IOE 691: Approximation & Online Algorithms

Lecture Notes: Linear-Programming Methods

Instructor: Viswanath Nagarajan Scribe: Kevin J. Sung & more

A fourth technique in approximation algorithms is the use of *linear programs*. Linear programs (LPs) are optimization problems with a linear objective and linear constraints– these can be solved in polynomial time. Most NP problems can be easily formulated as *integer* linear programs: these are similar to LPs except that variables are restricted to integer values. So, a natural approach is to exploit the LP-relaxations of these problems where variables are allowed to take continuous values (which can be solved efficiently). There are various ways in which LPs are used: rounding LP solutions, primal-dual methods and Lagrangian relaxation.

1 Linear Programming

In general, linear programming (LP) can be expressed as:

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0, \end{aligned}$$

where x is a vector of n variables, c is the linear objective, A is an $m \times n$ matrix and b is an m dimensional vector. We assume (by scaling) that all entries in A, b, c are integer valued.

Theorem 1.1 *Any linear program in n variables and m constraints can be solved optimally in $poly(n, m, \log U)$ time, where U is the largest absolute value of entries in c, A, b.*

There are two methods to solve linear programs in polynomial time: ellipsoid and interior point. In practice, the simplex method is often used (although its runtime is not polynomial).

An integer program is similar to an LP, except that variables may be restricted to integer values. IPs are NP-hard to solve. However many approximation algorithms work by (1) formulating an integer program, (2) relaxing integrality to obtain an LP relaxation which can be solved efficiently, and (3) rounding the optimal LP solution to an integral solution.

2 Minimum cut

The *minimum cut* problem is defined as follows.

- Input: A directed graph $G = (V, E)$ with a cost $c_e \geq 0$ associated with each edge $e \in E$, and two vertices $s, t \in V$.
- Output: A subset $F \subseteq E$ such that if the edges in F are removed from G, the resulting graph has no path from s to t. Equivalently, output a subset S that contains s but not t; the edge-set F is then defined by $F = \{(u, v) \in E : u \in S, v \notin S\}$.

2.1 Integer program formulation

The minimum cut problem can be formulated as an integer program as follows:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ s.t. \quad & y_t = 1, \\ & y_s = 0, \\ & y_v \leq y_u + x_{uv}, \qquad \forall (u,v) \in E, \\ & x_e, y_v \in \{0,1\}, \qquad \forall e \in E, v \in V. \end{aligned}$$

The variables have the following interpretation: x_e indicates whether we choose e as part of the cut, and $y_v = 0$ indicates whether, after cutting edges, there is a path from s to v. The inequality constraint expresses the fact that if $(u,v) \in E$ and there is a path from s to u, then there must also be a path from s to v.

2.2 Linear program relaxation and its rounding

We can turn this into a linear program by replacing the constraint

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

with the constraint

$$0 \leq x_e, y_v \leq 1.$$

By solving this LP and "rounding" the solution, we can get an approximate solution to the IP, and hence to minimum cut.

Let (x,y) be an optimal solution to the LP, and consider the following rounding procedure:

1. Pick a number $\tau \in (0,1)$ uniformly at random.
2. Set $S_\tau = \{v : y_v < \tau\}$.

Note that this always returns a feasible solution. We will prove that this gives a 1-approximation, i.e., it always returns an optimal solution.

Lemma 2.1 *For all $e \in E$, $\Pr_\tau[e \text{ is cut}] \leq x_e$.*

Proof: Let $e = (u,v)$. Then e is cut if and only if $y_u < \tau < y_v$. The probability of this occuring is $y_v - y_u$, and this is at most x_{uv}, by one of the constraints of the LP. ■

Theorem 2.1 *The rounding procedure described above returns an optimal solution.*

Proof: Let α be the optimal value of the IP, i.e., the minimum cost of a cut, and let β be the optimal value of the LP, so that $\beta \leq \alpha$. The expected cost of the cut is

$$\sum_{e \in E} c_e \cdot \Pr[e \text{ is cut}] \leq \sum_{e \in E} c_e x_e = \beta \leq \alpha.$$

where the first inequality comes from the lemma above. On the other hand, the expected cost must be at least α, since any cut has cost at least α. Therefore, the expected cost is equal to α. Since

the procedure can never return a cut with cost strictly less than α, it must always return a cut with cost exactly equal to α, i.e., an optimal cut. ■

Note that we can also choose any deterministic value for $\tau \in (0, 1)$ and use S_τ as the solution. This is also guaranteed to be optimal. So we also obtain a deterministic algorithm.

3 Facility location

The *facility location* problem is defined as follows.

- Input: A set F of m facilities and a set C of n clients, along with an opening-cost $f_i \geq 0$ associated with facility i, and a connection-cost $d_{ij} \geq 0$ between each pair of a facility i and client j.
- Output: A subset of facilities $H \subseteq F$ to open that minimizes the total cost

$$\sum_{i \in H} f_i + \sum_{j \in C} \min_{i \in H} d_{ij}.$$

3.1 Linear program relaxation

The facility location problem can be formulated as an integer program as follows:

$$\begin{aligned} \min \quad & \sum_{i \in F} f_i x_i + \sum_{i \in F, j \in C} d_{ij} y_{ij} \\ s.t. \quad & \sum_{i \in F} y_{ij} = 1, \qquad \forall j \in C, \\ & y_{ij} \leq x_i, \qquad \forall i \in F, j \in C \\ & x_i, y_{ij} \in \{0, 1\}, \qquad \forall i \in F, j \in C. \end{aligned}$$

The variables have the following interpretation: x_i indicates whether to open facility i, and y_{ij} indicates whether facility i is assigned to client j, i.e., whether it is the closest open facility. The equality constraint expresses the fact that each client is assigned exactly one facility, and the inequality constraint expresses the fact that a facility must be open to be assigned to a client.

We can turn this into a linear program by replacing the constraint

$$x_i, y_{ij} \in \{0, 1\}$$

with the constraint

$$x_i, y_{ij} \geq 0.$$

For a solution T, we will denote by $\text{cost}(T)$ the objective value of T.

3.2 Special case: set cover

We first observe that this problem encompasses the set cover problem. Given a set cover instance specified by m sets $\{S_i\}_{i=1}^m$ with costs $\{f_i\}$ and n elements, we can create a facility location instance

by letting F be the sets and C the elements, and setting d_{ij} to be 0 if $j \in i$ and ∞ otherwise. The optimal choice of facilities then corresponds to an optimal set cover. We first describe a randomized rounding procedure for the special case of set cover. In this case, the LP looks like:

$$\begin{aligned} \min \quad & \sum_{i \in F} f_i \cdot x_i \\ s.t. \quad & \sum_{i\,:\,j \in S_i} x_i \geq 1, \qquad \forall j \in C, \\ & x_i \geq 0, \qquad \forall i \in F. \end{aligned}$$

Random rounding. Let x be an optimal solution to this LP, and consider the following rounding procedure: pick each set $i \in F$ independently with probability $\min\{1, \alpha \cdot x_i\}$, where $\alpha \approx \ln n$. Let $T \subseteq F$ be the chosen subset in the rounding step.

Ensuring a feasible solution (alteration). Note that solution T may not cover all elements due to the randomness. We now augment the solution by adding one set for each element that is not covered by T. For each element $j \in C$, let $\ell(j)$ index the min-cost set that contains j; so $f_{\ell(j)}$ is the min-cost to cover just element j. Then, we pick the sets $\ell(j)$ for all elements j that are *not* covered by T. Let $R = \{\ell(j) : j \text{ not covered by } T\}$ denote the chosen sets in this "alteration" step. The final solution consists of sets $T \cup R$. Clearly, our solution is always feasible.

Lemma 3.1 $\mathbb{E}[\text{cost}(T)] \leq \alpha \cdot \mathsf{LP}$, *where* LP *is the optimal value of the LP.*

Proof: For any set $i \in F$, we have $\Pr[i \in T] \leq \alpha \cdot x_i$. So,

$$\mathbb{E}[\text{cost}(T)] = \sum_{i \in F} f_i \cdot \Pr[i \in T] \leq \alpha \sum_{i \in F} f_i \cdot x_i = \alpha \cdot \mathsf{LP}.$$

This bounds the cost in the rounding step. ∎

Lemma 3.2 *For any* $j \in C$, $\Pr[j \text{ is not covered by } T] \leq e^{-\alpha}$.

