Chapter 5:
Random sampling and randomized rounding of LP's

Definition 1. *A randomized α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution for which the expected value is within a factor α of the optimal value.*

To show that a randomized algorithm ALG is an α-approximation algorithm we need to show three things:

[1] The algorithm runs in polynomial time.

[2] The algorithm always produces a feasible solution.

[3] The expected value is within a factor α of the value of an optimal solution.

In Section 1.7 we have seen a randomized algorithm for set cover that did not satisfy [2]. There, it was shown that the algorithm produces a feasible solution with *high probability*. Another example are Las Vegas algorithms which always find a feasible solution but the running time is only *polynomial in expectation*. In this chapter we only use the definition above.

Section 5.1: Max Sat and Max Cut

Max Sat

An example of an instance of the Maximum Satisfiability problem:

$$x_1 \vee x_2, \qquad \neg x_1, \qquad x_2 \vee \neg x_2 \vee x_3, \qquad \neg x_2 \vee x_4, \qquad \neg x_2 \vee \neg x_4.$$

Some notation:

- There are $n = 4$ *boolen variables* $x_i \in \{\texttt{true}\ , \texttt{false}\ \}$.
- The first *clause* is $x_1 \vee x_2$ and the number of clauses is $m = 5$.
- The third clause has 3 *literals* but only 2 variables.
- x_i is called a *positive* literal and $\neg x_i$ is *negative* literal.
- $x_i = \texttt{true} \ \Leftrightarrow \neg x_i = \texttt{false}$.
- A clause is `true` or 'satisfied' if at least one of the literals is true.

The goal in the maximum satisfiability problem is to find a `true` /`false` - assignment of the variables such that the number of satisfied clauses is maximized. (In a decision variant of the problem, called *satisfiability* (SAT), the question is wether there is an assignment that satisfies *all* clauses. That problem was the first problem to be shown $\mathcal{NP}$-complete, Cook 1971, Levin 1973).

Algorithm Set all variables independently to true with probability 1/2.

Theorem 1. *The algorithm is a 1/2-approximation for the Max Sat problem.*

Proof. Let l_j be the number of literals in clause C_j. Then,

$$\Pr(C_j \text{ is satisfied}) = 1 - (1/2)^{l_j} \geq 1/2. \tag{1}$$

Let W be the total number of satisfied clauses. Then,

$$\mathbb{E}[W] = \sum_{j=1}^{m} \Pr(C_j \text{ is satisfied}) \geqslant \frac{1}{2} m \geqslant \frac{1}{2} \textsc{Opt}.$$

□

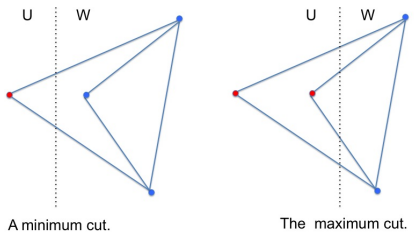
In the *weighted* version of the problem, each clause j is given a weight w_j and the goal is to maximize the total weight of the satisfied clauses. If all weights are 1 then we have exactly the unweighted version. The algorithm and proof work exactly the same:

$$\mathbb{E}[W] = \sum_{j=1}^{m} w_j \Pr(C_j \text{ is satisfied}) \geqslant \frac{1}{2} \sum_{j=1}^{m} w_j \geqslant \frac{1}{2} \textsc{Opt}.$$

Remark If each clause contains at least k literals then from the proof above we see that the approximation guarantee is at least $1 - (1/2)^k$.

Max Cut

The Max Cut problem is the maximization version of the Min Cut problem. It is known that the minimum cut in a graph can be found in polynomial time by solving a maximum flow problem (min-cut max-flow theorem). Finding the maximum cut in a graph is an $\mathcal{NP}$-hard problem.



Algorithm Assign each vertex independently and uniformly at random to one of the two sides.

