm has already fixed variables in Z_b to b and that it processes variable x in round k. If it sets x to 0, the adversary introduces equivalences between x and all variables in Z_0 and inequivalences between x and all variables in Z_1. These insertions are consistent with the data items presented in the previous rounds, since the algorithm cannot distinguish between equivalences and inequivalences. The insertions are also consistent with the current data item of x, where falsified (in-)equivalences appear again as unit clauses. Thus, the weight of unit clause (x) is unchanged, whereas the weight of $(\overline{x})$ increases by $k-1$. The adversary may drop any data item. Therefore, it can enforce that the weight of (x) is at least $k-1$, but at most G, and additionally guarantee that the weight of $(\overline{x})$ is bounded by G. If x is set to 1, the argumentation is analogous.

The assignment of the adversary is obtained by flipping the assignment of the algorithm. As a consequence, the adversary also satisfies all (in-)equivalences between variables fixed in phase (I).

Now we deal with the variables of phase (II). How does the adversary justify that the weights of unit clauses at the beginning of phase (II), i.e. in round $G+1$, always equal G? The adversary now drops any data item with an initial weight combination different from $(G,0),(0,G)$. For any variable with weight combination $(G,0)$ for (x) and $(\overline{x})$ resp. the adversary inserts equivalences with all variables in Z_0 and inequivalences with all variables in Z_1. Observe that the adversary never drops such a variable in phase (I), since its weight combination equals $(G,k-1)$ in round k. In particular, the final weight after phase (I) is G for both unit clauses. As before, the case of weights 0 for (x) and G for $(\overline{x})$ is treated analogously.

During phase (II) the adversary observes two consecutive assignments x,y made by the algorithm. Let us assume inductively, that the adversary only allows variables with weight combination $(G+m,G+m)$ at the beginning of round $G+2m$. Assume w.l.o.g. that the algorithm fixes variable x to 1.

Case 1: Variable y has weight combination $(G+m+1,G+m)$ and the algorithm fixes variable y to 0. The adversary sets both variables to 1.

Observe that y has unit weights $(G+m,G+m)$ at the beginning of round $G+2m$ and the weight of unit clause (y) increases by 1 according to the case assumption. The adversary inserts an equivalence between x and y. Hence, y indeed has weight combination $(G+m+1,G+m)$ at the beginning of round $G+2m+1$. Moreover, the adversary inserts equivalences between x,y and all variables of phase (II) that it set to 1 and inequivalences between x,y and all variables of phase (II) that it set to 0.

Contrary to the adversary the algorithm treated x and y differently. Any untouched variable z is later connected with x and y via equivalences only or inequivalences only. Hence both unit weights of z increase by 1 and we can conclude the inductive argument.

Case 2: Variable y has weight combination $(G+m+1,G+m)$ and the algorithm fixes variable y to 1. The adversary sets x to 1 and y to 0.

Observe that y has unit weights $(G+m,G+m)$ at the beginning of round $G+2m$ and the weight of unit clause (y) increases by 1 according to the case assumption. Again the adversary inserts an equivalence between x and y, but in this case satisfies only one out of two clauses. Hence, y indeed has weight combination $(G+m+1,G+m)$ at the beginning of round $G+2m+1$. This time, the adversary inserts equivalences between $x,\overline{y}$ and all variables of phase (II) that it set to 1 and inequivalences between $x,\overline{y}$ and all variables of phase (II) that it set to 0.

Contrary to the adversary the algorithm treated x and y identically. Any untouched variable z is later connected with x and $\overline{y}$ via equivalences only or inequivalences only. Hence both unit weights of z increase by 1. And we can conclude the inductive argument in this case as well.

The cases that variable y has weight combination $(G+m,G+m+1)$ in round $G+2m+1$ is treated analogously.

What is the total weight satisfied by the assignment of the adversary? Only at most $n-G$ (in-)equivalences of phase (II) are falsified and hence the adversary satisfies at least $\binom{n}{2}-(n-G)$ (in-)equivalences, i.e. at least $n\cdot(n-1)-(n-G)$ 2-clauses of weight 1.

For every variable of phase (II) the unit clause of initial weight G is satisfied: the adversary satisfies units of phase (II) variables with a combined weight of $(n-G)\cdot G$.

