A Survey of Algorithms for Capacitated k-median Problems

Khoa Trinh

University of Maryland, College Park, MD 20742
Email: khoa@cs.umd.edu

Abstract

k-median is a classic optimization problem which has been studied extensively in the last few decades. The current best known approximation ratio for this problem is $(2.675 + \epsilon)$. A more generalized version of this problem also considers the capacities of the given facilities. That is, each facility cannot serve more clients than its own capacity. Unfortunately, there is no known constant-factor approximation algorithm for this problem. We survey some state-of-the-art techniques to approximate the Capacitated k-median problem that may violate the hard constraint by a factor of $(1 + \epsilon)$. Note that the natural LP relaxation for this problem is still unbounded even when we are allowed to violate the cardinality constraint by a factor of $2 - \epsilon$. The new improvements come from several techniques to strengthen the LP as well as a novel clustering algorithm.

1 Introduction

In k-median problem, we have a set of facilities $\mathcal{F}$, a set of clients $\mathcal{C}$, a symmetric distance metric d on $\mathcal{F} \cup \mathcal{C}$, and a number k. The goal is to open at most k facilities so that the total connection cost from each client to the closest, open facility is minimized. This problem is known to be NP-hard. The current best known approximation algorithm for k-median is a $(2.675 + \epsilon)$-approximation algorithm [2]. In the capacitated version, each facility $i \in \mathcal{F}$ also has a capacity $u_i > 0$. We need to make sure that facility i only serves at most u_i clients in our solution.

One of the reasons making this problem difficult to obtain a constant approximation ratio is that its natural linear program relaxation has unbounded gap. Most of the previous algorithms for capacitated k-median either violate the capacity constraint or the cardinality constraint. For example, Byrka et. al. give an $O(1/\epsilon^2)$-approximation algorithm by violating the capacity constraint by a factor of $(2+\epsilon)$ in [1].

In this survey, we will discuss two new results by Shi Li on Capacitated k-median (CKM) problems. In Section 2, we review an old result for the Minimum Knapsack problem and introduce the idea of Relaxed Separation Oracle by Carr et. al. [3]. This technique allows us to round a fractional solution into an integral one even when there is no efficient separation oracle for the LP. In Section 3, we discuss Shi Li's $O(\exp(1/\epsilon^2))$-approximation algorithm for the Hard Uniform Capacitated k-median while opening at most $(1 + \epsilon)k$ facilities [5]. In Section 4, we discuss an improved algorithm by Shi Li which has an approximation ratio of $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon}\right)$ [4]. The drawback is that this algorithm may need to open 2 copies of a facility.

2 Relaxed Separation Oracle

2.1 Minimum Knapsack Problem

Given a set I of items and a demand D. Each item $i \in I$ has size $s_i \geq 0$ and value $v_i \geq 0$. The problem is to choose a subset X of I such that the total value of items in X is at least the demand D and the total size is minimized. The LP relaxation of the problem is as follows.

$$\begin{aligned} \text{minimize } & c(x) = \sum_{i \in I} s_i x_i \\ \text{subject to } & \sum_{i \in I} v_i x_i \geq D \\ & x_i \geq 0 \end{aligned}$$

2.2 Knapsack Cover Inequalities

Let x be any feasible, integral solution and A be any subset of I. We observe that if the items in A are given for free, the remaining chosen items in $I \setminus A$ should be still feasible on the residual instance with demand $D - v(A)$. In other words, we have the following valid inequality

$$\begin{aligned} \sum_{i \in I \setminus A} v_i x_i &= \sum_{i \in I} v_i x_i - \sum_{i \in A} v_i x_i \\ &\geq D - \sum_{i \in A} v_i x_i \\ &\geq D - v(A). \end{aligned}$$

We can strengthen this inequality by assuming that the value of each item is no more than the (residual) demand:

$$\sum_{i \in I \setminus A} \min\{v_i, D - v(A)\} x_i \geq D - v(A).$$

Now, we have a strengthened LP relaxation:

$$\begin{aligned} \text{minimize } & c(x) = \sum_{i \in I} s_i x_i \\ \text{subject to } & \sum_{i \in I \setminus A} \min\{v_i, D - v(A)\} x_i \geq D - v(A) \qquad , \forall A \subseteq I, v(A) \leq D \\ & x_i \geq 0 \end{aligned}$$

2.3 2-approximation Algorithm

Let x be a fractional solution to the strengthened LP with cost $c(x)$. We will round x to an integral solution whose cost is at most $2c(x)$. Let $A = \{i : 1/2 \leq x_i \leq 1\}$. We will take all items in A. If $v(A) \geq D$, then we are done. Note that the cost of choosing items in A is at most $2c(x)$. Assume that $v(A) \leq D$. Then

$$\sum_{i \in I \setminus A} \min\{v_i, D - v(A)\} x_i \geq D - v(A).$$

Consider the residual instance $I \setminus A$. Recall that $x_i < 1/2$ for all $i \in I \setminus A$. Sort these items in the decreasing order of new value:

$$\min\{v_1, D - v(A)\} \geq \min\{v_2, D - v(A)\} \geq \cdots \geq \min\{v_k, D - v(A)\}.$$

Let r be the (minimum) integer such that rx_i is also integral $\forall i \in I \setminus A$.

- We create r empty buckets $B_1, \ldots, B_r$ and distribute the items into these buckets as follows. We create $r(2x_1)$ copies of item 1 and put these copies into the first $r(2x_1)$ buckets, then put $r(2x_2)$ copies of item 2 into the next $r(2x_2)$ buckets (modulo r) and so on.
- Then the algorithm will just return the following set

$$\arg\min_{B_j \cup A} \sum_{i \in B_j \cup A} s_i.$$

Note that

- Since $r(2x_i) < r$, no bucket contains duplicate items.
- Let $v_A(B_j) = \sum_{i \in B_j} \min\{v_i, D - v(A)\}$. By ordering, we have

  $$v_A(B_1) \geq v_A(B_2) \geq \ldots \geq v_A(B_r).$$

  Moreover, the difference $v_A(B_1) - v_A(B_r)$ is at most $D - v(A)$ (the worst case is when B_1 contains one more item than B_r). We claim that

  $$v_A(B_r) \geq D - v(A).$$

  Otherwise, we have

  $$v_A(B_1) \leq v_A(B_r) + D - v(A) < 2(D - v(A)),$$

  which implies that

  $$\sum v_A(B_i) < 2r(D - v(A)).$$

  This is a contradiction because

  $$\sum v_A(B_i) = r \sum_{i \in I \setminus A} \min\{v_i, D - v(A)\}(2x_i) \geq 2r(D - v(A)).$$

- Thus, all the sets $B_1 \cup A, \ldots, B_r \cup A$ are feasible, integral solutions. Also,

  $$\sum_{i=1}^{r} \frac{x(B_i \cup A)}{r} \leq 2x.$$

  Therefore, the minimum cost of these sets is bounded by $2c(x)$.

2.4 Intuition

Consider the set A of items with "high" opening: $x_i \geq 1/2$. It is quite "easy" to choose these items since the cost of choosing all items in A is at most twice the fractional cost. The question is what to do with the remaining items in $I \setminus A$?

Now consider the residual instance $I \setminus A$ and let x' be x when restricted on this new instance. Since we already choose A, we want to satisfy the new demand $D' = D - v(A)$. WLOG, assume that item i in the new instance has value $v'_i = \min\{v_i, D'\}$. Indeed, we cannot freely take items in $I \setminus A$ because the opening can be close to 0 and we would not be able to bound the cost. Instead, we rely on the fact that

$$\sum_{i \in I \setminus A} v'_i x'_i \geq D'.$$

In other words, x' is a feasible, fractional solution to the new Minimum Knapsack instance. But why is it easier to round x'? Observe that the "opening" $x'_i < 1/2$ for all i. It is quite natural to scale up such a solution with small openings: let $\hat{x} = 2x'$ and note that $\hat{x}$ is still feasible because $\hat{x}_i \leq 1$, however, we have

$$\sum_{i \in I \setminus A} v'_i \hat{x}_i \geq 2D'.$$

Now it is clear that rounding $\hat{x}$ should be easier. This is because of the factor 2 of D'. In this new instance, given the fractional, feasible solution $\hat{x}$, we are allowed to violate the demand constraint by a factor of 2 during the rounding algorithm.

The rest can be done by using a bucketing algorithm. Let r be the (minimum) integer such that $r\hat{x}_i$ is also integral. Create r empty buckets $B_1, \ldots, B_r$. Sort all items in decreasing order of values: $v'_1 \geq v'_2 \geq \ldots \geq v'_k$. Then put $r\hat{x}_1$ items 1 into the first $r\hat{x}_1$ buckets, $r\hat{x}_2$ items 2 into the next $r\hat{x}_2$ buckets (modulo r) and so on. Finally, return the bucket with the minimum cost.

- Since $r\hat{x}_i \leq r$, no bucket contains duplicate items,
- It is easy to see that $v'(B_1) \geq \ldots \geq v'(B_r)$. We claim that $v'(B_r) \geq D'$ and hence, all buckets are feasible. Note that the difference $v'(B_1) - v'(B_r) \leq v'_1 \leq D'$ because B_1 may contain at most one more item than B_r. If $v'(B_r) < D'$ then $v'(B_1) < 2D'$ and

  $$\sum_{i=1}^{r} v'(B_i) < 2rD'.$$

  This is a contradiction because

  $$\sum_{i=1}^{r} v'(B_i) = \sum_{i \in I \setminus A} v'_i r\hat{x}_i \geq 2rD'.$$

- Finally, observe that $\hat{x}$ is a convex combination of these r integral solutions:

  $$\hat{x} = \sum_{i=1}^{r} \frac{x(B_i)}{r},$$

  which implies that the minimum cost can be bounded by $c(\hat{x}) \leq 2c(x')$.

2.5 Framework: Relaxed Separation Oracle

Now the question is "how to solve the strengthened LP relaxation". The given LP has exponential number of constraints and it is not clear if there is an efficient separation oracle for this LP. Interestingly, in the above algorithm, it suffices for x to satisfy the Knapsack Cover Inequality on a single set A.

In fact, all we need is a relaxed separation oracle: given a fractional solution x, it will

- either point out a violated constraint,
- or round x into an integral solution $\hat{x}$ where $c(x) \leq \alpha c(\hat{x})$ for some constant α.

The framework works as follows. First, we guess the true optimal LP value, say OPT. Then, in the ellipsoid method,

- if $c(x) > OPT$, return this violated constraint,
- else if we can round x into $\hat{x}$ so that $c(\hat{x}) \leq \alpha c(x)$, return $\hat{x}$,
- else return some violated constraint that does not allow us to round x.

3 Uniform Hard Capacitated k-median Problem

In this section, we consider Shi Li's algorithm to approximate the Uniform Hard Capacitated k-median using $(1+\epsilon)k$ facilities. In this problem, all facilities have the same capacity $u > 0$, and we are only allowed to open at most k facilities (at most one at each location). The natural LP relaxation for the problem is

$$\begin{aligned}
\text{minimize } & \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} d(i,j)x_{ij} \\
\text{subject to } & \sum_{i\in\mathcal{F}} x_{ij} = 1 \qquad ,\forall j \in \mathcal{C} \\
& \sum_{i\in\mathcal{F}} y_i \leq k \\
& x_{ij} \leq y_i \qquad ,\forall i \in \mathcal{F}, j \in \mathcal{C} \\
& \sum_{j\in\mathcal{C}} x_{ij} \leq u y_i \quad ,\forall i \in \mathcal{F} \\
& 0 \leq x_{ij}, y_i \leq 1.
\end{aligned}$$

3.1 Reduction to soft capacitated k-median problem with $\mathcal{F} = \mathcal{C}$

It turns out that if there is an α-approximation algorithm for the uniform soft capacitated k-median with $\mathcal{F} = \mathcal{C}$, then there is a $(1+4\alpha)$-approximation algorithm for the uniform hard capacitated k-median. To see this, consider any hard capacitated instance $(k, \mathcal{F}, \mathcal{C})$ and fix a α-approximate solution $\mathcal{C}'$ for the soft capacitated instance $(k, \mathcal{C}, \mathcal{C})$. Let OPT_{soft}, OPT_{hard} be the optimal costs of the soft and hard instances, respectively. Note that $OPT_{soft} \leq 2OPT_{hard}$ because, given any solution to the hard $(k, \mathcal{F}, \mathcal{C})$ instance,

we create a feasible solution of the soft $(k, \mathcal{C}, \mathcal{C})$ instance by moving the facility i to the nearest client j that it is serving. The cost of this solution can be at most twice the original cost.