Proof: By the random choice of T, for any element j,

$$\Pr[j \text{ is not covered by } T] = \prod_{i\,:\,j \in S_i} \Pr[i \notin T] = \prod_{i\,:\,j \in S_i} (1 - \min\{1, \alpha x_i\}).$$

If $\alpha x_i > 1$ for any i, then this is zero, and the lemma is proven. Otherwise,

$$\prod_{i\,:\,j \in S_i} (1 - \min\{1, \alpha x_i\}) \leq \prod_{i\,:\,j \in S_i} (1 - \alpha x_i) \leq \prod_{i\,:\,j \in S_i} e^{-\alpha x_i} = e^{-\alpha \sum_{i : j \in S_i} x_i} \leq e^{-\alpha}.$$

In the last inequality, we used the LP constraint $\sum_{i : j \in S_i} x_i \geq 1$. ∎

Lemma 3.3 $\mathbb{E}[\text{cost}(R)] \leq n e^{-\alpha} \cdot \mathsf{LP}$.

Proof: We first claim that $f_{\ell(j)} \leq \mathsf{LP}$ for each element j. Indeed, using the LP-constraint for covering element j, we have:

$$f_{\ell(j)} = \min_{i : j \in S_i} f_i \leq \sum_{i : j \in S_i} f_i \cdot x_i \leq \mathsf{LP}.$$

Now, observe that $\text{cost}(R) = \sum_{j \in C} f_{\ell(j)} \cdot \mathbf{1}[j \text{ not covered by } T]$. So,

$$\mathbb{E}[\text{cost}(R)] = \sum_{j \in C} f_{\ell(j)} \cdot \Pr[j \text{ not covered by } T] \leq e^{-\alpha} \sum_{j \in C} f_{\ell(j)} \leq ne^{-\alpha} \cdot \mathsf{LP},$$

where the first inequality is by Lemma 3.2. ■

Using Lemmas 3.1 and 3.3, the expected cost of our solution is at most $(\alpha + ne^{-\alpha}) \cdot \mathsf{LP}$. Setting $\alpha = \ln n$, we obtain:

Theorem 3.1 *There is an LP-rounding algorithm for set cover that (i) always produces a feasible solution and (ii) has expected cost at most* $(1 + \ln n) \cdot \mathsf{LP}$.

3.3 The general case

We now return to the general facility location problem and show that we can get a $(1 + \ln n)$-approximation here as well. Let x be an optimal solution to the LP for the general problem. Let $\alpha \approx \ln n$ as before. Consider the following rounding procedure:

1. For each facility $i \in F$, independently pick a number $\tau_i \in [0, 1]$ uniformly at random.
2. **Random rounding.** Open facility $i \in F$ if and only if $\alpha \cdot x_i > \tau_i$. Let $T \subseteq F$ denote the facilities opened in this step.
3. For each client $j \in C$ and facility $i \in F$, connect j to i if and only if $\alpha \cdot y_{ij} > \tau_i$. Let $T' \subseteq C$ denote the clients that are connected (to at least one facility) after this step.
4. **Alteration.** For each client $j \in C \setminus T'$, connect it to facility $\ell(j)$ where

$$\ell(j) = \arg\min_{i \in F} (f_i + d_{ij}), \quad \forall j \in C.$$

   Let $R \subseteq F$ denote the facilities opened in this step.

Note that the algorithm opens all facilities in $R \cup F$. We first show:

Lemma 3.4 *The above algorithm returns a feasible solution with probability one.*

Proof: Consider any client $j \in T'$ that gets connected to some facility i in step 3. Note that $x_i \geq y_{ij}$ by the LP-constraint, so we have $\alpha x_i \geq \alpha y_{ij} > \tau_i$. This implies that facility $i \in T$ by the choice in step 2. So client j is indeed connected to an open facility.

Now consider any other client $j \in C \setminus T'$. We explicitly open facility $\ell(j)$ and connect j to it in step 4. So, all clients in C are connected to open facilities. ■

The rest of the analysis is analogous to set cover.

Lemma 3.5 *The expected cost of the solution at the end of step 3 is at most* $\alpha \cdot \mathsf{LP}$.

Proof: After step 3, our solution opens facilities T and connects each client in T'. Consider first the expected facility cost $\mathbb{E}[\sum_{i \in T} f_i] = \sum_{i \in F} f_i \cdot \Pr[\alpha x_i > \tau_i] \leq \alpha \sum_{i \in F} f_i \cdot x_i$.

Now consider the expected connection cost for clients T',

$$\sum_{j \in C} \sum_{i \in F} d_{ij} \cdot \Pr[\alpha y_{ij} > \tau_i] \leq \alpha \sum_{j \in C} \sum_{i \in F} d_{ij} \cdot y_{ij}.$$

Note that this cost may even account for multiple connections from each client. Adding the facility and connection costs, the lemma follows. ∎

Lemma 3.6 *For any client $j \in C$, $\Pr[j \notin T'] \leq e^{-\alpha}$.*

Proof: By the random choice of connections in step 3, for any client j,

$$\Pr[j \notin T'] = \Pr[j \text{ not connected to any facility}] = \prod_{i \in F} \Pr[\alpha y_{ij} \leq \tau_i] = \prod_{i \in F} (1 - \min\{1, \alpha x_i\}).$$

If $\alpha x_i > 1$ for any i, then this is zero, and the lemma is proven. Otherwise,

$$\prod_{i \in F} (1 - \min\{1, \alpha y_{ij}\}) = \prod_{i \,:\, j \in S_i} (1 - \alpha y_{ij}) \leq \prod_{i \in F} e^{-\alpha y_{ij}} = e^{-\alpha \sum_{i \in F} y_{ij}} \leq e^{-\alpha}.$$

In the last inequality, we used the LP constraint $\sum_{i \in F} y_{ij} \geq 1$. ∎

Lemma 3.7 *The expected cost incurred in step 4 is at most $ne^{-\alpha} \cdot \mathsf{LP}$.*

Proof: For each $j \in C$ let $L_j := f_{\ell(j)} + d_{\ell(j),j}$ denote the min-cost for covering client j alone; recall the definition of $\ell(j)$. We first claim that $L_j \leq \mathsf{LP}$ for each client j. Indeed, using the LP-constraint for covering element j, we have:

$$L_j = \min_{i \in F} (f_i + d_{ij}) \leq \sum_{i \in F} (f_i + d_{ij}) \cdot y_{ij} \leq \sum_{i \in F} (f_i \cdot x_i + d_{ij} \cdot y_{ij}) \leq \mathsf{LP}.$$

The first inequality used the constraint $\sum_{i \in F} y_{ij} \geq 1$ and the second inequality used $x_i \geq y_{ij}$.

Now, observe that the cost in step 4 is at most $\sum_{j \in C \setminus T'} L_j$. So the expectation is

$$\sum_{j \in C} L_j \cdot \Pr[j \notin T'] \leq e^{-\alpha} \sum_{j \in C} L_j \leq ne^{-\alpha} \cdot \mathsf{LP},$$

where the first inequality is by Lemma 3.6. ∎

Using Lemmas 3.5 and 3.7, the expected cost of our solution is at most $(\alpha + ne^{-\alpha}) \cdot \mathsf{LP}$. Setting $\alpha = \ln n$, we obtain:

Theorem 3.2 *There is an LP-rounding algorithm for facility location that (i) always produces a feasible solution and (ii) has expected cost at most $(1 + \ln n) \cdot \mathsf{LP}$.*

4 Metric Facility Location

We now consider an important special case of the general facility location problem. Consider the clients and facilities as nodes in a network. Let the set of *all* nodes be $V = F \cup C$, where F are the facilities and C are the clients. We are now interested in the situation that the connection costs d form a *metric*, for example Euclidean distance or travel times in a network. Formally, this means $d_{ij} = d_{ji}$ for all $i, j \in V$ and $d_{ij} \leq d_{ik} + d_{jk}$ for all $i, j, k \in V$. To reduce notation, we use the shorthand $d(v, S) = \min_{u \in S} d_{vu}$ for any node $v \in V$ and subset $S \subseteq V$.