Now consider the variable fixed in round k of phase (I). Remember that for a variable x set to 0 by the algorithm, the adversary inserts equivalences between x and all variables in Z_0 as well as inequivalences between x and all variables in Z_1. Thus, only the weight of $(\overline{x})$ has been increased by the assignments the algorithm made so far in phase (I). Moreover, remember that the adversary drops all variables with a unit weight smaller than $k-1$ at the beginning of round k, thus the unit clause (x) has initial weight at least $k-1$. We obtained the assignment of the adversary by flipping the assignment of

the algorithm and hence the adversary satisfies (x). The case that the algorithm sets variable x to 1 is treated analogously. Thus, the weight of satisfied unit clauses for phase (I) variables is at least $\sum_{k=1}^{G} k-1 = \sum_{k=1}^{G-1} k = \frac{G\cdot(G-1)}{2}$.

The assignment presented by the adversary satisfies clauses of total weight at least

$$\begin{aligned} & n\cdot(n-1) - (n-G) + (n-G)\cdot G + \frac{G\cdot(G-1)}{2} \\ = \quad & n^2 + nG - \frac{1}{2}G^2 - 2n + \frac{1}{2}G. \end{aligned} \tag{2.3}$$

Combining (2.2) and (2.3) and then choosing $G := \alpha \cdot n$ yields

$$\frac{\text{Sat}}{\text{Opt}} \leq \frac{\frac{3}{4}n^2 + \frac{1}{2}nG + \frac{1}{4}G^2 - \frac{n}{2}}{n^2 + nG - \frac{1}{2}G^2 - 2n + \frac{1}{2}G} = \frac{\frac{3}{4} + \frac{1}{2}\alpha + \frac{1}{4}\alpha^2 - o(1)}{1 + \alpha - \frac{1}{2}\alpha^2 - o(1)}$$

as an upper bound on the approximation ratio. A simple calculation shows that the ratio is minimized for $\alpha = \frac{\sqrt{33}-5}{4}$ and that the approximation ratio converges to at most $\frac{\sqrt{33}+3}{12}$. □

3 A Refined Analysis of the Slack–Algorithm

In this section we present a refinement of the Slack–Algorithm [13]. In what follows we assume that a CNF formula ϕ as well as weights w_k for clauses $k \in \phi$ are given. Let $L_+(k)$ be the set of positive and $L_-(k)$ be the set of negative literals of clause k. If π is a *fractional* assignment, then

$$\pi(k) = \sum_{l \in L_+(k)} \pi_l + \sum_{l \in L_-(k)} (1 - \pi_{\overline{l}})$$

is the score of π on k and $\sum_{k\in\phi} w_k \cdot \min\{1, \pi(k)\}$ is the score of π on formula ϕ. We call a fractional assignment optimal for ϕ iff its score is optimal among fractional assignments and denote its score by Opt.

THEOREM 3.1. *Let W be the combined weight of all clauses with optimal (fractional) score* Opt. *Then*

$$\mathbb{E}[\text{Sat}] \geq \frac{2\text{Opt} + W}{4} \geq \frac{3}{4}\text{Opt},$$

where $\mathbb{E}[\text{Sat}]$ *is the expected clause weight satisfied by the Slack–Algorithm.*

Since Opt is at least as large as the score of an integral assignment, the approximation ratio of the Slack–Algorithm is at least $\frac{3}{4}$. Also, better approximation ratios are achievable for a highly contradictory formula, i.e., if Opt is small in comparison to the total weight W.

The algorithm processes the variables in arbitrary order and decides instantly. Remember that if the variable x is to be decided, we denote by fanin (fanout) the weight of all clauses of length at least two that contain the literal x (resp. $\overline{x}$). For the weights $w_x, w_{\overline{x}}$ of the unit clauses $x, \overline{x}$ and $\Delta = \text{fanin} + 2w_x + \text{fanout} + 2w_{\overline{x}}$ we define

$$q_0 := \text{prob}[x = 0] = \frac{2w_{\overline{x}} + \text{fanout}}{\Delta} \quad \text{and} \quad q_1 := \text{prob}[x = 1] = \frac{2w_x + \text{fanin}}{\Delta}$$

as the canonical assignment probabilities.

An Overview of the Analysis. We aim for an approximation ratio of $\frac{2\text{Opt}+W}{4}$, where Opt is the weight of all clauses satisfied by some fixed fractional optimal assignment π and W is the total clause weight. Goemans and Williamson [11] showed that their LP based algorithm satisfies each clause k with probability at least $\frac{3}{4}\pi(k)$. The Slack–Algorithm, however, cannot give such a guarantee for individual clauses. For an example consider the following formula $(x \vee y), (\overline{x})$ where the first clause has a weight of w and the unit clause of 1. The Slack–Algorithm satisfies the unit clause with probability $\frac{2}{2+w}$ only, whereas its fractional score is 1. Our analysis follows the different approach of [13].