Now we construct the following bipartite graph $G = (\mathcal{C}' \cup \mathcal{F}, E)$ as follows:

- let $\mathcal{M}$ be the minimum cost matching of the bipartite graph on $(\mathcal{C} \cup \mathcal{F})$ where the cost of the edge $(j, i) : j \in \mathcal{C}, i \in \mathcal{F}$ is equal to the distance $d(i, j)$, subject to the constraint that each client is matched exactly one time and each facility can be matched with at most u clients,
- recall that $\mathcal{C}'$ is the set of open facilities in the α-approximate solution (there can be multiple open facilities at a location but $|\mathcal{C}'| \leq k$). We consider the matching $\mathcal{M}'$ between $\mathcal{C}'$ and $\mathcal{C}$, where each client is matched with the open facilities that serves it in the solution.
- finally, we combine $\mathcal{M}$ and $\mathcal{M}'$ to get G: there is an edge (j, i) in G, where $j \in \mathcal{C}'$ and $i \in \mathcal{F}$ iff there exists a client $k \in \mathcal{C}$ such that k is matched with with i in $\mathcal{M}$ and with j in $\mathcal{M}'$. We assign the cost $d(i, k) + d(k, j)$ to this edge.

Note that G may can contain multi-edges, and $|E| = |C|$. (We can think of each edge as client.) Also, by construction, the degree of all vertices is at most G. Now we will transform this bipartite graph so that the number of matched vertices in $\mathcal{F}$ is at most k. This can be done as follows:

- while there exists even cycles, decompose it into 2 disjoint, equal sets of odd and even edges. Remove the set of edges with larger cost and double multiplicities of edges in the other set,
- now consider any connected component which is a tree. If the tree contain any 2 vertices $i_1, i_2 \in \mathcal{F}$ such that their degrees are *strictly* less than u, again, we decompose the even path from i_1 to i_2 and apply the above process. We repeat this process until no changes can be made.
- finally, G will only trees where each tree may contain at most one vertex only the right with degree less than u.

It is clear that this transformation does not increase the cost and each facility can only be matched at most u times. We claim that the number of matched facilities in $\mathcal{F}$ is now at most k and we can take this as a feasible solution. To see this, consider any tree T with at least one edge. Assume that T has e edges. By the above property, we have

$$(|T \cap \mathcal{F}| - 1)u < e \leq |T \cap \mathcal{C}'|u,$$

which implies

$$|T \cap \mathcal{F}| \leq |T \cap \mathcal{C}'|.$$

The claim follows by taking the sum over all trees and notice that $|\mathcal{C}'| \leq k$. How about the connection cost of this solution? Note that the degree of vertices in $\mathcal{C}'$ is still preserved. It means that, for each client k, we can serve it via the route: $k \to$ assigned facility in $\mathcal{C}' \to$ some matched facility in $\mathcal{F}$. Indeed, the total cost is at most

$$\begin{aligned} cost(C') + cost(G) &\leq cost(\mathcal{C}') + cost(\mathcal{M}) + cost(\mathcal{M}') \\ &= 2cost(C') + cost(\mathcal{M}) \\ &\leq 2\alpha OPT_{soft} + OPT_{hard} \\ &\leq 4\alpha OPT_{hard} + OPT_{hard} = (1 + 4\alpha)OPT_{hard}. \end{aligned}$$

3.2 A simple 6-approximation which opens $2k$ facilities

We first apply the standard filtering algorithm on $\mathcal{C}$ as follows. Let $\ell > 1$ be a parameter to be determined. Initially, $\mathcal{C}' := \emptyset$. We consider clients in increasing order of its fractional connection cost. For each client j in this order, we add j into $\mathcal{C}'$ and remove j and all all other clients k such that $d(j,k) \leq 2\ell \max\{C_j, C_k\}$. We then assign facilities in $\mathcal{F}$ to the closest cluster center in $\mathcal{C}'$, breaking ties arbitrarily. Let F_j denote the set of facilities assigned to $j \in \mathcal{C}'$.

Claim 1. *The set $\mathcal{C}'$ of cluster centers satisfies the following properties:*

- *for any $j, j' \in \mathcal{C}'$, $d(j,j') \geq 2\ell \max\{C_j, C_k\}$,*
- *$y(F_j) \geq 1 - 1/\ell$ for all $j \in \mathcal{C}'$.*

Algorithm 1 Simple rounding algorithm

$\alpha_j \leftarrow 0$ for all $j \in \mathcal{C}'$
for $j \in \mathcal{C}$ **do**
  for $i \in \mathcal{F} : x_{ij} > 0$ **do**
    $k \leftarrow$ the cluster center of i (i.e. $i \in F_k$)
    $\alpha_k \leftarrow \alpha_k + x_{ij}$
  end for
end for
for $j \in \mathcal{C}'$ **do**
  open $\lceil \alpha_j / u \rceil$ facilities at location j
end for
find the minimum matching between the set of open facilities and all clients, where each client is matched once and each facility is matched at most u times.
return the min-cost matching

Idea: Let us imagine that each client in $\mathcal{C}$ has a unit of demand and we want to transfer the demand into some open facility. Then the connection cost will be equal to "moving cost" that is required to transfer all demands to the open facilities in our solution. In the following algorithm, we first move all demands from clients j to its serving facilities i in the LP solution. Obviously, this would cost OPT_f. Then, the demand from each facility i will be sent to its cluster center. Finally, we divide the amount of demands at a center by u to obtain the number of facilities to be open at this site. Observe that the total demand at a cluster center can be bounded by the volume of that cluster. Moreover, by construction, the volume of any cluster is at least $1/2$, and we only loose a factor of at most 2 in the size of the solution.

Analysis: We claim that such a matching exists due to the integrality of matching polytope. Consider the fractional matching x' on $\mathcal{C}' \cup \mathcal{C}$ by setting $x'_{ij} = x_{F_i,j}$, for all $i \in \mathcal{C}', j \in \mathcal{C}$. The cost of x' (also called the moving cost as we *move* $x_{F_i,j}$ units of demand from j to i) is equal to $\sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{C}'} d(i,j) x_{F_i,\mathcal{C}}$ which will be analyzed in a moment. Note that the total flow out of each client is exactly 1 and the total flow into each facility j is exactly α_j. By opening $\lceil \alpha_j / u \rceil$ facilities at site j, we guarantee that the amount of incoming flow of j can be as large as $\lceil \alpha_j / u \rceil \cdot u > \alpha_j$. Thus, one can round x' into an integral matching where each client is matched at most once and each facility is matched at most u times.

We first bound the total connection cost. By integrality of b-matching polytope, the connection cost is at most

$$\begin{aligned}
\sum_{j\in\mathcal{C}}\sum_{k\in\mathcal{C}'} d(j,k)\sum_{i\in F_k} x_{ij} &\le \sum_{j\in\mathcal{C}}\sum_{k\in\mathcal{C}'}\sum_{i\in F_k} x_{ij}(d(i,j)+d(i,k)) \\
&\le \sum_{j\in\mathcal{C}}\sum_{k\in\mathcal{C}'}\sum_{i\in F_k} x_{ij}(d(i,j)+d(i,i')) \\
&\le \sum_{j\in\mathcal{C}}\sum_{k\in\mathcal{C}'}\sum_{i\in F_k} x_{ij}(2d(i,j)+d(i',j)) \\
&\le \sum_{j\in\mathcal{C}}\sum_{k\in\mathcal{C}'}\sum_{i\in F_k} x_{ij}(2d(i,j)+2\ell C_j) \\
&= 2OPT_f + 2\ell\sum_{j\in\mathcal{C}} C_j \sum_{k\in\mathcal{C}'}\sum_{i\in F_k} x_{ij} \\
&= 2OPT_f + 2\ell OPT_f = 2(\ell+1)OPT_f,
\end{aligned}$$

where i' is the cluster center of j. Moreover, for each bundle F_j, by LP constraints and construction, we open at most

$$\left\lceil \frac{\sum_{j\in\mathcal{C}}\sum_{i\in F_j} x_{ij}}{u} \right\rceil \le \lceil y(F_j)\rceil$$

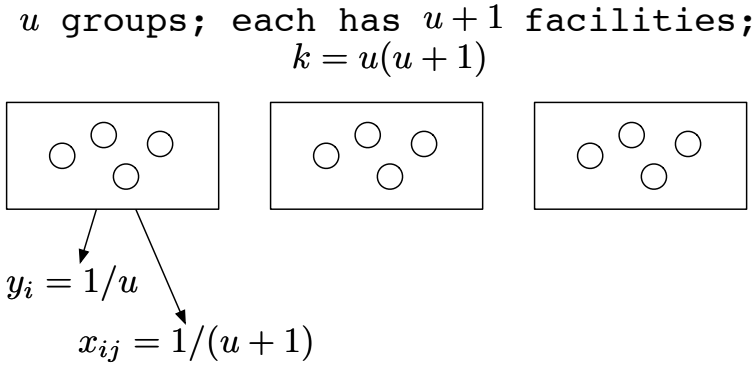
facilities at j. Since

$$\max_{j\in\mathcal{C}'} \frac{\lceil y(F_j)\rceil}{y(F_j)} \le \max_{y\ge 1-1/\ell} \frac{\lceil y\rceil}{y} \le 2,$$

if we set $\ell = 2$. Then the number of open facilities is at most $2\sum_{j\in\mathcal{C}'} y(F_j) \le 2k$. Increasing ℓ does not improve the factor of 2 because $\lim_{\epsilon\to 0}\frac{\lceil 1+\epsilon\rceil}{1+\epsilon} = 2$.

3.3 Integrality gap of the natural LP relaxation

Consider the following instance: $k = u+1, \mathcal{F} = \mathcal{C}$, and $|\mathcal{F}| = |\mathcal{C}| = n = u(u+1)$. (Each facility has capacity u.) The clients are partitioned into u groups of size $u+1$. The distance between any 2 clients is 0 if they are in the same group and 1 otherwise.



Then

- The cost of any integral solution is greater than 1 even when we are allowed to open $2k - 3 = 2(u + 1) - 3 = 2u - 1$ facilities. Indeed, there are u groups, and we cannot afford to open 2 facilities in all groups.
- On the other hand, the fractional cost is exactly 0. This can be achieved by setting $y_i = 1/u$ for all i and $x_{ij} = 1/(u + 1)$ for all i, j within the same group. Indeed, the number of open facilities is $n/u = u + 1 = k$. Each client can connect to itself and other u clients in the group (each with the extent $1/(u + 1)$).

3.4 Strengthening the natural LP with Rectangle Constraints

In the above bad instance, the LP solution will open $(1/u)(u + 1) = 1 + 1/u$ facilities for each group and use these to serve all $u + 1$ facilities in the group. The consequence is that no client would need to connect to some facility outside, causing the LP value to be equal to zero. Can we somehow rule out this possibility? Indeed, any integral would open either one or two facilities per group:

- 1 facility can serve u facilities in the group,
- 2 facilities can serve $u + 1$ facilities in the group (not $2u$).

While it is not possible to avoid opening a fractional number of facilities per group, we can still add some linear constraint to rule out the above event. In particular, we can use linear interpolation on the number of facilities in $[1, 2]$ to get the number of clients which can be served. For example, with $y \in [1, 2]$ open facilities, we require that only $u + y - 1$ clients in the group can be served (and other clients have to connect to outside facilities, hence making the LP value greater than zero). Indeed, this constraint suffices to get rid of the bad instance as only $u + 1 + 1/u - 1 = u + 1/u$ clients in a group are served by $1 + 1/u$ facilities.