We use the same LP relaxation as for the general facility location problem. But, we will exploit the metric structure to obtain a better 4-approximation algorithm.

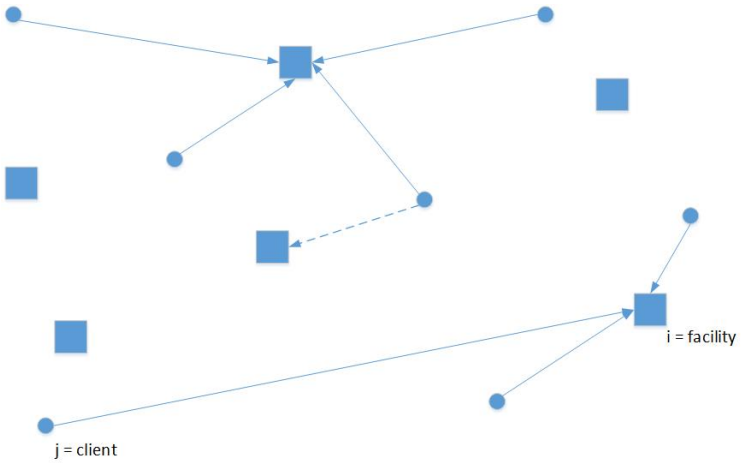

Figure 1: Example of metric facility locations

4.1 Rounding Algorithm

Let (x, y) denote an optimal LP solution. For any vertex v and radius r, define the *ball*

$$B(v, r) \;=\; \{u \in V : d(v, u) \leq r\}.$$

Let L_j denote the the LP connection cost of client $j \in C$, i.e.,

$$L_j = \sum_{i \in F} d_{ij}\ y_{ij},$$

where we also have $\sum_{i \in F} y_{ij} = 1$.

We will use the ball around each client j with radius αL_j where $\alpha > 1$ will be set later. We use $B(j, \alpha L_j) \triangleq B_j$. See Figure 2.

Now, the rounding algorithm is as follows.

1. Sort clients by increasing order of their connection costs, so that $L_1 \leq L_2 \leq L_3 \leq ... \leq L_{|C|}$;
2. Greedily pick a maximal set of disjoint balls in the order $1, 2, ...|C|$. Let I index the chosen balls, so that $B_j \cap B_k = \emptyset$ for all $j, k \in I$ and $j \neq k$;
3. Open the cheapest facility $\pi(j)$ in each $\{B_j, j \in I\}$. Let $S = \{\pi(j) : j \in I\}$.

4.2 Analysis

Let $F^* = \sum_{i \in F} f_i x_i$ denote the facility cost in the LP solution, and $D^* = \sum_{i \in C, j \in F} d_{ij} y_{ij}$ its connection cost. We first bound the facility cost of the rounded solution.

Lemma 4.1 *The facility cost* $\sum_{i \in S} f_i \leq \frac{F^*}{1 - \frac{1}{\alpha}}$.

Proof: We will show $f_{\pi(j)} \leq \frac{1}{1 - \frac{1}{\alpha}} \sum_{k \in B_j} f_k y_{jk}$ for each $j \in C$. The lemma would then follow by by summing over all $j \in I$ and using the disjointness of $\{B_j : j \in I\}$.

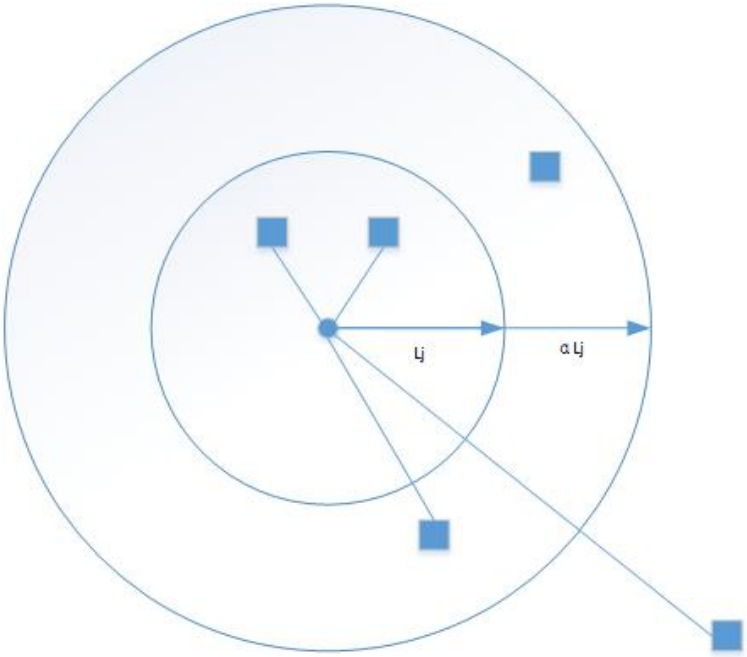

Figure 2: Relation of J_j and αL_j

We claim that $\sum_{k\in B_j} y_{jk} \geq 1 - \frac{1}{\alpha}$ for each $j \in C$. If not, then we must have $\sum_{k\notin B_j} y_{jk} > \frac{1}{\alpha}$, which implies:

$$L_j = \sum_{i\in F} d_{ji} y_{ij} \geq \sum_{k\notin B_j} y_{kj}(\alpha L_j) > \alpha \frac{L_j}{\alpha} = L_j,$$

a contradiction!

Now we have:

$$f_{\pi(j)} = \min_{k\in B_j} f_k \leq \frac{\sum_{k\in B_j} f_k y_{jk}}{\sum_{k\in B_j} y_{jk}} \leq \sum_{k\in B_j} f_k \frac{y_{jk}}{1-\frac{1}{\alpha}}$$

As discussed above, this implies the lemma. ■

Next, we bound the connection cost.

Lemma 4.2 *For each client j, the distance $d(j,S) \leq 3\alpha L_j$. So the connection cost $\leq 3\alpha D^*$.*

Proof: Suppose $j \in I$. Then $d(j,S) \leq \alpha L_j$ as we open some facility in B_j.

Now suppose $j \notin I$. Then there is some $p < j$ with $B_p \cap B_j \neq \emptyset$ and $p \in I$. Let v denote some vertex in $B_p \cap B_j$. Using triangle inequality,

$$d(j,p) \leq d(j,v) + d(v,p) \leq \alpha L_j + \alpha L_p \leq 2\alpha L_j,$$

where the last inequality used the greedy ordering on clients. This implies:

$$d(j,\pi(p)) \leq d(j,p) + d(p,\pi(p)) \leq 3\alpha L_j.$$

So in either case we have $d(j,S) \leq 3\alpha L_j$. By adding over all $j \in C$, the total connection cost is at most $3\alpha D^*$. ■

Based on above lemmas, we have:

Theorem 4.1 *There is a 4-approximation algorithm for metric facility location.*

Proof: We optimize the choice of the parameter $\alpha > 1$. From the above, we know that the facility cost $\leq \frac{1}{1-\frac{1}{\alpha}} F^*$. And the connection cost $\leq 3\alpha L_j$. Then we have:

$$\mathsf{ALG} \leq \frac{1}{1-\frac{1}{\alpha}} F^* + 3\alpha D^* \leq \max(\frac{\alpha}{\alpha-1}, 3\alpha) \cdot (F^* + D^*).$$

Setting $\alpha = \frac{1}{3} + 1$, we get $\mathsf{ALG} \leq 4(F^* + D^*)$. We see that the overall cost is at most 4 times optimal. ■

5 Congestion Minimization

Given a directed graph $G = (V, E)$, a set of k vertices $s_1, \ldots, s_k$ called *sources*, and a set of k vertices $t_1, \ldots, t_k$ called *destinations*, a *routing* is a set of paths $\mathcal{P} = \{P_i\}_{i=1,\ldots,k}$ such that P_i is a path from s_i to t_i. Given a routing $\mathcal{P}$ and an edge $e \in E$, the *load* of e, denoted by $\text{load}_{\mathcal{P}}(e)$, is the number of paths $P_i \in \mathcal{P}$ using the edge e. The *congestion* of a routing $\mathcal{P}$, denoted by $\text{cong}(\mathcal{P})$, is the maximum load among all edges:

$$\text{cong}(\mathcal{P}) = \max_{e \in E} \text{load}_{\mathcal{P}}(e).$$

Definition 5.1 *The **multicommodity routing problem** is an optimization problem which takes as inputs a directed graph $G = (V, E)$, k sources $s_1, \ldots, s_k \in V$, and k destinations $t_1, \ldots, t_k \in V$, and outputs a routing $\mathcal{P}$ minimizing* $\text{cong}(\mathcal{P})$.