Let "Sat" and "Unsat" be the random variables indicating the total weight of satisfied and falsified clauses after fixing *all* variables. Then the algorithm achieves the desired performance if $\mathbb{E}[\text{Sat}] \geq \frac{2\text{Opt}+W}{4}$ or equivalently if $\mathbb{E}[\text{Sat}] + 3\mathbb{E}[\text{Sat}] - 2\text{Opt} - W \geq 0$. But

$$\begin{aligned} \mathbb{E}[\text{Sat}] + 3\mathbb{E}[\text{Sat}] - 2\text{Opt} - W &= \mathbb{E}[\text{Sat}] - 3 \cdot (W - \mathbb{E}[\text{Sat}]) - 2 \cdot (\text{Opt} - W) \\ &= \mathbb{E}[\text{Sat}] - 3\mathbb{E}[\text{Unsat}] - 2 \cdot (\text{Opt} - W) \end{aligned} \tag{3.4}$$

holds and we have to guarantee that the right hand side is nonnegative. We call $W - \text{Opt}$ the *initial contradiction*; observe that $W - \text{Opt}$ coincides with the weight of all clauses multiplied with the fraction to what extend a clause is *falsified* by the optimal fractional assignment π.

We perform a step–by–step analysis of the algorithm. Imagine the algorithm processing a variable x. After assigning a binary value to x, some contradictions are resolved by terminally falsifying or satisfying clauses. However new contradictions may be introduced. In particular, let the random variable c be the partially contradictory weight of all clauses *before* fixing x and c' be the same quantity *after* fixing x.

We show that the right hand side of (3.4) is nonnegative by establishing an invariant which holds whenever fixing a variable. In particular, if "sat" and "unsat" are the random variables expressing the total weight of clauses satisfied, resp. falsified when fixing x, then we choose assignment probabilities such that the invariant

$$\mathbb{E}[\text{sat} - 3 \cdot \text{unsat} - 2(c' - c)] \geq 0. \tag{3.5}$$

holds. Where is the problem in determining appropriate assignment probabilities $p_0 = \text{prob}[x = 0]$, $p_1 = \text{prob}[x = 1]$ such that (3.5) holds? We have no problem computing $\mathbb{E}[\text{sat}]$ and $\mathbb{E}[\text{unsat}]$, since

$$\mathbb{E}[\text{sat}] = p_0(w_{\overline{x}} + \text{fanout}) + p_1(w_x + \text{fanin}), \quad \mathbb{E}[\text{unsat}] = p_0 w_x + p_1 w_{\overline{x}} \tag{3.6}$$

holds. However computing $\mathbb{E}[c' - c]$ will in general be infeasible. But we may bound the unknown quantity $\mathbb{E}[c' - c]$, assuming that we know π_x, the fractional value of x in the optimum π.

LEMMA 3.1. *Assume that the assignment probabilities are given by* $p_0 = \text{prob}[x = 0]$, $p_1 = \text{prob}[x = 1]$ *and that* $\pi_x \in [0; 1]$ *is the optimal fractional assignment of variable* x. *The increase in contradiction is bounded as follows:*

$$\mathbb{E}[c' - c] \leq \pi_x \cdot (p_0 \text{fanin} - w_{\overline{x}}) + (1 - \pi_x)\,(p_1 \text{fanout} - w_x)\,.$$

Proof. Let k be a clause containing the literal x, so k is a clause contributing to fanin. If k is satisfied by fixing x, which happens with probability p_1, then no new contradictions are introduced.

In case k is (temporarily) falsified, however, the contradiction is increased by at most $\pi_x \cdot w_k$, where w_k is the weight of clause k. Hence, the increase related to fanin is at most $p_0 \cdot \pi_x \cdot \text{fanin}$. The expected gain in contradiction related to fanout follows analogously.

Before x is fixed, the unit clauses $x, \overline{x}$ contribute $(1 - \pi_x) \cdot w_x$ and $\pi_x \cdot w_{\overline{x}}$ to all contradictions. Irrespective of the particular assignment, both unit clauses are removed from the set of clauses, and hence these contradictions are resolved. □

How to Adjust the Assignment Probabilities? W.l.o.g. we assume $q_0 \leq q_1$ in the following and hence Johnson's algorithm would assign $x = 1$. The Slack–Algorithm increases the majority probability slightly, i.e., replaces q_1 by $p_1 = q_1 + \varepsilon$ and q_0 by $p_0 = q_0 - \varepsilon$. How can the algorithm determine ε? We evaluate invariant (3.5) after we replaced $\mathbb{E}[c - c']$ by the bound supplied in Lemma 3.1.