Generally, let $f(p, q)$ be the maximum number of clients in a group of size p can be served by opening q facilities. If $p, q \in \mathbb{Z}_+$, we have $f(p, q) = \min\{p, qu\}$. How can we extend this function for $q \in \mathbb{R}_+$. As discussed above, assuming $q \in \mathbb{R}_+$, we define $f(p, q)$ as follows:

- If $q \le \lfloor p/u \rfloor$ then $f(p, q) = \min\{p, qu\} = qu$,
- If $q \ge \lceil p/u \rceil$ then $f(p, q) = \min\{p, qu\} = p$,
- For $q \in [\lfloor p/u \rfloor, \lceil p/u \rceil]$, we use linear interpolation to connect two integer points $(\lfloor p/u \rfloor, u\lfloor p/u \rfloor)$ and $(\lceil p/u \rceil, p)$:
$$f(p, q) = u\lceil p/u \rceil + u(p/u - \lceil p/u \rceil)(q - \lceil p/u \rceil).$$

We rewrite the LP as follows.

$$\begin{aligned}
\text{minimize } & \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} d(i, j) x_{ij} \\
\text{subject to } & \sum_{i \in \mathcal{F}} x_{ij} = 1 \qquad , \forall j \in \mathcal{C} \\
& \sum_{i \in \mathcal{F}} y_i \le k \\
& x_{\mathcal{B},\mathcal{J}} \le f(|\mathcal{J}|, y_{\mathcal{B}}) \qquad , \forall \mathcal{B} \subseteq \mathcal{F}, \mathcal{J} \subseteq \mathcal{C} \\
& 0 \le x_{ij}, y_i \le 1.
\end{aligned}$$

3.5 High-level ideas

In the normal k-median problem, we can simply assign a client to the nearest open facility. However, in this capacitated version, clients are required to be carefully assigned to some facility without violating the capacity constraint. In other words, the assignment variables x_{ij} are also important in this problem. The above 6-approximation algorithm suggests a good strategy: for each client j, move all the demands x_{ij} to the cluster center of i. We can think of the result of this process as a fractional matching where the total weight of incidence edges of a client is exactly one. The integrality of any matching polytope implies that there is a integral solution whose cost is at most the current fractional cost.

Recall that the LP constraint guarantees a bound on the number of open facilities. For each $j \in \mathcal{C}'$, let α_j be the total mass that have been moved to j in the above process and scaled down by u. Let $\beta_j := y(F_j)$ be the (fractional) number of open facilities inside the ball F_j. By construction and the fact that $\sum_{j \in \mathcal{C}'} x_{ij} \le u y_i \ \ \forall i \in \mathcal{F}$, we have $\alpha_j \le \beta_j$.

From now on, we shall call α_j the demand of j and β_j the supply of j. Indeed, we have $\sum_j \beta_j \le k$. In the simple 6-approximation algorithm, we open $\lceil \alpha_j \rceil$ facilities at each location j and show that

$$\sum_{j \in \mathcal{C}'} \lceil \alpha_j \rceil \le 2 \sum_{j \in \mathcal{C}'} \beta_j \le 2k,$$

when $\ell = 2$. How can we improve the factor of 2? In fact, if $\lceil \alpha_j \rceil \approx \beta_j$ or $\alpha_j < \lfloor \beta_j \rfloor$ then we do not lose much when opening $\lceil \alpha_j \rceil$ facilities. Otherwise, $\lceil \alpha_j \rceil / \beta_j$ is large and we have to move the demand from j to some other cluster center. This is our strategy: to move the demand from *bad* cluster centers j to some nearby cluster centers. The challenging part of this process is that we have to bound the connection cost as moving amount f of demand from j to k would incur a cost of $(uf)d(j,k)$. One natural idea is to partition $\mathcal{C}'$ into groups of nearby centers so that the cost of moving the demand among centers within a group is not too large. Because of some technical issues, we would need to bound the cost of moving demand from a subset of centers to some outside centers.

Generally, when shall we move the demand from a set $\mathcal{A}$ of cluster centers to other centers in $\mathcal{C}' \setminus \mathcal{A}$ and how can we bound the incurring cost? Let $\mathcal{S} := \bigcup_{j \in \mathcal{A}} F_j$ be the set of facilities claimed by centers in $\mathcal{A}$. Define

$$y'_{\mathcal{S}} := \sum_{i \in \mathcal{S}, j \in \mathcal{C}} \frac{x_{ij}}{u},$$

and

$$y_{\mathcal{S}} := \sum_{i \in \mathcal{S}} y_i.$$

Recall that we have the demand-supply constraint: $y'_{\mathcal{S}} \le y_{\mathcal{S}}$. Indeed, there is no need to transfer demand from $\mathcal{A}$ if either $y'_{\mathcal{S}} \le \lfloor y_{\mathcal{S}} \rfloor$ or $\lceil y_{\mathcal{S}} \rceil - y_{\mathcal{S}}$ is small. For example, if $\lceil y_{\mathcal{S}} \rceil - y_{\mathcal{S}} \le 1/\ell$ then

$$\frac{\lceil y'_{\mathcal{S}} \rceil}{y_{\mathcal{S}}} \le \frac{\lceil y_{\mathcal{S}} \rceil}{y_{\mathcal{S}}} \le 1 + \frac{1}{y_{\mathcal{S}} \ell} \le 1 + \frac{1}{\ell(1 - 1/\ell)},$$

since $y_{\mathcal{S}} \ge 1 - 1/\ell$.

Therefore, the two main conditions on transferring the demand from $\mathcal{A}$ to $\mathcal{C}' \setminus \mathcal{A}$ are (1) $y'_S \geq \lfloor y_S \rfloor$ and (2) $\lceil y_S \rceil - y_S$ is *large*, say $\lceil y_S \rceil - y_S \geq 1/\ell$. If these conditions are satisfied, then we have the following bound derived from the Rectangle Constraints:

$$\begin{aligned} u(y'_S - \lfloor y'_S \rfloor) d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A}) &\leq \frac{1}{\lceil y_S \rceil - y_S}(4D_S + (4\ell + 2)D'_S) \\ &\leq O(\ell)(D_S + \mathcal{D}'_S) \end{aligned}$$

where

- $d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A}) = \min_{j \in \mathcal{A}, k \in \mathcal{C}' \setminus \mathcal{A}} d(j,k)$ is distance to the nearest facility outside $\mathcal{A}$,
- $D_S = \sum_{i \in S} \sum_{j \in \mathcal{C}} x_{ij} d(i,j)$ and $D'_S = \sum_{i \in S} \sum_{j \in \mathcal{C}} x_{ij} C_j$,
- the LHS $u(y'_S - \lfloor y'_S \rfloor) d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A})$ is simply the cost of moving the excessive amount of demand $(y'_S - \lfloor y'_S \rfloor)$ from $\mathcal{A}$ to the closest center in $\mathcal{C}' \setminus \mathcal{A}$,
- if we take sum of the RHS over disjoint sets of facilities, we get $D_{\mathcal{F}} = D'_{\mathcal{F}} = OPT_f$.

To this end, Shi Li suggests the use of *neighborhood trees*. Basically, we partition $\mathcal{C}'$ into several *almost* disjoint sets. The vertices inside each set form a rooted tree with certain properties. Focus on a tree $T = (V, E)$ with root r. Initially, each cluster center $j \in V$ has demand $\alpha_j = y'_{F_j}$ and supply $\beta_j = y_{F_j}$. We shall redistribute the demand and supply within T such that

- $\alpha_j \leq \beta_j \ \ \forall j$, at any time during the process,
- the total demand and supply are not changed,
- $\lceil \alpha_j \rceil \leq \beta_j + 1/\ell \ \ \forall j \neq r$.

Note that while we may redistribute the supply within T, this is for the analysis only and does not incur any cost. After the moving process, we simply open $\lceil \alpha_j \rceil$ facilities at location j as before. It is easy to see that the last property suffices to bound the number of open facilities to within $(1+\epsilon)k$. Note that $\beta_V \geq |V|(1 - 1/\ell) > (|V| - 1)(1 - 1/\ell)$. The number of open facilities in T is

$$\begin{aligned} \sum_{j \in V} \lceil \alpha_j \rceil &\leq \sum_{j \in V, j \neq r} (\beta_j + 1/\ell) + \alpha_r + 1 \\ &\leq \sum_{j \in V, j \neq r} (\beta_j + 1/\ell) + \beta_r + 1 \\ &\leq \beta_V + 1 + (|V| - 1)/\ell \\ &\leq \beta_V + 1 + \frac{\beta_V}{\ell - 1} \\ &= \beta_V \left(1 + 1/\beta_V + \frac{1}{\ell - 1}\right). \end{aligned}$$

The term $1/\beta_V$ can be ignored if we choose $|V|$ large enough. In fact, we will create the tree T with at least ℓ vertices so that $1/\beta_V \leq \frac{1}{(\ell-1)(1-1/\ell)}$.

Now we have all the main ingredients. How can we combine all these ideas to achieve the above goal while still being able to bound the cost? Shi Li suggests a clever recursive moving process. The edges of T is sorted and carefully grouped into sets $E_1, E_2, \ldots, E_h$ such that

- the edges in group E_i is shorter than or equal to edges in E_{i+1},
- for any group E_i, the difference between the longest and shortest edges is bounded:

$$\frac{e_{max}}{e_{min}} \leq \exp(O(|V|)).$$

Then the tree is further partitioned into many level-sets as follows. Let $E_{\leq i} = \bigcup_{j \leq i} E_j$. Each subtree of T induced by edges in $E_{\leq i}$ is called a level-i set. Indeed, the (only) level-h set is exactly T. Note that these level-sets form a laminar family. As mentioned, the redistribution process is done in a recursive manner: when processing some level-i set $\mathcal{A}$, all of its level-$(i-1)$ subsets are already *processed*. A level-$(i-1)$ set $\mathcal{A}' \subseteq \mathcal{A}$ is *processed* if either

- all centers $j \in \mathcal{A}'$ satisfy: $\lceil \alpha_j \rceil \leq \beta_j + 1/\ell$ (this is our ultimate goal),
- indeed, the mentioned condition cannot be always achieved. But if this cannot be done, by inductive hypothesis, we can redistribute the demand and supply in such a way that $\alpha_j = \beta_j \in \mathbb{Z}$ for all j except a *bad* center v, for which $\alpha_v < \beta_v$ (and maybe $\lceil \alpha_v \rceil > \beta_v + 1/\ell$).

Main idea: Now when processing $\mathcal{A}$, we would consider all of its level-$(i-1)$ subsets $\mathcal{A}'$. If $\mathcal{A}'$ satisfies the first condition, there is no need to move any mass out of $\mathcal{A}'$. Otherwise, there is one vertex v such that $\lceil \alpha_v \rceil > \beta_v - 1/\ell$ and all other vertices have balanced demand and supply. We shall round both α_v and β_v down to $\lfloor \alpha_v \rfloor$. In other words, we collect an amount $\alpha_v - \lfloor \alpha_v \rfloor = y'_{\mathcal{S}} - \lfloor y'_{\mathcal{S}} \rfloor$ of demand from $\mathcal{A}$, where $\mathcal{S}$ is the facilities claimed by centers in $\mathcal{A}'$, and move it to $\mathcal{A} \setminus \mathcal{A}'$ in the attempt of making $\mathcal{A}$ balanced. Finally, we can bound the moving cost:

- The moving distance is indeed at most the sum of all edges in $\mathcal{A}$. Let e_{min} be the shortest edge in E_i then $e_{max} \leq \exp(O(|V|))e_{min}$. Thus, the total distance can be bounded by

$$\exp(O(|V|))e_{min}.$$

- The neighborhood tree has another useful property that

$$e_{min} \leq 2d(\mathcal{A}', \mathcal{C}' \setminus \mathcal{A}'),$$

- Recall that we want to move $y'_{\mathcal{S}} - \lfloor y'_{\mathcal{S}} \rfloor$ demand out of $\mathcal{A}'$. Also, the fact that $\lceil \alpha_v \rceil > \beta_v + 1/\ell$ and all other vertices are balanced imply that

$$\lceil y_{\mathcal{S}} \rceil - y_{\mathcal{S}} \geq \lceil y'_{\mathcal{S}} \rceil - y_{\mathcal{S}} = \lceil \alpha_v \rceil - \beta_v > 1/\ell$$

  Thus, we can apply the Rectangle Bound on the moving distance:

$$\begin{aligned} cost &\leq \exp(O(|V|)) \cdot ((y'_{\mathcal{S}} - \lfloor y'_{\mathcal{S}} \rfloor)u) \cdot e_{min} \\ &\leq \exp(O(|V|)) \cdot ((y'_{\mathcal{S}} - \lfloor y'_{\mathcal{S}} \rfloor)u) \cdot d(\mathcal{A}', \mathcal{C}' \setminus \mathcal{A}') \\ &\leq \exp(O(|V|))(\mathcal{D}_{\mathcal{S}} + \mathcal{D}'_{\mathcal{S}}). \end{aligned}$$

- Recall that by taking the sum of $\mathcal{D}_{\mathcal{S}}$ or $\mathcal{D}'_{\mathcal{S}}$ on disjoint sets $\mathcal{S}$, we get OPT_f.
- Our final ingredient is to choose V carefully so that $|V| \leq \ell^2$ so that we only lose a factor of $O(\exp(\ell^2))$.