This is also known as the congestion minimization problem. One application is in chip design. We need to connect k sources $s_1, \ldots, s_k$ to k destinations $t_1, \ldots, t_k$ by wires on the chip. Naturally, we want to minimize the number of wires going into a single region, which exactly corresponds to the objective of the multicommodity routing problem where you want to minimize the number of paths using a single edge/node.

We present an $O(\log n)$-approximation algorithm for the multicommodity routing problem using randomized rounding technique. The algorithm consists of the following three steps.

1. IP formulation: Formulate the problem as an integer program, and solve the corresponding linear program relaxation.
2. $s \to t$ fractional path decomposition: Given a fractional $s_i \to t_i$ path obtained from Step 1, decompose it to many $s_i \to t_i$ paths each of which is assigned a probability. So for each $i = 1, \ldots, k$, we obtain a probability distribution $\mathcal{D}_i$ on all $s_i \to t_i$ paths.
3. Randomized rounding: For each $i = 1, \ldots, k$, sample $P_i \sim \mathcal{D}_i$. We show that with high probability we have an $O(\log n)$-approximation.

5.1 Integer Program Formulation

For each $i = 1, \ldots, k$ and each $e \in E$, we use the following binary variable to define if path P_i uses edge e:

$$x_{ie} = \begin{cases} 1 & \text{if } e \in P_i \\ 0 & \text{otherwise} \end{cases}, \qquad x_{ie} \in \{0, 1\}.$$

We need to make sure that the set $\{x_{ie}\}_{e \in E}$ defines a valid path for each i, and we impose the following *flow constraint* to achieve this.

$$\sum_{e=(*,v)} x_{ie} - \sum_{e=(v,*)} x_{ie} = \begin{cases} -1 & v \text{ is a source} \\ 1 & v \text{ is a destination} \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V \text{ and } \forall i = 1, \ldots, k.$$

The multicommodity routing problem can be formulated by the following integer program.

$$\begin{aligned} \min \quad & y \\ s.t. \quad & y \geq \sum_{i=1}^{k} x_{ie}, \qquad \forall e \in E & \text{(A')} \\ & \sum_{e=(*,v)} x_{ie} - \sum_{e=(v,*)} x_{ie} = \begin{cases} -1 & v \text{ is a source} \\ 1 & v \text{ is a destination} \\ 0 & \text{otherwise} \end{cases}, \qquad \forall v \in V \text{ and } \forall i = 1, \ldots, k, & \text{(B')} \\ & x_{ie} \in \{0, 1\} \qquad \forall e \in E \text{ and } \forall i = 1, \ldots, k. & \text{(C')} \end{aligned}$$

As before, we relax the integral constraint (C') to

$$x_{ie} \in [0, 1] \qquad \forall e \in E \text{ and } \forall i = 1, \ldots, k. \qquad \text{(C)}$$

so that the integer program becomes a linear program.

However, the integrality gap here is large. Consider a directed graph $G = (V, E)$ with $m + 2$ vertices named $s_1, v_1, \ldots, v_m, t_1$ and $2m$ edges $(s_1, v_1), \ldots, (s_1, v_m), (v_1, t_1), \ldots, (v_m, t_1)$ (so that there is only one source-destination pair). The optimal solution to the relaxed linear program would assign $x_{1e} = \frac{1}{m}$ for every edge, which implies $y = \frac{1}{m}$. However, it is easy to verify that $y = 1$ in the optimal solution to the original integer program. The integrality gap is $m = n - 2 = \Theta(n)$, which is very large! To handle this, we add another trivial constraint $y \geq 1$. The resultant relaxed linear program is as follows.

$$\begin{aligned} \min \quad & y \\ s.t. \quad & y \geq \sum_{i=1}^{k} x_{ie}, \qquad \forall e \in E & \text{(A)} \\ & \sum_{e=(*,v)} x_{ie} - \sum_{e=(v,*)} x_{ie} = \begin{cases} -1 & v \text{ is a source} \\ 1 & v \text{ is a destination} \\ 0 & \text{otherwise} \end{cases}, \qquad \forall v \in V \text{ and } \forall i = 1, \ldots, k, & \text{(B)} \\ & x_{ie} \in [0, 1] \qquad \forall e \in E \text{ and } \forall i = 1, \ldots, k. & \text{(C)} \\ & y \geq 1. & \text{(D)} \end{aligned}$$

Upon receiving an optimal solution to the linear program, for each i, we call the set $\{x_{ie}\}_{e\in E}$ a *fractional $s_i \to t_i$ path*. An fractional $s_i \to t_i$ path $\{x_{ie}\}_{e\in E}$ represents an s_i-t_i *flow* on G, rather than a single path. In the next subsection, we will show how to decompose a fractional $s_i \to t_i$ path to a probability distribution of all $s_i \to t_i$ paths.

5.2 Compute $s \to t$ Fractional Path Decomposition

Fixing i and defining $z_e = x_{ie}$, if $\{x_{ie}\}_{e\in E}$ satisfies condition (B), then $\{z_e\}_{e\in E}$ is a fractional path. In this subsection, we consider a generic fractional $s \to t$ path $\{z_e\}_{e\in E}$, which can represent any fractional $s_i \to t_i$ path (with $z_e = x_{ie}$, $s = s_i$ and $t = t_i$) in the previous subsection.

Lemma 5.1 ($s \to t$ Fractional Path Decomposition) *For any fractional $s \to t$ path $\{z_e\}_{e\in E}$ satisfying*

$$\sum_{e=(*,v)} z_e - \sum_{e=(v,*)} z_e = \begin{cases} -1 & \text{if } v = s \\ 1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V, \tag{1}$$

we can find in polynomial time at most $|E|$ $s \to t$ paths $Q_1, \ldots, Q_r$ with multipliers $\lambda_1, \ldots, \lambda_r$ such that

1. $\sum_{j=1}^{r} \lambda_j = 1$;
2. $\sum_{j=1}^{r} \lambda_j \mathbf{1}_{Q_j}(e) \le z_e$ *for each* $e \in E$.[1]

We first provide some informal intuitive arguments before the rigorous proof of Lemma 5.1.

We view each fractional $s \to t$ path as a flow. To find a decomposition in Lemma 5.1, we find an $s \to t$ path with as much flow as possible, and then find the residual flow with this new found path deducted. We repeat the same process on the residual graph iteratively, until there is no flow coming out of s (or there is no flow going into t) in the residual flow. In each iteration j, the path found is Q_j, and the corresponding maximum flow on this path is λ_j.

