CS 787: Advanced Algorithms

LP-Based Approximations

Instructor: Dieter van Melkebeek

We present methods for constructing approximation algorithms to NP-hard optimization problems using linear programming (LP). We typically obtain an LP relaxation of the problem by formulating it as an integer linear program and dropping the integrality constraints. We then solve the LP relaxation exactly, and apply various rounding strategies (including randomized rounding and iterative rounding) to obtain a valid integral solution that is close to optimal. Alternately, we exploit LP duality and employ a primal-dual approach, or consider a Lagrangian relaxation to deal with difficult constraints.

1 Basic Approach

We explain the basic approach at a high level and illustrate it with Set Cover. Many NP-hard combinatorial optimization problems can be formulated as LPs with additional integrality constraints. The fact that there are integrality constraints makes these LPs much harder to solve. LP with integrality constraints is called Integer Linear Programming (ILP) and is an NP-complete problem.

Example: Set Cover:
Given a set of elements $[m] \doteq \{1, 2, \ldots, m\}$, a family $\mathcal{S}$ set of n subsets $S_1, S_2, \ldots, S_n \subseteq [m]$ whose union equals $[m]$, and weights $w_1, w_2, \ldots, w_n$.
Goal: Find a subfamily of $\mathcal{S}$ whose union equals $[m]$ and has smallest totall weight.

We formulate Set Cover as an ILP by introducing an indicator variable x_i for every $i \in [n]$, with the intended meaning that $x_i = 1$ if S_i is taken up into the cover, and 0 otherwise.

Objective Function: $\min \sum_{i=1}^{n} w_i x_i$ (minimize the total weight of the sets in the cover)
Constraints: $(\forall\, j \in [m])\ \sum_{i:j \in S_i} x_i \geq 1$ (cover all elements).
Integrality constraint: $(\forall\, i \in [n])\, x_i \in \{0, 1\}$ (every set is either in the set cover or not).

More explicitly, the integrality constraint can be rewritten as the conjunction of the linear constraints $0 \leq x_i \leq 1$ and the mere integrality constraint that x_i is integer. Alternately, we could just require that $x_i \geq 0$ and that x_i be integer. ⊠

The process to construct an approximately optimal solution to this problem has two steps:

1. Look at the relaxation of the problem: remove integrality constraints and use linear constraints. We can then solve the problem optimally using a LP algorithm.

$$OPT_{LP} \leq OPT \tag{1}$$

   OPT_{LP} is the optimal solution of the relaxed LP, which can be constructed efficiently.

2. Exploit above to construct a valid solution that also satisfies the integrality constraints and has cost at most $\rho \cdot OPT_{LP}$.

This way of constructing approximations puts limitations on which approximation ratios you can achieve.

Definition 1 (Integrality Gap)**.**

$$\gamma(n) = \max_{|I|=n} \frac{OPT(I)}{OPT_{LP}(I)} \tag{2}$$

$\gamma(n)$ is a lower bound for $\rho(n)$. Suppose you could realize a $\rho(n) < \gamma(n)$, then you could reach a solution less than OPT for some instance I.

Example: For our LP-relaxation of Set Cover one can show that

$$\gamma(n) = \Theta(\log n).$$

⊠

Now we introduce two approaches for solving step (2) from above:

- Rounding
  Round an optimal solution for the LP. Rounding must be done in a way that does not violate the linear constraints and in addition satisfies the integrality constraints and does not deteriorate the objective function by too much.
- Primal - Dual
  Construct some possibly invalid integral primal solution and a valid dual solution iteratively. Use the value of the dual solution as a bound on OPT_{LP} instead of OPT_{LP} itself. This may lead to a more efficient algorithm because you do not have to make a call to a black box LP algorithm. On the other hand, this approximation may have a worse approximation ratio ρ.

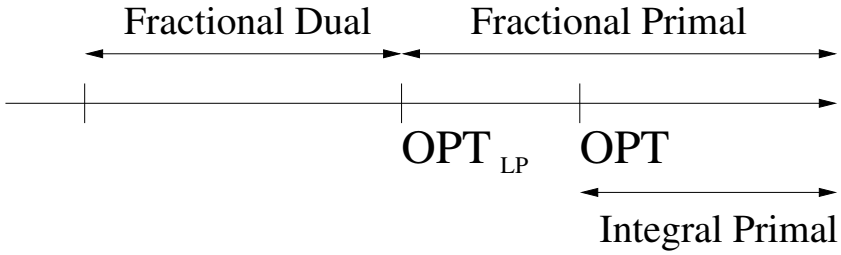


Figure 1: Solution space of optimization algorithm

Example: Set Cover

$$\begin{array}{ccc} \textbf{Primal} & & \textbf{Dual} \\ \min \sum_{i=1}^{n} w_i x_i & & \max \sum_{j=1}^{m} y_j \\ (\forall j) \sum_{i:j \in S_i} x_i \geq 1 \, (y_j) & \Longleftrightarrow & (\forall i) \sum_{j \in S_i} y_j \leq w_i \, (x_i) \\ x_i \geq 0 & & y_j \geq 0 \end{array}$$

The primal problem is a covering problem; to cover a universe with the minimum possible cost. The dual problem is a packing problem; to pack as much as possible into each set S_i (maximum of w_i per set).

⊠

We now look at applying the two paradigms for step (2) to Set Cover.

1.1 Simple rounding

Let x^* be an optimal solution to the LP relaxation. Try to distill an integral solution. Consider the maximum number of times an element can occur over all sets, $f = \max_j |\{i : j \in S_i\}|$. At least one term in $\sum_{i:j \in S_i} x_i \geq 1$ is at least $1/f$; round all terms greater or equal to $1/f$ up to 1 and the rest down to 0. More formally, set

$$x_i = \begin{cases} 1 & \text{if } x_i^* \geq 1/f \\ 0 & \text{otherwise.} \end{cases}$$

or equivalently, let

$$I = \{i : x_i^* \geq 1/f\}.$$

Determining I can be done in polynomial time.

Claim 1. *I is a set cover.*

Proof. For every element, at least one of the terms in the linear constraints of the primal problem is at least $\frac{1}{f}$ that is, it is covered by I. □

Claim 2. *The solution obtained by rounding is at most $f \cdot OPT_{LP}$.*

Proof.

$$\begin{aligned} \sum_{i \in I} w_i &\leq f \cdot \sum_{i \in I} w_i x_i^* \quad (\text{since for } i \in I,\ x_i^* \geq \frac{1}{f}) \\ &= f \cdot OPT_{LP} \end{aligned}$$

□

So this is a factor-f approximation algorithm for the SET COVER problem, which is a generalization of the Vertex Cover(VC) problem. For VC, the vertices are the sets and the edges are the elements of the universe. Note that $f = 2$ for VC, because each edge is covered by two vertices.

This method of rounding works well for the SET COVER problem, but may not apply well to all problems in general. So let us look at other methods.

1.2 Rounding based on Dual

Let y^* be the optimal solution for the dual problem. Let us construct J' in the following way.

$$I' = \{i : \text{ dual constraint for } x_i \text{ is binding for } y^*\}.$$

This means that the dual constraint corresponding to x_i is an equality: $\sum_{j \in S_i} y_j^* = w_i$.

Claim 3. *I' is a valid set cover.*

Proof. We claim that $I \subset I'$, which implies the claim because I is a valid set cover.

Consider an element $i \in I$. Membership to I means that $x_i^* \geq \frac{1}{f}$, so the dual corresponding to x_i^* is binding (by Complementary Slackness). So, $i \in I'$, and $I \subseteq I'$. □

Note that this solution is no better than the previous method since the weight of I' is at least as large as the weight of I.