3.6 Neighborhood Trees and Geometric Grouping

In this section, we will discuss how to build neighborhood trees. Recall that our goal is to partition $\mathcal{C}'$ into trees and, for a fixed tree $T = (V, E)$, the edges of T are sorted and grouped into $E_1, \ldots, E_h$ such that

- $\ell \le |V| \le \ell^2$,
- for any group E_i: $e_{max}/e_{min} \le \exp(O(|V|))$, where e_{max}, e_{min} are the lengths of the longest and shortest edges in E_i, respectively,
- let $E_{\le i} = E_1 \cup \ldots \cup E_i$ and $\mathcal{A} \subseteq V$ be any maximal subtree using edges in $E_{\le i}$, we want

$$d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A}) \ge \min_{e \in E_{i+1}} e/2.$$

One way to achieve the third property is to construct *neighborhood* trees such that for any non-root vertex v, we have

$$d(v, p(v)) = \min d(v, \mathcal{C}' \setminus \Lambda(v)),$$

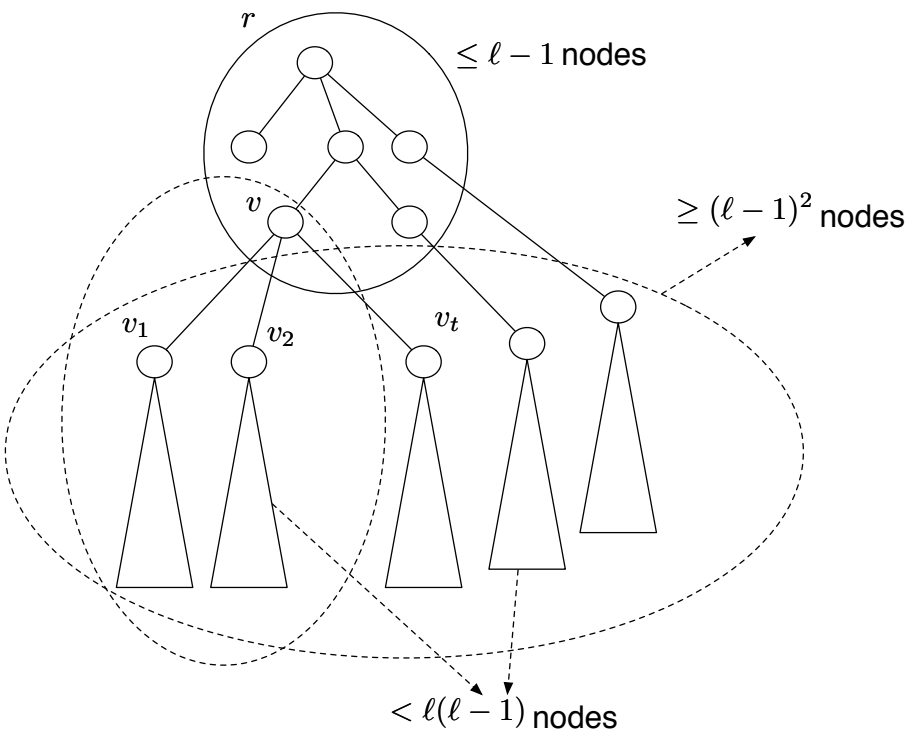
where $p(v)$ is the parent of v and Λ is the set of vertices of the subtree rooted at v. We will see how to exploit this later. Now we show how to partition $\mathcal{C}'$ into neighborhood trees such that the size of each tree is in $[\ell, \ell^2]$. The algorithm is as follows:

- initially, each node is considered a tree. While there exists a tree T rooted at v with size less than ℓ, let $r = \arg\min_{r \in \mathcal{C}' \setminus \Lambda(v)} d(v, r)$ and then "hang" T onto r. It is easy to check that all trees are neighborhood trees with size at least ℓ after this step.
- next, we need to *break* some large trees whose size exceeds ℓ^2. Fix any large tree $T = (V, E)$ where $|V| > \ell^2$. Recall that, initially, T only contains the root r. Then T is *growing* gradually by absorbing smaller trees (of size less than ℓ). Let us shrink such a tree into a supernode for a moment and let the size of a supernode be the size of the original tree. Now T is composed of supernodes. We find a *deepest* supernode r such that the size of the subtree rooted at r is at least $\ell(\ell - 1)$, i.e. the sizes of subtrees of its children are less than $\ell(\ell - 1)$.

  Now the total number of (original) nodes in the subtree of r but not belonging to the supernode r is at least $\ell(\ell - 1) - (\ell - 1) = (\ell - 1)^2$. These nodes will be hanged into at most $\ell - 1$ nodes in r. Thus, there exists a node $v \in r$ such that the total number of nodes hanged into v is at least $(\ell - 1)^2/(\ell - 1) = \ell - 1$. Let children of v be $v_1, \ldots, v_t$. Then we greedily build a tree T' consisting of v and some subtrees rooted at $v_1, v_2, \ldots, v_m$ such that T' has at least ℓ nodes and at most ℓ^2 nodes. This can be done due to the fact that the size of subtree of v is at least $(\ell - 1) + 1 = \ell$ and the size of $v_1, \ldots, v_t$ is less than $\ell(\ell - 1)$.

  We break T into T' and $T \setminus T'$. It can be checked easily that both trees are still neighborhood tree. By repeating the process, the final trees will have size in $[\ell, \ell^2]$.

Now fix any tree $T = (V, E)$ with root r. Assuming for the moment that we have a partition $E = E_1 \cup \ldots \cup E_h$, we will use the definition of neighbor trees to derive the third property: $d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A}) \ge \min_{e \in E_{i+1}} e/2$ for any maximal connected component induced by $E_{\le i}$. There is a technical problem here: if $\mathcal{A}$ contains the root then we do not know how to lower-bound $d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A})$. This is because we have



broken large trees and the distance from the root to its previous parent could be very small compared to distance between vertices within the subtree. Thus, we have to add another constraint that the root $r \notin \mathcal{A}$. See the following figure for notations.

In this figure, blue edges are in $E_{\leq i}$. Let $L = \sum_{e \in E_{\leq i}} e$ be the total length of edges in $E_{\leq i}$. Let v be the root of the subtree induced by $\mathcal{A}$. Note that $\mathcal{C}' \setminus \mathcal{A} = (\mathcal{C}' \setminus \Lambda(v)) \cup (\Lambda(v) \setminus \mathcal{A})$. Let e_{min} be the length of the shortest edge in E_{i+1}. Consider the following two cases:

- Assume $d(x, x') = d(\mathcal{A}, \mathcal{C}' \setminus \Lambda(v))$. Then we have

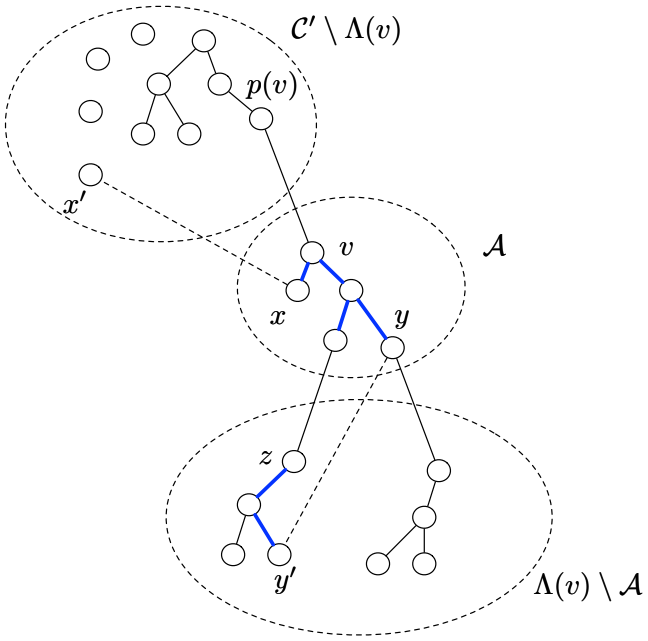
$$\begin{aligned} e_{min} &\leq d(v, p(v)) \leq d(v, x') \\ &\leq d(v, x) + d(x, x') \\ &\leq L + d(x, x'). \end{aligned}$$

- Assume $d(y, y') = d(\mathcal{A}, \Lambda(v) \setminus \mathcal{A})$. Let z be the highest vertex such that the path from y' to z only uses edges in E_{i+1}. We have

$$\begin{aligned} e_{min} &\leq d(z, v) \\ &\leq d(v, y) + d(y, y') + d(y', z) \\ &\leq L + d(y, y'). \end{aligned}$$

Thus, we have proved that

$$d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A}) \geq e_{min} - \sum_{e \in E_{\leq i}} e.$$



Therefore, to get $d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A}) \geq e_{min}/2$, it suffices to require that $e_{min} \geq 2 \sum_{e \in E_{\leq i}} e$. This suggests the following geometric grouping scheme:

- sort all edges $e_1 \leq e_2 \leq \ldots \leq e_{|V|-1}$ and initialize $E_1 = \emptyset, j = 1$, and $h = 1$,
- for each e_j, if $e_j < 2(e_1 + \ldots + e_{j-1})$ then $E_h \leftarrow E_h \cup \{e_j\}$. Else, we create a new level-set $E_{h+1} \leftarrow \{e_j\}$ and increase $h \leftarrow h + 1$.

Finally, we prove that, in any group E_i, $e_{max}/e_{min} \leq \exp(O(|V|))$. Assume E_i contains the following edges:

$$e_{min} \leq e'_1 \leq e'_2 \leq \ldots \leq e'_t = e_{max}.$$

Let $L = \sum_{e \in E_{\leq i-1}} e$. Then, by construction, we have $e_{min} \geq 2L$ and

$$e'_1 \leq 2(L + e_{min}) \leq 3e_{min}.$$

We prove by induction that $e'_j \leq 3^j \cdot e_{min}$ as follows.

$$\begin{aligned} e'_j &\leq 2(L + e_{min} + e'_1 + \ldots + e'_{j-1}) \\ &\leq 3e_{min} + 2 \cdot 3^1 \cdot e_{min} + \ldots + 2 \cdot 3^{j-1} \cdot e_{min} \\ &\leq 2e_{min}(1/2 + 1 + 3^1 + \ldots + 3^{j-1}) \\ &= \left(1 + 2\frac{3^j - 1}{3 - 1}\right) e_{min} = 3^j \cdot e_{min}. \end{aligned}$$

Since E_i contains at most $|V| - 1$ edges, $e_{max} = \exp(O(|V|))e_{min}$.

3.7 Rounding Rectangle LP solution

Focus on any neighborhood tree $T = (V, E)$. As discussed before, we want to move the demands and supplies among vertices of T. Note that moving supplies does not incur any cost. Our goal is to make these vertices *balanced*, i.e. $\lceil \alpha_j \rceil \le \beta_j + 1/\ell$ for as many $j \in V$ as possible. In fact, we may afford to have a constant number of *unbalanced* vertices.