LEMMA 3.2. *Assume that $q_0 \leq q_1$. For assignment probabilities $p_0 = q_0 - \varepsilon$, $p_1 = q_1 + \varepsilon$ we get*

$$\begin{aligned} & \mathbb{E}[\,\mathrm{sat} - 3 \cdot \mathrm{unsat} - 2(c' - c)] \\ \geq\ & (q_1 - q_0) \cdot ((q_1 - q_0) \cdot \Delta + (1 - 2\pi_x)(w_x + w_{\overline{x}})) \\ & + \varepsilon \cdot (2(q_1 - q_0) \cdot \Delta - (1 - 2\pi_x)(\mathrm{fanin} + \mathrm{fanout}))\,. \end{aligned} \tag{3.7}$$

We give the proof in Sect. A in the appendix. The algorithm has to find an ε^* for which (3.7) is nonnegative, regardless of the value π_x. As we show in the proof of the following lemma, the algorithm may set $\pi_x = 1$ (resp. $\pi_x = 0$ for $q_0 > q_1$) and compute the smallest ε for which the right side of (3.7) is nonnegative. Observe that the probabilities are adjusted only if the slack of the decision is small compared to the weights of the unit clauses, i.e. $(q_1 - q_0) \cdot \Delta < w_x + w_{\overline{x}}$.

LEMMA 3.3. *Let ε^* denote the adjustment determined by the algorithm.*
(a) Then the right side of (3.7) is nonnegative regardless of the value of π_x and hence local invariant (3.5) holds in each step.
(b) The assignment probabilities are well defined, i.e. $0 \leq \varepsilon^* \leq q_0$.

The proof was moved to Sect. B in the appendix. This concludes the proof of Theorem 3.1.

Acknowledgement. The author would like to thank Georg Schnitger for his helpful comments.

References

[1] M. Alekhnovich, A. Borodin, J. Buresh-Oppenheim, R. Impagliazzo, and A. Magen. Toward a model for backtracking and dynamic programming. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:38, 2009.

[2] S. Angelopoulos and A. Borodin. Randomized priority algorithms. *Theor. Comput. Sci.*, 411(26-28):2542–2558, 2010.

[3] A. Avidor, I. Berkovitch, and U. Zwick. Improved approximation algorithms for MAX NAE-SAT and MAX SAT. In T. Erlebach and G. Persiano, editors, *WAOA*, volume 3879 of *Lecture Notes in Computer Science*, pages 27–40. Springer, 2005.

[4] Y. Azar, I. Gamzu, and R. Roth. Submodular Max-SAT. In *ESA*, 2011.

[5] A. Borodin, J. Boyar, K. S. Larsen, and N. Mirmohammadi. Priority algorithms for graph optimization problems. *Theor. Comput. Sci.*, 411(1):239–258, 2010.

[6] A. Borodin, M. N. Nielsen, and C. Rackoff. (Incremental) priority algorithms. *Algorithmica*, 37(4):295–326, 2003.

[7] J. Chen, D. K. Friesen, and H. Zheng. Tight bound on Johnson's algorithm for maximum satisfiability. *J. Comput. Syst. Sci.*, 58(3):622–640, 1999.

[8] K. P. Costello, A. Shapira, and P. Tetali. Randomized greedy: new variants of some classic approximation algorithms. In *SODA*, pages 647–655, 2011.

[9] S. Davis and R. Impagliazzo. Models of greedy algorithms for graph problems. *Algorithmica*, 54(3):269–317, 2009.

[10] L. Engebretsen. Simplified tight analysis of Johnson's algorithm. *Inf. Process. Lett.*, 92(4):207–210, 2004.

[11] M. X. Goemans and D. P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7(4):656–666, 1994.

[12] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[13] M. Poloczek and G. Schnitger. Randomized variants of Johnson's algorithm for MAX SAT. In *SODA*, pages 656–663, 2011.

[14] A. van Zuylen. Simpler 3/4 approximation algorithms for MAX SAT. Submitted.

[15] A. C.-C. Yao. Lower bounds by probabilistic arguments. In *FOCS*, pages 420–428. IEEE, 1983.

[16] C. K. Yung. Inapproximation result for exact Max-2-SAT. Unpublished manuscript.