Claim 4. *The solution obtained by rounding using the dual is at most f times OPT_{LP}*

Proof.

$$\begin{aligned}\sum_{i \in I'} w_i &= \sum_{i \in I'} \sum_{j \in S_i} y_j^* \\ &\leq f \cdot \sum_{j=1}^{m} y_j^* \\ &= f \cdot OPT_{LP}\end{aligned}$$

The first step follows from the fact that for every x_i with $i \in I'$, the corresponding constraint in the dual is tight, i.e., $w_i = \sum_{j \in S_i} y_j^*$. The second step follows from the fact that y_j^* can occur a maximum of f times. □

So, for the Set Cover problem, rounding based on dual also gives a factor-f approximation algorithm. Now let us see another way of approximating LP problems, without actually making any calls to the LP black box.

1.3 Primal-Dual Algorithm

For A' all that we needed was:

1. An integral primal solution that is not necessarily feasible, namely a set I that may not cover all elements of the universe.
2. For every i such that $x_i > 0$, the corresponding dual constraint is binding.
3. y is a feasible dual solution.

These were the only conditions we needed for the proof of Claim 4. In particular, y need not be an optimal dual solution – as soon as y is feasible, we know that the objective function is no more than OPT_{LP}, which is all we need for the proof of Claim 4.

At each point in the algorithm below, $I' = \{i : i\text{th dual constraint is binding for } y\}$.

Algorithm 1: Primal-Dual Algorithm for Set Cover using LP
Input: An instance of the Set Cover problem (SC)
Output: The approximate Set Cover I' of SC
PRIMAL_DUAL(SC)
(1) **while** $(\exists j)$ $j \notin \cup_{i \in I'} S_i$
(2) Increase y_j such that at least one of the dual
(3) constraints containing y_j becomes binding
(4) **return** I'

The condition in the while loop is true only if there exists some j which is not covered by I'. In this case, all the dual constraints with y_j are strict inequalities, otherwise the constraint would be binding and j would belong to some S_i for $i \in I'$. So we can increase y_j without violating the constraints.

Note that the above is a polynomial time algorithm. Consider the following reasoning: once a constraint is binding in the algorithm, it remains so since $y_j's$ can only increase. Therefore, there will be at most m iterations of the while loop. Also, at the end of the algorithm, all three required conditions are met which implies this algorithm produces a factor-f approximation.

Let us see how the required conditions are met:

- Condition 1 is satisfied since it is the halt condition of the while loop
- Condition 2 is satisfied by the definition of I'.
- Condition 3 is satisfied since the algorithm starts with a feasible solution for y and increases it without violating any of the constraints.

1.4 LP view of greedy approximation

The LP-based approximations we obtained so far all yield an approximation factor of f. Note that f can be as large as n. Earlier we developed a greedy algorithm that yields the best known factor, namely logarithmic.

Recall that in the greedy algorithm for the set cover problem, elements are added in the order $u_1, u_2, \ldots, u_m$. The price of adding an element u_j is defined as

$$PRICE(u_j) = \text{Marginal cost per element at the time } u_j \text{ is added.}$$

The cost of the greedy solution is,

$$\sum_{j=1}^{m} PRICE(u_j)$$

Given the interpretation of the dual variables as marginal increase in the objective per unit of tightening the corresponding primal constraint, it is plausible to set $PRICE(u_j)$ to be y_j. However, this may violate the packing constraints of the dual. To fix this, we can divide all the y_j's by a big enough constant factor h so as to get a feasible dual solution. This is known as *dual fitting*.

The cost of the greedy solution is

$$\begin{aligned}\sum_{j=1}^{m} PRICE(u_j) &= h \cdot \sum_{j=1}^{m} y_j \\ &\leq h \cdot OPT_{LP} \\ &\leq h \cdot OPT\end{aligned}$$

Now we need to find a small value of h that is guaranteed to work.

For a fixed i, let $S_i = \{u_{j_1}, u_{j_2}, \ldots, u_{j_{|S_i|}}\}$, where $j_1 < j_2 < \ldots < j_{|S_i|}$. As we argued when we covered the greedy algorithm,

$$\begin{aligned}PRICE(u_{j_k}) &= \text{marginal cost for } S_i \text{ when } u_{j_k} \text{ is considered} \\ &\leq \frac{w_i}{|S_i| - k + 1}\end{aligned}$$

therefore,

$$\sum_{k=1}^{|S_i|} PRICE(u_{j_k}) \leq H_{|S_i|} \cdot w_i.$$

Therefore, we can see that the constraint for S_i in the dual formulation issatisfied if $h \geq H_{|S_i|}$ for every i.

By setting $h = H_g$, where $g = \max |S_i|$, we show that the greedy solution gives us a factor H_g approximation. Since $g \leq m$, we obtain a *rho*-approximation with $\rho = H_m \sim \ln(m)$.

2 Randomized Rounding

We'll discuss the MAX SAT problem to show how randomized rounding can be used to obtain LP approximations.

MAX SAT problem: Given m clauses $c_1, c_2, \ldots, c_m$ on n variables and weights $w_1, w_2, \ldots, w_m$; the goal is to find an assignment that maximizes W , where W is defined as

$$\sum_{j:c_j \text{ is satisfied}} w_j$$

2.1 A Trivial Approximation Algorithm

A trivial approximation algorithm would be to set each variable x_i to 1 with probability $\frac{1}{2}$ independently. Then,

$$\Pr[c_j \text{ is satisfied}] = 1 - \left(\frac{1}{2}\right)^{|c_j|}$$

Therefore,

$$\begin{aligned} E[W] &= \sum_{j=1}^{m} w_j \left(1 - \left(\frac{1}{2}\right)^{|c_j|}\right) \\ &\geq \frac{1}{2}\sum_{j=1}^{m} w_j \\ &\geq \frac{1}{2} OPT \end{aligned} \tag{3}$$

Note that the approximation factor improves as the minimum clause size gets larger. For MAX 3-SAT we get a factor of $\frac{7}{8}$. It can be shown that this factor is tight for MAX 3-SAT.

The above only gives us the expected value of W, this does not guarantee that we'll get a approximate solution every time. One way to take care of this is to find the probability of getting the above bound. If the probability is greater that $\frac{1}{2} + \frac{1}{poly(.)}$, we can run the algorithm multiple times and choose the best solution. The other way is to derandomize using conditional expectations. The derandomization should involve polynomial steps. This can be done by finding $E[W|x_i = 0]$ and $E[W|x_i = 1]$. Choose the setting which gives $E[W] \geq \frac{1}{2}\sum_{j=1}^{m} w_j$ and continue setting the other variables in the same manner.

We'll now improve this approximation factor by setting a variable x_i to 1 with a probability different from $\frac{1}{2}$. This is similar to flipping bent coins.

2.2 Flipping bent coins

We'll set each variable x_i independently such that

$$\Pr[x_i = 1] = p$$

where $p \geq \frac{1}{2}$. Without loss of generality, assume that for each x_i,

$$\sum_{c_j = x_i} w_j \geq \sum_{c_j = \overline{x_i}} w_j$$

Therefore,

$$\begin{aligned} E\left[\text{Weight of unit clauses that are satisfied}\right] &= \sum_{c_j \text{ is a positive unit clause}} p w_j + \\ &\quad \sum_{c_j \text{ is a negative unit clause}} (1-p) w_j \\ &\geq \sum_{c_j \text{ is a positive unit clause}} p w_j \end{aligned}$$

The probability that a non unit clause is satisfied is given as,

$$\begin{aligned} \Pr\left[c_j \text{ is satisfied}\right] &= 1 - \left(p^{|N_j|}\left(1-p\right)^{|P_j|}\right) \\ &\geq 1 - p^2 \end{aligned}$$

Therefore,

$$E\left[\text{Weight of non unit clauses that are satisfied}\right] \geq \sum_{c_j \text{ is a non unit clause}} \left(1-p^2\right) w_j$$

Therefore,

$$\begin{aligned} E[W] &= E\left[\text{Weight of unit clauses that are satisfied}\right] + \\ &\quad E\left[\text{Weight of non unit clauses that are satisfied}\right] \\ &\geq \min\left(p,(1-p^2)\right) \sum_{c_j \text{ is non unit or positive unit}} w_j \\ &\geq \min\left(p,(1-p^2)\right) OPT \end{aligned}$$

because,

$$OPT = \sum_{c_j \text{ is positive unit}} w_j - \sum_{c_j \text{ is negative unit}} w_j$$

This works whenever $p \geq \frac{1}{2}$ but it works best when $p = 1-p^2$ which is when $p = .608$. So we have a factor of .608.