Fix any level-i set $\mathcal{A}$. Recall that a level-$(i-1)$ set $\mathcal{A}' \subseteq \mathcal{A}$ is *processed* if either

- all centers $j \in \mathcal{A}'$ satisfy: $\lceil \alpha_j \rceil \le \beta_j + 1/\ell$ (this is our ultimate goal),
- or, $\alpha_j = \beta_j \in \mathbb{Z}$ for all j except a *bad* center v, for which $\alpha_v < \beta_v$ (and maybe $\lceil \alpha_v \rceil > \beta_v + 1/\ell$).

If $r \in \mathcal{A}$ then we cannot afford to move demands out of $\mathcal{A}$ because we have no bound on $d(\mathcal{A}, \mathcal{C}' \setminus \mathcal{A})$. In this case, we will try to make all $j \neq r$ *balanced* instead. If $r \notin \mathcal{A}$, we want $\mathcal{A}$ to have one of the above two properties as well. To process $\mathcal{A}$, we focus on the level-$(i-1)$ subsets of $\mathcal{A}$. The first step is to collect demands and supplies from these subsets. Initialize $S_{demand} \leftarrow 0, S_{supply} \leftarrow 0$. For each subset $\mathcal{A}' \subset \mathcal{A}$ of level $i-1$:

1. We first collect supply from all vertices $j \in \mathcal{A}'$ or $j \in \mathcal{A}$ such that $\lceil \alpha_j \rceil < \beta_j$:

$$S_{supply} \leftarrow S_{supply} + \beta_j - \lceil \alpha_j \rceil$$

$$\beta_j \leftarrow \lceil \alpha_j \rceil$$

2. Then if there is a bad center v and $\lceil \alpha_v \rceil > \beta_v + 1/\ell$, then we collect its demand and supply:

$$S_{demand} \leftarrow S_{demand} + \alpha_v - \lfloor \alpha_v \rfloor$$

$$S_{supply} \leftarrow S_{supply} + \beta_v - \lfloor \alpha_v \rfloor$$

$$\alpha_v \leftarrow \lfloor \alpha_v \rfloor, \quad \beta_v \leftarrow \lfloor \alpha_v \rfloor$$

After this step, we have (1) $S_{demand} \le S_{supply}$ and (2) $\beta_j \le \lceil \alpha_j \rceil \le \beta_j + 1/\ell$. If $\mathcal{A}$ contains the root r, then we simply move the collected demand and supply into r and we are done. (Note that if $\mathcal{A}' \subset \mathcal{A}$ contains the root, then there is no need to move collect demands and supplies from $\mathcal{A}'$.)

Assume that $r \notin \mathcal{A}$. Then we will try to make as many $\alpha_j = \beta_j \in \mathbb{Z}$ as possible by redistributing the collected S_{demand} and S_{supply}. This can be done by first increasing some fractional α_j until it is equal to β_j. Then we increase both α_j, β_j until both of them are integral. Note that we maintain the inequality $S_{demand} \le S_{supply}$ at the end of the process. If this can be done successfully, then we move the remaining demand and supply to some arbitrary vertex and the second property is satisfied. Otherwise, we get stuck because we ran out of demand during the process ($S_{demand} = 0$) first. If this is the case, we still maintain the property $\beta_j \le \lceil \alpha_j \rceil \le \beta_j + 1/\ell$ for all j. Finally, moving the remaining S_{supply} to any vertex does not violate this property either.

How to bound the moving cost? One important thing is that we only collect demand from *bad / unbalanced* centers and each level-$i-1$ set $\mathcal{A}'$ only contains at most one such bad center. By the algorithm, we only move at most $\alpha(\mathcal{A}') - \lfloor \alpha(\mathcal{A}') \rfloor = y'_{\mathcal{S}} - \lfloor y_{\mathcal{S}} \rfloor$ units of demand out of $\mathcal{A}'$.

4 Non-uniform Soft Capacitated k-median Problem

Now we consider the non-uniform capacitated k-median problem, in which each facility i has a capacity $u_i > 0$. As the capacities are different, the hard CKM problem is not equivalent to the soft version with $\mathcal{F} = \mathcal{C}$ anymore. It is still an open question whether we can approximate hard CKM problem by opening $(1+\epsilon)k$ facilities within some constant factor. In this section, we will discuss Shi Li's new algorithm which opens at most $(1+\epsilon)k$ facilities (at most 2 facilities per location) and gives an approximation ratio of $O\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}\right)$. For simplicity, given a soft CKM instance, we make k copies of each facility so that the indicator $y_i \in [0,1]$ for all i in the LP relaxation:

$$\begin{aligned}
\text{minimize } & \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} d(i,j)x_{ij} \\
\text{subject to } & \sum_{i\in\mathcal{F}} x_{ij} = 1 \qquad ,\forall j \in \mathcal{C} \\
& \sum_{i\in\mathcal{F}} y_i \le k \\
& x_{ij} \le y_i \qquad ,\forall i \in \mathcal{F}, j \in \mathcal{C} \\
& \sum_{j\in\mathcal{C}} x_{ij} \le u_i y_i \quad ,\forall i \in \mathcal{F} \\
& 0 \le x_{ij}, y_i \le 1.
\end{aligned}$$

4.1 A simple $(4, 11)$-approximation algorithm

Again, we first filter the set $\mathcal{C}$ of clients to get $\mathcal{C}'$ containing cluster centers with radius $4\max\{C_j, C_{j'}\}$. We also move the demands to cluster centers as in the $(2,6)$-approximation algorithm for the soft CKM problem with $\mathcal{F} = \mathcal{C}$. Then each center $j \in \mathcal{C}'$ has $x_{F_j,C}$ units of demand. The moving cost up to this point is at most $6OPT_f$. However, we cannot open any facility at this site because j may not be in $\mathcal{F}$.

Now the trick is to move the demand $x_{F_j,C}$ from each cluster center j back to facilities in F_j. Let α_i be the amount of demand to be moved into facility i. We use the following LP as a guide to reroute demands:

$$\begin{aligned}
\text{minimize } & \sum_{i\in\mathcal{F}_j} \alpha_i d(i,j) \\
\text{subject to } & \sum_{i\in\mathcal{F}_j} \alpha_i = x_{F_j,C} \\
& \sum_{i\in F_j} \alpha_i/u_i \le y(F_j) \\
& \alpha_i \in [0, u_i] \qquad ,\forall i \in \mathcal{F}_j
\end{aligned}$$

Let α^* be any vertex solution of this LP. It is easy to see that α^* has at most 2 fractional values. Note that objective function of the LP gives us the moving cost. Moreover, it is easy to check that setting $\alpha_i = x_{i,C},\ \ \forall i \in \mathcal{F}_j$ yields a feasible solution. Thus, the moving cost of α^* is at most $\sum_{i\in\mathcal{F}_j} x_{i,C}d(i,j)$.

Summing this upper-bound over all cluster centers gives $5OPT_f$:

$$\begin{aligned}\sum_{j\in\mathcal{C}'}\sum_{i\in F_j} x_{i,C}d(i,j) &= \sum_{j\in\mathcal{C}'}\sum_{i\in F_j}\sum_{k\in\mathcal{C}} x_{ik}d(i,j)\\ &\le \sum_{j\in\mathcal{C}'}\sum_{i\in F_j}\sum_{k\in\mathcal{C}} x_{ik}(d(i,k)+4C_k)\\ &= \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{C}} x_{ik}d(i,k) + 4\sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{C}} x_{ik}C_k = 5OPT_f,\end{aligned}$$

where, in the second inequality, we use the fact that $d(i,j) \le d(i,q) \le d(i,k) + d(k,q) \le d(i,k) + 4C_k$ with q being the cluster center of k. Thus, the total moving cost is at most $11OPT_f$.

As before, we open $\lceil \alpha_i^*/u_i \rceil$ facilities at each site i. Again, the first constraint guarantees that one can round the corresponding fractional matching without any loss. Bounding the number of open facilities in F_j is now easy due to the fact that α^* is almost integral:

$$\sum_{i\in F_j} \lceil \alpha_i^*/u_i \rceil \le \lfloor y(F_j) \rfloor + 2.$$

Since

$$\frac{\lfloor y(F_j)\rfloor + 2}{y(F_j)} \le \max_{y\ge 1/2} \frac{\lfloor y\rfloor + 2}{y} = 4,$$

for all j, we open at most $4k$ facilities in total.

4.2 High-level ideas

In the previous section, we describe a $(4, 11)$-approximation algorithm. Basically, there are three main steps. First, the original demands are moved from clients to facilities. The moving cost in this step is obviously equal to OPT_f. Next, the demands from all facilities are moved to the corresponding, closest cluster centers. Finally, for each cluster, we redistribute the demands back to facilities inside that cluster by using an LP as a guidance. Moreover, this LP has "almost" integral vertex solutions which allows us to open at most 2 more facilities per cluster. The total moving costs in both second and third steps are at most $5OPT_f$ via simple distance bounds. We loose a factor of 4 in the number of open facilities because the opening volume of each cluster can be as small as $1/2$. How can we reduce this factor to $(1 + \epsilon)$?

Observe that if we can somehow get a set of cluster centers with volume at least $1/\epsilon$, then the idea of using an LP to redistribute the demands within this set while opening some small extra number of facilities, say c, works fine in this case. To be more precise, we will first move all demands from cluster centers in this set to some specific center j^* and then redistribute the demands from j^* to facilities claimed by some cluster center in this set. Note that the ratio of open facilities is now bounded by $\max_{y\ge 1/\epsilon}(\lfloor y\rfloor + c)/y \le (1 + c\epsilon)$. The main difficulty is to bound the connection cost. In the $(4, 11)$-approximation, it is relatively easy to bound the cost when moving demands within a single cluster. However, in the approach, we may have to move the demands from one cluster center to another. How can we bound this cost?

Let us consider a simplified case where we want to move all demands out of a group J of cluster centers. Recall that $F(J) = \bigcup_{j\in J} F_j$ and $x_{F(J),\mathcal{C}}$ is the total demand to be *moved out* of J. For simplicity, we further assume that the centers in J are *close* to each other and these distances can be bounded by $d(J, \mathcal{C}' \setminus J)$ so that the moving cost in this case can be bounded by

$$B = O(1) \times d(J, \mathcal{C}' \setminus J) \times x_{F(J),\mathcal{C}}.$$

We want to upper-bound this in terms of OPT_f. Unfortunately, there is no easy answer for this and the following quantity will be useful:

$$\pi(J) := \sum_{j\in\mathcal{C}} x_{F(J),j}(1 - x_{F(J),j}).$$

Note that if $F(J)$ is *isolated*, i.e. all clients are either served by J or $\mathcal{C} \setminus J$ (but not both), then $\pi(J) = 0$. Intuitively, this measures the number of clients that are served by both J and $\mathcal{C} \setminus J$. The quantity $\pi(J)$ is an important threshold for $x_{F(J),j}$ that indicates whether B can be bounded by OPT_f. We can prove the following technical bound:

$$\begin{aligned}
d(J, \mathcal{C}' \setminus J)\pi(J) &= d(J, \mathcal{C}' \setminus J) \sum_{j\in\mathcal{C}} x_{F(J),j}(1 - x_{F(J),j}) = d(J, \mathcal{C}' \setminus J) \sum_{j\in\mathcal{C}} x_{F(J),j} \sum_{i'\in\mathcal{F}\setminus F(J)} x_{i'j} \\
&= \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J), i'\in\mathcal{F}\setminus F(J)} x_{i,j}x_{i'j}d(J, \mathcal{C}' \setminus J) \le \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J), i'\in\mathcal{F}\setminus F(J)} x_{i,j}x_{i'j}(d(\mathcal{C}' \setminus J, i') + d(i', J)) \\
&\le \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J), i'\in\mathcal{F}\setminus F(J)} x_{i,j}x_{i'j}(2d(i', J)) \\
&\le 2 \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J), i'\in\mathcal{F}\setminus F(J)} x_{i,j}x_{i'j}(d(i', j) + d(j, i) + d(i, J)) \\
&= 2 \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J)} x_{i,j} \sum_{i'\in\mathcal{F}\setminus F(J)} x_{i'j}(d(i', j) + d(j, i) + d(i, J)) \\
&\le 2 \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J)} x_{i,j} \sum_{i'\in\mathcal{F}} x_{i'j}(d(i', j) + d(j, i) + d(i, J)) \\
&\le 2 \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J)} x_{i,j} \sum_{i'\in\mathcal{F}} x_{i'j}(d(i', j) + d(j, i) + d(i, \sigma(i))) \\
&= 2 \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J)} x_{i,j}(C_j + d(j, i) + d(i, \sigma(i))) \\
&\le 2 \sum_{j\in\mathcal{C}, i\in\mathcal{F}(J)} x_{ij}(5C_j + 2d(j, i)) \le O(1) \times (D(F(J)) + D'(F(J))).
\end{aligned}$$

where $D(F(J)) = \sum_{i\in F(J), j\in\mathcal{C}} x_{ij}C_j$ and $D'(F(J)) = \sum_{i\in F(J), j\in\mathcal{C}} x_{ij}d(i, j)$. (Recall that the sum of D, D' on disjoint sets of facilities will be at most OPT_f.) Now there are two cases:

- Case $x_{F(J),\mathcal{C}} \le O(1) \times \pi(J)$: we say that J is non-concentrated and, by the above bound, we have

$$B \le O(1)(D(F(J)) + D'(F(J))).$$

- Case $x_{F(J),\mathcal{C}} > O(1) \times \pi(J)$: we say that J is concentrated. In this case, B cannot be bounded in terms of D, D'. However, Shi Li suggests a very clever way to make J become non-concentrated. It can be shown that, if the opening mass $y(F(J))$ is not too large, say $O(1)$, then we can *guess* some subset of facilities inside $F(J)$ to be open and connect most of the clients in J to these facilities so that the remaining demand is small, and hence can be bounded by $\pi(J)$. Intuitively, if the LP solution says that opening $y(F(J))$ facilities is enough, then there exists a facility set of size $\lfloor y(F(J)) \rfloor$ which serves *most* of the local clients locally.