An example illustrating this process is shown in Figure 3. The algorithm is explained with each iteration as follows:

1. the input fractional $s \to t$ path (viewed as a flow), with the number on each edge indicates the value of z_e (it is straightforward to check the flow constraint (1) holds at all the five vertices s, a, b, c, t);
2. path $Q_1 = s \to a \to t$ is found and removed, with the maximum possible flow $\lambda_1 = 0.3$;
3. path $Q_2 = s \to a \to b \to t$ is found and removed, with the maximum possible flow $\lambda_2 = 0.2$;
4. path $Q_3 = s \to b \to t$ is found and removed, with the maximum possible flow $\lambda_3 = 0.1$;

[1] Given a universe U, a subset $S \subseteq U$ and an element $x \in U$, the indicator function $\mathbf{1}_S : U \to \{0, 1\}$ is defined as

$$\mathbf{1}_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}.$$

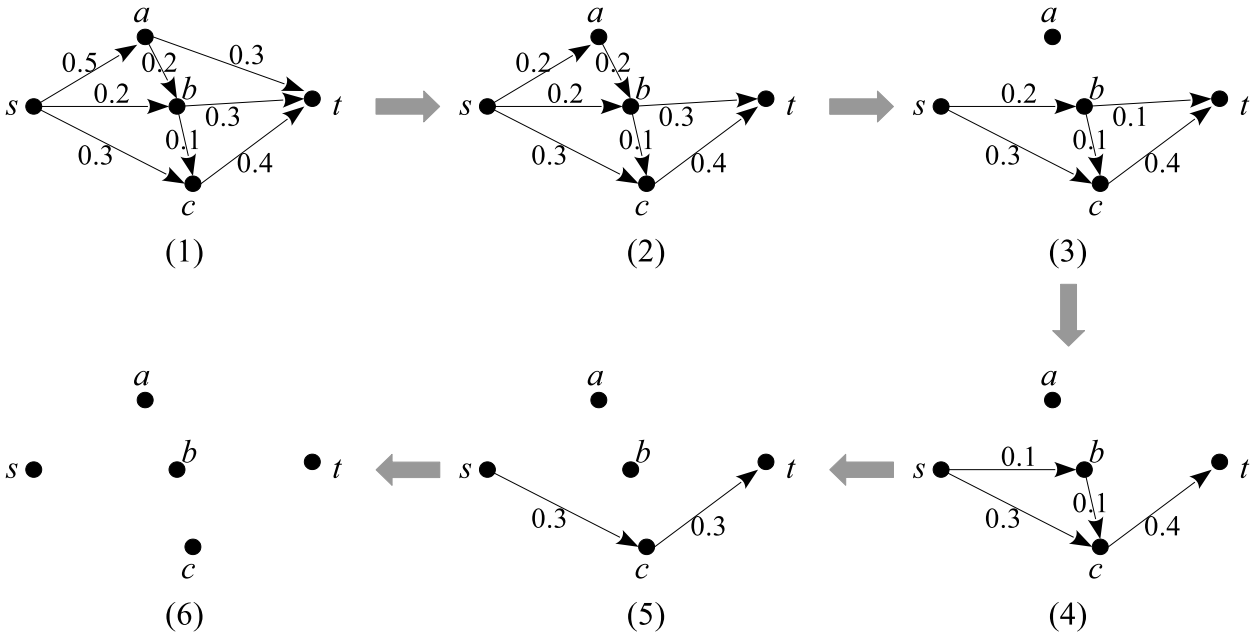

Figure 3: An example of fractional $s \to t$ path decomposition

5. path $Q_4 = s \to b \to c \to t$ is found and removed, with the maximum possible flow $\lambda_4 = 0.1$;
6. path $Q_5 = s \to c \to t$ is found and removed, with the maximum possible flow $\lambda_5 = 0.3$;

It is routine to check that both requirements 1 and 2 in Lemma 5.1 hold. Intuitively, since a fractional $s \to t$ path is a flow sending 1 unit of flow from s to t, requirement 1 is guaranteed as we have cleared all the flows coming out of s; requirement 2 is also satisfied as each removed path Q_j has flow $\lambda_j \leq z_e$ for each $e \in Q_j$.

Proof: [Lemma 5.1] Given a fractional $s \to t$ path $\{z_e\}_{e \in E}$, the algorithm below computes a decomposition into integral paths.

Algorithm 1 Convex decomposition of $s - t$ flow

$r \leftarrow 0$
while $\sum_{e=(s,*)} z_e > 0$ **do**
  $r \leftarrow r + 1$
  find an arbitrary $s \to t$ path, let it be Q_r
  $\lambda_r = \min_{e \in Q_r} z_e$; ▷ compute the maximum possible flow on Q_r
  for each $e \in Q_r$ **do**
    $z_e \leftarrow z_e - \lambda_r$; ▷ deduct Q_r from the flow
  end for
end while
$E \leftarrow E \setminus \{e \in E : z_e = 0\}$; ▷ we only look at edge e with $z_e > 0$

First, we will show that Algorithm 1 output at most $|E|$ paths, i.e., $r \leq |E|$, which implies the running time is polynomial as we find a path in each iteration. Second, we will show that both requirements 1 and 2 in Lemma 5.1 are satisfied.

To show $r \leq |E|$, it is enough to see that, in each while-loop iteration, at least one edge e is removed

from E in Step 9 (notice that the while-loop will certainly terminate if all edges are removed), and Step 5 and Step 7 make sure this.

To show requirements 1 and 2, let $z_e^{(j)}$ be the value of z_e after iteration j, and let $z_e^{(0)} = z_e$. Algorithm 1 (Step 5 to Step 8 in particular) implies that $z_e^{(j)} = z_e^{(j-1)} - \lambda_j \mathbf{1}_{Q_j}(e)$. The while-loop terminates when $\sum_{e=(s,*)} z_e = 0$, which implies $z_e^{(r)} = 0$ for all edges e coming out of s. Then the following calculation implies requirement 1:

$$1 = \sum_{e=(s,*)} z_e^{(0)} = \sum_{e=(s,*)} \sum_{j=1}^{r} \left(z_e^{(j-1)} - z_e^{(j)} \right) = \sum_{e=(s,*)} \sum_{j=1}^{r} \lambda_j \mathbf{1}_{Q_j}(e) = \sum_{j=1}^{r} \lambda_j \sum_{e=(s,*)} \mathbf{1}_{Q_j}(e) = \sum_{j=1}^{r} \lambda_j,$$

where the last equality uses the fact $\sum_{e=(s,*)} \mathbf{1}_{Q_j}(e) = 1$, which is because Q_j at each iteration is *an $s \to t$ path*, making Q_j use exactly one edge that comes out of s. As for requirement 2, for any $e \in E$, we have

$$0 \le z_e^{(r)} = z_e^{(0)} - \sum_{j=1}^{r} \lambda_j \mathbf{1}_{Q_j}(e) = z_e - \sum_{j=1}^{r} \lambda_j \mathbf{1}_{Q_j}(e),$$

which implies $\sum_{j=1}^{r} \lambda_j \mathbf{1}_{Q_j}(e) \le z_e$. ■

As an important remark to Lemma 5.1, the decomposition defines a *probability distribution* over all the $s \to t$ paths. In particular, in this distribution, each Q_j is sampled with probability λ_j.

5.3 Randomized Rounding of Each Path

Coming back to our multicommodity routing problem, for each fractional $s_i \to t_i$ path output by the relaxed linear program, we apply Lemma 5.1 and obtain a probability distribution over all $s_i \to t_i$ paths. We denote this distribution by $\mathcal{D}_i$. In this subsection, we aim to show that the randomized algorithm sampling $P_i \sim \mathcal{D}_i$ independently for each $i = 1, \ldots, k$ achieves approximation ratio $O(\log n)$ with high probability.

Let $\{y, \{x_{ie}\}\}$ be the output of the relaxed linear program.

Firstly, the lemma below is a corollary of Lemma 5.1.

Lemma 5.2 *For any $e \in E$ and $i = 1, \ldots, k$, $\Pr_{P_i \sim \mathcal{D}_i}[e \in P_i] \le x_{ie}$.*

Proof: Let $Q_1, \ldots, Q_r$ be all the $s_i \to t_i$ paths obtained by Algorithm 1, and without loss of generality let $Q_1, \ldots, Q_s$ (with $s \le r$) be those containing the edge e. Recall that each λ_j in Lemma 5.1 can be interpreted as the probability that Q_j is sampled. We have

$$\Pr_{P_i \sim \mathcal{D}_i}[e \in P_i] = \sum_{j=1}^{s} \Pr[Q_j \text{ is sampled}] = \sum_{j=1}^{s} \lambda_j = \sum_{j=1}^{r} \lambda_j \mathbf{1}_{Q_j}(e) \le x_{ie},$$

where the last inequality is due to requirement 2 of Lemma 5.1. ■

Chernoff Bound. The following is a very useful "tail" bound in probability.