So how can we improve on this further? We can pick a bias differently for each variable.

2.3 Using separate probabilities for every variable

This is where linear programming comes in. The following is a linear programming relaxation of MAX SAT.

$$\max \sum_{j=1}^{m} w_j y_j \tag{4}$$

s.t.

$$y_j \leq \sum_{i \in P_j} z_i + \sum_{i \in N_j} (1-z_i) \tag{5}$$

and

$$y_j \leq 1 \tag{6}$$

$$0 \leq z_i \leq 1 \tag{7}$$

We will now solve this linear program relaxation to get values for the y^*'s and z^*'s. We will set x_i independently such that

$$\Pr(x_i = 1) = z_i^* \tag{8}$$

The probability that c_j is not satisfied is the probability that every variable gets the wrong sign. This can be written as

$$\prod_{i \in P_j} (1-z_i^*) * \prod_{i \in N_j} z^* \tag{9}$$

which, by the geometric mean *leq* arithmetic mean inequality can be bounded as

$$\leq \left[\frac{1}{|c_j|}\left(\sum_{i\in P_j}(1-z_i^*)+\sum_{i\in N_j}z_i^*\right)\right]^{|c_j|} \tag{10}$$

$$\leq \left(1-\frac{y_j^*}{|c_j|}\right)^{|c_j|} \tag{11}$$

We would like to make the equation above linear in y_j so that we can relate it to our objective function.

$$Pr[c_j \text{ is satisfied}] \geq 1-\left(1-\frac{y_j^*}{|c_j|}\right)^{|c_j|} \tag{12}$$

$$\geq \left[1-\left(1-\frac{1}{|c_j|}\right)^{|c_j|}\right] * y_j^* \tag{13}$$

$$\geq \left(1-\frac{1}{e}\right) * y_j^* \tag{14}$$

Therefore the expected weight can be written as

$$E[W] = \sum_{j=1}^{m} w_j * Pr[c_j \text{ is satisfied}] \tag{15}$$

$$\geq \left(1-\frac{1}{e}\right) * \sum_{j=1}^{m} w_j * y_j^* = \left(1-\frac{1}{e}\right) * OPT_{LP} \tag{16}$$

$$\geq \left(1-\frac{1}{e}\right) * OPT = .632 * OPT \tag{17}$$

So this gives a factor of .632 * OPT which is a little better than .618 * OPT. The interesting thing to note is that this approach actually behaves better as the clauses get shorter which is the opposite behavior of our previous approach which behaves better as clauses get longer.

We would like to get the best of both worlds, and to accomplish this we can run both algorithms and take the best of the two. Therefore the expected value that a clause is satisfied by an assignment chosen is $\geq \frac{1}{2}$ (E[c_j is satisfied by ALG1] + E[c_j is satisfied by ALG2]). This can be written as

$$\geq \frac{1}{2}\left[1-\left(\frac{1}{2}\right)^{|c_j|}+\left(1-\left(1-\frac{1}{|c_j|}\right)^{|c_j|}\right) * y_j^*\right] \tag{18}$$

$$\geq \frac{3}{4} * y_j^* \tag{19}$$

We can therefore get an approximation ratio of $\frac{3}{4}$ which is the best we can do with this approach. Consider the equations:

$$x_1 \vee x_2 \tag{20}$$

$$x_1 \vee \overline{x}_2 \tag{21}$$

$$\overline{x}_1 \vee x_2 \tag{22}$$

$$\overline{x}_1 \vee \overline{x}_2 \tag{23}$$

OPT = 3, and for our linear program, we can set all $z_i = \frac{1}{2}$ and all $y_j = 1$ which will get us $OPT_{LP} = 4$ Therefore we have an example that shows this approximation is tight.

3 Iterative Rounding

In general, however, rounding may not always lead to a valid solution. We present a different rounding technique called Iterative Rounding which handles this issue in some cases.

As before we will solve the LP Relaxation exactly and then round up those components which are larger than 1/2. The additional step is to incorporate the rounded solution in the LP and iterate until a valid solution is found. We will apply Iterative Rounding to obtain a factor 2 approximation for the Survivable Network Design Problem.

The Survivable Network Design problem is defined as follows:

Given: undirected graph $G = (V, E)$, edge weights $w_e \geq 0$, requirements $r(u, v)$ $\forall u, v \in V$

Goal : Find a subgraph $F \subseteq E$ s.t $\forall u, v \in V$ the number of edge-disjoint paths between u and v in (V, F) is $\geq r(u.v)$ and

$$\sum_{e \in F} w_e \text{ is minimized}$$

Notes:

- We can think of the graph G as a network where the vertices V are routers and the edges E are links between the routers. For each pair of routers u and v, we would like a guarantee of a variety of different routes between the routers so that if a single router in the network fails, u and v will still be connected in the network.
- We will assume that $F = E$ is a solution that works.

  We can efficiently verify whether there exists r disjoint paths from (u, v) using the max-flow algorithm. To do so, set all edge capacities to 1, find the max flow using u as the source and v as the sink and verify that the flow is $\geq r$. We can use this verification to check if $F = E$ is a valid solution by considering all pairs of vertices (u, v).

3.1 LP relaxation

Unlike previous examples we have seen, the ILP formulation that corresponds to the Survivable Network Design problem is not immediately clear. The formulation will be based on the following lemma:

Lemma 1. *For $x_e \geq 0$ and $r(u, v) \geq 0$, consider graph G with edge capacities x_e. Then, $\forall u, v \in V$ there is a flow in G from u to v of value $\geq r(u, v)$.* $(*)$

$$\Updownarrow$$

$$(\forall S \subseteq V) \sum_{e \in \delta(S)} x_e \geq \max_{u \in S, v \in \bar{S}} r(u, v)$$

where $\delta(S)$ is the set of edges that cross the cut between S and $\bar{S}$.

Proof.

($\Rightarrow$) This direction follows from the max-flow min-cut theorem. Fix a cut S. If there exists a flow in G, $\forall(u,v)$ of value $\geq r(u,v)$, then the sum of the edge capacities crossing S must be $\geq \max_{u\in S, v\in\bar{S}} r(u,v)$.

($\Leftarrow$) This direction also follows from the max-flow min-cut theorem. Fix any two vertices u, v. The right hand side implies that the flow between (u,v) is $\geq r(u.v)$.

□

Notes:

- If $x_e \in \{0,1\}$ and $r(u,v)$ is integral then $(*)$ is exactly what we need to find for the survivable network design problem
- The above lemma applies even if the x_e's and r's are non-integers.

In the ILP below, x_e will indicate whether $e \in F$. Let $f_0(S) = \max_{u\in S, v\in\bar{S}} r(u,v)$. The ILP formulation below exactly characterizes our problem for $f = f_0$.

$$\min \sum_{e\in E} w_e x_e$$

$$s.t(\forall S \subseteq V) \sum_{e\in\delta(S)} x_e \geq f(S)$$

$$x_e \in \{0,1\}$$

The LP relaxation is obtained by replacing the constraint $x_e \in \{0,1\}$ with $0 \leq x_e \leq 1$.