In the above discussion, there are two main sub-problems we still need to solve:

- First, we need a new clustering algorithm which is able to partition the set $\mathcal{C}'$ into disjoint sets. The first requirement is that each set T in this partition should have volume large enough, say at least $\Theta(1/\epsilon)$. In addition, to bound the moving cost, we further partition T into disjoint *groups* so that the volume of each group is $O(1)$. Then we will be able transfer the demands among these groups as discussed before.

- Second, for a given non-concentrated group $J \subseteq \mathcal{C}'$ with volume small enough, we need to *pre-open* some facilities in $F(J)$ and *pre-assign* clients to these facilities so that (1) the remaining demand is small enough and J becomes non-concentrated, (2) the cost of pre-assignment is not too big, and (3) the number of *pre-opened* facilities is *small* so that we can still open some more facilities in $F(J)$ without exceeding the fractional value $y(F(J))$ by too much. This can be done by using a configuration LP.

4.3 Preprocessing step

4.3.1 Configuration LP

As discussed above, we have to deal with some concentrated group J having a large amount of demand: $x_{F(J),\mathcal{C}} > O(1) \times \pi(J)$. The idea is to open some *small, good* subset of facilities in $F(J)$ and pre-assign some clients in J so that the residual instance would be simplified and J would become non-concentrated. Actually, this can be done if the total mass $y(F(J)) = O(1)$ (in other words, the LP says that we only need to open a few facilities in the optimal LP solution). In this paper, let us also fix any optimal integral solution.

To this end, for each subset $F \subseteq \mathcal{F}$, we define the following variables:

- let $\mathcal{S} := \{S \subseteq F : |S| \leq b\}$ for some constant b to be determined. Let $\perp$ denote "any subset of F of size greater than b". We also view $\perp$ as a set and say that $i \in \perp$ for all $i \in F$. Let $\tilde{\mathcal{S}} := \mathcal{S} \cup \{\perp\}$.

- for every $S \in \tilde{S}$, let z_S^F denote that event that, in the optimal solution, the set of open facilities in F is exactly S,

- for every $S \in \tilde{S}$ and $i \in F$, let $z_{S,i}^F$ denote the event that $z_S^F = 1$ and $y_i = 1$,

- for every $S \in \tilde{S}$, $i \in F$, and $j \in \mathcal{C}$, let $z_{S,i,j}^F$ denote the event that $z_{S,i}^F = 1$ and j is connected to i in the optimal solution.

The following constraints will be valid:

- exactly one set $S \in \tilde{S}$ will be chosen: $\sum_{S\in\tilde{S}} z^F_S = 1$,
- if i is open then it should belong to some set S: $\sum_{S\in\tilde{S}:i\in S} z^F_{S,i} = y_i$ for all $i \in F$,
- if j is connected to i then $z^F_{S,i,j} = 1$ for one set $S \in \tilde{\mathcal{S}}$: $\sum_{S\in\tilde{S}:i\in S} z^F_{S,i,j} = x_{ij}$ for all $i \in F$ and $j \in \mathcal{C}$,
- the following constraints follows by definition: $0 \le z^F_{S,i,j} \le z^F_{S,i} \le z^F_S$, $z^F_{S,i} = z^F_S$ (if $S \neq \perp$), and $\sum_{i\in S} z^F_{S,i,j} \le z^F_S$, for all $S \in \tilde{S}, j \in \mathcal{C}$,
- capacity constraints: $\sum_{j\in\mathcal{C}} z^F_{S,i,j} \le u_i z^F_{S,i}$ for all $S \in \tilde{S}, i \in S$,
- if $z^F_\perp = 1$, then more than b facilities are open in S: $\sum_{i\in F} z^F_{\perp,i} \ge b z^F_\perp$.

Note that the configuration LP has exponentially many variables and constraints. However, we can still apply the Relaxed Separation Oracle because we only want the above constraints to hold for polynomially many sets $F \subseteq \mathcal{F}$. The idea is as follows:

- We first rewrite the LP so that it only contains x, y variables. For any subset $F \subseteq \mathcal{F}$, the above constraints can be rewritten as $M\vec{z} \ge b + M'\vec{x} + M''\vec{y}$, where M, M', M'' are some matrices, b is a column vector, and $\vec{x}, \vec{y}, \vec{z}$ are column vectors containing x, y, z variables. Now, for a fixed solution $(\vec{x}, \vec{y})$, this inequality holds for some vecotr $\vec{z}$ iff for all $\vec{g}$ that $\vec{g}^T M = 0$, we have $\vec{g}^T(b + M'x + M''y) \le 0$. Thus, we can define an LP in which there is a constraint $\vec{g}^T(b + M'x + M''y) \le 0$ for each $\vec{g}$ such that $\vec{g}^T M = 0$.
- The new LP contains exponentially many x, y variables and infinite number of constraints. However, we do not really need to solve it. In fact, our algorithm will need the additional constraints to hold for a polynomial number of subset $F \in \mathcal{F}$. Now, for such an subset F, we can easily verify if there exists a vector $\vec{g}$ such that $\vec{g}^T M = 0$ and $\vec{g}^T(b + M'x + M''y) > 0$. If yes, we return this violated constraint for the Oracle. Otherwise, we will be able to round it into a good integral solution.

4.3.2 Pre-assignment

Lemma 1. *Let ℓ be some constant to be determined and $\ell' = O(\ell \log \ell)$ be some large enough constant. Suppose J is any set of cluster centers such that $x_{F(J),\mathcal{C}} > \ell' \times \pi(J)$ (i.e., J is non-concentrated) and the opening volume is small enough: $y(F(J)) \le 2\ell$. If all the related constraints for the set $F(J)$ in the configuration LP are satisfiable, then we can pre-open a set $S \subseteq F(J)$ and pre-assign a set $R \subseteq \mathcal{C}$ of clients to S so that*

1. *each facility in S only serves at most u_i clients,*
2. *J becomes concentrated: $x_{F(J),\mathcal{C}\setminus R} \le \ell'\pi(J)$,*
3. *After opening S, we can still afford to open some $\frac{x_{F(J),\mathcal{C}\setminus R}}{x_{F(J),\mathcal{C}}} y(F(J))$ facilities in $F(J)$:*

$$\frac{x_{F(J),\mathcal{C}\setminus R}}{x_{F(J),\mathcal{C}}} y(F(J)) + |S| \le (1 + 1/\ell) y(F(J)),$$

4. the pre-assignment cost is at most $\ell' D(F(J))$.

Proof. Let $z^{F(J)}$ be the vector of the z's variables related to $F(J)$. Since we only the set $F(J)$, we will omit the superscript $F(J)$ while using these variables in this proof and write F instead of $F(J)$. Also, this will be a probabilistic proof. Recall that $\sum_{S\in\tilde{S}} z_S = 1$. It means that we have a distribution on subsets of size at most b. We will randomly choose a set S according to distribution and show that the claimed properties holds with positive probability.

We have to avoid the the event $\perp$ because we do not have any useful information about which facilities should be open in this case. Let $\mathcal{E}$ denote the *good* event that S is not $\perp$. By LP constraints, We have

$$\begin{aligned}\sum_{S\in\mathcal{S}} z_S|S| + bz_\perp &\le \sum_{S\in\mathcal{S}}\sum_{i\in S} z_{S,i} + \sum_{i\in F} z_{\perp,i} \\ &= \sum_{S\in\tilde{S}}\sum_{i\in S} z_{S,i} = \sum_{i\in F}\sum_{S\in\tilde{S}:i\in S} z_{S,i} = \sum_{i\in F} y_i = y(F).\end{aligned}$$

Thus, if we set b large enough, then $\mathcal{E}$ will happen with positive probability. We now condition on the event $\mathcal{E}$. Let $q := \Pr[\mathcal{E}]$. Also, define $w_{ij} := z_{S,i,j}/z_S$ for all $i \in S$ and $j \in \mathcal{C}$. By the LP constraints, it is to check that w is a valid fractional matching between S and C. We can now do a dependent rounding on the bipartite graph with weight w and obtains a random matching M. Let R be the the clients in $\mathcal{C}$ that are matched in M. Indeed, property (1) holds since we preserve the degrees of all vertices.

Now fix any client $j \in \mathcal{C}$. We have

$$\begin{aligned}\Pr[j \textit{ is matched}] &= \Pr[j \textit{ is matched} \wedge \mathcal{E}] \\ &= \sum_{S\neq\perp} z_S \sum_{i\in S} w_{ij} = \sum_{S\neq\perp} z_S \sum_{i\in S} z_{S,i,j}/z_S \\ &= \sum_{S\neq\perp, i\in S} z_{S,i,j} = \sum_{i\in F}\sum_{S\in\mathcal{S}:i\in S} z_{S,i,j} \\ &= x_{F,j} - \sum_{i\in\perp} z_{\perp,i,j} \ge x_{F,i} - z_\perp = x_{F,i} - 1 + q.\end{aligned}$$

Thus,

$$\begin{aligned}\Pr[j \text{ is not matched}|\mathcal{E}] &= 1 - \Pr[j \text{ is matched}|\mathcal{E}] \\ &= 1 - \frac{\Pr[j \textit{ is matched} \wedge \mathcal{E}]}{\Pr[\mathcal{E}]} \\ &\le 1 - \frac{x_{F,j} - 1 + q}{q} = \frac{1 - x_{F,j}}{q}.\end{aligned}$$

Now we can bound the remaining demand:

$$\begin{aligned}\mathrm{E}[x_{F,\mathcal{C}\setminus R}|\mathcal{E}] &= \sum_{j\in\mathcal{C}} x_{F,j} \Pr[j \text{ is not matched}|\mathcal{E}] \\ &\le \sum_{j\in\mathcal{C}} x_{F,j}\left(\frac{1 - x_{F,j}}{q}\right) = \frac{\pi(J)}{q}.\end{aligned}$$

Observe that the pre-assignment cost is exactly the cost of the matching M. For any $i \in F, j \in \mathcal{C}$, we have

$$\begin{aligned}\Pr[j \text{ is matched to } i] &= \Pr[j \text{ is matched to } i \wedge \mathcal{E}] \\ &= \sum_{S \neq \perp} z_S w_{ij} = \sum_{S \neq \perp} z_{S,i,j} \leq x_{ij}.\end{aligned}$$

Thus, we can bound the total pre-assignment cost:

$$\begin{aligned}\mathrm{E}[\text{cost of } M|\mathcal{E}] &= \sum_{i \in F, j \in \mathcal{C}} d(i,j) \Pr[j \text{ is matched to } i|\mathcal{E}] \\ &= \sum_{i \in F, j \in \mathcal{C}} d(i,j) \Pr[j \text{ is matched to } i]/\Pr[\mathcal{E}] \\ &\leq \frac{1}{q} \sum_{i \in F, j \in \mathcal{C}} d(i,j) x_{ij} = D(F)/q.\end{aligned}$$

By Markov bound, we have $\Pr[x_{F,\mathcal{C}\setminus R}|\mathcal{E}] \geq 3\pi(J)/q] \leq 1/3$ and $\Pr[\text{cost of } M \geq 3D(F)/q|\mathcal{E}] \leq 1/3$. Thus, with probability at least $1/3$, we can avoid these two bad events and properties (2) and (4) hold if we set $\ell' > 3/q$.

