Approximation Algorithms III

Maximum Satisfiability

Input: Set $\mathcal{C}$ of clauses over n Boolean variables, nonnegative weights w_c for each clause $c \in \mathcal{C}$.

Output: A truth assignment to the Boolean variables that maximizes the weight of satisfied clauses.

- Special case: MAX-kSAT (each clause is of size at most k).
- Even MAX-2SAT is NP-hard.

A first algorithm:

1. Set each Boolean variable to be TRUE independently with probability 1/2.
2. Output the resulting truth assignment.

Lemma 1. *Let W_c be a random variable that denotes the weight contributed by clause c. If c contains k literals, then* $\mathrm{E}[W_c] = (1 - 2^{-k})w_c$.

Proof:

- Clause c is not satisfied iff all literals are set to FALSE.
- The probability of this event is 2^{-k}.
- $\mathrm{E}[W_c] = w_c \cdot \Pr[c \text{ is satisfied}]$.

□

Theorem 2. *The first algorithm has an expected performance guarantee of* 1/2.

Proof:

- By linearity of expectation,

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

□

Derandomizing via the method of conditional expectations:

- Note that $\mathrm{E}[W] = \frac{1}{2} \cdot \mathrm{E}[W|x_1 = \mathrm{T}] + \frac{1}{2} \cdot \mathrm{E}[W|x_1 = \mathrm{F}]$.
- Also, we can compute $\mathrm{E}[W|x_1 = \{\mathrm{T}, \mathrm{F}\}]$ in polynomial time.

- We choose the truth assignment with the larger conditional expectation, and continue in this fashion:

- $\mathrm{E}[W|x_1 = a_1, \ldots, x_i = a_i] = \frac{1}{2} \cdot \mathrm{E}[W|x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = \mathrm{T}] \quad + \frac{1}{2} \cdot \mathrm{E}[W|x_1 = a_1, \ldots, x_i = a_i, x_{i+1} = \mathrm{F}]$.

- After n steps, we get a deterministic truth assignment of weight at least $\frac{1}{2} \cdot \mathrm{OPT}$.

An integer programming formulation:

$$
\begin{aligned}
\max \quad & \sum_{c \in \mathcal{C}} w_c y_c \\
\text{s.t.} \quad & \sum_{i \in c^+} x_i + \sum_{i \in c^-} (1 - x_i) \geq y_c & c \in \mathcal{C} \\
& y_c \in \{0, 1\} & c \in \mathcal{C} \\
& x_i \in \{0, 1\} & i = 1, \ldots, n
\end{aligned}
$$

And its linear programming relaxation:

$$
\begin{aligned}
\max \quad & \sum_{c \in \mathcal{C}} w_c y_c \\
\text{s.t.} \quad & \sum_{i \in c^+} x_i + \sum_{i \in c^-} (1 - x_i) \geq y_c & c \in \mathcal{C} \\
& 0 \leq y_c \leq 1 & c \in \mathcal{C} \\
& 0 \leq x_i \leq 1 & i = 1, \ldots, n
\end{aligned}
$$

Randomized rounding:

1. Solve the LP relaxation. Let (x^*, y^*) denote the optimal solution.
2. FOR $i = 1$ TO n
3. Independently set variable i to TRUE with probability x_i^*.
4. Output the resulting truth assignment.

Lemma 3. *If c contains k literals, then*

$$\mathrm{E}[W_c] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) w_c y_c^*.$$

Proof:

- We may assume that $c = (x_1 \vee \ldots \vee x_k)$.

- The probability that not all $x_1, \dots x_k$ are set to FALSE is

$$
\begin{aligned}
1 - \prod_{i=1}^{k} (1 - x_i^*) &\geq 1 - \left(\frac{\sum_{i=1}^{k} (1 - x_i^*)}{k} \right)^k && (1) \\
&= 1 - \left(1 - \frac{\sum_{i=1}^{k} x_i^*}{k} \right)^k && (2) \\
&\geq 1 - \left(1 - \frac{y_c^*}{k} \right)^k && (3)
\end{aligned}
$$

  where (1) follows from the arithmetic-geometric mean inequality and (3) follows from the LP constraint.