On first observation it seems that we won't be able to solve this LP in polynomial time since there are exponentially many subsets of V which would require us to handle exponentially many constraints. However, so solve an LP in polynomial time it suffices that the number of variables is polynomial, and that we can provide a separation oracle that runs in polynomial time.

For the problem that we are looking at, the input to the separation oracle will be a candidate x with $0 \leq x_e \leq 1 \forall e \in E$. To get a separation oracle we use the same trick that was described to verify if $F = E$ is a valid solution - run the max-flow algorithm with edge capacities set to the x_e's and verify that the flow is $\geq f(S)$. By lemma 1, if this flow cannot be realized we can find a cut S, which violates the constraints. Also note that since the LP above has one variable per edge, the number of variables is polynomial. Together with the separation oracle that runs in polynomial time, this allows us to solve the LP in polynomial time.

3.2 Iterative rounding algorithm and analysis

The general idea for the algorithm is as follows. We hope the LP given above will produce some $x_e \geq \frac{1}{2}$, which we will round up to 1. It turns out this will be the case if $f(S)$ is weakly supermodular (defined below). After rounding, we will modify the original LP and iterate until we find a valid solution. We will see that to modify the LP we will only need to change equation $f(S)$.

Definition 2 (Weakly Super-Modular). *$f : 2^V \to \mathbb{R}$ is weakly super-modular if $f(V) = 0$ and $\forall A, B \subseteq V$*

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B) \text{ OR}$$
$$f(A) + f(B) \leq f(A - B) + f(B - A)$$

Note that a super-modular function satisfies both conditions.

Lemma 2. *f_0 is weakly super-modular.*

The proof is left as an exercise. Our algorithm will depend on the following key lemma whose proof we omit.

Lemma 3 (Key Lemma). *For any weakly super-modular f each vertex of the linear program given above (and therefore any optimal solution to the linear program) has at least one e s.t $x_e \geq \frac{1}{2}$*

This leads to the following algorithm:

Algorithm 2: Iterative Rounding for Survivable Network Design
Input: An instance of the Survivable Network problem (SN)
Output: The approximate subgraph F of SN
ITERATIVE_ROUNDING(SN)
(1) $F \leftarrow \emptyset$
(2) **while** (F is not a valid solution to SN)
(3) $x \leftarrow$ solution of LP for $G = (V, E - F)$ with
(4) $f(S) = f_0(S) - |\delta(S) \cap F|$
(5) $F \leftarrow F \cup \{e | x_e \geq \frac{1}{2}\}$
(6) **return** F

Correctness

- The first question to ask is whether the above algorithm will halt. The answer is yes, provided each $f(S)$ is weakly super-modular. If this is the case then in each iteration we will round at least one edge up to 1. This edge will be added to F. Since $F = E$ is a valid solution (by assumption), we will iterate for at most $|E|$ steps.

  To see that each $f(S)$ is weakly super-modular, note that $\forall F \subseteq E$, $g(S) = |\delta(S) \cap F|$ is sub-modular. This is straightforward to verify given the following definition.

  Definition 3 (Sub-modular). *$g : 2^V \to \mathbb{R}$ is sub-modular if $g(V) = 0$ and $\forall A, B \subseteq V$*

  $$g(A) + g(B) \geq g(A \cup B) + g(A \cap B) \text{ AND}$$
  $$g(A) + g(B) \geq g(A - B) + g(B - A)$$

  Since $g(S)$ is sub-modular and we get $f(S)$ by subtracting $g(S)$ from $f_0(S)$ in step (4) of the algorithm it ensures that $f(S)$ remains weakly super-modular (subtracting a sub-modular function from a weakly super-modular one yields a weakly super-modular one).

- Can we solve each LP in polynomial time (i.e., construct separation oracles efficiently)? The answer here is yes. We use the same trick that was used to produce the separation oracle for the original problem - use the max flow algorithm by setting the edge capacities to 1 for edges in current solution F and verifying that the flow is $\geq f(S)$.

Approximation ratio We argue that the algorithm gives a factor-2 approximation.

Lemma 4. *The following is true of the F that is returned by the algorithm*

$$\sum_{e \in F} w_e \leq 2 \sum_{e \in E} w_e x_e^{(0)}$$

where $x^{(0)}$ is the solution to the LP in the first iteration of the loop.

Notice that the right hand side of the above inequality is equal to OPT_{LP} (the optimal solution of the original LP relaxation), and we know that $OPT_{LP} \leq OPT$. This lemma implies that the algorithm is a factor-2 approximation.

Proof. We will prove by induction on the number of iterations of the while loop. Suppose the while loop runs for k iterations. We will assume that $F = \emptyset$ is not a valid solution, so the base case will be $k = 1$.

base case: For $k = 1$, the loop only runs one time, so this case is similar to the vertex cover factor 2 approximation algorithm. The algorithm is a factor 2 approximation algorithm because each x_e that is taken is increased by at most a factor of 2.

induction step: For this case, we can assume $k \geq 2$. We will analyze the algorithm in terms of two phases:

A) The first iteration of the while loop. This phase adds F_0 to F, and the solution to the LP denoted $LP^{(0)}$ in this phase is $x^{(0)}$.

B) All remaining iterations of the while loop. This phase adds F_1 to F, and the first iteration in this phase will have a LP denoted $LP^{(1)}$ with solution $x^{(1)}$.

First let us consider the A) phase:

$$\sum_{e \in F_0} w_e \leq 2 \sum_{e \in F_0} w_e x_e^{(0)}$$

because all we have done after phase A) is to round up x_e such that $e \in F_0$ from at least $1/2$ to 1.

Now we analyze the B) phase. Assume the lemma holds for any instance that takes k iterations, and consider an instance $G = (V, E)$ that takes $k + 1$ iterations. Notice that the B) phase is equivalent to running the algorithm on $G_1 = (V, E - F_0)$ and initial function $f_1(S) = f_0(S) - |\delta(S) \cap F_0|$. Running the algorithm on this instance will require k iterations, so the induction hypothesis says that

$$\sum_{e \in F_1} w_e \leq 2 \sum_{e \in E - F_0} w_e x_e^{(1)}.$$

The following key observation will allow us to conclude the analysis for the B) phase.

Observation 1 (Key Observation) $x^{(0)}$ restricted to $E - F_0$ is a feasible solution to $LP^{(1)}$.

Proof. Fix a set $S \in V$. Because $x^{(0)}$ is a solution to $LP^{(0)}$, it satisfies the constraints of $LP^{(0)}$, namely

$$\sum_{e \in E, e \in \delta(S)} x_e^{(0)} \geq f_0(S).$$

Because each of the variables in $x^{(0)}$ is ≤ 1

$$\sum_{e \in E - F_0, e \in \delta(S)} x_e^{(0)} + |\delta(S) \cap F_0| \geq \sum_{e \in E - F_0, e \in \delta(S)} x_e^{(0)} + \sum_{e \in F_0, e \in \delta(S)} x_e^{(0)} = \sum_{e \in E, e \in \delta(S)} x_e^{(0)} \geq f_0(S).$$

Rearranging terms, we get that

$$\sum_{e \in E - F_0, e \in \delta(S)} x_e^{(0)} \geq f_0(S) - |\delta(S) \cap F_0| = f_1(S)$$

which means that $x^{(0)}$ is satisfies the constrains of $LP^{(1)}$. □

Combining this with the fact that any feasible solution to $LP^{(1)}$ is $\geq$ the optimal solution, and our induction hypothesis, we get that

$$\sum_{e \in F_1} w_e \leq 2 \sum_{e \in E - F_0} w_e x_e^{(1)} \leq 2 \sum_{e \in E - F_0} w_e x_e^{(0)}.$$