A Proof of Lemma 3.2

Remember that "fanin" and "fanout" are the weight of all clauses of length at least two that contain the literal x, resp. $\overline{x}$, and that $w_x, w_{\overline{x}}$ are the weights of the unit clauses $x, \overline{x}$. Moreover, we defined $\Delta = \text{fanin} + 2w_x + \text{fanout} + 2w_{\overline{x}}$ and the canonical assignment probabilities as

$$q_0 := \text{prob}[x = 0] = \frac{2w_{\overline{x}} + \text{fanout}}{\Delta} \quad \text{and} \quad q_1 := \text{prob}[x = 1] = \frac{2w_x + \text{fanin}}{\Delta}.$$

To prove Lemma 3.2 we have to show

$$\begin{aligned} \mathbb{E}[\,\text{sat} - 3 \cdot \text{unsat} - 2(c' - c)] \;\geq\; & (q_1 - q_0) \cdot ((q_1 - q_0) \cdot \Delta + (1 - 2\pi_x)(w_x + w_{\overline{x}})) \\ & + \varepsilon \cdot (2(q_1 - q_0) \cdot \Delta - (1 - 2\pi_x)(\text{fanin} + \text{fanout}))\,. \end{aligned}$$

Proof. First remember that

$$\begin{aligned} \text{sat} &= p_0(\text{fanout} + w_{\overline{x}}) + p_1(\text{fanin} + w_x) \\ \text{unsat} &= p_0 w_x + p_1 w_{\overline{x}}. \end{aligned}$$

We apply the upper bound of Lemma 3.1 for $c' - c$, the change in contradiction, to get

$$\begin{aligned} & \mathbb{E}[\text{sat} - 3 \cdot \text{unsat} - 2(c' - c)] \\ \geq\; & p_0(\text{fanout} + w_{\overline{x}}) + p_1(\text{fanin} + w_x) - 3(p_0 w_x + p_1 w_{\overline{x}}) \\ & - 2 \cdot \pi_x\,(p_0 \cdot \text{fanin} - w_{\overline{x}}) - 2 \cdot (1 - \pi_x)\,(p_1 \cdot \text{fanout} - w_x) \\ =\; & q_0 \cdot (\text{fanout} + w_{\overline{x}} - 3w_x) + q_1 \cdot (\text{fanin} + w_x - 3w_{\overline{x}}) \\ & - 2 \cdot \pi_x\,(q_0 \cdot \text{fanin} - w_{\overline{x}}) - 2 \cdot (1 - \pi_x)\,(q_1 \cdot \text{fanout} - w_x) \\ & + \varepsilon\,((1 + 2\pi_x)\text{fanin} - (3 - 2\pi_x)\text{fanout} + 4w_x - 4w_{\overline{x}})\,, \\ =\; & q_1 \cdot (\text{fanin} - 2\text{fanout} + w_x - 3w_{\overline{x}}) + q_0 \cdot (\text{fanout} + w_{\overline{x}} - 3w_x) + 2w_x \\ & + \pi_x \cdot (2q_1\text{fanout} - 2q_0\text{fanin} + 2w_{\overline{x}} - 2w_x) \\ & + \varepsilon \cdot ((1 + 2\pi_x)\text{fanin} - (3 - 2\pi_x)\text{fanout} + 4w_x - 4w_{\overline{x}})\,. \end{aligned}$$

We use the definition of p_0, p_1 in the first equality and show all linear terms in π_x in the last equality. Next we consider the three terms of the right hand side, the term not depending on ε or π_x, the term depending on π_x but not on ε and the rest. Observe first that $q_1 \cdot (2w_{\overline{x}} + \text{fanout}) = q_0 \cdot q_1 \cdot \Delta = q_0 \cdot (2w_x + \text{fanin})$ follows from the definition of q_0, q_1 and therefore we obtain

$$q_1\text{fanout} - q_0\text{fanin} = 2q_0 w_x - 2q_1 w_{\overline{x}}. \tag{A.1}$$

As a consequence we can rewrite, again by using the definition of q_0, q_1 and the fact that $q_0 + q_1 = 1$ holds,

$$\begin{aligned} & q_1 \cdot (\text{fanin} - 2\text{fanout} + w_x - 3w_{\overline{x}}) + q_0 \cdot (\text{fanout} + w_{\overline{x}} - 3w_x) + 2w_x \\ =\; & q_1 \cdot \Delta \cdot (q_1 - q_0) - q_1 \cdot \text{fanout} - q_1 \cdot (w_x + w_{\overline{x}}) \\ & - q_0 \cdot \Delta \cdot (q_1 - q_0) + q_0 \cdot \text{fanin} - \; q_0 \cdot (w_x + w_{\overline{x}}) + 2w_x \\ =\; & \Delta(q_1 - q_0)^2 - q_1 \cdot \text{fanout} + q_0 \cdot \text{fanin} - (q_0 + q_1) \cdot (w_x + w_{\overline{x}}) + 2w_x \\ \overset{(A.1)}{=}\; & \Delta(q_1 - q_0)^2 - 2q_0 w_x + 2q_1 w_{\overline{x}} + w_x - w_{\overline{x}} \\ =\; & \Delta(q_1 - q_0)^2 + (q_1 - q_0)(w_x + w_{\overline{x}}). \end{aligned} \tag{A.2}$$

