## Warm-up

**Problem 1.** Check your understanding: recall their definitions, and summarise the key differences between an LP and an ILP.

**Solution 1.**

**LP:** optimize a linear function subject to linear constraints on a continuous domain. Linear programming is P-complete, meaning that (the optimisation version of) every decision problem that can be solved in polynomial time can be formulated as an LP. There exist efficient algorithms to solve LP problems.

**ILP:** ILP has the extra constraints that some of its variables have to be an integer. More general and allows to formulate more problems than LPs, but hard to solve in general.

**Problem 2.** Formulate MAX-CUT as an ILP.

a) Give its LP relaxation, and suggest a randomised rounding strategy.

b) Show that $y^* = (1, 1, \ldots, 1)$ and $x^* = (1/2, 1/2, \ldots, 1/2)$ is always an optimal solution to the LP relaxation.

c) What does your rounding scheme become in this case?

**Solution 2.** An ILP is given below:

$$\text{maximize: } \sum_{e \in E} w_e y_e$$

subject to:

$$\begin{aligned} y_e &\leq x_u + x_v && \forall e = (u, v) \in E \\ y_e &\leq 2 - (x_u + x_v) && \forall e = (u, v) \in E \\ x_v &\in \{0, 1\} && \forall v \in V \\ y_e &\in \{0, 1\} && \forall e \in E \end{aligned}$$

The LP relaxation is straightforward. Now, setting $v \in S$ with probability $x_v$ independently for each $v \in V$ means that an edge $e = (u, v) \in E$ is part of the cut with probability $x_u(1 - x_v) + x_v(1 - x_u) = x_u + x_v - 2x_u x_v$.

But for the optimal solution $x^* = (1/2, 1/2, \ldots, 1/2)$ (which might be the thing that solving the LP returns: we cannot promise it would return *another* optimal solution), this just means choosing to put each vertex in $S$ independently with probability $1/2$, and the expected value of the cut is simply $\frac{1}{2} \sum_{e \in E} w_e$. This is the simple randomised algorithm we saw a while back, and going through ILP and LP relaxation brings us nothing!

**Problem 3.** Describe how to derandomise the 3/4-approximation algorithm for MAX-SAT given in class.

**Solution 3.** We will use the method of conditional expectations. Since it is a "best-of-two" algorithm, it suffices to show that both randomised algorithm can be derandomised.

For Theorem 47, we can show that going through $x_1, \ldots, x_n$ and choosing the realization that maximises expectation gives $\frac{1}{2}$-approximation through the below inequality and applying it iteratively:

$$\begin{aligned}
&\max\{\mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) \mid x_1 = 0], \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) \mid x_1 = 1]\} \\
&\geqslant \quad \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n)] \\
&= \quad \frac{1}{2} \cdot \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) | x_1 = 0] + \frac{1}{2} \cdot \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) | x_1 = 1].
\end{aligned}$$

(And evaluating the expectation is straightforward in polynomial time.)

Similarly, one can show the same for Theorem 48,

$$\begin{aligned}
&\max\{\mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) | x_1 = 0], \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) | x_1 = 1]\} \\
&\geqslant \quad \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) | x_1 = 0] \cdot (1 - y_1^*) + \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n) | x_1 = 0] \cdot (y_1^*) \\
&= \quad \mathbb{E}[\text{Value}_\phi(x_1, \ldots, x_n)]
\end{aligned}$$

Someone suggested during the tutorial that one can also go through all the random seeds: to make it efficient, we need to check that $k$-wise independence is sufficient, where $k$ is the number of variables in the largest clause of the instance. It takes $O(k \log n)$ random bits to construct $k$-wise independence, and so iterating them takes $n^{O(k)}$ time. Without a bound on the length of the largest clause, however, we could have $k = \Theta(n)$!

## Problem solving

**Problem 4.** Consider the KNAPSACK problem, where the goal is to select a subset of $n$ items that fit in the knapsack (which can only store total weight $W$) in order to maximise total value, where item $i$ has value $v_i \geq 0$ and weight $w_i > 0$.

a) Give the corresponding ILP.

b) Provide the LP relaxation, which corresponds to the *Fractional* Knapsack.

c) Solve the LP relaxation (using, e.g., Matlab with the function linprog) on the following set of 10 items, with weight limit $W = 20$: $(v_i, w_i) = (i^2, i)$, $1 \leq i \leq 10$. See how this changes as you vary $W$ from 20 to 55.

d) Compare to the solution obtained by the Greedy algorithm for Fractional Knapsack.

e) Compare to the optimal solution of the ILP (for $W = 20$, then varying $W$ as before), also obtained by solving the ILP (on Matlab, with the function intlinprog).