Adding the components from phases A) and B), we have that

$$\sum_{e \in F} w_e \leq 2 \sum_{e \in E} w_e x_e^{(0)}$$

or in other words, the algorithm is a factor 2 approximation algorithm. □

4 Primal-Dual Approach

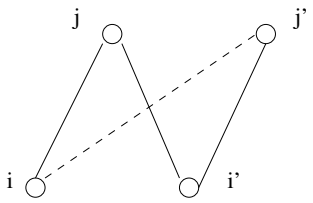
We now revisit in more detail the primal-dual approach to developing LP-based approximation algorithms. Recall that when we can apply this approach, we eliminate the need to use a linear program solver as a black box, potentially giving us a large efficiency benefit. This time, we will apply the primal-dual method in a more complicated way than we have seen before. Specifically, we will use it to develop a factor 3 approximation algorithm for the metric facility location problem. We begin by providing a description of the problem and its formulation into a linear program. We then develop our primal-dual algorithm in two attempts. The first will give us the general idea, but not quite provide an upper bound on our cost. The second attempt will build on the result from the first to obtain a factor 3 approximation algorithm.

We first formally define the (Metric) Facility Location Problem. **Given:** A set F of facilities with f_i representing the cost to open facility i, a set C of clients with c_{ij} representing the cost to serve client j from facility i.
Goal: Find a subset of facilities $I \subseteq F$ and an assignment of clients to facilities $\phi : C \to I$ such that the total cost given by the following equation is minimized.

$$\sum_{i \in I} f_i + \sum_{j \in C} c_{\phi(j)j}$$

The *metric* variant of the facility location problem adds the additional requirement that the c_{ij}'s obey the triangle inequality. More precisely, we assume the following: For all facilities i and i', and for all clients j and j',

$$c_{ij'} \leq c_{ij} + c_{i'j} + c_{i'j'}.$$



4.1 LP relaxation

Following our approach for developing approximation algorithms using linear programming, we now develop an integer linear program (ILP) for this problem, where variables x_i indicate whether facility i is open, and variables y_{ij} indicate whether client j is served by facility i.
Our ILP is as follows:

$$\min \quad \sum_{i \in F} f_i x_i + \sum_{\substack{i \in F \\ j \in C}} c_{ij} y_{ij} \tag{24}$$

$$\text{subject to} \quad \sum_{i \in F} y_{ij} \geq 1 \qquad \forall j \in C \tag{25}$$

$$y_{ij} \leq x_i \qquad \forall i \in F, j \in C \tag{26}$$

$$x_i \in \{0, 1\} \qquad \forall i \in F \tag{27}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i \in F, j \in C \tag{28}$$

Constraint (25) ensures that every client is assigned to some facility. Constraint (26) enforces that every client is served by an open facility.

We now look at the LP relaxation of above formulated ILP. As we've done before, we introduce inequalities instead of integral constraints in our LP relaxation. We rewrite constraints (27) and (28) as follows:

$$x_i \geq 0 \qquad \forall i \in F \tag{29}$$

$$y_{ij} \geq 0 \qquad \forall i \in F, j \in C \tag{30}$$

We note that we can simply use $x_i, y_{ij} \geq 0$ instead of $0 \leq x_i, y_{ij} \leq 1$ in the relaxation, since an optimal solution would never have these variable set to larger than 1. This is useful, since removing constraints in the primal will reduce the number of variables we need to deal with in the dual.

Dual LP We use the standard procedure for obtaining the dual to get the following LP:

$$\max \quad \sum_{j \in C} u_j \tag{31}$$

$$\text{subject to} \quad \sum_{j \in C} v_{ij} \leq f_i \qquad \forall i \in F \tag{32}$$

$$u_j - v_{ij} \leq c_{ij} \qquad \forall i \in F, j \in C \tag{33}$$

$$u_j \geq 0 \qquad \forall j \in C \tag{34}$$

$$v_{ij} \geq 0 \qquad \forall i \in F \tag{35}$$

Constraint (32) in the dual corresponds to the x_i variables in the primal, and constraint (33) corresponds to the y_{ij} variables. Note that without loss of generality, we can substitute constraints (33) and (35) with the following:

$$v_{ij} = \max(0, u_j - c_{ij}) \qquad \forall i \in F, j \in C \tag{36}$$

This is because the v_{ij}'s do not appear in the dual's objective function. Thus, if we have some setting of u_j's such that it is possible to assign the v_{ij}'s and make the solution feasible, the equation above will realize this assignment without affecting our objective value.

The intuitive meaning of a dual is not always apparent. Since we typically minimize some objective in the primal, we consider that we are trying to minimize the amount we spend in achieving some goal. In the dual, we take the position of a service provider who would like to charge as much as possible subject to some constraints. In this problem, the intuitive meaning of the dual is obvious. The dual maximizes the prize charged to clients, which is the sum of two components. The first component, v_{ij}, represents the charge used to pay for the opening of facilities. The other component, u_j, is a per client connection charge.

4.2 First primal-dual attempt

In this section, we make a first attempt at using the primal-dual method to obtain an approximation algorithm for the metric facility location problem. We will find that our approach is not quite sufficient, and will refine it in the next section to obtain a factor 3 approximation algorithm.

We start with a solution that is feasible in the dual and in-feasible integral in the primal. We then iterate, at each step making our dual solution more optimal and/or making our primal solution more feasible. We do this until we have a solution that is feasible in the primal. Applying the primal-dual method to the current problem, we want to relate the cost of the primal to the cost of the dual. The two ideas that constitute our first attempt are outlined below:

1. We will only assign a client to a facility i if the facility is already open and $c_{ij} \leq u_j$. This will guarantee that our total assignment cost obeys:

$$\sum_{j \in C} c_{\phi(j)j} \leq \sum_{j \in C} u_j \leq OPT_{LP}$$

Note that $\sum_{j \in C} u_j$ is simply the dual objective function, and, since the dual is solution feasible, is a lower bound on the LP relaxation's optimal solution.

2. We will only open a facility in the primal if its corresponding constraint in the dual is tight. We will then modify the dual variables in order to improve dual solution by assigning clients to facilities just opened.

We now present our first attempt algorithm. In the algorithm description, the set I will represent the set of "fully paid for" facilities, and the set J represent will represent the set of assigned clients.

$$I := \{i \in F \mid \sum_{j \in C} v_{ij} = f_i\}$$
$$J := \{j \in C \mid (\exists\, i \in I)\ u_i \geq c_{ij}\}$$

Algorithm 3: First attempt at metric facility location approximation.
Input: An instance of metric facility location, (F, C, c).
Output: A set I of open facilities, and assignment ϕ of facilities to clients.
FIRST_ATTEMPT(F, C, c)
(1) $(u) \leftarrow 0$
(2) **while** $J \neq C$
(3) uniformly increase each u_j for $j \in \bar{J}$ until I or J changes
(4) **return** I, ϕ

The main advantage of this algorithm is that we are not using a linear program solver as a black box, and are thus avoiding its expensive operation. In each iteration, the set J can change because some j can now pay for its connection to some i already in I, and the set I can change because a new facility gets paid for.

Termination It is not immediately apparent that this algorithm will halt but follows from the fact that I and J cannot shrink. One of two cases will happen each time we uniformly increase a set of u_j's. The first case is that for some client u_j is increased to exceed c_{ij}, and thus we will add the client to J. The second case is that some facility constraint becomes tight, in which case that facility is added to I. Since increasing the u_j's as done in our algorithm will always result in one of these two actions, and these can only be done a finite number of times before $J = C$. Hence, the algorithm is guaranteed to halt.