We are done with the first term. For the second one we apply (A.1) again and obtain

$$\begin{aligned} & \pi_x \cdot (2q_1\text{fanout} - 2q_0\text{fanin} + 2w_{\overline{x}} - 2w_x) = \pi_x \cdot (4q_0 w_x - 4q_1 w_{\overline{x}} + 2w_{\overline{x}} - 2w_x) \\ (\text{A.3}) \qquad = \; & 2\pi_x(q_0 - q_1)(w_x + w_{\overline{x}}). \end{aligned}$$

And finally for the third term we again use the definition of q_0, q_1

$$\begin{aligned} & \varepsilon \cdot ((1 + 2\pi_x)\text{fanin} - (3 - 2\pi_x)\text{fanout} + 4w_x - 4w_{\overline{x}}) \\ = \; & \varepsilon \cdot (2\Delta \cdot (q_1 - q_0) - \text{fanin} - \text{fanout} + 2\pi_x \cdot (\text{fanin} + \text{fanout})) \\ (\text{A.4}) \qquad = \; & \varepsilon \cdot (2\Delta \cdot (q_1 - q_0) - (1 - 2\pi_x) \cdot (\text{fanin} + \text{fanout})). \end{aligned}$$

The claim is now an immediate consequence of (A.2), (A.3) and (A.4). □

B Proof of Lemma 3.3

We want to determine an ε such that the right hand side of (3.7), that we give here for convenience,

$$\begin{aligned} \mathbb{E}[\,\text{sat} - 3 \cdot \text{unsat} - 2(c' - c)] \; \geq \; & (q_1 - q_0) \cdot ((q_1 - q_0) \cdot \Delta + (1 - 2\pi_x)(w_x + w_{\overline{x}})) \\ & + \varepsilon \cdot (2(q_1 - q_0) \cdot \Delta - (1 - 2\pi_x)(\text{fanin} + \text{fanout}))\,. \end{aligned}$$

is nonnegative. If $\pi_x > \frac{1}{2}$ holds, then the right hand side may be negative for $\varepsilon = 0$, in which case ε has to be increased. If however $\pi_x \leq \frac{1}{2}$ holds, then the right side of (3.7) is nonnegative for $\varepsilon = 0$, since we assume $q_1 \geq q_0$.

Remark 1. *Now let us assume that the right hand side is negative for $\varepsilon = 0$ and $\pi_x > \frac{1}{2}$. If μ is the root of the right hand side, then*

$$\mu = \frac{-(q_1 - q_0) \cdot (q_1 - q_0) \cdot \Delta - (q_1 - q_0)(1 - 2\pi_x)(w_x + w_{\overline{x}})}{2(q_1 - q_0) \cdot \Delta - (1 - 2\pi_x)(\text{fanin} + \text{fanout})}.$$

Hence $\mu = \frac{-a_1 - a_2(1-2\pi_x)}{b_1 - b_2(1-2\pi_x)}$ with nonnegative coefficients a_1, a_1, b_1, b_2. But then μ, as a function of π_x, is monotone increasing, since its first derivative

$$\begin{aligned} \mu' \; = \; & \frac{2a_2(b_1 - b_2(1 - 2\pi_x)) - 2b_2(-a_1 - a_2(1 - 2\pi_x))}{(b_1 - b_2(1 - 2\pi_x))^2} \\ = \; & \frac{2a_2 b_1 + 2b_2 a_1}{(b_1 - b_2(1 - 2\pi_x))^2} \\ \geq \; & 0. \end{aligned}$$

Set $\varepsilon^* = \mu$, if $\pi_x = 1$, and μ is always bounded by ε^*. Thus, if we can make sure that the right hand side of (3.7) stays nonnegative for $\varepsilon = \varepsilon^*$ when $\pi_x \leq \frac{1}{2}$, then the right hand side is nonnegative in either case. Since the right hand side is nonnegative for $\pi_x = \frac{1}{2}$ and since it depends linearly on π_x, it suffices to show that it is nonnegative for $\pi_x = 0$.

Thus we have to verify two claims, namely

1. to show that the left hand side of (3.7) is nonnegative for $\varepsilon = \varepsilon^*$ and $\pi_x = 0$. We have set $\varepsilon^* = \mu$, assuming $\pi_x = 1$.
2. And that $0 \leq \varepsilon^* \leq q_0$ holds.