Theorem 2. *The algorithm is a 1/2-approximation for the MAX CUT problem.*

Proof. The probability that an edge (i, j) ends up in the cut is exactly 1/2. Let Z be the total number of edges in the cut found by the algorithm. Then

$$\mathbb{E}[Z] = \sum_{(i,j) \in E} \Pr((i,j) \text{ in the cut}) = \frac{1}{2}|E| \geqslant \frac{1}{2} \textsc{Opt}.$$

□

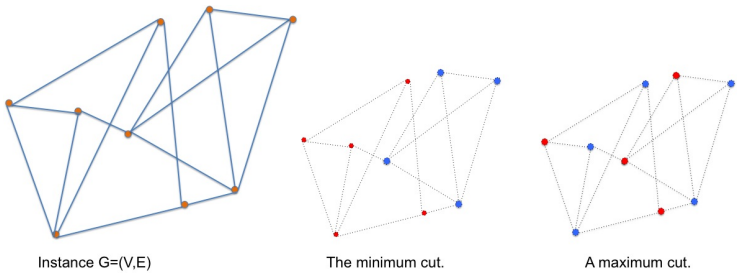


Figure 1: *Finding a maximum cut is in general much harder than finding a minimum cut. The graph $G = (V, E)$ has 15 edges. It is fairly easy to see that the minimum cut has value 2. The maximum cut, has value 11. The red vertices and blue vertices indicate the two sides, U and W, of the cut. You can think of the problem as coloring the vertices with two colors so as to minimize/maximize the number of edges with different colored endpoints.*

In the *weighted* version of the problem each edge (i, j) has a given weight w_{ij} and the goal is to maximize the total weight of the edges in the cut. The algorithm and proof work exactly the same:

$$\mathbb{E}[Z] = \sum_{(i,j)\in E} w_{ij}\Pr((i,j) \text{ in the cut}) = \frac{1}{2}\sum_{(i,j)\in E} w_{ij} \geqslant \frac{1}{2}\textsc{Opt}.$$

Section 5.2: Derandomization.

Sometimes, a randomized algorithm can easily be *derandomized*. This is the case for the algorithms of the previous section. The idea of the derandomization is to make our choices one by one and each time making a choice which gives the highest expected objective value. It is best explained by an example. In the max cut algorithm, the vertices are assigned independently at random. To derandomize the algorithm, assign the vertices one by one in arbitrary order, for example, in the order $v_1, \ldots, v_n$. Assume we have already assigned the vertices $v_1, \ldots, v_i$ and denote the assignment by S_i. Denote

$$\mathbb{E}[Z|S_i]$$

as the expected size of the cut if the first i vertices are assigned as in S_i and the others are assigned uniformly at random. Similarly, let

$$\mathbb{E}[Z|S_i \text{ and } v_{i+1} \to U], \quad \text{and} \quad \mathbb{E}[Z|S_i \text{ and } v_{i+1} \to W]$$

denote the expected size of the cut given the assignement S_i plus the assignement of v_{i+1} while the other vertices are assigned uniformly at random.

From probability theory we know that in general for a stochastic variable Z and event B it holds that $\mathbb{E}[Z] = \mathbb{E}[Z|B]\Pr(B) + \mathbb{E}[Z|\overline{B}]\Pr(\overline{B})$. In this case

$$\begin{aligned}\mathbb{E}[Z|S_i] &= \mathbb{E}[Z|S_i \text{ and } v_{i+1} \to U] \cdot \Pr(v_{i+1} \to U) \; + \\ &\quad \mathbb{E}[Z|S_i \text{ and } v_{i+1} \to W] \cdot \Pr(v_{i+1} \to W) \\ &= \mathbb{E}[Z|S_i \text{ and } v_{i+1} \to U] \cdot \frac{1}{2} \; + \; \mathbb{E}[Z|S_i \text{ and } v_{i+1} \to W] \cdot \frac{1}{2}.\end{aligned}$$

The equality above implies that either

$$\mathbb{E}[Z|S_i \text{ and } v_{i+1} \to U] \geqslant \mathbb{E}[Z|S_i] \;\text{ or }\; \mathbb{E}[Z|S_i \text{ and } v_{i+1} \to W] \geqslant \mathbb{E}[Z|S_i].$$

In the first case we assign v_{i+1} to U and assign it to W otherwise. Denote the extended assignment by S_{i+1}. Then we achieved that

$$\mathbb{E}[Z|S_{i+1}] \geqslant \mathbb{E}[Z|S_i].$$

If we do this for all vertices $v_1, \ldots, v_n$ then we end up with the assignment S_n and conclude that the value of the cut found by this derandomized algorithm is

$$\mathbb{E}[Z|S_n] \geqslant \mathbb{E}[Z|S_{n-1}] \geqslant \cdots \geqslant \mathbb{E}[Z|S_1] \geqslant \mathbb{E}[Z] \geqslant \frac{1}{2}\textsc{Opt}.$$

Section 5.3: Flipping biased coins

We have seen that setting each variable to true with probability 1/2 satisfies at least half the number of clauses in expectation. Can we do better?

$$C_1 = x_1, \quad C_2 = \neg x_1$$

The example shows that no algorithm can do better than $m/2$. But, for this example the optimal value is 1 and any assignment is optimal. So can we do better than 1/2 times the optimal value? Yes! In this section we give the first example.