Proof:

- The function $g(y) := 1 - \left(1 - \frac{y}{k}\right)^k$ is concave.
- In addition, $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{k}\right)^k$.
- Therefore, for $y \in [0, 1]$, $g(y) \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) y$.
- Hence, $\Pr[c \text{ is satisfied }] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) y_c^*$.

□

Thus,

- Randomized rounding is a $\left(1 - \left(1 - \frac{1}{k}\right)^k\right)$-approximation algorithm for MAX-kSAT.
- Randomized rounding is a $(1 - \frac{1}{e})$-approximation algorithm for MAX-SAT.

k	Simple algorithm	Randomized rounding
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684
5	0.969	0.672

Theorem 4. *Given any instance of MAX-SAT, we run both algorithms and choose the better solution. The (expected) performance guarantee of the solution returned is* $3/4$*.*

Proof:

- It suffices to show that $\frac{1}{2}\left(\mathrm{E}[W_c^1] + \mathrm{E}[W_c^2]\right) \geq \frac{3}{4} w_c y_c^*$.
- Assume that c has k clauses.
- By the first lemma, $\mathrm{E}[W_c^1] \geq \left(1 - 2^{-k}\right) w_c y_c^*$.

- By the second lemma, $\mathrm{E}[W_c^2] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) w_c y_c^*$.
- Hence, $\frac{1}{2}\left(\mathrm{E}[W_c^1] + \mathrm{E}[W_c^2]\right) \geq \frac{3}{4} w_c y_c^*$.

□

- Note that this argument also shows that the integrality gap is not worse than 3/4.
- The following example shows that this is tight:
- Consider $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$.
- $x_i = 1/2$ and $y_c = 1$ for all i and c is an optimal LP solution.
- On the other hand, $\mathrm{OPT} = 3$.

Bin Packing

Input: n items of size $a_1, \ldots, a_n \in (0, 1]$.

Output: A packing of items into unit-sized bins that minimizes the number of bins used.

Theorem 5. *The* Bin-Packing Problem *is NP-complete.*

Proof:

- Reduction from Partition:

Input: n numbers $b_1, \ldots, b_n \geq 0$.

?: Does there exist $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} b_i = \sum_{i \notin S} b_i$?

- Define $a_i := \frac{2b_i}{\sum_{j=1}^n b_j}$, for $i = 1, \ldots, n$.
- Obviously, there exists a partition iff one can pack all items into two bins.

□

Corollary 6. *There is no α-approximation algorithm for* Bin Packing *with $\alpha < 3/2$, unless* $\mathrm{P} = \mathrm{NP}$.

First Fit:

- "Put the next item into the first bin where it fits. If it does not fit in any bin, open a new bin."
- This is an obvious 2-approximation algorithm:

- If m bins are used, then at least $m-1$ bins are more than half full. Therefore,

$$\sum_{i=1}^{n} a_i > \frac{m-1}{2}.$$

Since $\sum_{i=1}^{n} a_i$ is a lower bound, $m-1 < 2 \cdot \mathrm{OPT}$. The result follows.

Theorem 7. *For any $0 < \epsilon < 1/2$, there is an algorithm that runs in time polynomial in n and finds a packing using at most $(1+2\epsilon)\mathrm{OPT}+1$ bins.*

Step 1:

Lemma 8. *Let $\epsilon > 0$ and $K \in \mathbb{Z}_+$ be fixed. The bin-packing problem with items of size at least ϵ and with at most K different item sizes can be solved in polynomial time.*

Proof:

- Let the different item sizes be $s_1, \ldots, s_l$, for some $l \leq K$.
- Let b_i be the number of items of size s_i.
- Let $T_1, \ldots, T_N$ be all ways in which a single bin can be packed:

$$\Big\{T_1, \ldots, T_N\Big\} = \Big\{(k_1, \ldots, k_m) \in \mathbb{Z}_+^m : \sum_{i=1}^{m} k_i s_i \leq 1\Big\}.$$

- We write $T_j = (t_{j1}, \ldots, t_{jm})$.
- Then bin packing is equivalent to the following IP:

$$\begin{array}{lll} \min & \displaystyle\sum_{j=1}^{N} x_j & \\ \text{s.t.} & \displaystyle\sum_{j=1}^{N} t_{ji} x_j \geq b_i & i = 1, \ldots, m \\ & x_j \in \mathbb{Z}_+ & j = 1, \ldots, n \end{array}$$

- Since N is constant (each bin fits at most $1/\epsilon$ many items, and there are only K different item sizes), this is an IP in fixed dimension, which can be solved in polynomial time.

□

Step 2:

Lemma 9. *Let $\epsilon > 0$ be fixed. The bin-packing problem with items of size at least ϵ has a $(1+\epsilon)$-approximation algorithm*

Proof:

- Let I be the given instance. Sort the n items by nondecreasing size.

- Partition them into $K := \lceil 1/\epsilon^2 \rceil$ groups each having at most $Q := \lfloor n\epsilon^2 \rfloor$ items.
- Construct a new instance, J, by rounding up the size of each item to the size of the largest item in its group.
- Note that J has at most K different item sizes.
- By the previous lemma, we can find an optimal packing for J in polynomial time.
- Clearly, this packing is also feasible for the original item sizes.
- We construct another instance, J', by rounding down the size of each item to the size of the smallest item in its group.
- Clearly, $\mathrm{OPT}(J') \leq \mathrm{OPT(I)}$.
- Observe that a feasible packing for J' yields a feasible packing for all but the largest Q items of J.
- Therefore, $\mathrm{OPT}(J) \leq \mathrm{OPT}(J') + Q \leq \mathrm{OPT}(I) + Q$.
- Since each item has size at least ϵ, $\mathrm{OPT}(I) \geq n\epsilon$.
- Thus, $Q = \lfloor n\epsilon^2 \rfloor \leq \epsilon \mathrm{OPT}(I)$.

□

Step 3: Proof of the Theorem

- Let I' be the instance obtained by ignoring items of size $< \epsilon$.
- By the previous lemma, we can find a packing for I' using at most $(1+\epsilon)\mathrm{OPT}$ bins.
- We then pack the items of size $\leq \epsilon$ in a First-Fit manner into the bins opened for I'.
- If no additional bins are need, we are done.
- Otherwise, let M be the total number of bins used.
- Note that all but the last bin must be full to the extent of at least $1 - \epsilon$.
- Therefore, the sum of item sizes in I is at least $(M-1)(1-\epsilon)$.
- Since this is a lower bound on $\mathrm{OPT}(I)$, we get

$$M \leq \frac{\mathrm{OPT}(I)}{1-\epsilon} + 1 \leq (1+2\epsilon)\mathrm{OPT}(I) + 1.$$

□

Performance Guarantees

- The *absolute performance ratio* for an approximation algorithm A for a minimization problem Π is given by
$$R_A := \inf\{r \geq 1 : \frac{A(I)}{\mathrm{OPT}(I)} \leq r \text{ for all instances } I \in \Pi\}.$$

- The *asymptotic performance ratio* for an approximation algorithm A for a minimization problem Π is given by
$$\begin{aligned} R_A^\infty := \inf\{r \geq 1 : \text{ for some } N \in \mathbb{Z}_+, \frac{A(I)}{\mathrm{OPT}(I)} \leq r \\ \text{for all } I \in \Pi \text{ with } \mathrm{OPT}(I) \geq N\}. \end{aligned}$$

- The last theorem gives an APTAS (i.e., an asymptotic polynomial-time approximation scheme) for Bin Packing.