# Chapter 6

# Satisfiability

The SATISFYABILITY problem asks if a certain given Boolean formula has a satisfying assignment, i.e., one that makes the whole formula evaluate to true. There is a related optimization problem called MAXIMUM SATISFIABILITY. The goal of this chapter is to develop a deterministic 3/4-approximation algorithm. We first give a corresponding randomized algorithm which will then be derandomized.

We are given the Boolean *variables* $X = \{x_1, \ldots, x_n\}$, where each $x_i \in \{0, 1\}$. A *literal* $\ell_i$ of the variable $x_i$ is either $x_i$ itself, called a *positive* literal, or its negation $\bar{x}_i$ with truth value $1 - x_i$, called a *negative* literal. A *clause* is a disjunction $C = (\ell_1 \vee \cdots \vee \ell_k)$ of literals $\ell_j$ of $X$; their number $k$ is called the *size* of $C$. For a clause $C$ let $S_C^+$ denote the set of its positive literals; similarly $S_C^-$ the set of its negative literals. Let $\mathcal{C}$ denote the set of clauses. A Boolean formula in *conjunctive form* is a conjunction of clauses $F = C_1 \wedge \cdots \wedge C_m$. Each vector $x \in \{0, 1\}^n$ is called a *truth assignment*. For any clause $C$ and any such assignment $x$ we say that $x$ *satisfies* $C$ if at least one of the literals of $C$ evaluates to 1.

The problem MAXIMUM SATISFIABILITY is the following: We are given a formula $F$ in conjunctive form and for each clause $C$ a weight $w_C$, i.e., a weight function $w : \mathcal{C} \to \mathbb{N}$. The objective is to find a truth assignment $x \in \{0, 1\}^n$ that maximizes the total weight of the satisfied clauses. As an important special case: If we set all weights $w_C$ equal to one, then we seek to maximize the number of satisfied clauses.

Now we introduce for each clause $C$ a variable $z_C \in \{0, 1\}$ which takes the value one if and only if $C$ is satisfied under a certain truth assignment $x$. Now we can formulate this problem as a mathematical program as follows:

---

**Problem 6.1** MAXIMUM SATISFIABILITY

---

*Instance.* Formula $F = C_1 \wedge \cdots \wedge C_m$ with $m$ clauses over the $n$ Boolean variables $X = \{x_1, \ldots, x_n\}$. A weight function $w : \mathcal{C} \to \mathbb{N}$.

*Task.* Solve the problem

$$\begin{aligned} \text{maximize} \quad & \text{val}(z) = \sum_{C \in \mathcal{C}} w_C z_C, \\ \text{subject to} \quad & \sum_{i \in S_C^+} x_i + \sum_{i \in S_C^-} (1 - x_i) \geq z_C \quad C \in \mathcal{C}, \\ & z_C \in \{0, 1\} \quad C \in \mathcal{C}, \\ & x_i \in \{0, 1\} \quad i = 1, \ldots, n. \end{aligned}$$

---

The algorithm we aim for is a combination of two algorithms. One works better for small clauses, the other for large clauses. Both are initially randomized but can be *derandomized* using the method of conditional expectation, i.e., the final algorithm is deterministic.

## 6.1 Randomized Algorithm

For each variable $x_i$ we define the random variable $X_i$ that takes the value one with a certain probability $p_i$ and zero otherwise. This induces, for each clause $C$, a random variable $Z_C$ that takes the value one if $C$ is satisfied under a (random) assignment and zero otherwise.

### Algorithm for Large Clauses

Consider this algorithm RANDOMIZED LARGE: For each variable $x_i$ with $i = 1, \ldots, n$, set $X_i = 1$ independently with probability $1/2$ and $X_i = 0$ otherwise. Output $X = (X_1, \ldots, X_n)$.

Define the quantity

$$\alpha_k = 1 - 2^{-k}.$$

**Lemma 6.1.** *Let $C$ be a clause. If $\mathrm{size}(C) = k$ then*

$$\mathbb{E}[Z_C] = \alpha_k.$$

*Proof.* A clause $C$ is not satisfied, i.e., $Z_C = 0$ if and only if all its literals are set to zero. By independence, the probability of this event is exactly $2^{-k}$ and thus

$$\mathbb{E}[Z_C] = 1 \cdot \Pr[Z_C = 1] + 0 \cdot \Pr[Z_C = 0] = 1 - 2^{-k} = \alpha_k$$

which was claimed. □

**Theorem 6.2.** *In expectation, the algorithm* RANDOMIZED LARGE *is a $1/2$-approximation algorithm for* MAXIMUM SATISFIABILITY.