An obvious approach is to set the variables independently to true with probability $p \in]0, 1[$.

If $p < 0.5$ then the ratio is worse than 0.5 for the single clause instance: $C = x_1$.
If $p > 0.5$ then the ratio is worse than 0.5 for the single clause instance: $C = \neg x_1$.

This implies that flipping a biased coin won't work if we do this independently of the instance. We will see that it is enough to let p depend only on the unit clauses.

Let's consider the weighted version of MaxSat. In fact, the algorithm and analyis are easier formulated for the weighted version. For example, we may assume w.l.o.g. that no two clauses are the same since we can turn two equal clauses into one by adding the two weights. Let $w(x_i)$ and $w(\neg x_i)$ be the weight of, respectively, clause x_i and $\neg x_i$. If the instance has no such clause then we assume it has zero weight.

Algorithm 3 Choose $p \geq 1/2$. Set x_i to true with probability p if $w(x_i) \geq w(\neg x_i)$ and set x_i to true with probability $1 - p$ otherwise.

Theorem 3. *The approximation factor of the algorithm is at least* $\min(p, 1-p^2)$*, which is*

$$\frac{1}{2}(\sqrt{5} - 1) \approx 0.62 \quad \textit{for } p = \frac{1}{2}(\sqrt{5} - 1).$$

Proof. Let Z_1 be the total weight of unit clauses satisfied by the algorithm and let Z_2 be the total weight of the other clauses satisfied by the algorithm. Fix an optimal solution and let Z_1^* and Z_2^* be the correspinding weights for the optimal solution. First, we show that

$$\mathbb{E}[Z_1] \geq pZ_1^*. \tag{2}$$

Assume $w(x_i) \geq w(\neg x_i)$. The contribution of these two clauses in the optimal solution is at most $w(x_i)$ since the clauses cannot both be satisfied. On the other hand, the expected satisfied weight by the algorithm is $pw(x_i)+(1-p)w(\neg x_i) \geq pw(x_i)$.

Similarly, if $w(x_i) \leq w(\neg x_i)$ then the contribution of these two clauses in the optimal solution is at most $w(\neg x_i)$. On the other hand, the expected satisfied weight by the algorithm is $(1 - p)w(x_i) + pw(\neg x_i) \geq pw(\neg x_i)$.

Next, we show that

$$\mathbb{E}[Z_2] \geq (1 - p^2)Z_2^*. \tag{3}$$

Since $p \geq 1/2$, any literal is false with probability at most p. If C_j has l_j literals then

$$\Pr(C_j \text{ is satisfied}) \geq 1 - p^{l_j} \geq 1 - p^2 \quad \text{for } l_j \geq 2.$$

Hence,

$$\mathbb{E}[Z_2] \geq (1 - p^2) \sum_{j:l_j \geq 2} w_j \geq (1 - p^2)Z_2^*.$$

From (2) and (3) we see that the expected value of the solution is

$$\mathbb{E}[Z_1]+\mathbb{E}[Z_2] \geq pZ_1^*+(1-p^2)Z_2^* \geq \min(p, 1-p^2)(Z_1^*+Z_2^*) = \min(p, 1-p^2)\textsc{Opt}.$$

□

Section 5.4: Randomized Rounding

For each clause C_j let P_j be the indices of the variables x_i that occur positively in the clause, and let N_j be the indices of the variables x_i that are negated in the clause. The following mixed ILP is an exact formulation of the Max Sat problem.

$$\begin{aligned}
\text{(ILP)}\ \max \quad & Z = \sum_{j=1}^{m} w_j z_j \\
s.t. \quad & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geqslant z_j && \text{for all } j = 1 \dots m, \\
& y_i \in \{0,1\} && \text{for all } i = 1 \dots n, \\
& z_j \in \{0,1\} && \text{for all } j = 1 \dots m.
\end{aligned}$$

For the relaxation, replace the last two constraints by $0 \leq y_i \leq 1$ and $0 \leq z_j \leq 1$. (Note that only changing last constraint by $0 \leq z_j \leq 1$ will still give an integral optimal solution.)