Theorem 5.1 *Let $X_1, \ldots, X_n$ be n independent random variables with $X_i \in [0,1]$ for each i. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. For any $\delta > 0$,*

$$\Pr[X > (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \tag{2}$$

$$< e^{-\frac{1}{3}\delta^2\mu}, \qquad \text{(if in addition } \delta < 1\text{)}$$

and for any $\delta \in (0,1)$,

$$\Pr[X < (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} < e^{-\frac{1}{2}\delta^2\mu}. \tag{3}$$

Let $\delta = 10 \log n$, and the Chernoff bound yields the following lemma.

Lemma 5.3 *For each $e \in E$, we have* $\Pr\left[\mathrm{load}_{\mathcal{P}}(e) > (1+\delta)y\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{y} \leq \frac{1}{n^4}$.

Proof: Fix an edge $e \in E$. Define the random variable $X_i = \mathbf{1}_{P_i}(e)$ (the randomness comes from $P_i \sim \mathcal{D}_i$). Then $\mathrm{load}_{\mathcal{P}}(e) = \sum_{i=1}^{k} X_i$. Let $\mu = \mathbb{E}[\sum_{i=1}^{k} X_i]$, and by Lemma 5.2,

$$\mu = \mathbb{E}\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} \mathbb{E}[X_i] = \sum_{i=1}^{k} \Pr_{P_i \sim \mathcal{D}_i}[e \in P_i] \leq \sum_{i=1}^{k} x_{ie} \leq y.$$

We add some dummy deterministic random variables $X_{k+1}, X_{k+2}, \ldots, X_{\ell}$ such that $X_{\imath} \in [0,1]$ for each $\imath = k+1, \ldots, \ell$ and $\bar{\mu} := \sum_{i=1}^{\ell} X_i = y$.

Since X_i's are independent random variables and $X_i \in [0,1]$, we can apply the Chernoff bound (Theorem 5.1), and obtain

$$\Pr\left[\mathrm{load}_{\mathcal{P}}(e) > (1+\delta)y\right] \leq \Pr\left[\sum_{i=1}^{\ell} X_i > (1+\delta)y\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{y}.$$

Since $y \geq 1$ (constraint (D) in the linear program) and $\frac{e^{\delta}}{(1+\delta)^{1+\delta}} < 1$,

$$\Pr\left[\mathrm{load}_{\mathcal{P}}(e) > (1+\delta)y\right] \leq \frac{e^{\delta}}{(1+\delta)^{1+\delta}} < \left(\frac{e}{\delta}\right)^{\delta} < \left(\frac{1}{\log n}\right)^{10 \log n} = \frac{1}{n^{10 \log \log n}} < \frac{1}{n^4},$$

which concludes the proof of the lemma. ∎

Finally, we are ready to show that the randomized algorithm is an $O(\log n)$-approximation.

Theorem 5.2 *With probability at least $1 - \frac{1}{n^2}$, $\mathrm{cong}(\mathcal{P}) = O(\log n) \cdot \mathsf{OPT}$, where OPT is the optimal congestion.*

Proof: Applying a union bound over all edges, we have

$$\begin{aligned}
\Pr\left[\mathrm{cong}(\mathcal{P}) > (10 \log n + 1) \cdot \mathsf{OPT}\right] &\leq \sum_{e \in E} \Pr\left[\mathrm{load}_{\mathcal{P}}(e) > (1+\delta)\mathsf{OPT}\right] \\
&\leq \sum_{e \in E} \frac{1}{n^4} \qquad \text{(Lemma 5.3)} \\
&\leq \frac{1}{n^2}.
\end{aligned}$$

Thus, with probability at least $1 - \frac{1}{n^2}$, $\text{cong}(\mathcal{P}) \leq (10 \log n + 1) \cdot \mathsf{OPT} = O(\log n) \cdot \mathsf{OPT}$. ■

We can obtain approximation guarantee better than that in Theorem 5.2. Using Chernoff bounds (with different parameters), we can obtain the following sharper bounds for the same algorithm:

- An $O\left(\frac{\log n}{\log \log n}\right)$-approximation.
- If ALG is the algorithm's congestion then with high probability $\mathsf{ALG} \leq 2 \cdot \mathsf{OPT} + O(\log n)$.

Finally, the following theorem shows that this approximation guarantee is tight.

Theorem 5.3 [CGKT07] *Assume that $NP \not\subseteq BPTIME(n^{O(\log \log n)})$. Then, there is no $o(\frac{\log n}{\log \log n})$-approximation algorithm for congestion minimization.*

6 Primal-Dual method

Suppose we want to solve an optimization problem that is formulated as a "covering" integer program:

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax \geq b, \\ & x \geq 0, \\ & x \text{ binary}. \end{aligned}$$

Its LP relaxation is obtained by dropping the integrality requirement: we call this the primal LP. Then it has an associated "dual" LP which is as follows:

$$\begin{aligned} \max \quad & b^T y \\ s.t. \quad & y^T A \leq c^T, \\ & y \geq 0. \end{aligned}$$

Each constraint in primal gives rise to a variable in dual (y in this case), and each variable in primal gives rise to a constraint in dual.

The primal-dual method is an approach that incrementally constructs an integer primal and fractional dual solution. The method ensures that the cost of the primal solution is at most some factor α times the dual solution, which implies an approximation ratio of α.

Start with an empty primal solution $F = \emptyset$ and dual solution $y = 0$.
while F is not feasible **do**
  Increase y in a suitable way until the dual constraint gets tight for some i.
  Add element i to the primal solution, i.e. $F \leftarrow F \cup i$.
end while

Often we also need to modify the final solution F (in some simple way) to prove a good approximation ratio. One particularly useful aspect of this approach is that it does not rely on actually solving an LP: so primal-dual algorithms are often faster than LP-rounding algorithms.

7 k-Sparse Set Cover

The k-sparse set cover problem is defined as follows. We are given a set of elements E and n subsets $S_i \subseteq E$. There is cost $c_i \geq 0$ for each subset S_i. Let k denote the maximum number of sets containing any element, i.e. $k = \max_{e \in E} |i : e \in S_i|$. The goal is to select some subsets to cover the set E with a minimum cost.

Note that in the vertex-cover problem, $k = 2$. In the homework, a k-approximation algorithm is developed via LP rounding. Here, we will provide another k-approximation algorithm via Primal-Dual approach without solving the corresponding LP directly.

Recall that we can formulate the LP relaxation of the k-sparse set cover problem as follows, where x_i denotes whether set $S_i \subseteq E$ is selected or not.

$$\begin{aligned} (P) \quad \min \quad & \sum_{i=1}^{n} c_i x_i \\ s.t. \quad & \sum_{i: e \in S_i} x_i \geq 1, \qquad \forall e \in E, \\ & \mathbf{x} \geq 0. \end{aligned}$$

By assigning variable y_e to each of the constraint, the dual problem of the above (P) is

$$\begin{aligned} (D) \quad \max \quad & \sum_{e \in E} y_e \\ s.t. \quad & \sum_{e \in S_i} y_e \leq c_i, \qquad \forall i \in [n], \\ & \mathbf{y} \geq 0. \end{aligned}$$

Let P^* denote the optimal primal objective value, OPT is the optimal set cover cost. The following observation is a restatement of weak duality.

Observation 7.1 *Any dual feasible solution has an objective value upper bounded by the optimal primal objective, i.e., $D \leq P^* \leq OPT$.*

Now we formally state the primal-dual algorithm for k-sparse set cover problem.

Algorithm 2 Primal-Dual algorithm for k-sparse set cover problem

1: Start with primal feasible solution $F = \emptyset$ and dual feasible solution $\mathbf{y} = 0$
2: **while** F is not feasible **do**
3: Pick an element e that is not covered by F
4: Increase the dual variable y_e as much as possible while maintaining dual feasibility. This process stops when $\sum_{e \in S_i} y_e = c_i$ for some set S_i.
5: Add set S_i to F, i.e., $F \leftarrow F \cup \{i\}$
6: **end while**

The main technical result is the following theorem.

Theorem 7.1 *The above algorithm is a k-approxiamtion algorithm for k-sparse set cover problem*

To analyze the algorithm, we first show the following critical lemma.