*Proof.* By linearity of expectation, Lemma 6.1, and $\mathrm{size}(C) \geq 1$ we have

$$\mathbb{E}[\mathrm{val}(Z)] = \sum_{C \in \mathcal{C}} w_C \mathbb{E}[Z_C] = \sum_{C \in \mathcal{C}} w_C \alpha_{\mathrm{size}(C)} \geq \frac{1}{2} \sum_{C \in \mathcal{C}} w_C \geq \frac{1}{2} \mathrm{val}(z^*)$$

where $(x^*, z^*)$ is an optimal solution for MAXIMUM SATISFIABILITY. We have used the obvious bound $\mathrm{val}(z^*) \leq \sum_{C \in \mathcal{C}} w_C$. □

### Algorithm for Small Clauses

Maybe the most natural linear programming relaxation of the problem is:

$$\begin{aligned} \text{maximize} \quad & \mathrm{val}(z) = \sum_{C \in \mathcal{C}} w_C z_C, \\ \text{subject to} \quad & \sum_{i \in S_C^+} x_i + \sum_{i \in S_C^-} (1 - x_i) \geq z_C \quad C \in \mathcal{C}, \\ & 0 \leq z_C \leq 1 \quad C \in \mathcal{C} \\ & 0 \leq x_i \leq 1 \quad i = 1, \ldots, n. \end{aligned}$$

In the sequel let $(\bar{x}, \bar{z})$ denote an optimum solution for this LP.

Consider this algorithm Randomized Small: Determine $(\bar{x}, \bar{z})$. For each variable $x_i$ with $i = 1, \ldots, n$, set $X_i = 1$ independently with probability $\bar{x}_i$ and $X_i = 0$ otherwise. Output $X = (X_1, \ldots, X_n)$.

Define the quantity

$$\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k .$$

**Lemma 6.3.** *Let $C$ be a clause. If* $\text{size}(C) = k$ *then*

$$\mathbb{E}\left[Z_C\right] = \beta_k \bar{z}_C.$$

*Proof.* We may assume that the clause $C$ has the form $C = (x_1 \vee \cdots \vee x_k)$; otherwise rename the variables and rewrite the LP.

The clause $C$ is satisfied if $x_1, \ldots, x_k$ are not all set to zero. The probability of this event is

$$\begin{aligned} 1 - \Pi_{i=1}^k (1 - \bar{x}_i) &\geq 1 - \left(\frac{\sum_{i=1}^k (1 - \bar{x}_i)}{k}\right)^k \\ &= 1 - \left(1 - \frac{\sum_{i=1}^k \bar{x}_i}{k}\right)^k \\ &\geq 1 - \left(1 - \frac{\bar{z}_C}{k}\right)^k . \end{aligned}$$

Above we firstly have used the arithmetic-geometric mean inequality, which states that for non-negative numbers $a_1, \ldots, a_k$ we have

$$\frac{a_1 + \cdots + a_k}{k} \geq \sqrt[k]{a_1 \cdot \cdots \cdot a_k}.$$

Secondly the LP guarantees the inequality $\bar{x}_1 + \cdots + \bar{x}_k \geq \bar{z}_C$.

Now define the function $g(t) = 1 - (1 - t/k)^k$. This function is concave with $g(0) = 0$ and $g(1) = 1 - (1 - 1/k)^k$ which yields that we can bound

$$g(t) \geq t(1 - (1 - 1/k)^k) = t\beta_k$$

for all $t \in [0, 1]$.

Therefore

$$\Pr\left[Z_C = 1\right] \geq 1 - \left(1 - \frac{\bar{z}_C}{k}\right)^k \geq \beta_k \bar{z}_C$$

and the claim follows. □

**Theorem 6.4.** *In expectation, the algorithm* Randomized Small *is a* $1-1/e$*-approximation algorithm for* Maximum Satisfiability.

*Proof.* The function $\beta_k$ is decreasing with $k$. Therefore if all clauses are of size at most $k$, then by Lemma 6.3

$$\mathbb{E}\left[\text{val}(Z)\right] = \sum_{C \in \mathcal{C}} w_C \mathbb{E}\left[Z_C\right] \geq \beta_k \sum_{C \in \mathcal{C}} w_C \bar{z}_C = \beta_k \text{val}(\bar{z}) \geq \beta_k \text{val}(z^*),$$

where $(x^*, z^*)$ is an optimal solution for Maximum Satisfiability. The claim follows since $(1 - 1/k)^k < 1/e$ for all $k \in \mathbb{N}$. □

**3/4-Approximation Algorithm**

Consider the algorithm Randomized Combine: With probability 1/2 run Randomized Large otherwise run Randomized Small.

**Lemma 6.5.** *Let $C$ be a clause, then*

$$\mathbb{E}[Z_C] \geq \frac{3\bar{z}_C}{4}.$$

*Proof.* Let the random variable $B$ take the value zero if the first algorithm is run, one otherwise. For a clause $C$ let $\text{size}(C) = k$. By Lemma 6.1 and $\bar{z}_C \leq 1$

$$\mathbb{E}[Z_C \mid B = 0] = \alpha_k \geq \alpha_k \bar{z}_C.$$

and by Lemma 6.1

$$\mathbb{E}[Z_C \mid B = 1] \geq \beta_k \bar{z}_C.$$

Combining we have

$$\mathbb{E}[Z_C] = \mathbb{E}[Z_C \mid B = 0] \Pr[B = 0] + \mathbb{E}[Z_C \mid B = 1] \Pr[B = 1] \geq \frac{\bar{z}_C}{2}(\alpha_k + \beta_k).$$

Inspection shows that $\alpha_k + \beta_k \geq 3/2$ for all $k \in \mathbb{N}$. □

**Theorem 6.6.** *In expectation, the algorithm* Randomized Combine *is a 3/4-approximation algorithm for* Maximum Satisfiability.