Now the trickiest part is to prove property (3). To do this, we may need to redefine the event $\mathcal{E}$ (the above arguments still remain the same.) Suppose the above properties hold, we have

$$x_{B,\mathcal{C}\setminus R} \leq 3\pi(J)/q \leq 3x_{B,\mathcal{C}}/(\ell' q)$$

or, $x_{B,\mathcal{C}\setminus R}/x_{B,\mathcal{C}} \leq 3/(\ell' q)$ .Thus, it suffices to show that, in this case,

$$\frac{3}{\ell' q} y(F) + |S| \leq (1 + 1/\ell) y(F).$$

Note that the factor $(1 + 1/\ell)$ make our task easier and does not affect the analysis of the main algorithm later. Let $Y := (1 + 1/\ell)y(F)$. Since $y(F) \geq \sum_{S \in \mathcal{S}} z_S |S| + b z_\perp$, we have

$$y(F)/\ell = Y - y(F) \leq \sum_{S \in \mathcal{S}} z_S (Y - |S|) + z_\perp (Y - b) = \sum_{S \in \mathcal{S}} z_S (Y - |S|),$$

if we set $b := Y \leq 2\ell(1 + 1/\ell)$. Note that, by the choice of b, $|S| \leq Y$. We now have a lower bound on the weighted sum of $(Y - |S|)$, the maximum possible number of open facilities after opening S. Indeed, we want to lower bound this quantity.

The idea is to divide all possible values of $(Y - |S|)$ in to different ranges, choose a particular range L, and redefine the event $\mathcal{E}$ as "getting S such that $Y - |S|$ lies in this range." Shi Li suggests the following geometric grouping. We assign each set S a rank: if $Y - |S| \in [2^{t-1}, 2^t)$ for some integer $t \geq 1$ then $\mathrm{rank}(S) := t$. Indeed, the ranks are integer numbers in $[0, \delta]$ where $\delta \leq \log(Y) + 1 = O(\log \ell)$ is the maximum rank. Then, by grouping the sets S by rank, we have

$$\begin{aligned}\sum_{t \in [0,\delta]} 2^t \left(\sum_{S \in \mathcal{S}: \mathrm{rank}(S) = t} z_S \right) &= \sum_{S \in \mathcal{S}} z_S 2^{\mathrm{rank}(S)} \\ &\geq \sum_{S \in \mathcal{S}} z_S (Y - |S|) \\ &\geq y(F)/\ell.\end{aligned}$$

So there must exists a rank $t^* \in [0, \delta]$ such that

$$2^{t^*} \left(\sum_{S \in \mathcal{S}: \mathrm{rank}(S) = t^*} z_S \right) \geq \frac{y(F)}{\delta \ell}.$$

Now we define the event $\mathcal{E}$ as "getting S such that $\mathrm{rank}(S) \in [2^{t^*-1}, 2^{t^*}]$". Then, we have

- the probability of $\mathcal{E}$ can be lower-bounded:

$$q = \Pr[\mathcal{E}] = \sum_{S \in \mathcal{S}: \mathrm{rank}(S) = t^*} z_S \geq \frac{y(F)}{\delta \ell 2^{t^*}} \geq \Omega\left(\frac{1}{\delta \ell}\right) = \Omega\left(\frac{1}{\ell \log \ell}\right).$$

- finally,

$$Y - |S| \geq 2^{t^*-1} \geq \frac{y(F)}{\delta \ell q} \geq \frac{3y(F)}{\ell' q},$$

  if we set $\ell' := \Theta(1/q) = \Theta(\ell \log \ell)$ large enough so that $\ell' > 3/q$ and $\ell' \geq \delta \ell$.

□

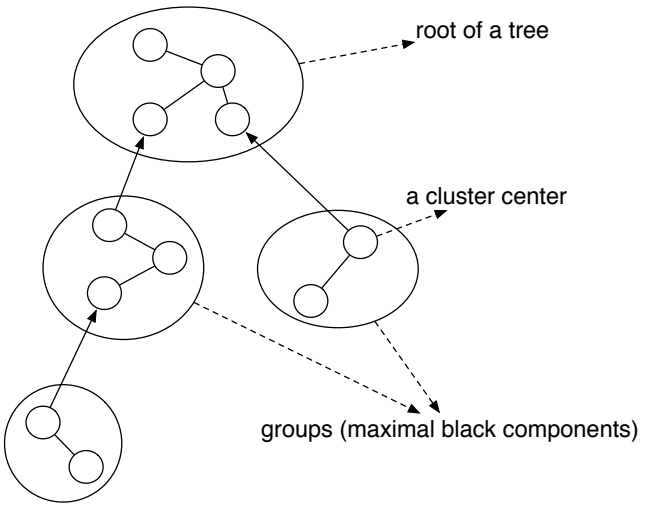
4.4 A new clustering algorithm

This algorithm is quite similar to the Kruskal's algorithm. Recall that we want to partition the set $\mathcal{C}'$. We first sort all edges in $\binom{\mathcal{C}'}{2}$ in increasing order by their lengths. Initially, let $\mathcal{J} := \{\{j\} : j \in \mathcal{C}'\}$ and $\ell > 0$ be an integer to be determined. We repeat the following process:

- Consider the next edge $e = (u, v)$ such that u and v are in different connected components in $\mathcal{J}$. Let J_u and J_v denote the connected components containing u, v, respectively.
- If the opening volumes $y(F(J_u))$ and $y(F(J_v))$ are both at least ℓ, we add a *white* edge between u and v.
- If the opening volumes $y(F(J_u))$ and $y(F(J_v))$ are both less than ℓ, we add a *black* edge between u and v.
- If $y(F(J_u)) < \ell$ and $y(F(J_v)) \geq \ell$, we add a *directed, grey* edge from u to v.
- Then we merge two components J_u and J_v together (i.e., remove J_u, J_v from $\mathcal{J}$ and add $J_u \cup J_v$ into $\mathcal{J}$). We repeat this process until all vertices in $\mathcal{C}'$ are connected when treating grey edges as undirected edges.

Now we remove all the white edges and call a maximal connected component using only black edges a *group*. What we have now is a forest of directed trees with vertices in $\mathcal{C}'$. The following figure illustrates a single directed tree.

By construction, if group J is a root of some tree, then its opening volume $y(F(J))$ will be at least ℓ. Thus, the total mass of any tree should be at least ℓ. This is exactly what we want: we can now transfer

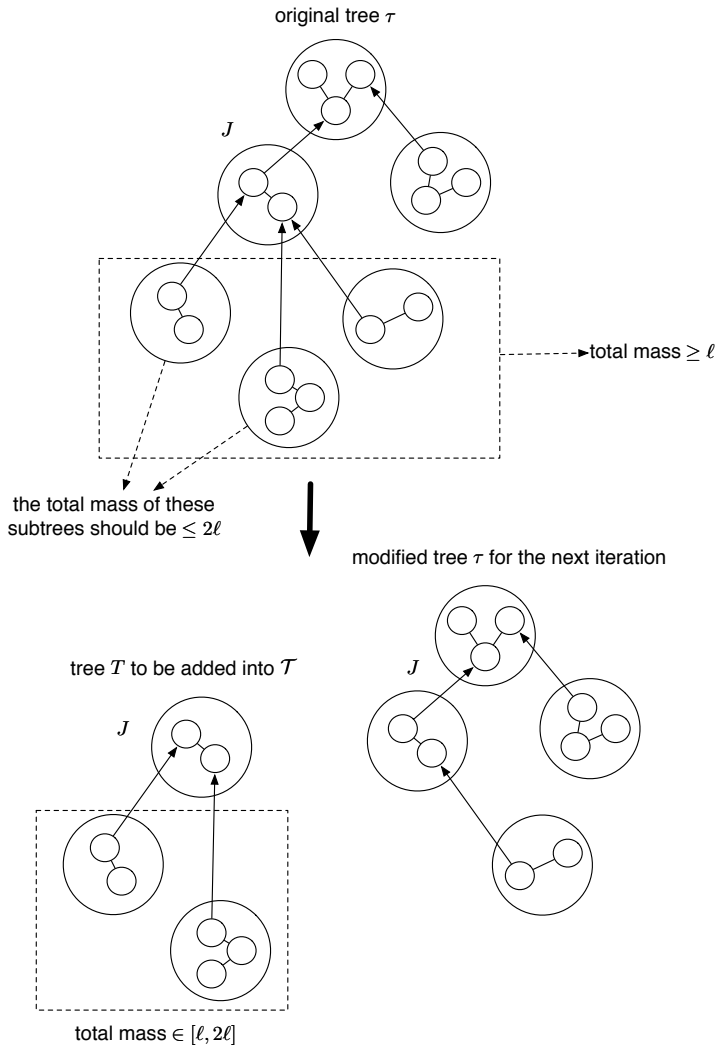


the mass among the groups in each tree and can afford to open some extra constant number of facilities. Unfortunately, the diameter of such a tree could be too big. Therefore, it is necessary to further break the original trees into smaller trees so that we can bound the moving cost when transferring demands among groups in these trees.

The decomposing process works as follows. Let us fix any original tree τ. Set $\mathcal{T}_\tau := \emptyset$. If there is no group J in τ such that the total mass of subtrees rooted at children of J (not including the mass of J) is at least ℓ, then let T be this tree and we simply add it into $\mathcal{T}_\tau$. Now suppose such a group J exists, we pick J as the deepest group with this property. Then the total mass at subtrees rooted at children of J will be at most 2ℓ (recall that the mass of any internal group is at most ℓ). We then use a simple greedy algorithm to choose a subset of children of J such that the total of subtrees rooted at the chosen children is in the range $[\ell, 2\ell]$. Let T be the tree induced by J and these subtrees. We add T into $\mathcal{T}_\tau$, remove all the chosen subtrees from τ, and repeat the process.

Properties and distance bounds: For any tree τ, the decomposing process will break τ into a collection $\mathcal{T}_\tau$ of trees with the following properties. Let us take any tree $T \in \mathcal{T}_\tau$. Abusing the notation, let $F(\tau), F(T)$ denote the set of facilities claimed by some cluster center in τ and T, respectively.

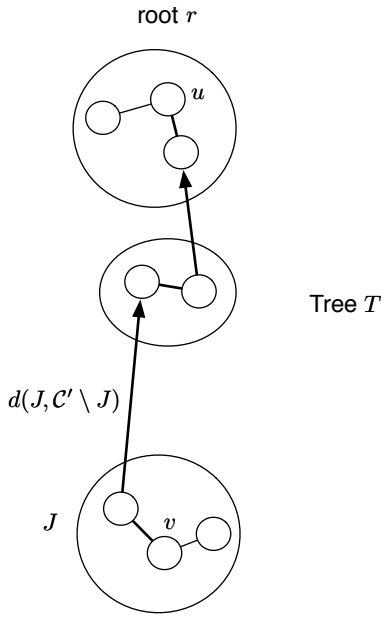
- Note that each group J will appear as an internal of some tree in $\mathcal{T}_\tau$ at most once. Since the total mass of internal groups of each tree (not including the last tree in the above process) is at least ℓ, we have
$$|\mathcal{T}_\tau| \leq y(F(\tau))/\ell + 1.$$
- If T is not the last tree, then the total mass of $F(T)$ is at most $\ell + 2\ell = 3\ell$. Thus, there can be at most $3\ell/(1/2) = 6\ell$ cluster centers in T since the mass of a single cluster is at least $1/2$. If T is the last tree, then either the mass of the root is at most 2ℓ or the root contains only a single big cluster. In either cases, the total mass of the internal vertices is at most ℓ. Thus, there are at most $\max\{(\ell + 2\ell)/(1/2), \ell/(1/2) + 1\} \leq 6\ell$ cluster centers in this tree.
- Now suppose r is the root of T and J is an arbitrary group other than r in T. Let u, v be arbitrary cluster centers in J and r, respectively. We want to bound the distance $d(u, v)$.