Lemma 7.1 *$Cost(F) \leq k \sum_{e \in E} y_e$, where $Cost(F)$ denotes the primal cost computed by the algorithm and y_e is the dual solution found by the algorithm.*

Proof: Since for each S_i picked by the algorithm, we have $c_i = \sum_{e \in S_i} y_e$ holds until the end of the algorithm, we have

$$\begin{aligned} Cost(F) &= \sum_{i \in F} c_i \\ &= \sum_{i \in F} \sum_{e \in S_i} y_e \\ &= \sum_{e \in E} y_e | i \in F : e \in S_i | \\ &\leq k \sum_{e \in S_i} y_e, \end{aligned}$$

where the third equality holds by interchanging the order of summation. The last inequality is by definition of the k-sparse set cover instance. ■

Since the dual solution computed by the algorithm is always feasible, we have

$$Cost(F) \leq k \sum_{e \in E} y_e \leq kP^* \leq k \cdot OPT$$

by applying Observation 7.1 and Lemma 7.1.

8 Steiner Forest Problem

Given a graph $G = (V, E)$ along with cost $c_e \geq 0$ on each edge $e \in E$. Let $\{s_i, t_i\}_{i=1}^k$ be pairs of terminals which are given as input. The Steiner forest problem is to select some edges from graph G such that s_i is connected to t_i for each $i \in [k]$ while minimizing the total cost.

The main theorem is stated as follows.

Theorem 8.1 *There is a 2-approximation algorithm for Steiner forest problem.*

Definition 8.1 *We say a vertex set $S \in 2^V$ is **active** if and only if $|S \cap \{s_i, t_i\}| = 1$ for some $i \in [k]$. Let $\mathcal{A} \subseteq 2^V$ be the collection of all **active** vertex sets.*

We first provide an LP relaxation to the Steiner forest problem. Let x_e denote whether an edge e is selected. Consider the following LP, where $\delta S = \{(u, v) \in E : u \in S, v \notin S\}$ denotes the edges at the boundary of S.

$$\begin{aligned} (P) \quad \min \quad & \sum_{e \in E} c_e x_e \\ s.t. \quad & \sum_{e \in \delta S} x_e \geq 1, \qquad \forall S \in \mathcal{A}, \\ & \mathbf{x} \geq 0. \end{aligned}$$

By assigning variable y_S to each of the constraint, the dual problem of the above (P) is

$$\begin{aligned} (D) \quad \max \quad & \sum_{S\in\mathcal{A}} y_S \\ s.t. \quad & \sum_{S\in\mathcal{A}, e\in\delta S} y_e \le c_e, \qquad \forall e \in E, \\ & \mathbf{y} \ge 0. \end{aligned}$$

Now we formally state the primal-dual algorithm for Steiner forest problem.

Algorithm 3 Primal-Dual algorithm for Steiner forest problem

1: Start with primal feasible solution $F = \emptyset \subseteq E$ and dual feasible solution $\mathbf{y} = 0$
2: **while** F is not feasible **do**
3: Let $\mathcal{B}$ denotes the connected components in F and $\mathcal{C} \subseteq \mathcal{B}$ consists of active components.
4: Increase the dual solution y_S for each $S \in \mathcal{C}$ until $c_e = \sum_{S\in\mathcal{A}, e\in\delta S} y_S$ for some edge $e \in E$.
5: Add edge e to F, i.e., $F \leftarrow F \cup \{e\}$
6: **end while**
7: Output union of s_i-t_i paths in F (denoted by R)

Observation 8.1 *The solution R is a feasible Steiner forest and $\mathbf{y}$ is a feasible dual solution.*

To analyze the algorithm, we need to show the following critical lemma.

Lemma 8.1 $cost(R) = \sum_{e\in R} c_e \le 2\sum_{S\in\mathcal{A}} y_S$.

Proof: Since for each $e \in R$, the algorithm guarantees $c_e = \sum_{S\in\mathcal{A}, e\in\delta S} y_S$, therefore,

$$\begin{aligned} cost(R) &= \sum_{e\in R} c_e \\ &= \sum_{e\in R} \sum_{S\in\mathcal{A}, e\in\delta S} y_S \\ &= \sum_{S\in\mathcal{A}} y_S \cdot |R \cap \delta S| \triangleq P(\mathbf{y}) \end{aligned}$$

Let $D(\mathbf{y}) \triangleq \sum_{S\in\mathcal{A}} y_S$, It suffices to show that $\frac{dP}{dt} \le 2\frac{dD}{dt}$, where time t is ticking whenever dual solution $\mathbf{y}$ increases. Note that $\frac{dy_S}{dt} = 1$ when S is active and $\frac{dy_S}{dt} = 0$ otherwise. We conclude that $\frac{dD}{dt} = \sum_{S\in\mathcal{A}} \frac{dy_S}{dt} = |\mathcal{C}|$ and $\frac{dP}{dt} = \sum_{S\in\mathcal{A}} \frac{dy_S}{dt}|\delta S \cap R| = \sum_{S\in\mathcal{C}} |\delta S \cap R|$. Define an auxiliary graph H that treats each connected component in $\mathcal{B}$ as a single node and includes all edges in R that are *added after time t*. Note that for any $S \in \mathcal{B}$, we have $deg_H(S) = |\delta S \cap R|$. Also, H is a forest because F is a forest where each $S \in \mathcal{B}$ is connected internally.

We now have several simple observations:

Lemma 8.2 *The nodes of zero degree in H must be inactive connected components.*

Otherwise, the solution R must be infeasible since it contains active connected component without an edge going out. Let $\mathcal{B}' = \mathcal{B} \setminus \{S \in \mathcal{B} : deg_H(S) = 0\}$ and $\mathcal{I} = \mathcal{B}' \setminus \mathcal{C}$. Note that H is a forest on the nodes $\mathcal{B}'$ and all these nodes have degree at least one.

Lemma 8.3 *Every leaf node in H must be active.*

Otherwise, suppose an inactive leaf node $S_I \in \mathcal{I}$ has only one edge e connected to a connected component $S \in \mathcal{B}'$. Then the edge e will not be used in any s_i-t_i path of F: any such path must cross S_I exactly once which is not possible as S_I is inactive. This contradicts with the fact that $e \in R$.

Since H has no cycles, we must have $\sum_{S\in\mathcal{B}'} deg_H(S) \leq 2|\mathcal{B}'|$. Note that Lemma 8.3 implies $\sum_{S\in\mathcal{I}} deg_H(S) \geq 2|\mathcal{I}|$, we have

$$\begin{aligned}
\frac{dP}{dt} &\leq \sum_{S\in\mathcal{C}} deg_H(S) \\
&= \sum_{S\in\mathcal{B}'} deg_H(S) - \sum_{S\in\mathcal{I}} deg_H(S) \\
&\leq 2|\mathcal{B}'| - 2|\mathcal{I}| \\
&= 2|\mathcal{C}| \\
&= 2\frac{dD}{dt}.
\end{aligned}$$

Thus, the lemma is proved. ■

Finally, according to Lemma 8.1 and the feasibility of both the primal and dual solution, we have $cost(R) \leq 2\sum_{S\in\mathcal{A}} y_S \leq 2P^* \leq 2OPT$, which completes the proof of Theorem 8.1.

9 Generalized Assignment (Iterative Rounding)

We now consider the generalized assignment problem. There are m machines and n jobs. Each job must be assigned to some machine. If job $j \in [n]$ is assigned to machine $i \in [m]$, it requires processing time p_{ij} and incurs cost c_{ij}. Moreover, there is a common budget T such that the total processing time assigned to each machine has cannot exceed this budget. It is required to find an assignment in which no machine exceeds its budget and the total cost is minimized.

Let x_{ij} indicate the assignment of job j to machine i. A natural IP formulation is:

$$\begin{array}{lll}
\min & \sum_{i\in[m],j\in[n]} c_{ij}x_{ij} & \\
s.t. & \sum_{j\in[n]} p_{ij}x_{ij} \leq T, & \forall i \in [m] \\
& \sum_{i\in[m]} x_{ij} = 1, & \forall j \in [n] \\
& x_{ij} \in \{0,1\}, & \forall i \in [m], j \in [n].
\end{array}$$

This is an NP-hard problem. Our goal is to find a bi-criteria approximation – achieving the optimal cost with a small processing time violation.