*Proof.* This follows from Lemma 6.5 and linearity of expectation. □

## 6.2 Derandomization

The notion of *derandomization* refers to "turning" a randomized algorithm into a deterministic one (possibly at the cost of additional running time or deterioration of approximation guarantee). One of the several available techniques is the method of *conditional expectation.*

We are given a Boolean formula $F = C_1 \wedge \cdots \wedge C_m$ in conjunctive form over the variables $X = \{x_1, \ldots, x_n\}$. Suppose we set $x_1 = 0$, then we get a formula $F_0$ over the variables $x_2, \ldots, x_n$ after simplification; if we set $x_1 = 1$ then we get a formula $F_1$.

**Example 6.7.** Let $F = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_3) \wedge (x_1 \vee \bar{x}_4)$ where $X = \{x_1, \ldots, x_4\}$.

$$\begin{aligned} x_1 = 0: &\quad F_0 = (x_2) \wedge (x_4) \\ x_1 = 1: &\quad F_1 = (x_3) \end{aligned}$$

Applying this recursively, we obtain the tree $T(F)$ depicted in Figure 6.1. The tree $T(F)$ is a complete binary tree with height $n+1$ and $2^{n+1}-1$ vertices. Each vertex at level $i$ corresponds to a setting for the Boolean variables $x_1, \ldots, x_i$. We label the vertices of $T(F)$ with their respective conditional expectations as follows. Let $X_1 = a_1, \ldots, X_i = a_i \in \{0, 1\}$ be the outcome of a truth assignment for the variables $x_1, \ldots, x_i$. The vertex corresponding to this assignment will be labeled

$$\mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i].$$

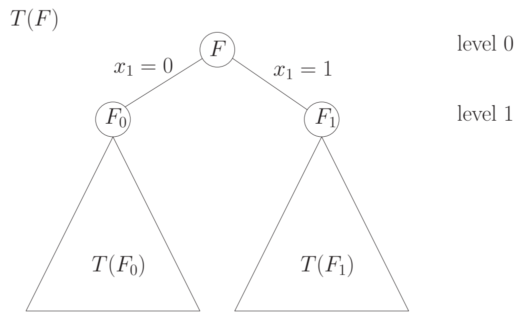


Figure 6.1: Derandomization tree for a formula $F$.

If $i = n$, then this conditional expectation is simply the total weight of clauses satisfied by the truth assignment $x_1 = a_1, \ldots, x_n = a_n$.

The goal of the remainder of the section is to show that we can find deterministically in polynomial time a path from the root of $T(F)$ to a leaf such that the conditional expectations of the vertices on that path are at least as large as $\mathbb{E}[\text{val}(Z)]$. Obviously, this property yields the desired: We can construct determistically a solution which is at least as good as the one of the randomized algorithm in expectation.

**Lemma 6.8.** *The conditional expectation*

$$\mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i]$$

*of any vetex in $T(F)$ can be computed in polynomial time.*

*Proof.* Consider a vertex $X_1 = a_1, \ldots, X_i = a_i$. Let $F'$ be the Boolean formula obtained from $F$ by setting $x_1, \ldots, x_i$ accordingly. $F'$ is in the variables $x_{i+1}, \ldots, x_n$.

Clearly, by linearity of expectation, the expected weight of any clause of $F'$ under any random truth assignment to the variables $x_{i+1}, \ldots, x_n$ can be computed in polynomial time. Adding to this the total weight of clauses satisfied by $x_1, \ldots, x_i$ gives the answer. □

**Theorem 6.9.** *We can compute in polynomial time a path from the root to a leaf in $T(F)$ such that the conditional expectation of each vertex on this path is at least $\mathbb{E}[\text{val}(Z)]$.*

*Proof.* Consider the conditional expectation at a certain vertex $X_1 = a_1, \ldots, X_i = a_i$ for setting the next variable $X_{i+1}$. We have that

$$\begin{aligned}
&\mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i] \\
&\qquad = \mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i, X_{i+1} = 0] \Pr[X_{i+1} = 0] \\
&\qquad\qquad + \mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i, X_{i+1} = 1] \Pr[X_{i+1} = 1] .
\end{aligned}$$

We show that the two conditional expectations with $X_{i+1}$ can *not* be both strictly smaller than $\mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i]$. Assume the contrary, then we have

$$\begin{aligned}
&\mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i] \\
&\qquad < \mathbb{E}[\text{val}(Z) \mid X_1 = a_1, \ldots, X_i = a_i] (\Pr[X_{i+1} = 0] + \Pr[X_{i+1} = 1])
\end{aligned}$$

which is a contradiction since $\Pr[X_{i+1} = 0] + \Pr[X_{i+1} = 1] = 1$.

This yields the existence of such a path can by Lemma 6.8 it can be computed in polynomial time. □

The derandomized version of a randomized algorithm now simply executes these proofs with the probability distribution as given by the randomized algorithm.