Verification of the First Claim. If $\varepsilon^* = 0$, the right hand side of (3.7) is nonnegative for $\pi_x = 0$ and obviously $0 = \varepsilon^* \leq q_0$ holds. Thus only the case $\varepsilon^* > 0$ remains to be discussed. We first evaluate the right hand side of (3.7) for $\pi_x = 0$ (assuming $\varepsilon = \varepsilon^*$) and get

$$\begin{aligned}
& (q_1 - q_0) \cdot (\Delta \cdot (q_1 - q_0) + w_x + w_{\overline{x}}) + \varepsilon^* \cdot (2\Delta \cdot (q_1 - q_0) - (\text{fanin} + \text{fanout})) \\
= \; & (q_1 - q_0) \cdot (\Delta \cdot (q_1 - q_0) + w_x + w_{\overline{x}}) \\
& + \frac{-\Delta \cdot (q_1 - q_0)^2 + (q_1 - q_0)(w_x + w_{\overline{x}})}{2\Delta \cdot (q_1 - q_0) + \text{fanout} + \text{fanin}} \cdot (2\Delta \cdot (q_1 - q_0) - (\text{fanin} + \text{fanout})) \\
= \; & \Delta \cdot (q_1 - q_0)^2 \cdot \left(1 - \frac{2\Delta \cdot (q_1 - q_0) - (\text{fanin} + \text{fanout})}{2\Delta \cdot (q_1 - q_0) + (\text{fanin} + \text{fanout})}\right) \\
& + (q_1 - q_0) \cdot (w_x + w_{\overline{x}}) \cdot \left(1 + \frac{2\Delta \cdot (q_1 - q_0) - (\text{fanin} + \text{fanout})}{2\Delta \cdot (q_1 - q_0) + (\text{fanin} + \text{fanout})}\right) \\
= \; & \Delta \cdot (q_1 - q_0)^2 \cdot \frac{2(\text{fanin} + \text{fanout})}{2\Delta \cdot (q_1 - q_0) + (\text{fanin} + \text{fanout})} \\
& + (q_1 - q_0) \cdot (w_x + w_{\overline{x}}) \cdot \frac{4\Delta \cdot (q_1 - q_0)}{2\Delta \cdot (q_1 - q_0) + (\text{fanin} + \text{fanout})}.
\end{aligned}$$

But then the right hand side of (3.7) is nonnegative, since we assumed $q_1 \geq q_0$.

Verification of the Second Claim. It remains to check that $0 \leq \varepsilon^* \leq q_0$ holds. If $\varepsilon^* = 0$, then we are done.

Thus we assume that the right side of (3.7) is negative for $\varepsilon = 0$ and $\pi_x = 1$. But then $\Delta \cdot (q_1 - q_0) \in (0, w_x + w_{\overline{x}})$ holds and in particular ε^* is nonnegative. It remains to show

$$\varepsilon^* = \frac{-\Delta \cdot (q_1 - q_0)^2 + (q_1 - q_0)(w_x + w_{\overline{x}})}{2\Delta \cdot (q_1 - q_0) + \text{fanout} + \text{fanin}} \overset{!}{\leq} q_0 = \frac{\text{fanout} + 2w_{\overline{x}}}{\Delta}. \tag{B.5}$$

We first observe that (B.5) is equivalent with

$$\begin{aligned}
& -\Delta^2 \cdot (q_1 - q_0)^2 + \Delta \cdot (q_1 - q_0)(w_x + w_{\overline{x}}) \\
\overset{!}{\leq} \; & (\text{fanout} + 2w_{\overline{x}})\,(2\Delta \cdot (q_1 - q_0) + \text{fanout} + \text{fanin}).
\end{aligned}$$

We move $-\Delta^2 \cdot (q_1 - q_0)^2$ from the left to the right hand side and combine this term with

$$(\text{fanout} + 2w_{\overline{x}}) \cdot 2\Delta \cdot (q_1 - q_0) = \Delta \cdot (q_1 - q_0) \cdot (2\text{fanout} + 4w_{\overline{x}})$$

to obtain x. Since $\Delta \cdot (q_1 - q_0) = 2w_x + \text{fanin} - (2w_{\overline{x}} + \text{fanout})$ by definition of the assignment probabilities q_0, q_1, we obtain the equivalent requirement

$$\begin{aligned}
\Delta \cdot (q_1 - q_0)(w_x + w_{\overline{x}}) \; &\leq \; \Delta \cdot (q_1 - q_0)(2w_{\overline{x}} + \text{fanout} + \text{fanin} + 2w_x) \\
& \quad + (\text{fanout} + 2w_{\overline{x}})(\text{fanout} + \text{fanin}) \\
0 \; &\leq \; \Delta \cdot (q_1 - q_0) \cdot \Delta + (\text{fanout} + \text{fanin})(\text{fanout} + 2w_{\overline{x}})
\end{aligned}$$

and the last line follows by the definition of Δ. This completes the proof of Lemma 3.3. □