Algorithm :
Step 1. Solve the LP-relaxation $\rightarrow y^*, z^*, Z^*_{LP}$.
Step 2. Set each variable x_i to true with probability y^*_i.

Theorem 4. *The algorithm gives a* $(1 - \frac{1}{e})$*-approximation,* $(1 - \frac{1}{e} \approx 0.63)$.

Proof. Consider an arbitrary clause C_j and let l_j be the number of literals in it.

$$\begin{aligned}
\Pr(C_j \text{ not sat.}) \quad &= \quad \prod_{i \in P_j} (1 - y^*_i) \prod_{i \in N_j} y^*_i \\
&\leq_{(1)} \quad \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y^*_i) + \sum_{i \in N_j} y^*_i \right) \right]^{l_j} \\
&=_{(2)} \quad \left[1 - \frac{1}{l_j} \left(\sum_{i \in P_j} y^*_i + \sum_{i \in N_j} (1 - y^*_i) \right) \right]^{l_j} \\
&\leq_{(3)} \quad \left[1 - \frac{1}{l_j} z^*_j \right]^{l_j}
\end{aligned}$$

(1) since the arithmetic mean is at least the geometric mean (Fact 5.8 in book)
(2) rearranging and using that $|P_j| + |N_j| = l_j$
(3) follows from the LP-inequality.

From the inequality above:

$$\Pr(C_j \text{ is sat.}) \quad \geqslant \quad 1 - \left[1 - \frac{1}{l_j} z^*_j \right]^{l_j}$$

If we can derive a bound of the form $\Pr(C_j \text{ is sat.}) \geqslant \alpha z_j^*$ for some constant α then by adding weights and taking the sum over all j we get an α-approximation. Observe that the function $f(z) = 1 - (1 - \frac{1}{l_j} z)^{l_j}$ is concave on $[0, 1]$. Thus,

$$f(z) \geqslant f(0) + (f(1) - f(0))\, z = \left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) z.$$

$$\Rightarrow \quad \Pr(C_j \text{ is sat.}) \quad \geq \quad \left(1 - \left[1 - \frac{1}{l_j}\right]^{l_j}\right) z_j^*. \tag{4}$$

Now use that the right side above is more than $\left(1 - \frac{1}{e}\right) z_j^*$ fo any integer $l_j \geq 1$.

$$\mathbb{E}[W] = \sum_{j=1}^{m} w_j \Pr(C_j \text{ is sat.}) > \sum_{j=1}^{m} w_j \left(1 - \frac{1}{e}\right) z_j^* = \left(1 - \frac{1}{e}\right) Z_{LP}^* \geqslant \left(1 - \frac{1}{e}\right) \textsc{Opt}.$$

□

Section 5.5: Choosing the better of two solutions

Here, we see that if we apply for any instance both the algorithms of Section 5.4 and 5.1 and take the best of the two solutions, then the approximation ratio is better than what we have seen so far. Let W_1 and W_2 be the weight of the solution for the algorithm of, respectively, Section 5.4 and 5.1.

Theorem 5. $\mathbb{E}[\max(W_1, W_2)] \geqslant \frac{3}{4}\textsc{Opt}$.

Proof. From the Equations (1) of 5.1 and (4) of 5.4 it follows that

$$\begin{aligned}
\mathbb{E}[\max(W_1, W_2)] &\geqslant \frac{1}{2}\mathbb{E}[W_1] + \frac{1}{2}\mathbb{E}[W_2] \\
&\geqslant \frac{1}{2}\sum_{j=1}^{m}(1 - 2^{-l_j})w_j + \frac{1}{2}\sum_{j=1}^{m}\left(1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right) w_j z_j^*. \\
&\geqslant \frac{1}{2}\sum_{j=1}^{m}\left((1 - 2^{-l_j}) + 1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right) w_j z_j^*.
\end{aligned}$$

The last inequality above follows from $z_j^* \leqslant 1$. Note that

$$(1 - 2^{-l_j}) + 1 - \left(1 - \frac{1}{l_j}\right)^{l_j} \begin{cases} = 3/2 & \text{for } l_j = 1 \text{ or } l_j = 2, \\ > \frac{7}{8} + 1 - 1/e > 3/2 & \text{for } l_j \geqslant 3. \end{cases}$$