- First, observe that the lengths of directed edges on the path from J to r are non-increasing. Suppose J_1, J_2, J_3 are three consecutive groups on this path: there are edges from J_1 to J_2 and from J_2 to J_3. Notice that when the algorithm add a directed edge from J_1 to J_2, J_2 is already a part of some connected component with total mass greater than ℓ. Thus, this edge should be added after the edge from J_2 to J_3 and hence should have greater length.
- Second, for any group J' on the path from J to r, the length of any black edge in this connected component should be at most the length of any grey in-coming or out-coming edges. This is because any undirected, black edge in J' should have been added before any directed, grey edge to/from any vertex in J'.
- Third, the length from J to its parent group is exactly $d(J, \mathcal{C}' \setminus J)$. Consider the time when this edge is first added by the algorithm. At this time, J has been formed and no more black edges will be included to this component. The property follows by greedy choice of the algorithm.
- From these observations, the length of any single edge on the path from u to v is at most

$d(J, \mathcal{C}' \setminus J)$. Since there are at most 6ℓ cluster centers in T, we have

$$d(u, v) \leq 6\ell \cdot d(J, \mathcal{C}' \setminus J).$$



4.5 Rounding algorithm for CKM

In this section, we describe a rounding algorithm that round the LP solution into an integral one. Recall that after the clustering step, we have a collection of edge disjoint trees $\Upsilon = \bigcup_{\tau \in \Lambda} \mathcal{T}_\tau$ where Λ is the forest of original trees without using any white edge. Moreover, the family of non-root clusters of all trees in Υ and the root clusters of original trees in Λ form a partition of $\mathcal{F}$. As discussed before, to move all demands out of a group J, we need to make sure that $x_{F(J),\mathcal{C}}$ is bounded in terms of $\pi(J)$. This can be done via Lemma 1.

Main algorithm:

- Pre-assignment phase: for each group J which is *either* not a root of some original tree in Λ *or* a root with total mass at most 2ℓ, apply the algorithm in Lemma 1 to J (or return an infeasible constraint to the Relaxed Separation Oracle if the related constraints cannot be satsfied.) Let $\tilde{C}$ denote the set of remaining unconnected clients.

- Rounding phase: we first move all demands from clients to facilities and then to cluster centers as in the $(4, 11)$-approximation algorithm. Now we process each tree in Λ separately. For each tree $T \in \Lambda$ with root group r. Let $V(T)$ be the family of groups in T, excluding r. Also, let t denote the number of pre-opened facilities in $F(V)$. Let v be some arbitrary cluster center in the root r. We move all demands from centers in V to v. Then we move the demands from v back to facilities

in $F(V)$ using the following LP as a guide.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in F(V(T))} \alpha_i d(v,i) \\
\text{subject to} \quad & \sum_{i \in F(V(T))} \frac{\alpha_i}{u_i} + t \leq (1+1/\ell) y(F(V(T))) \\
& \alpha(F(V(T))) = x_{F(V(T)),\tilde{C}} \\
& \alpha_i \in [0, u_i] \qquad ,\forall i \in F(V(T))
\end{aligned}$$

Let α^* be any vertex solution of this LP. We move α_i^* units of demand from v to facility i. Next, if r is also a root of some original tree in Λ, there are two cases.

- if r has total mass at most 2ℓ, then r contains at most 4ℓ cluster centers. In this case, we again pick an arbitrary center v, move all the demands to v and then redistribute the demands from v to facilities in $F(r)$ by a similar LP.
- if r has total mass greater than 2ℓ, the group r must contain exactly one cluster center. We shall redistribute the demand from this center to its local facilities as in the simple approximation algorithm.

Finally, we open a facility at i if $\alpha_i^* > 0$.

4.6 Analysis

Bounding the number of open facilities: Fix any tree $\tau \in \Upsilon$ with root r. For a tree T, let $V(T)$ denote the set of non-root groups in T. Recall that the sets $V(T)$ for $T \in \mathcal{T}_\tau$ and r are mutual disjoint and form a partition of $F(\tau)$. Consider the iteration in which we process the tree $T \in \mathcal{T}_\tau$. It is clear that α^* would contain at most two fractional values. Thus, by the LP constraint, we only open

$$\sum_{i \in F(V(T))} \lceil \alpha_i/u_i \rceil + t \leq \sum_{i \in F(V(T))} \alpha_i/u_i + t + 2 \leq (1+1/\ell) y(F(V(T))) + 2$$

facilities in $F(V(T))$. By similar argument, we also open at most $(1+1/\ell)y(F(r))+2$ facilities in $F(r)$. In total, we open at most

$$\begin{aligned}
(1+1/\ell)y(F(\tau)) + 2|\mathcal{T}_\tau| &\leq (1+1/\ell)y(F(\tau)) + 2(y(F(\tau))/\ell + 1) \\
&= (1+3/\ell)y(F(\tau)) + 2 \\
&\leq (1+5/\ell)y(F(\tau))
\end{aligned}$$

facilities for the tree τ. (The final inequality follows because $y(F(\tau)) \geq \ell$.) Summing this bound over all $\tau \in \Gamma$, the total number of open facilities is at most $(1+5/\ell)y(\mathcal{F}) = (1+5/\ell)k$.

Cost analysis: It is clear that moving the demands to cluster centers would cost $5OPT_f$ as in the simple approximation algorithm. Now fix any tree $\tau \in \Lambda$. We will analyze the cost of the rounding phase for trees in $\mathcal{T}_\tau$. Consider the iteration in which some tree $T \in \mathcal{T}_\tau$ is processed. The moving cost for this step is exactly $\sum_{i \in F(V(T))} \alpha_i^* d(v,i)$.

Now observe that $\alpha := \left(\alpha_i = \frac{x_{F(J_{\sigma(i)}),\tilde{\mathcal{C}}}}{x_{F(J_{\sigma(i)}),\mathcal{C}}} x_{i,\mathcal{C}},\ i \in F(V(T)) \right)$ would give a feasible solution:

- since $x_{i,\mathcal{C}} \leq u_i y_i \leq u_i$ and $0 \leq \frac{x_{F(J_{\sigma(i)}),\tilde{\mathcal{C}}}}{x_{F(J_{\sigma(i)}),\mathcal{C}}} \leq 1$, we have $\alpha_i \in [0, u_i]$,
- we have

$$\begin{aligned}\alpha(F(V(T))) &= \sum_{J \in V(T)} \sum_{i \in J} \frac{x_{F(J),\tilde{\mathcal{C}}}}{x_{F(J),\mathcal{C}}} x_{i,\mathcal{C}} \\ &= \sum_{J \in V(T)} x_{F(J),\tilde{\mathcal{C}}} = x_{F(V(T)),\tilde{\mathcal{C}}}.\end{aligned}$$

- finally,

$$\begin{aligned}\sum_{i \in F(V(T))} \frac{\alpha_i}{u_i} + t &= \sum_{J \in V(T)} \sum_{i \in F(J)} \frac{\alpha_i}{u_i} + t \\ &= \sum_{J \in V(T)} \sum_{i \in F(J)} \frac{x_{F(J),\tilde{\mathcal{C}}}}{x_{F(J),\mathcal{C}}} \frac{x_{i,\mathcal{C}}}{u_i} + t \\ &\leq \sum_{J \in V(T)} \left(\sum_{i \in F(J)} \frac{x_{F(J),\tilde{\mathcal{C}}}}{x_{F(J),\mathcal{C}}} y_i + t_J \right) \\ &= \sum_{J \in V(T)} \left(\frac{x_{F(J),\tilde{\mathcal{C}}}}{x_{F(J),\mathcal{C}}} y(F(J)) + t_J \right) \\ &\leq \sum_{J \in V(T)} (1 + 1/\ell) y(F(J)) = (1 + 1/\ell) y(F(V(T))),\end{aligned}$$

  where t_J is the number of pre-opened facilities when preprocessing the group J. The final inequality follows by Lemma 1.

Now, fix any group $J \in V(T)$. We have

$$\begin{aligned}\sum_{i \in F(J)} \alpha_i d(v, i) &= \sum_{i \in F(J)} \frac{x_{F(J),\tilde{\mathcal{C}}}}{x_{F(J),\mathcal{C}}} x_{i,\mathcal{C}} d(v, i) \\ &\leq \sum_{i \in F(J)} \frac{x_{F(J),\tilde{\mathcal{C}}}}{x_{F(J),\mathcal{C}}} x_{i,\mathcal{C}} (d(v, \sigma(i)) + d(\sigma(i), i)) \\ &\leq \sum_{i \in F(J)} x_{i,\mathcal{C}} d(\sigma(i), i) + \sum_{i \in F(J)} \frac{x_{F(J),\tilde{\mathcal{C}}}}{x_{F(J),\mathcal{C}}} x_{i,\mathcal{C}} (6\ell) d(J, \mathcal{C} \setminus J) \\ &= \sum_{i \in F(J)} x_{i,\mathcal{C}} d(\sigma(i), i) + (6\ell) x_{F(J),\tilde{\mathcal{C}}} d(J, \mathcal{C} \setminus J).\end{aligned}$$

Using a similar argument in the simple approximation algorithm, we can bound the first term:

$$\sum_{i \in F(J)} x_{i,\mathcal{C}} d(\sigma(i), i) \leq 5(D(F(J)) + D'(F(J))).$$

For the second term, note that $x_{F(J),\tilde{\mathcal{C}}} \leq \ell'\pi(J)$ after the preprocessing step. Therefore, by the distance bound in the previous section, we have

$$\begin{aligned}(6\ell)x_{F(J),\tilde{\mathcal{C}}}d(J,\mathcal{C}\setminus J) &\leq (6\ell\ell')\pi(J)d(J,\mathcal{C}\setminus J)\\ &= O(\ell\ell')(D(F(J))+D'(F(J))).\end{aligned}$$

Now the moving cost for T can be easily bounded:

$$\begin{aligned}\sum_{i\in F(V(T))}\alpha_i^* d(v,i) &\leq \sum_{i\in F(V(T))}\alpha_i d(v,i)\\ &= \sum_{i\in F(V(T))}\alpha_i d(v,i)\\ &= \sum_{J\in V(T)}\sum_{i\in F(J)}\alpha_i d(v,i)\\ &\leq O(\ell\ell')(D(F(V(T)))+D'(F(V(T)))).\end{aligned}$$

Next, for the root group r of τ, we can use a similar argument to show that the cost of redistributing demands in this group is at most $O(\ell\ell')(D(F(r))+D'(F(r)))$. Thus, by taking the sum over all tree $T\in\tau$ and this bound, the moving cost when processing τ is at most

$$O(\ell\ell')(D(F(\tau))+D'(F(\tau))).$$

Then, summing this bound over all $\tau\in\Gamma$, the total connection cost is at most

$$O(\ell\ell')(D(\mathcal{F})+D'(\mathcal{F})) = O(\ell\ell')OPT_f.$$

By setting $\ell = 5/\epsilon$, the algorithm opens $(1+\epsilon)k$ facilities and the approximation ratio is $O\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}\right)$.

References

[1] J. Byrka, K. Fleszar, B. Rybicki, and J. Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 722–736. SIAM, 2015.

[2] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. An improved approximation for *k*-median, and positive correlation in budgeted optimization. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 737–756, 2015.

[3] R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2000, pages 106–115. Society for Industrial and Applied Mathematics, 2000.

[4] S. Li. Approximating capacitated k-median with $(1+\epsilon)k$ open facilities. *ArXiv e-prints*, Nov. 2014.

[5] S. Li. On uniform capacitated k-median beyond the natural lp relaxation. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 696–707. SIAM, 2015.