Analysis Note that the effect of increasing the dual u_j variables uniformly guarantees that clients that are served later are charged at least as much. Thus, suppose client j_1 is served before j_2, then $u_{j_1} \leq u_{j_2}$.
The total facility opening cost is given by:

$$\sum_{i \in I} f_i = \sum_{i \in I} \sum_{j \in C} v_{ij} \tag{37}$$
$$= \sum_{j \in C} \sum_{i \in I \cap N(j)} v_{ij} \tag{38}$$

Equation (38) is obtained by rearranging the summation and using only the neighborhood of each client in the summation in the second term in equation (37). The neighborhood of client j denoted by $N(j)$ is the set of facilities where j can be served.

$$N(j) = \{i \in F \mid u_i \geq c_{ij}\} \tag{39}$$

The neighborhood relation changes over time, but once a facility enters the neighborhood of some client, it will remain there for the duration of the algorithm. In the next section, we will operate on the final state of the neighborhood relation in modifying our solution. Similarly to neighborhood of a client, the strong neighborhood of client j denoted by $\tilde{N}(j)$ is defined as:

$$\tilde{N}(j) = \{i \in F \mid u_i > c_{ij}\} \tag{40}$$

Applying our earlier observation that $v_{ij} = \max(0, u_j - c_{ij})$ to equation (38), we get:

$$\sum_{i \in I} f_i = \sum_{j \in C} \sum_{i \in I \cap N(j)} \max(0, u_j - c_{ij})$$

$(u_j - c_{ij})$ will always be non-negative, since i is in the neighborhood of j (i.e., $u_j \geq c_{ij}$). We can thus drop the max() from the above equation.

$$\sum_{i \in I} f_i = \sum_{j \in C} \sum_{i \in I \cap N(j)} (u_j - c_{ij}) \tag{41}$$

This result looks promising in two ways. First, suppose $|I \cap N(j)| \leq 1$, then the total facility opening cost in the primal is bounded by OPT_{LP} as shown below:

$$\sum_{i \in I} f_i \leq \sum_{j \in C} u_j \leq OPT_{LP} \tag{42}$$

Secondly, suppose that for all j, $|N(j) \cap I| = 1$, then our choice of ϕ is fixed, since each client can only be assigned to one facility. Thus, we have:

$$\sum_{i \in I} f_i = \sum_{j \in C} (u_i - c_{\phi(j)j})$$

$$\sum_{i \in I} f_i + \sum_{j \in C} c_{\phi(j)j} = \sum_{j \in C} u_i$$

This would imply an optimal integral solution to our problem, and this gives us a factor 1 approximation algorithm! Of course, we cannot hope to realize this unless $\mathsf{P} = \mathsf{NP}$. This analysis does, however, hint at the modification that we need to make. The condition $|N(j) \cap I| = 1$ is very unlikely to be satisfied for all $j \in C$. We address this aspect in the next section.

5 Second primal-dual attempt

The first attempt yields $I \subseteq F$ such that each $j \in C$ has at least one neighbor in I, where i and j are neighbors if j can pay for the connection cost to i (in other words, if $u_j \geq c_{ij}$). If we then connect

each j to one of its neighbors, the problem is that clients may be contributing to the opening cost of facilities they're not assigned to. We argued that if for each client $j \in C$, $|N(j) \cap I| \leq 1$, we can also bound the total facility opening cost by OPT_{LP}. If we look carefully at that argument from the previous section, we can see that the same conclusion holds for the strong neighborhood, i.e., if for each client $j \in C$, $|\tilde{N}(j) \cap I| \leq 1$, we have the same bound of OPT_{LP}. In the second attempt, we will add two phases to our first attempt.

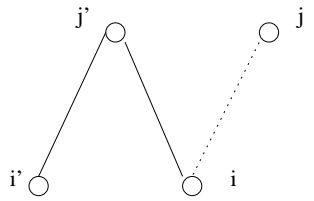
5.1 First phase: Pruning

In the first phase, we will construct the set of facilities $I' \subseteq I$ in a greedy way so as to ensure that no j has more than one strong neighbor in I'. We go over the set I in the order in which facilities where opened in the first attempt, and then add them to I' only if $|I' \cap \tilde{N}(j)| \leq 1$ for all $j \in C$.

5.2 Second phase: Re-assignment

The problem with constructing this subset I' is that some clients might be left without a facility assigned. Thus, in this phase we reassign these "abandoned" clients (i.e., clients that were served by removed facilities in $I - I'$) to facilities in I' without increasing our cost by too much.

Let i be a facility in $I - I'$. Let client j be a client assigned to facility i in the first attempt, i.e., $\phi(j) = i$. We want to reassign this "abandoned" client to some facility in I'. There has to be an $i' \in I'$ and $j' \in C$ such that $\{i, i'\} \in \tilde{N}(j')$. We reassign client j to facility i'. The following figure illustrates the scenario, where the dotted line represents a neighborhood edge, and the solid lines represent strong neighborhood edges:



We also need to calculate the new cost of serving client j. We can compute an upper bound for this new cost $c_{i'j}$ using the triangle inequality as follows:

$$\begin{aligned} c_{i'j} &\leq c_{i'j'} + c_{ij'} + c_{ij} \\ &\leq u_{j'} + u_{j'} + u_j \end{aligned}$$

Claim 5. $u_{j'} \leq u_j$

Proof. The value of u_j is the time at which client j gets served, i.e., the first time j can afford the connection cost to an open facility. If $\phi(j) = i$ then u_j is no earlier than the opening time of facility i. On the other hand, since j' makes a positive contribution to the opening cost of i', this means that the time $u_{j'}$ that j' gets served is no later than the opening time of facility i'. Since i' is opened before i, the claim follows. □

Using the above claim, the upper bound of the re-assignment cost $c_{i'j}$ becomes:

$$c_{i'j} \leq 3u_j \tag{43}$$

Once this client j is reassigned, we find the next "abandoned" client by the procedure described earlier, and reassign it. This is repeated until we have reassigned every "abandoned" client.

5.3 Analysis

Now, we turn to the analysis of this technique. Let $\phi' : C- > I'$ denote the final assignment. By construction, if $j \in C$ makes a positive contribution v_{ij} to the opening cost f_i of a facility $i \in I'$, then j gets served from i, i.e., $\phi(j) = \phi(j') = i$. Let C' denote the union of all such elements over all $i \in I'$. It follows that the total cost of facility opening is given by:

$$\sum_{i \in I'} f_i = \sum_{j \in C'} (u_j - c_{\phi'(j)j}). \tag{44}$$

The total cost of facility assignment is given by the following equation, where the summation in C can be split into two summations on disjoint sets C' and $C - C'$.

$$\begin{aligned}
\sum_{\substack{i \in F \\ j \in C}} c_{ij} y_{ij} = & \sum_{j \in C} c_{\phi'(j)j} \\
&= \sum_{j \in C'} c_{\phi'(j)j} + \sum_{j \in C - C'} c_{\phi'(j)j} \\
&\leq \sum_{j \in C'} c_{\phi'(j)j} + \sum_{j \in C - C'} 3u_j \\
&\leq \sum_{j \in C'} 3c_{\phi'(j)j} + \sum_{j \in C - C'} 3u_j
\end{aligned} \tag{45}$$

Adding equation (45) and 3 times equation (44), we get the following:

$$3 \sum_{i \in I'} f_i + \sum_{j \in C} c_{\phi'(j)j} \leq 3 \sum_{j \in C} u_j \tag{46}$$

By dropping $2 \sum_{i \in I'} f_i$ on the left hand side, we get:

$$\begin{aligned}
\sum_{i \in I'} f_i + \sum_{j \in C} c_{\phi'(j)j} &\leq 3 \sum_{j \in C} u_j \\
&\leq 3 \, OPT_{LP}
\end{aligned}$$

Thus, we have a factor 3 approximation algorithm for metric facility location problem. We note that this factor 3 approximation is not the best that we know of. The current best known factor is 1.735, which can be obtained by using a more complicated rounding scheme than we have presented thus far. We also know that the best factor we can hope to achieve if $\mathsf{P} \neq \mathsf{NP}$ is 1.427.