Thus,

$$\mathbb{E}[\max(W_1, W_2)] \geqslant \frac{1}{2}\sum_{j=1}^{m}\frac{3}{2}w_j z_j^* = \frac{3}{4}\sum_{j=1}^{m} w_j z_j^* = \frac{3}{4}Z_{LP}^* \geqslant \frac{3}{4}\textsc{Opt}.$$

□

Section 5.6: Non-linear randomized rounding

Section 5.5 showed a 3/4-approximation algorithm by taking the best of two solution. Here it is shown that the same ratio can be obtained if we round the LP of section 5.4 in a more sophisticated way. Instead of setting x_i to true with probability y_i^*, we set it to true with probability $f(y_i^*)$ for some appropriate function f. It turns out that this works for any function f between the following bounds.

$$1 - 4^{-y} \leqslant f(y) \leqslant 4^{y-1}. \tag{5}$$

Algorithm :
Step 1. Solve the LP-relaxation of Section 5.4 $\rightarrow y^*, z^*, Z_{LP}^*$.
Step 2. Set each variable x_i to true with probability $f(y_i^*)$ for some function $f(y)$ satisfying (5).

Theorem 6. *The algorithm above is a 3/4-approximation for (weighted) Max Sat.*

Proof.

$$\begin{aligned}
\Pr(C_j \text{ is not sat.}) &= \prod_{i \in P_j} (1 - f(y_i^*)) \prod_{i \in N_j} f(y_i^*) \\
&\leqslant \prod_{i \in P_j} 4^{-y_i^*} \prod_{i \in N_j} 4^{y_i^* - 1} \\
&= 4^{-(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} 1 - y_i^*)} \\
&\leqslant 4^{-z_j^*}
\end{aligned}$$

From the above:

$$\Pr(C_j \text{ is sat.}) \geqslant 1 - 4^{-z_j^*}.$$

Just as in Section 5.4, we replace the inequality above by a (weaker) inequality from which the approximation ratio follows immediately. Note that $g(z) = 1 - 4^{-z}$ is a concave function on $[0, 1]$. Thus,

$$g(z) \geqslant g(0) + (g(1) - g(0))z = 1 - \frac{1}{4}z \geq \frac{3}{4}z, \quad \text{for } 0 \leqslant z \leqslant 1.$$

$$\Rightarrow \quad \mathbb{E}[W] = \sum_{j=1}^{m} w_j \Pr(C_j \text{ is sat.}) \geqslant \sum_{j=1}^{m} w_j \frac{3}{4} z_j^* = \frac{3}{4} Z_{LP}^* \geqslant \frac{3}{4} \textsc{Opt}.$$

□

Section 5.7: Prize-collecting Steiner tree

Section 4.4, gave a 3-approximation algorithm by LP-rounding. A vertex i was added to the tree if $y_i^* \geqslant \alpha$ where $\alpha = 2/3$. Now we take $\alpha \in [\gamma, 1]$ uniformly at random. The value γ is chosen appropriately later.

Algorithm :
Step 1: Solve the LP-relaxation of Section 4.4 $\rightarrow \quad x^*, y^*, Z_{LP}^*$.
Step 2: Take $\alpha \in [\gamma, 1]$ uniformly at random.
Let $U = \{i \mid y_i^* \geqslant \alpha\}$. Construct a Steiner tree T on U.

Lemma 1. *The expected connection cost for the Steiner tree T is*

$$\mathbb{E}[\sum_{e \in T} c_e] \leqslant \frac{2}{1-\gamma} \ln \frac{1}{\gamma} \sum_{e \in E} c_e x_e^*.$$

Proof. By Lemma 4.6 from the book, the expected connection cost is at most

$$\mathbb{E}\left[\frac{2}{\alpha} \sum_{e \in E} c_e x_e^*\right] = \mathbb{E}\left[\frac{2}{\alpha}\right] \sum_{e \in E} c_e x_e^*.$$

Now use that $\mathbb{E}\left[\frac{2}{\alpha}\right] = \frac{1}{1-\gamma} \int_{\alpha=\gamma}^{\alpha=1} \frac{2}{x\alpha} d\alpha = \frac{2}{1-\gamma} \ln \frac{1}{\gamma}$. □

Lemma 2. *The expected total penalty cost is at most*

$$\frac{1}{1-\gamma} \sum_{i \in V} 1 - y_i^*$$

Proof. Let $U = \{i \in V : y_i^* \geqslant \alpha\}$. Note that $\Pr(i \notin U) = \begin{cases} 1 & \text{if } y_i^* \leqslant \gamma, \\ \frac{1-y_i^*}{1-\gamma} & \text{if } y_i^* \geqslant \gamma. \end{cases}$

