Randomized Algorithms

MAX-SAT

Maximum Satisfiability (MAX-SAT)

- *Given* a conjunctive normal form formula f on Boolean variables $x_1,\ldots, x_n$, and nonnegative weights w_c, for each clause c of f.
- *Find* a truth assignment to the Boolean variables that maximizes the total weight of satisfied clauses.

Clauses

- Each clause is a disjunction of literals; each literal being either a Boolean variable or its negation. Let size(c) denote the **size** of clause c, i.e., the number of literals in it. We will assume that the sizes of clauses in f are arbitrary.
- A clause is said to be satisfied if one of the unnegated variables is set to true or one of the negated variables is set to false.

Terminology

- Random variable W will denote the total weight of satisfied clauses.
- For each clause $c \in f$, random variable W_c denotes the weight contributed by clause c to W.

$$W = \sum_{c \in f} W_c,$$

$$\mathrm{E}\left[W_c\right] = w_c \cdot \Pr\left[c = 1\right].$$

Johnson's Algorithm

0) **Input** $(x_1, \ldots, x_n, f, w: f \to \mathbf{Q}^+)$
1) Set each Boolean variable to be True independently with probability 1/2.
3) **Output** the resulting truth assignment, say τ.

A good algorithm for large clauses

- For $k \geq 1$, define $\alpha_k = 1 - 2^{-k}$.
- **Lemma 11.1** If size(c)=k, then $E[W_c] = \alpha_k w_c$.

  Proof. Clause c is not satisfied by τ iff all its literals are set to False. The probability of this event is 2^{-k}.
- **Corollary 11.2** $E[W] \geq$ ½ OPT.

  Proof. For $k \geq 1$, $\alpha_k \geq$ ½. By linearity of expectation,

$$\mathbf{E}[W] = \sum_{c \in f} \mathbf{E}[W_c] \geq \frac{1}{2} \sum_{c \in f} w_c \geq \frac{1}{2} \mathrm{OPT}.$$

Conditional Expectation

- Let $a_1, \ldots, a_i$ be a truth assignment to $x_1, \ldots, x_i$.

- **Lemma 11.3**

  $E[W | x_1 = a_1, \ldots, x_i = a_i]$ can be computed in polynomial time.

Proof

- Let an assignment of variables $x_1,\ldots, x_i$ is fixed say $x_1 = a_1,\ldots, x_i = a_i$.
- Let φ be the Boolean formula, on variables $x_{i+1},\ldots, x_n$, obtained for this node via self-reducibility.
- The expected weight of satisfied clauses of φ under a 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 of f already satisfied by the partial assignment $x_1 = a_1,\ldots, x_i = a_i$ gives the answer.

Derandomazing

- **Theorem 11.4**

  We can compute, in polynomial time, an assignment $x_1=a_1,\ldots, x_n=a_n$ such that $W(a_1,\ldots, a_n) \geq \mathrm{E}[W]$.

Proof

- $E[W| x_1=a_1,\ldots, x_i=a_i] = E[W| x_1=a_1,\ldots, x_i=a_i, x_{i+1}= \text{True}]/2 +$ $+ E[W| x_1=a_1,\ldots, x_i=a_i, x_{i+1}= \text{False}]/2$
- It follows that either $E[W| x_1=a_1,\ldots, x_i=a_i, x_{i+1}= \text{True}] \geq E[W| x_1=a_1,\ldots, x_i=a_i]$, or $E[W| x_1=a_1,\ldots, x_i=a_i, x_{i+1}= \text{False}] \geq E[W| x_1=a_1,\ldots, x_i=a_i]$.
- Take an assignment with larger expectation.
- The procedure requires n iterations. Lemma 11.3 implies that each iteration can be done in polynomial time.

Remark

- Let us show that the technique outlined above can, in principle, be used to derandomize more complex randomized algorithms. Suppose the algorithm does not set the Boolean variables independently of each other. Now,
- $\mathrm{E}[W \mid x_1=a_1,\ldots, x_i=a_i] =$

$\mathrm{E}[W \mid x_1=a_1,\ldots, x_i=a_i, x_{i+1}= \text{True}] \cdot \Pr[x_{i+1}= \text{True} \mid x_1=a_1,\ldots, x_i=a_i] +$

$\mathrm{E}[W \mid x_1=a_1,\ldots, x_i=a_i, x_{i+1}= \text{False}] \cdot \Pr[x_{i+1}= \text{False} \mid x_1=a_1,\ldots, x_i=a_i]$.

- The sum of the two conditional probabilities is again 1, since the two events are exhaustive.