6 Lagrangian Relaxation

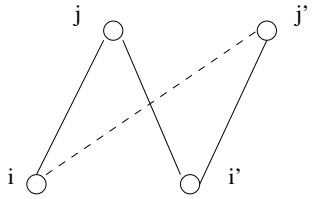
The techniques we described so far for obtaining LP-based approximations often work well when the constraints are localized but not when they are global. For handling the latter, one can use Lagrangian relaxation, a technique to move global constraint to the objective function alleviating the difficulty in solving. We will use this technique to develop a factor 6 approximation for the k-Median problem.

We first formally define k-Median problem.

Given: A set F of facilities, a set C of clients, cost c_{ij} to serve client j to facility i, and $k \in \mathbb{N}$.
Goal: Find a subset of facilities $I \subseteq F$ to open with $|I| \leq k$, and an assignment of clients to facilities $\phi : C \to I$ such that the following total cost is minimized:

$$\sum_{j \in I} c_{\phi(j)j}$$

This problem is similar to some other problems discussed before: such as k-center problem, and facility location problem. The *metric* variant of the problem adds the additional constraint that the c_{ij}'s obey the triangle inequality. As observed in the Primal-Dual LP approximation lecture, the triangle inequality applies as follows to this case:



$$c_{ij'} \leq c_{ij} + c_{i'j} + c_{i'j'}$$

6.1 Relaxations

We develop an integer linear program (ILP) for the k-Median problem using variables x_i to indicate whether facility i is open, and variables y_{ij} to indicate whether client j is served by facility i.

The ILP is formulated as follows:

$$\min \quad \sum_{\substack{i \in F \\ j \in C}} c_{ij} y_{ij} \tag{47}$$

$$\text{subject to} \quad \sum_{i \in F} y_{ij} \geq 1 \qquad \forall j \in C \tag{48}$$

$$y_{ij} \leq x_i \qquad \forall i \in F, j \in C \tag{49}$$

$$\sum_{i \in F} x_i \leq k \tag{50}$$

$$x_i \in \{0, 1\} \qquad \forall i \in F \tag{51}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i \in F, j \in C \tag{52}$$

Constraint (48) ensures all clients are served by at least one facility. Constraint (49) enforces that the clients are served only by open facilities. Constraint (50) ensures at most k facilities are opened.

LP relaxation Solving the formulated ILP is difficult, we relax the integral constraints to linear form, which transforms the NP-hard ILP problem to relaxed LP that can be solved in polynomial time. Integral constraints (51) and (52) are transformed to linear constraints as follows:

$$x_i \geq 0 \qquad \forall i \in F \tag{53}$$

$$y_{ij} \geq 0 \qquad \forall i \in F, j \in C \tag{54}$$

Note that the above constraints are not upper bounded by 1 (i.e., $0 \leq x_i, y_{ij} \leq 1$), since in the optimal solution these variable will never be set to greater than 1.

Lagrangian relaxation (LPLR) Constraint (50) in LPkM is a "global" constraint, which makes solving the problem more difficult. To ease solving problems with such constraints, we utilize a technique called Lagrangian relaxation, where "global" constraints are removed and moved to the objective function with assigned weights (the Lagrangian multipliers). Thus, applying Lagrangian relaxation to LPkM, we obtain the following relaxed LP:

$$\min \quad \sum_{\substack{i \in F \\ j \in C}} c_{ij} y_{ij} + \lambda (\sum_{i \in F} x_i - k) \tag{55}$$

$$\text{subject to} \quad \sum_{i \in F} y_{ij} \geq 1 \qquad \forall j \in C \tag{56}$$

$$y_{ij} \leq x_i \qquad \forall i \in F, j \in C \tag{57}$$

$$x_i \geq 0 \qquad \forall i \in F, j \in C \tag{58}$$

$$y_{ij} \geq 0 \qquad \forall i \in F, j \in C \tag{59}$$

λ in the objective function (55) is the Lagrangian multiplier. For a fixed $\lambda > 0$, ignoring the constant $-k\lambda$, this formulation becomes the facility location problem with $f_i \equiv \lambda$.

6.2 First attempt

Our first idea for an efficient algorithm is to run the primal-dual approximation algorithm for this instance of facility location problem. The dual for LPLR is obtained using the standard technique to generate the dual, where the variables u_i and v_{ij} correspond to constraints (56) and (57) in the primal (LPkM) respectively. Thus, dual is as follows:

$$\max \quad \sum_{j \in C} u_j - kw \tag{60}$$

$$\text{subject to} \quad \sum_{j \in C} v_{ij} \leq w \qquad \forall i \in F \tag{61}$$

$$u_j - v_{ij} \leq c_{ij} \qquad \forall i \in F, j \in C \tag{62}$$

$$u_j \geq 0 \qquad \forall j \in C \tag{63}$$

$$v_{ij} \geq 0 \qquad \forall i \in F, j \in C \tag{64}$$

$$w \geq 0 \tag{65}$$

Thus, running the primal-dual approximation algorithm from the previous lecture yields the integral solution to the primal (LPLR), i.e., the subset of I of open facilities and the mapping ϕ of

which facility serves which client. By (46) we have that

$$\sum_{j\in C} c_{\phi(j)j} + 3\sum_{i\in I} f_i \leq 3\sum_{j\in C} u_j \tag{66}$$

$$\sum_{j\in C} c_{\phi(j)j} \leq 3\left(\sum_{j\in C} u_j - \sum_{i\in I} f_i\right) \tag{67}$$

$$= 3\left(\sum_{j\in C} u_j - \lambda|I|\right) \tag{68}$$

Now, if $k = |I|$, we have a factor 3 approximation:

$$\sum_{j\in C} c_{\phi(j)j} \leq 3\left(\sum_{j\in C} u_j - \lambda k\right) \tag{69}$$

$$\leq 3\, OPT_{LPkM} \tag{70}$$

However, this only works for $k = |I|$. If $|I| < k$, the inequality in equation (69) becomes invalid. If $|I| > k$, it is not clear how to convert it to a feasible solution to LPkM.

7 Second attempt

We observe that the value of λ is under our control. Also, notice that for $\lambda = 0$ we have $I = F$, and for $\lambda = |C|c_{max}$ we have $|I| = 1$. Thus, we can do a binary search on λ in the interval $[0, |C|c_{max}]$ to arrive at the case where $|I| = k$. Unfortunately, there is no guarantee that there exists a value of λ such that $|I| = k$.

Instead, we find (λ_1, λ_2) with their corresponding solution $(I_1,\ I_2)$ and (ϕ_1, ϕ_2), and their corresponding primal and dual variable $(x^{(1)}, x^{(2)})$, $(y^{(1)}, y^{(2)})$, $(u^{(1)}, u^{(2)})$, and $(v^{(1)}, v^{(2)})$ such that:

$$\lambda_1 \leq \lambda_2 \leq \lambda_1 + \delta \tag{71}$$

$$|I_1| > k > |I_2| \tag{72}$$

$$\sum_{j\in C} c_{\phi_1(j)j} \leq 3\left(\sum_{j\in C} u_j^{(1)} - \lambda_1|I_1|\right) \tag{73}$$

$$\sum_{j\in C} c_{\phi_2(j)j} \leq 3\left(\sum_{j\in C} u_j^{(2)} - \lambda_2|I_2|\right) \tag{74}$$

By (72) we can write k as a convex combination of $|I_1|$ and $|I_2|$: $k = \alpha_1|I_1| + \alpha_2|I_2|$ where $\alpha_1, \alpha_2 \geq 0$ and $\alpha_1 + \alpha_2 = 1$. Let $(x, y, u, v) = \alpha_1(x^{(1)}, y^{(1)}, u^{(1)}, v^{(1)}) + \alpha_2(x^{(2)}, y^{(2)}, u^{(2)}, v^{(2)})$.