In both cases, the probability is at most $(1-y_i^*)/(1-\gamma)$. The expected penalty cost is at most

$$\sum_{i \in V} \pi_i \Pr(i \notin U) \leqslant \sum_{i \in V} \pi_i \frac{1-y_i^*}{1-\gamma} = \frac{1}{1-\gamma} \sum_{i \in V} \pi_i (1-y_i^*).$$

□

Form the two lemmas we conclude that the total expected cost of the solution is at most

$$\max\left\{\frac{2}{1-\gamma} \ln \frac{1}{\gamma},\ \frac{1}{1-\gamma}\right\} Z_{LP}^*.$$

The maximum above is $1/(1-e^{-1/2}) \approx 2.54$ for $\gamma = e^{-1/2}$.

Derandomization The algorithm makes only one random decision: it chooses α at random. So the method of computing conditional expectations of Section 5.3 is not very helpful here. It only tells us to pick an α which gives the best objective value, but there are infinitely many α's to try. An obvious approach is to show that trying a polynomial number of value's for α is enough. In our case, we only need $n + 1 = |V| + 1$ different values of α. To see this, label the vertices such that $y_1^* \leqslant \cdots \leqslant y_n^*$. Then, the solution of the algorithm is the same for all values $\alpha \in]y_i^*, y_{i+1}^*[$. In other words, when we let α vary from γ till 1 then the outcome of the rounding changes only when α becomes y_i^* for some i. It is enough to try for α all values $y_1^*, \ldots, y_n^*, 1$ and take the best solution.

Algorithm :
Step 1: Solve the LP-relaxation of Section 4.4 $\rightarrow$ x^*, y^*, Z_{LP}^*.
Step 2: For all $\alpha \in \{y_1^*, \ldots, y_n^*, 1\}$ do the following:
Let $U = \{i \mid y_i^* \geqslant \alpha\}$. Construct a Steiner tree T on U.
Step 3: Return the best solution found.

Section 5.8: Uncapacitated facility location.

In Section 4.4 we have seen a 4-approximation algorithm for the uncapacitated facility location problem. The algorithm was to solve an LP-relaxation and then assigning clients one by to the cheapest neighboring facility in the support graph. The main difference here is that in stead of taking the cheapest facility, we take a facility at random using the values x_{ij} from the LP as probability distribution. First, let us repeat (just for completeness) part of Section 4.4:

$$\begin{aligned}
\text{(ILP)} \quad \min \quad & Z = \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in D} c_{ij} x_{ij} & \\
s.t. \quad & \sum_{i \in F} x_{ij} = 1 & \text{for all } j \in D, \\
& x_{ij} \leqslant y_i & \text{for all } i \in F, j \in D, \\
& x_{ij} \in \{0,1\} & \text{for all } i \in F, j \in D, \\
& y_i \in \{0,1\} & \text{for all } i \in F.
\end{aligned}$$

In the LP-relaxation, replace the binary constraint by $x_{ij} \geqslant 0$ and $y_i \geqslant 0$. The dual of the LP-relaxation is:

$$\begin{aligned}
\text{(D)} \quad \max \quad & Z = \sum_{j \in D} v_j & \\
s.t. \quad & \sum_{j \in D} w_{ij} \leqslant f_i & \text{for all } i \in F, \\
& v_j - w_{ij} \leqslant c_{ij} & \text{for all } i \in F, j \in D, \\
& w_{ij} \geqslant 0 & \text{for all } i \in F, j \in D, \\
& (v_i \text{ is free}). &
\end{aligned}$$

Let x^*, y^* be optimal primal solution and let v^*, w^* be optimal dual solution. By complementary slackness, we have the following lemma.

Lemma 3. *If $x^*_{ij} > 0$, then $c_{ij} = v^*_j - w^*_{ij} \leqslant v^*_j$.*

- *Support graph* of x^*: There is an edge if $x^*_{ij} > 0$.
- $N(j)$ is the set of neighbors of client $j \in D$ in the support graph.
- $N^2(j)$ is the set of all neighbors of neighbors of client $j \in D$.