$\Pr[x_{i+1}= \text{True} \mid x_1=a_1,\ldots, x_i=a_i]+\Pr[x_{i+1}= \text{False} \mid x_1=a_1,\ldots, x_i=a_i]=1.$

Conclusion

- So, the conditional expectation of the parent is still a convex combination of the conditional expectations of the two children.
- Thus,

either $E[W | x_1=a_1, \ldots, x_i=a_i, x_{i+1}= \text{True}] \geq E[W | x_1=a_1, \ldots, x_i=a_i]$,

or $E[W | x_1=a_1, \ldots, x_i=a_i, x_{i+1}= \text{False}] \geq E[W | x_1=a_1, \ldots, x_i=a_i]$.

- If we can determine, in polynomial time, which of the two children has a larger value, we can again derandomize the algorithm.

Flipping biased coins

- How might we improve the randomized algorithm for MAX SAT? We will show here that biasing the probability with which we set x_i is actually helpful; that is, we will set x_i true with some probability not equal to 1/2.

No negated unit clauses

- To do this, it is easiest to start by considering only MAX SAT instances with no negated unit clauses. We will later show that we can remove this assumption. Suppose now we set each x_i to be true independently with probability $p > 1/2$. As in the analysis of the previous randomized algorithm, we will need to analyze the probability that any given clause is satisfied.

- **Lemma 11.5**

If each x_i is set to true with probability $p > 1/2$ independently, then the probability that any given clause is satisfied is at least $\min(p, 1-p^2)$ for MAX SAT instances with no negated unit clauses.

Proof

- If the clause is a unit clause, then the probability the clause is satisfied is p, since it must be of the form x_i, and the probability x_i is set true is p.
- If the clause has length at least two, then the probability that the clause is satisfied is $1 - p^a(1 - p)^b$, where a is the number of negated variables in the clause and b is the number of unnegated variables in the clause and $a + b \geq 2$.
- Since $p > 1/2 > 1 - p$, this probability is at least $1 - p^{a+\mathrm{b}} \geq 1 - p^2$, and the lemma is proved.

Best performance guarantee

- We can obtain the best performance guarantee by setting $p = 1 - p^2$. This yields $p = 1/2\,(\sqrt{5} - 1) \approx 0.618$. Lemma 11.5 immediately implies the following theorem.
- **Theorem 11.6**

Setting each x_i to true with probability $_{\text{p}}$ independently gives a randomized $\min(p, 1 - p^2)$-approximation algorithm for MAX SAT instances with no negated unit clauses.

$$\mathbf{E}\left[W\right] = \sum_{c \in f} \mathbf{E}\left[W_c\right] \geq \min\left(p, 1 - p^2\right) \sum_{c \in f} w_c \geq \min\left(p, 1 - p^2\right) \mathrm{OPT}.$$

General case

- We would like to extend this result to all MAX SAT instances.
- To do this, we will use a better bound on OPT than $\sum_{c \in f} w_c$.
- Assume that for every i the weight of the unit clause x_i appearing in the instance is at least the weight of the unit clause $\bar{x}_i$.
- This is without loss of generality since we could negate all occurrences of x_i if the assumption is not true. Let v_i be the weight of the unit clause $\bar{x}_i$ if it exists in the instance, and let v_i be zero otherwise.

New upper bound

- **Lemma 11.7**

$$OPT \leq \sum_{c \in f} w_c - \sum_{\bar{x}_i \in f} v_i.$$

Proof. For each i, the optimal solution can satisfy exactly one of x_i and $\bar{x}_i$. Thus the weight of the optimal solution cannot include both the weight of the clause x_i and the clause $\bar{x}_i$. Since v_i is the smaller of these two weights, the lemma follows.

0.618-approximation algorithm

- **Theorem 11.8**

We can obtain a randomized $\frac{\sqrt{5}-1}{2}$ -approximation algorithm for MAX SAT.

Proof

- Let φ be the Boolean formula obtained from f by deleting all negated unit clauses.
- Thus, $\sum_{c \in \varphi} w_c = \sum_{c \in f} w_c - \sum_{\bar{x}_i \in f} v_i$.
- Set each x_i to be *true* independently with probability $p = \frac{\sqrt{5}-1}{2}$.

$$\mathbf{E}[W] = \sum_{c \in f} \mathbf{E}[W_c] = \sum_{c \in f} w_c \Pr[c = 1]$$

$$\geq \sum_{c \in \varphi} w_c \Pr[c = 1] \geq p \sum_{c \in \varphi} w_c = p\left(\sum_{c \in f} w_c - \sum_{\bar{x}_i \in f} v_i\right) \geq p \cdot OPT.$$

- This algorithm can be derandomized using the method of conditional expectations.