Claim 6. *(u, v) expanded with $w = \lambda_2$ is a feasible dual solution for LPkM.*

Proof. This follows directly from the convexity of the feasible region, and the fact that $\lambda_2 \geq \alpha_1\lambda_1 + \alpha_2\lambda_2$. □

Lemma 5. $\alpha_1 \sum_{j \in C} c_{\phi_1(j)j} + \alpha_2 \sum_{j \in C} c_{\phi_2(j)j} \leq 3\, OPT_{LPkM} + 3\delta|F|$

Proof.

$$\begin{aligned}
\sum_{j \in C} c_{\phi_1(j)j} &\leq 3 \left(\sum_{j \in C} u_j^{(1)} - \lambda_1 |I_1| \right) \\
&= 3 \left(\sum_{j \in C} u_j^{(1)} - (\lambda_1 + \lambda_2 - \lambda_2) |I_1| \right) \\
&= 3 \left(\sum_{j \in C} u_j^{(1)} - \lambda_2 |I_1| \right) + 3(\lambda_2 - \lambda_1)|I_1| \\
&= 3 \left(\sum_{j \in C} u_j^{(1)} - \lambda_2 |I_1| \right) + 3\delta |I_1|
\end{aligned}$$

On taking the convex combination of this inequality and inequality (74), we get:

$$\begin{aligned}
\alpha_1 \sum_{j \in C} c_{\phi_1(j)j} + \alpha_2 \sum_{j \in C} c_{\phi_2(j)j} &\leq 3 \left(\sum_{j \in C} u_j - \lambda_2 k \right) + 3\alpha_1 \delta |I_1| \\
&\leq 3\, OPT_{LPkM} + 3\delta |F|
\end{aligned}$$

□

Since we get a fractional solution, we use randomized rounding to obtain I with $|I| = k$ and then connect each j with a cheapest connection to I. To simplify notation, let $k_1 = |I_1|$ and $k_2 = |I_2|$. Note that $\alpha_1 = (k - k_2)/(k_1 - k_2)$ and $\alpha_2 = (k_1 - k)/(k_1 - k_2)$.

Here is how we perform the randomized rounding to obtain I with $|I| = k$. We first construct I_1' and I_2' as follows. Let P (for "projection") contain the closest element in I_1 for each element of I_2, and arbitrary additional elements of I_1 so as to obtain a subset P of I_1 of size exactly k_2. Let R (for "random") denote a random subset of $k - k_2$ elements from $I_1 - P$. We set $I_1' = P \cup R$ and $I_2' = I_2 \cup R$.

To create I, we flip a biased coin such that with probability α_1 we pick I_1' as I, and with probability α_2 we pick I_2' as I. We then serve j from the cheapest facility in I.

Claim 7. *Consider a fixed client j in C, and let c denote the service cost for j under the randomized construction of I, c_1 the service cost under I_1, and c_2 the service cost under I_2. Then $E[c] \leq (1 + \max(\alpha_1, \alpha_2))(\alpha_1 c_1 + \alpha_2 c_2)$.*

Proof. Recall that we pick $I = I_1'$ with probability α_1, and $I = I_2'$ with probability α_2. In each case, we serve j from the cheapest i in I. If we pick $I = I_2'$, we can always serve j from $\phi_2(j)$. If we pick $I = I_1'$, then we may not be able to serve j from $\phi_1(j)$ but we certainly can if $\phi_1(j)$ is in P. Thus, if $\phi_1(j)$ is in P, then $E[c] \leq \alpha_1 c_1 + \alpha_2 c_2$. This is because with probability α_1 we set $I = I_1'$ and we can serve j from $\phi_1(j)$ at cost c_1, and with probability α_2 we set $I = I_2'$ and we can serve j from $\phi_2(j)$ at cost c_2. In the sequel we only need to consider the situation where $\phi_1(j)$ is not in P. We distinguish between three cases.

Case 1: $\phi_1(j)$ is in R. This happens with probability exactly $(k-k_2)/(k_1-k_2) = \alpha_1$ (convince yourself of this fact.) Since R is always part of I, we can serve j from $\phi_1(j)$ at cost c_1. Thus, in this case $c \leq c_1$.

Case 2: $\phi_1(j)$ is not in R and we pick $I = I'_2$. This happens with probability $(1-\alpha_1)\alpha_2 = (\alpha_2)^2$. In this case we can serve j from $\phi_2(j)$ at most c_2, so $c \leq c_2$.

Case 3: $\phi_1(j)$ is not in R and we pick $I = I'_1$. This happens with probability $(1-\alpha_1)\alpha_1 = \alpha_1\alpha_2$. In this case we can serve j from the facility f in I_1 closest to $\phi_2(j)$ at cost c_{fj}. By the triangle inequality, $c_{fj} \leq c_{f\phi_2(j)} + c_{\phi_2(j)j}$. We know that $c_{f\phi_2(j)} \leq c_{\phi_1(j)\phi_2(j)}$ (by the choice of f), and that $c_{\phi_1(j)\phi_2(j)} \leq c_{\phi_1(j)j} + c_{\phi_2(j)j}$ (by the triangle inequality). Using the definition of c_1 and c_2, we conclude that in this case $c \leq c_1 + 2c_2$.

Combining the three cases we have that:

$$\begin{aligned} E[c] &\leq \alpha_1 c_1 + (\alpha_2)^2 c_2 + \alpha_1\alpha_2(c_1 + 2c_2) \\ &= \alpha_1(1+\alpha_2)c_1 + \alpha_2(1+\alpha_1)c_2 \\ &\leq (1 + \max(\alpha_1, \alpha_2))(\alpha_1 c_1 + \alpha_2 c_2) \end{aligned}$$

□

Based on the claim 7, the expectation of total cost is given by:

$$E[\text{Total Cost}] \leq (1 + \max(\alpha_1, \alpha_2))(3\ OPT + \delta|F|)$$

We know $\max(\alpha_1, \alpha_2) \leq 1 - \frac{1}{|F|}$, so:

$$\begin{aligned} E[\text{Total Cost}] &\leq \left(2 - \frac{1}{|F|}\right)(3\ OPT + 3\delta|F|) \\ &= 6\ OPT + 6|F|\delta - \frac{3\ OPT}{|F|} - 3\delta \\ &\leq 6\ OPT + 6|F|\delta - \frac{3\ OPT}{|F|} \\ &= 6\ OPT + 3\left(2|F|\delta - \frac{OPT}{|F|}\right) \end{aligned}$$

Thus, to arrive at factor 6 approximation algorithm, we need to pick δ such that:

$$\begin{aligned} 2|F|\delta - \frac{OPT}{|F|} &\leq 0 \\ \delta &\leq \frac{OPT}{2|F|^2} \end{aligned}$$

However, we know that $OPT \geq c_{min} \geq 0$. Thus, we pick:

$$\delta = \frac{c_{min}}{2|F|^2}$$

The number of steps in the binary search is determined by δ, since the binary search reduces the interval of search $[0, |C|c_{max}]$ by half every iteration until the interval size reaches δ. Thus, the number of iteration in the binary search is at most $\log_2\left(\frac{|C|c_{max}}{\delta}\right)$.

This way we arrive at a factor-6 polynomial time approximation algorithm.