**Solution 4.** Matlab code:

```
v = (1:10)^2; w=(1:10); W=20;
linprog(-v, w, [W], [],[], zeros(1,10), ones(1,10));
intlinprog(-v, 1:10, w, [W], [],[], zeros(1,10), ones(1,10));
```

a) Let $x_i \in \{0,1\}$ denote the deciding variable for picking $i$ into the knapsack.

$$\text{maximise} \sum_i v_i \cdot x_i$$

subject to

$$\sum_i w_i \cdot x_i \leqslant W.$$

$$x_i \in \{0,1\}.$$

b) Relax the previous $x_i \in [0,1]$.

c) (0, 0, 0, 0, 0, 0, 0, 0.125, 1, 1) for $W = 20$;
(0, 0, 0, 0, 0, 0, 0, 0.75, 1, 1) for $W = 25$.

d) Same results as greedy algorithm for Fractional Knapsack (sort the value from high to low by value-per-weight and pick from top): $\frac{v_1}{w_1} \geqslant \frac{v_2}{w_2} \geqslant \cdots \geqslant \frac{v_n}{w_n}$. So for this problem it is $\frac{n^2}{n} \geqslant \cdots \geqslant \frac{2^2}{2} \geqslant \frac{1^2}{1}$.

e) (1, 0, 0, 0, 0, 0, 0, 0, 1, 1)

**Problem 5.** Suppose the instance of MAX-SAT has no negated "unit clause" (that is, either a clause has length at least 2, or it is a non-negated variable $x_i$). Instead of setting each variable to 1 independently with probability $1/2$ in the "obvious" randomised algorithm, do the analysis when this is done with some (fixed) probability $p > 1/2$.

a) Show that this gives (in expectation) a $\min(p, 1-p^2)$-approximation.

b) Optimise the choice of $p$ to obtain the best approximation possible.

c) $(\star)$ Show how to remove the "no negated unit clause" assumption: let $S \subseteq [n]$ be the set of variables such that both the unit clause $\neg x_i$ *and* the unit clause $x_i$ exist in the instance $\phi$, and $T \subseteq [n]$ be the set of variables for which only the unit clause $\neg x_i$ is in $\phi$. Then consider the randomised rounding scheme with sets each variable $i$ independently to 1 with probability $p$ if $i \notin T$, and with probability $p$ (as before) otherwise, where $p$ is the value found in the previous subquestion. Show that $\text{opt}(\phi) \leq m - |S|$. Use this to conclude that $\mathbb{E}[\text{value}(\phi)] \geq p \cdot \text{opt}(\phi)$.

d) Compare this with the $1 - 1/e$ approximation guarantee obtained by LP rounding in the lecture.

**Solution 5.**

a) If a clause only has one variable, then the probability that it is satisfied is $p$ (it is non-negated). Suppose a clause has length $l$ with variables $x_1, \ldots, x_l$. The probability of satisfying it is $1 - p^a(1-p)^b$, where $a$ is the number of negated variables and $b$ is the number of non-negated variables. Since $1 > p > \frac{1}{2} > (1-p)$, we have that

$$p^a(1-p)^b < p^{a+b} \leqslant p^2.$$

So the probability of any one clause satisfying is at least $\min(p, 1-p^2)$. Denote $C_i$ the $i$-th clause. By linearity of expectation we can conclude that,

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{m} \mathbb{1}_{\{C_i=1\}}\right] &= \sum_{i=1}^{m} \mathbb{E}[\mathbb{1}_{\{C_i=1\}}] = \sum_{i=1}^{m} \Pr[\mathbb{1}_{\{C_i=1\}}] \\ &\geqslant m \cdot \min(p, 1-p^2) \\ &\geqslant \mathrm{opt}(\phi) \cdot \min(p, 1-p^2),\end{aligned}$$

where $\mathrm{opt}(\phi)$ is the value of an optimal solution for MAX-SAT of $\phi$.

b) Setting $p = 1 - p^2$ and solving for $p$, we get the optimal value $\frac{1}{2}\left(\sqrt{5}-1\right) \approx 0.618$.

c) For the variables in $S$, setting them to any value will fail at least one clause: $C_1 = (x_i), C_2 = (\neg x_i)$. Note that we assume all $m$ clauses are distinct (otherwise look at the weighted version). So, we have

$$\mathrm{opt}(\phi) \leqslant m - |S|.$$

Now by the rounding scheme we have, for every clause involved in $T$, the probability of satisfying it is at least $p$ (equivalent to negating all these variables and solving the transformed $\phi'$). For clauses (there are exactly $2|S|$ of them) involved in $S$, we will exclude the ones with negated unit clause. Denote $S_1$ the non-negated unit clause involved in $S$ and $S_2$ the negated unit clause involved in $S$ – and note that $|S_1| = |S_2| = |S|$. And $U = [m] \backslash (S \cup T)$. We will write the expectation, setting $p = 1 - p^2 \approx .618$:

$$\begin{aligned}\mathbb{E}[\mathrm{value}(\phi)] = \sum_{i=1}^{m} \mathbb{E}[\mathbb{1}_{\{C_i=1\}}] &= \sum_{i \in T \cup S} \mathbb{E}[\mathbb{1}_{\{C_i=1\}}] + \sum_{i \in U} \mathbb{E}[\mathbb{1}_{\{C_i=1\}}] \\ &\geqslant \sum_{i \in T \cup S_1 \cup U} p \\ &= p(m - |S|) \geqslant p \cdot \mathrm{opt}(\phi).\end{aligned}$$

d) $1 - \frac{1}{e} \approx 0.632 > 0.618$, so the LP rounding gives a better bound.

