

Simpler Algorithms for Capacitated k -Median with $(1 + \epsilon)$ -Violations Using New Inequalities

Han Dai *

School of Computer Science,
Nanjing University,
Nanjing, Jiangsu Province, China.
handai@smail.nju.edu.cn

Shi Li *

School of Computer Science,
Nanjing University,
Nanjing, Jiangsu Province, China.
shili@nju.edu.cn

Abstract

We revisit the capacitated k -median problem, focusing on constant-factor approximation algorithms with either $(1 + \epsilon)$ -cardinality or capacity violations. Previous results [17, 28] achieve an $O_\epsilon(1)$ -approximation with $(1 + \epsilon)$ -violations via complex extended configuration linear programming formulations with the round-or-cut framework.

We propose a simpler, unified round-or-cut framework that handles both cardinality and capacity violation settings. Our main technical contribution is a mechanism that directly utilizes the costs of local instances to generate valid inequalities. Specifically, we partition the metric space into local pieces of size $O(1/\epsilon)$ as in previous work, and construct instances of capacitated clustering with penalties, which allow client demands to be unserved at a penalty cost. Solving these local instances yields lower bounds on the original LP cost, yielding valid LP constraints. In contrast, previous approaches rely on complex extended configuration LPs over connection variables x_{ij} 's and opening variables y_i 's.

As a secondary contribution, our approach achieves an improved running time of $\text{poly}(n, 1/\epsilon)$, bypassing the exponential dependence on $1/\epsilon$ required to solve the previous extended configuration LPs. Instead of solving the local instances optimally in time exponential in $1/\epsilon$, we obtain $O(1/\epsilon)$ -approximate solutions in $\text{poly}(n, 1/\epsilon)$ time.

1 Introduction

In the capacitated k -median (CKM) problem, we are given a facility set F , where every facility i is associated with a capacity $u_i \in \mathbb{Z}_{\geq 0}$, a client set C , an integer $k \in \mathbb{Z}_{\geq 0}$, and a metric d over $F \cup C$. The goal is to select a subset $S \subseteq F$ of at most k open facilities and a connection assignment $\sigma : C \rightarrow S$ such that $|\sigma^{-1}(i)| \leq u_i$ for every facility $i \in S$, and so as to minimize the connection cost $\sum_{j \in C} d(j, \sigma(j))$.

When $u_i = \infty$ for all $i \in F$, the CKM problem reduces to the classic (uncapacitated) k -median problem. Following the first constant-factor approximation algorithm by Charikar, Guha, Tardos and Shmoys [13], a long line of work has progressively improved the approximation ratio [21, 22, 30, 7, 8, 20, 11, 4, 16, 25, 26], with the current state-of-the-art standing at $2 + \epsilon$ [15, 6]. On the

*The work of HD and SL is supported by State Key Laboratory for Novel Software Technology, and New Cornerstone Science Foundation.

negative side, it is known to be NP-hard to approximate the problem within a factor better than $1 + 2/e$ [23, 24].

In contrast, obtaining a true constant-factor approximation for the capacitated k -median (CKM) problem remains a notoriously hard problem in approximation algorithms. The current best approximation ratio is $O(\log k)$ [2], achieved via the metric tree-embedding framework of Fakcharoenphol, Rao, and Talwar [18, 19]. Consequently, most prior work focuses on bicriteria approximation algorithms, which allow constant-factor violations on either the number of open facilities (the cardinality constraint) or the facility capacity constraints.

For the uniform capacitated k -median problem—where all facilities have the same capacity—Charikar, Guha, Tardos and Shmoys [12] gave a 16-approximation with a capacity violation factor of 3. Later, Li [27] presented an $O(1/\epsilon)$ -approximation with a $(2 + \epsilon)$ -capacity violation. For the general capacity setting, Chuzhoy and Rabani [14] obtained a 40-approximation with a capacity violation factor of 50. Byrka, Fleszar, Rybicki, and Spoerhase [5] subsequently achieved a constant-factor approximation with a $(3 + \epsilon)$ -capacity violation. On the cardinality-violation front, Aardal, Berg, Gijswijt and Li [1] designed a $(7 + \epsilon)$ -approximation algorithm that opens at most $2k$ facilities (a factor-2 cardinality violation).

Most bicriteria approximations for CKM rely on linear programming (LP) relaxations. However, the natural LP relaxation has an unbounded integrality gap, even if one permits up to a $(2 - \epsilon)$ -violation on either the cardinality or capacity constraint. Thus, to achieve an $O(1)$ -approximation via standard LP formulations, a violation factor of at least 2 is necessary—a barrier that holds even for uniform capacities.