A good algorithm for small clauses

- We design an integer program for MAX-SAT.
- For each clause $c \in f$, let $S_c^+ (S_c^-)$ denote the set of Boolean variables occurring nonnegated (negated) in c.
- The truth assignment is encoded by $\boldsymbol{y}$. Picking $y_i = 1$ ($y_i = 0$) denotes setting x_i to True (False).
- The constraint for clause c ensures that z_c can be set to 1 only if at least one of the literals occurring in c is set to True, i.e., if clause c is satisfied by the picked truth assignment.

ILP of MAX-SAT

$$\text{maximize} \quad \sum_{c \in f} w_c z_c$$

$$\text{s.t.} \quad \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c, \quad c \in f$$

$$z_c \in \{0, 1\}, \qquad c \in f$$

$$y_i \in \{0, 1\}, \qquad i = 1, \ldots, n$$

LP-relaxation of MAX-SAT

$$\text{maximize} \quad \sum_{c \in f} w_c z_c$$

$$\text{s.t.} \quad \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c, \quad c \in f$$

$$1 \geq z_c \geq 0, \qquad c \in f$$

$$1 \geq y_i \geq 0, \qquad i = 1, \ldots, n$$

Algorithm LP-MAX-SAT

0) **Input** $(x_1,\ldots, x_n, f, w\colon f \to \mathbf{Q}^+)$
1) Solve LP-relaxation of MAX-SAT.
   Let (y^*, z^*) denote the optimal solution.
2) Independently set x_i to True with probability y_i^*.
3) **Output** the resulting truth assignment, say τ.

Expected weight of disjunction

- For $k \geq 1$, define $\beta_k = 1 - (1 - 1/\mathrm{k})^k$.
- **Lemma 11.9**

  If size(c)=k, then $\mathrm{E}[W_c] \geq \beta_k w_c z^*(c)$.

Proof

We may assume w.l.o.g. that all literals in c appear nonnegated. Further, by renaming variables, we may assume