We will exploit "extreme point" properties of optimal LP solutions, summarized below.

Theorem 9.1 *[Extreme Point] For any linear program with bounded and non-empty feasible region $\{x : Ax \geq b\} \subseteq \mathbb{R}^n$, there exists an optimal solution $\bar{x}$ such that $\bar{A}\bar{x} = \bar{b}$ where $\bar{A} \in \mathbb{R}^{n\times n}$ is a rank-n sub-matrix of A and $\bar{b} \in \mathbb{R}^n$ is the corresponding sub-vector of b.*

We consider an iterative LP based algorithm. In each iteration, we keep track of the current jobs $N \subseteq [n]$, current machines $M \subseteq [m]$, current set of active edges $E \subseteq [m] \times [n]$ and the current

partial assignment $F \subseteq [m] \times [n]$. We solve the following LP in each such iteration.

$$\begin{array}{rll} \min & \sum_{i\in[m],j\in[n]} c_{ij}x_{ij} & \\ s.t. & \sum_{j\in[n]} p_{ij}x_{ij} \leq T, & \forall i \in M \\ & \sum_{i\in[m]} x_{ij} = 1, & \forall j \in N \\ & x_{ij} \in [0,1], & \forall (i,j) \in E \\ & x_{ij} = 1, & \forall (i,j) \in F \\ & x_{ij} = 0, & \forall (i,j) \in ([m] \times [n]) \setminus E \setminus F. \end{array}$$

Remark: In the LP above, x_{ij} is treated as a decision variable only if $(i,j) \in E$; if $(i,j) \notin E$ then x_{ij} is a fixed 0-1 value.

Now, we can describe our algorithm in detail.

Algorithm 4 LP based iterative algorithm

Set $N = [n], M = [m], E = [m] \times [n], F = \emptyset$
repeat
  Find an extreme point solution $\mathbf{x}$ of problem $LP(N, M, E, F)$.
  if there exists $(i,j) \in E$ such that $x_{ij} = 0$ **then** ▷ Case 1
    assign $x_{ij} = 0$
    drop edge $E = E - \{(i,j)\}$
  end if
  if there exists $(i,j) \in E$ such that $x_{ij} = 1$ **then** ▷ Case 2
    assign $x_{ij} = 1$
    add edge to partial assignment $F = F \cup \{(i,j)\}$
    drop edge $E = E - \{(i,j)\}$
    drop assignment constraint $N = N - \{j\}$
  end if
  if there exists $i \in M$ such that $|\delta_E(i)| \leq 1$ **then** ▷ Case 3
    drop budget constraint $M = M - \{i\}$
  end if
  if there exists $i \in M$ such that $|\delta_E(i)| = 2$ and $\sum_{(i,j)\in\delta_E(i)} x_{ij} \geq 1$ **then** ▷ Case 4
    drop budget constraint $M = M - \{i\}$
  end if
until $N = \emptyset$

Theorem 9.2 *[Bi-criteria Approximation] Our iterative LP based algorithm finds an integral assignment in polynomial time that violates each budget constraint by at most an additive* $p_{max} = \max_{ij} p_{ij}$ *and has cost at most* OPT.

The proof of Theorem 9.2 follows from the following two lemmas.

Lemma 9.1 *Assume that the algorithm terminates and produces assignment* $\{\bar{x}_{ij}\}_{i\in[m],j\in[n]}$. *Then,*

$$\begin{array}{rcl} \sum_j p_{ij}\bar{x}_{ij} & \leq & T + p_{\max}, \forall i \in [m] \\ \sum_{ij} c_{ij}\bar{x}_{ij} & \leq & OPT. \end{array}$$

Proof: Note that we only remove a job j from N when it gets assigned in the partial solution F. So, if the algorithm terminates, the assignment $\{\bar{x}_{ij}\}_{i\in[m],j\in[n]}$ corresponding to F must have assigned each job (integrally) to some machine. Note that the first LP with $N=[n], M=[m], E=[m]\times[n], F=\emptyset$ is a relaxed version of the generalized assignment problem: so its optimal value is at most OPT. Moreover, in each iteration we only relax the current $LP(N,M,E,F)$ further, which shows that the optimal value of the final LP is also at most OPT. Now notice that the value of the final LP is just $\sum_{ij} c_{ij}\bar{x}_{ij}$. Hence $\sum_{ij} c_{ij}\bar{x}_{ij} \le OPT$.

Now, we show that each budget constraint is violated by at most p_{max}. Consider any machine $i\in[m]$. At the termination time, let $load_{\mathsf{TRM}}(i)$ denote the total processing time on machine i. If at that time $i\in M$ (i.e., the budget constraint i is active), then $load_{\mathsf{TRM}}(i)=\sum_{j\in[n]} p_{ij}\bar{x}_{ij}\le T$. Otherwise, i has been dropped at either case 3 or case 4 in some iteration. Consider the point when i is dropped: let $load_{\mathsf{DRP}}(i)$ denote the load on i due to jobs currently assigned to it in F (the partial solution), and let E denote the current active edges. If i is dropped at case 3, we have $|\delta_E(i)|\le 1$ and $load_{\mathsf{DRP}}(i)\le T$. They together imply $load_{\mathsf{TRM}}(i)\le T+p_{max}$. If i is instead dropped at case 4, since $|\delta_E(i)|=2$ let's assume $\delta_E(i)=\{j,k\}$. The LP budget constraint implies that:

$$load_{\mathsf{DRP}}(i)+p_{ij}x_{ij}+p_{ik}x_{ik}\le T,$$

Also because $x_{ij}+x_{ik}\ge 1$, we then have, by simple algebra,

$$load_{\mathsf{TRM}}(i) \;\le\; load_{\mathsf{DRP}}(i)+p_{ij}+p_{ik}\le T+p_{max}.$$

■

Lemma 9.2 *The algorithm terminates in polynomial time.*

Proof: Since each case 1 – 4 makes our problem size strictly smaller, it is enough to show that, in each iteration, some case 1 – 4 must apply. We proceed by contradiction. We assume at some iteration that none of cases 1 – 4 apply to the extreme point solution $\mathbf{x}$ of $LP(N,M,E,F)$,. So we must have

1. $0<x_{ij}<1$ for all $(i,j)\in E$,
2. $|\delta_E(i)|\ge 2$ for all $i\in M$,
3. $|\delta_E(j)|\ge 2$ for all $j\in N$ (otherwise, constraint $\sum_{i\in[m]} x_{ij}=1$ forces some $x_{ij}=1$ for $i\in\delta_E(j)$).

Note (2) and (3) implies that

$$|E|\ge \frac{1}{2}(2M+2N)=M+N. \quad (\dagger)$$

On the other hand, note $\mathbf{x}\in\mathbb{R}^{|E|}$ as an extreme point solution has $|E|$ active/tight constrains (refer to Theorem 9.1), and (1) implies that constraints

$$\begin{array}{ll}\sum_{j\in[n]} p_{ij}x_{ij}\le T & ,\forall i\in M\\ \sum_{i\in[m]} x_{ij}=1 & ,\forall j\in N\end{array}$$

are the only possible $M+N$ active constrains, which shows that

$$|E|\le M+N. \quad (*)$$

Therefore, from (†) and $(*)$, we get $|E| = M + N$, which, together with (2) and (3), forces that $|M| = |N|$ and $\delta_E(i) = \delta_E(j) = 2$ for all $i \in M, j \in N$. Now consider

$$\sum_{i \in M} \sum_{j:(ij) \in E} x_{ij} = \sum_{j \in N} \sum_{i:(ij) \in E} x_{ij} = |N| = |M|,$$

which implies $x(\delta_E(i_0)) \geq 1$ for some $i_0 \in M$. Combined with $|\delta_E(i_0)| = 2$, it follows that case 4 applies for i_0, which leads to contradiction. ■

References

[CGKT07] Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, and Kunal Talwar. Hardness of routing with congestion in directed graphs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 165–178, 2007.