**Problem 6.** Show that one can also obtain (directly) an expected $\frac{3}{4}$-approximation to MAX-SAT by using only randomised rounding: in Algorithm 20, instead of having $x_i \sim \mathrm{Bern}(y_i^*)$ (independently), we will set set them independently to 1 with

probability

$$p_i := f(y_i^*),$$

where $f\colon [0,1] \to [0,1]$ is any function such that $1 - \frac{1}{4^x} \leq f(x) \leq \frac{1}{4^{1-x}}$.

a) Draw the plot of both upper and lower bounds on $f$, to see what the conditions look like (and that such functions $f$ do exist).

b) In what follows, we fix any such function $f$. With the notation of Theorem 48, show that, for any $1 \leq j \leq m$,

$$\Pr\left[\, C_j \text{ not satisfied} \,\right] \leq \frac{1}{4^{z_j^*}}$$

c) Deduce that, for any $1 \leq j \leq m$,

$$\Pr\left[\, C_j \text{ satisfied} \,\right] \geq \frac{3}{4} z_j^*$$

*Hint: use concavity.*

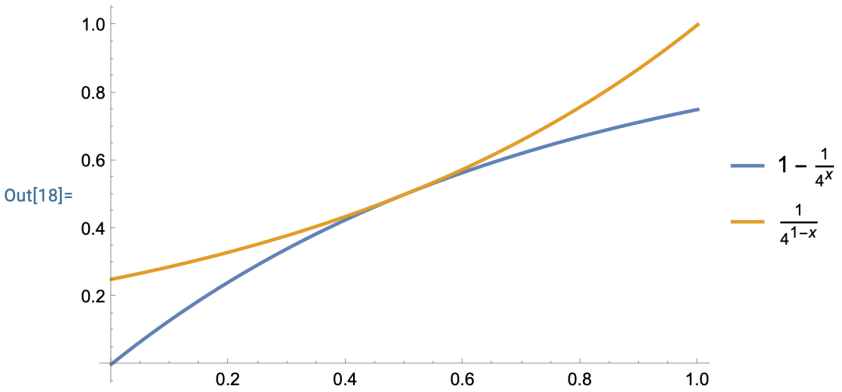
d) Conclude.

**Solution 6.**

a) In Mathematica:
Plot[{1 - 1/4^x, 1/4^(1 - x)}, {x, 0.0, 1}, PlotLegends ->"Expressions"]

```
In[18]:= Plot[{1 - 1 / 4^x, 1 / 4^(1 - x)}, {x, 0.0, 1},
          PlotLegends → "Expressions"]
```

Out[18]=



b) We have

$$\Pr[C_j \text{ not satisfied}] = \prod_{i:x_i \in C_j} (1 - f(y_i^*)) \cdot \prod_{i:\neg x_i \in C_j} f(y_i^*)$$

$$\begin{aligned}
\Pr[C_j \text{ not satisfied}] &= \prod_{i:x_i \in C_j} (1 - f(y_i^*)) \cdot \prod_{i:\neg x_i \in C_j} f(y_i^*) \\
&\leqslant \prod_{i:x_i \in C_j} \frac{1}{4^{y_i^*}} \cdot \prod_{i:\neg x_i \in C_j} \frac{1}{4^{1-y_i^*}} \\
&= 4^{-\left(\sum_{i:x_i \in C_j} y_i^* + \sum_{i:\neg x_i \in C_j} (1-y_i^*)\right)} \\
&\leqslant 4^{-z_j^*} = \frac{1}{4^{z_j^*}}.
\end{aligned}$$

c) Since $g(x) = 1 - 1/4^x$ is concave and $g(0) = 0$, we have that

$$\begin{aligned}
\Pr[C_j \text{ satisfied}] &\geqslant 1 - \frac{1}{4^{z_j^*}} = g(z_j^*) = g(z_j^* \cdot 1 + (1 - z_j^*) \cdot 0) \\
&\geqslant z_j^* \cdot g(1) + (1 - z_j^*) \cdot g(0) = \frac{3}{4} \cdot z_j^*.
\end{aligned}$$

d) By linearity of expectation and using the fact that the optimal solution to the LP is at least as good as the optimal solution to the ILP, which has value $\text{opt}(\phi)$:

$$\mathbb{E}[\text{value}(\phi)] \geqslant \frac{3}{4} \cdot \sum_{j=1}^{m} z_j^* \geqslant \frac{3}{4} \cdot \text{opt}(\phi).$$

Also fully detailed in the *Design of Approximation Algorithms* book by Williamson and Shmoys, Section 5.6 (freely available at `https://www.designofapproxalgs.com/book.pdf`).

## Advanced

**Problem 7.** Show that one can also obtain (directly) an expected $\frac{3}{4}$-approximation to MAX-SAT by using only randomised rounding with a *linear* function of $y_i^*$: in Algorithm 20, instead of having $x_i \sim \text{Bern}\, y_i^*$ (independently), set them independently to 1 with probability

$$p_i := \frac{y_i^*}{2} + \frac{1}{4}.$$