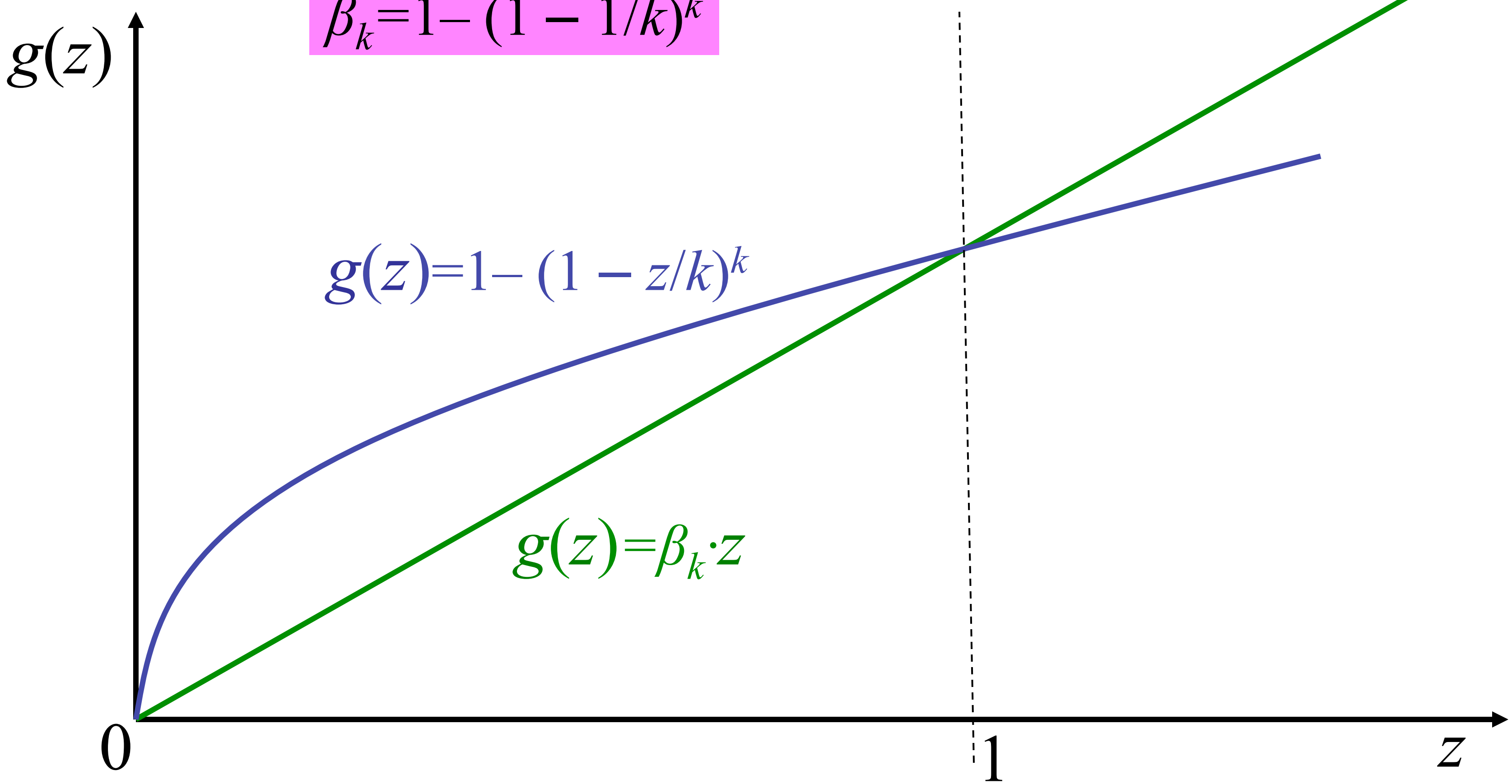
$$c = (x_1 \vee \ldots \vee x_k):$$

$$\Pr[c = True] = 1 - \prod_{i=1}^{k}(1 - y_i) \geq 1 - \left(\frac{\sum_{i=1}^{k}(1 - y_i)}{k}\right)^k =$$

$$= 1 - \left(1 - \frac{\sum_{i=1}^{k} y_i}{k}\right)^k \overset{LP}{\geq} 1 - \left(1 - \frac{z^*(c)}{k}\right)^k.$$

Proof

$\beta_k = 1 - (1 - 1/k)^k$



$$\Pr[c = True] \geq 1 - \left(1 - \frac{z^*(c)}{k}\right)^k \geq \beta_k z^*(c).$$

1–1/e

- **Corollary 11.10**

  $E[W] \geq \beta_k \text{OPT}$ (if all clauses are of size at most k).

- Proof. Notice that β_k is a decreasing function of k. Thus, if all clauses are of size at most k,

$$E[W] = \sum_{c \in f} E[W_c] \geq \beta_k \sum_{c \in f} w_c z_c^* = \beta_k \text{ OPT}.$$

(1–1/e)-factor approximation

- Since $\forall k \in Z^{+} : \left(1 - \frac{1}{k}\right)^{k} > \frac{1}{e}$, we obtain the following result .

Theorem 11.11

Algorithm LP-MAX-SAT is a (1–1/e)-factor algorithm for MAX-SAT.

A (¾)-factor algorithm

- We will combine the two algorithms as follows. Let b be the flip of a fair coin.
- If $b = 0$, run the Johnson algorithm, and if $b = 1$, run Algorithm LP-MAX-SAT.
- Let z^* be the optimal solution of LP on the given instance.
- **Lemma 11.12**

$$E[W_c] \geq (3/4) w_c z^*(c).$$

$\mathrm{E}[W_c] \geq (3/4) w_c z^*(c)$

- Let size(c)=k.
- $Л$ 8.5 $\Rightarrow \mathrm{E}[W_c|b{=}0] = \alpha_k w_c \geq \alpha_k w_c z^*(c)$
- $Л$ 8.9 $\Rightarrow \mathrm{E}[W_c|b{=}1] \geq \beta_k w_c z^*(c)$
- $\mathrm{E}[W_c] = (1/2)(\mathrm{E}[W_c|b{=}0] + \mathrm{E}[W_c|b{=}1]) \geq$
  $\geq (1/2) w_c z^*(c)(\alpha_k + \beta_k)$
- $\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = 3/2$
- $k \geq 3$, $\alpha_k + \beta_k \geq 7/8 + (1 - 1/\mathrm{e}) > 3/2$
- $\mathrm{E}[W_c] \geq (3/4) w_c z^*(c)$

$\mathrm{E}[W]$

By linearity of expectation,

$$\mathrm{E}\left[W\right] = \sum_{c\in f} \mathrm{E}\left[W_c\right] \geq \frac{3}{4}\sum_{c\in f} w_c z^*\left(c\right) = \frac{3}{4}\mathrm{OPT}_{LP} \geq \frac{3}{4}\mathrm{OPT},$$

Finally, consider the following deterministic algorithm.

Goemans-Williamson Algorithm

0. **Input** $(x_1, \ldots, x_n, f, w: f \to \mathbf{Q}^+)$
1. Use the Johnson algorithm to get a truth assignment, τ_1.
2. Use Algorithm LP-MAX-SAT to get a truth assignment, τ_2.
3. Output the better of the two assignments.

(3/4)-approximation

- **Theorem 11.13**

  Goemans-Williamson Algorithm is a deterministic factor 3/4 approximation algorithm for MAX-SAT.

Exercise

- Consider the following instance I of MAX-SAT problem.
  - Each clause has two or more literals.
  - If clause has exactly two literals it has at least one nonnegated variable.
- Consider Algorithm Random(p): set each Boolean variable to be True independently with probability p.
- Determine the value of p for which Algorithm Random(p) finds the best solution for the instance I in the worst case.