The deterministic algorithm:
For $k = 1, 2, \ldots$ until all clients are connected do:
Step 1: Among the unconnected clients, choose client j_k with smallest value $v^*_{j_k}$.
Step 2: Choose facility $i_k \in N(j_k)$ with smallest value f_{i_k}.
Step 3: Connect all still unconnected clients in $N^2(j_k)$ to facility i_k.

In the randomized version, Step 2 is changed by taking i_k at random. Consequently, we see that the analysis work out nicely if we also make a small change in Step 1. Define the *fractional connection cost* of client j as $C_j^* = \sum_{i \in F} c_{ij} x_{ij}^*$.

The randomized algorithm:
For $k = 1, 2, \ldots$ until all clients are assigned do:
Step 1: Among the unconnected clients choose client j_k with smallest $v_{j_k}^* + C_{j_k}^*$.
Step 2: Choose facility $i_k \in N(j_k)$ at random, where $\Pr(i = i_k) = x_{ij}^*$.
Step 3: Connect all still unconnected clients in $N^2(j_k)$ to facility i_k.

Theorem 7. *Algorithm above is a randomized 3-approximation algorithm.*

Proof. The expected opening cost for facility opened in iteration k is

$$\sum_{i \in N(j_k)} f_i x_{ij_k}^* \leqslant \sum_{i \in N(j_k)} f_i y_i^*.$$

The inequality follows from the LP-constraint. Step 3 of the algorithm ensures that $N(j_k) \cap N(j_{k'}) = \emptyset$ for any pair $j_k, j_{k'}$. Thus, the total expected opening cost is

$$\sum_k \sum_{i \in N(j_k)} f_i y_i^* \leqslant \sum_{i \in F} f_i y_i^*.$$

For the expected connection cost, consider an arbitrary iteration k. Given

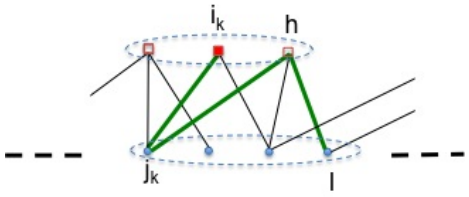


Figure 2: *Sketch of iteration k. Only the clients that get connected in this iteration are shown. Client l is an arbitrary client in $N^2(j_k)$.*

j_k, the neighborhoods $N(j_k)$ and $N^2(j_k)$ are fixed. For any client l that gets connected in that iteration (i.e., $l \in N^2(j_k)$) there is some $h \in N(j_k)$ such that h is a neighbor of l. The distances $c_{hj_k} + c_{hl}$ are *not* random variables. However, the distance $c_{i_k j_k}$ is a random variable since the facility i_k is chosen at random.

$$\mathbb{E}[c_{i_k j_k}] = \sum_{i \in N(j_k)} x_{ij_k}^* c_{ij_k} = \sum_{i \in F} x_{ij_k}^* c_{ij_k} = C_{j_k}^*.$$

The algorithm assigns client l to facility i_k. The expected connection cost for

client l is

$$\begin{aligned}
\mathbb{E}[c_{i_k l}] &\leq_{(1)} \mathbb{E}[c_{i_k j_k} + c_{h j_k} + c_{hl}] \\
&=_{(2)} \mathbb{E}[c_{i_k j_k}] + c_{h j_k} + c_{hl} \\
&= C^*_{j_k} + c_{h j_k} + c_{hl} \\
&\leq_{(3)} C^*_{j_k} + v^*_{j_k} + v^*_l \\
&\leq_{(4)} C^*_l + v^*_l + v^*_l
\end{aligned}$$

(1): Follows from the triangle inequality.
(2): Only the first is a random variable (given j_k).
(3): From the complementary slackness lemma.
(4): From Step 1. Note that j_k and l were both unassigned before j_k was chosen.

The expected value of the solution is the sum of expected opening cost and expected connection cost, which is bounded by

$$\begin{aligned}
& \sum_{i \in F} f_i y^*_i + \sum_{l \in D} C^*_l + 2 \sum_{l \in D} v^*_l \\
= & \left(\sum_{i \in F} f_i y^*_i + \sum_{l \in D} \sum_{i \in F} x^*_{il} c_{il} \right) + 2 \sum_{l \in D} v^*_l \\
= & \; Z^*_{LP} + 2 Z^*_{\mathrm{D}} \;\leqslant\; 3 Z^*_{LP} \;\leqslant\; 3\textsc{Opt}.
\end{aligned}$$

□