C An Improved Bound on Johnson's Algorithm

THEOREM C.1. *Denote the total weight of all clauses by W. If an optimal* fractional *assignment satisfies clauses of total weight* Opt*, then*

$$\text{Sat} \geq \frac{\text{Opt} + W}{3} \geq \frac{2}{3}\text{Opt},$$

where Sat *is the clause weight satisfied by the Johnson's algorithm.*

If the score of Opt is not better than an integral optimum for a particular formula, this bound matches the result of Engebretsen [10].

Remark. Moreover, we would like to point out that for giving this guarantee on the performance of Johnson's algorithm it is sufficient to merely double the weight of unit clauses instead of using the modified weight μ, where $\mu(c) = w_c \cdot 2^{-|c|}$ for each clause c with weight w_c and length $|c|$.

Proof. We perform a step–by–step analysis of Johnson's algorithm and prove with the help of Lemma 3.1 that the local invariant

$$\text{sat} - 2 \cdot \text{unsat} - (c' - c) \geq 0. \tag{C.6}$$

holds in each step. Remember that sat (unsat) is the weight of newly satisfied (resp. terminally falsified) clauses in the current step and $(c'-c)$ denotes the change in contradiction with respect to a fixed fractional optimum π.

Summing up (C.6) over all variables yields $\text{Sat} - 2 \cdot \text{Unsat} + (W - \text{Opt}) \geq 0$, where W denotes the total weight of all clauses and Opt is the score of an optimal fractional assignment. Observe that $W - \text{Opt}$ equals the initial contradiction with respect to π: these contradictions are resolved into Sat and Unsat, but not introduced by the algorithm. Thus, the claimed bound follows, since

$$\begin{aligned} & 3\text{Sat} - 2 \cdot (\text{Sat} + \text{Unsat}) + (W - \text{Opt}) \\ = \;& 3\text{Sat} - W - \text{Opt} \geq 0. \end{aligned}$$

We use that $\text{Sat} + \text{Unsat} = W$ holds.

Let's consider the step where variable x is decided. Following the notation of Sect. 3, let fanin (fanout) be the weight of all clauses of length at least two that contain the literal x (resp. $\overline{x}$). Moreover, denote by $w_x, w_{\overline{x}}$ the weights of the unit clauses $x, \overline{x}$.

Due to symmetry we may assume that the support of 1 is at least the support of 0, i.e. $\mu(x) \geq \mu(\overline{x})$. Thus, the assignment probabilities are $q_1 = 1, q_0 = 0$, since Johnson's algorithm always follows the majority of modified weight. Hence,

$$\begin{aligned} \text{sat} &= w_x + \text{fanin}, \\ \text{unsat} &= w_{\overline{x}}, \end{aligned}$$

and the change in contradiction follows by Lemma 3.1 as

$$c' - c \leq \pi_x \cdot (-w_{\overline{x}}) + (1 - \pi_x) \cdot (\text{fanout} - w_x).$$

We can rewrite the left side of the local invariant (C.6) as

$$\begin{aligned} & \text{sat} - 2 \cdot \text{unsat} - (c' - c) \\ \geq \;& w_x + \text{fanin} - 2w_{\overline{x}} - (\pi_x \cdot (-w_{\overline{x}}) + (1 - \pi_x) \cdot (\text{fanout} - w_x)) \\ = \;& (2 - \pi_x)w_x - (2 - \pi_x)w_{\overline{x}} + \text{fanin} - (1 - \pi_x)\text{fanout}. \end{aligned} \tag{C.7}$$

We require (C.7) nonnegative, hence we may multiply it by $\frac{2}{2-\pi_x}$, which is at least 1 because of $\pi_x \in [0; 1]$. We obtain the lower bound

$$\begin{aligned} & 2w_x - 2w_{\overline{x}} + \underbrace{\frac{2}{2 - \pi_x}}_{\geq 1} \text{fanin} - \underbrace{\frac{2 - 2\pi_x}{2 - \pi_x}}_{\leq 1} \text{fanout} \\ \geq \;& 2w_x - 2w_{\overline{x}} + \text{fanin} - \text{fanout} \end{aligned}$$

which coincides with $\mu(x) - \mu(\overline{x})$. This concludes the proof of Theorem C.1, since we assumed that the support of 1 is at least the support of 0. □