Li [29] was the first to break this factor-2 barrier, proposing an $\exp(O(1/\epsilon^2))$ -approximation algorithm for uniform CKM with a $(1 + \epsilon)$ -cardinality violation. To overcome the integrality gap, Li introduced an exponential family of *rectangle inequalities*. Because separating these inequalities efficiently is non-trivial, Li employed a round-or-cut framework: given a fractional solution that only satisfies the basic constraints, the algorithm either rounds it to an integer solution with cost at most α times the value of the LP solution (round), or identifies a violated rectangle inequality (cut). Coupled with the Ellipsoid method, this yields a polynomial-time α -approximation. Li [28] subsequently extended this framework to general capacities, achieving an $O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$ -approximation under a $(1 + \epsilon)$ -cardinality violation—providing an exponential improvement in the approximation ratio’s dependence on ϵ . Since rectangle constraints fail in the general capacity setting, the approach in [28] instead constructed exponentially many extended configuration LP relaxations with auxiliary variables.

Similarly, in the capacity-violation setting, Byrka, Rybicki, and Uniyal [9] developed an $O(1/\epsilon^2)$ -approximation algorithm for uniform CKM with a $(1 + \epsilon)$ -capacity violation. Demirci and Li [17] generalized this result to arbitrary capacities, achieving an $O(1/\epsilon^5)$ -approximation with a $(1 + \epsilon)$ -capacity violation using extended configuration LPs in [28]. Together, these results establish optimal $(1 + \epsilon)$ -violation factors in both settings, leaving the existence of a true $O(1)$ -approximation as the primary remaining open problem.

Regarding the running time, the uniform-capacity algorithms of [29] and [9] run in $\text{poly}(n, 1/\epsilon)$ time, whereas the general-capacity algorithms of [28] and [17] require $n^{\text{poly}(1/\epsilon)}$ time. Furthermore, there is a distinction between the *soft-capacitated* setting (where multiple copies of a facility may be opened, each counting toward k) and the *hard-capacitated* setting (where each facility can be opened at most once). Notably, the state-of-the-art results for the cardinality violation setting [28] apply only to the soft-capacitated variant.

1.1 Our Results

We present a simplified and unified bicriteria approximation framework for the capacitated k -median problem with general capacities, addressing the $(1 + \epsilon)$ -cardinality and $(1 + \epsilon)$ -capacity violation settings. Throughout, n denotes the total number of facilities and clients, and opt denotes the optimum cost of the CKM instance, without any violation.

Theorem 1.1. *For any constant $\epsilon > 0$, there exists a $\text{poly}(n, 1/\epsilon)$ -time algorithm that, given a capacitated k -median instance, outputs a solution opening at most $(1 + \epsilon)k$ facilities while satisfying all capacity constraints, with cost at most $O(1/\epsilon^5) \cdot \text{opt}$.*

Our second main result addresses the capacity-violation setting.

Theorem 1.2. *For any constant $\epsilon > 0$, there exists a $\text{poly}(n, 1/\epsilon)$ -time algorithm that, given a capacitated k -median instance, outputs a solution opening at most k facilities while connecting at most $\lceil (1 + \epsilon)u_i \rceil$ clients to each open facility i , with cost at most $O(1/\epsilon^6) \cdot \text{opt}$.*

1.2 Our Techniques

We now present the key ideas behind our simplified framework. To illustrate our core intuition, consider the integrality gap of the basic LP relaxation (LP (1)) in the cardinality-violation setting. Suppose we have two facilities, i and i' , and $u + 1$ clients co-located at the same point, where $u_i = u_{i'} = u$ for some large integer u . The LP can fractionally open $1 + 1/u$ facilities from $\{i, i'\}$ to serve all $u + 1 = (1 + 1/u)u$ clients at cost 0. By taking disjoint copies of this gadget, the LP can open k fractional facilities in total at cost 0. However, any integer solution must open $2k/(1 + 1/u)$ facilities to achieve cost 0, requiring a cardinality violation factor of $2/(1 + 1/u)$, which approaches 2 for large u .

More generally, a CKM instance can consist of far-apart “islands”, where the LP solution opens z fractional facilities per island for some small integer $z \ll 1/\epsilon$. Under a $(1 + \epsilon)$ -cardinality budget, we cannot afford to open $z + 1$ facilities on each island. Thus, one must guarantee that each island’s subproblem admits a small-cost integral solution with at most z open facilities. While *rectangle inequalities* suffice for uniform capacities [29], the general capacity setting becomes significantly more intricate: Li [28] introduced auxiliary variables to encode every valid configuration of open facilities on each island (up to size $\Theta(1/\epsilon)$), leading to a heavy configuration LP. The capacity violation setting exhibits a similar complexity behavior.

We take a simpler, direct approach. Rather than constructing complex extended LPs, we solve the local instance on each island directly to establish a valid lower bound on its client connection cost, and then enforce this bound via a valid linear constraint. To handle the more general case where a small fraction of demands from clients may be satisfied outside their local island, we formalize each local subproblem as a *capacitated clustering with penalty* problem, where unserved client demand incurs a penalty proportional to the distance from the island to outside. Furthermore, by approximating these local penalty instances within an $O(1/\epsilon)$ factor instead of solving them optimally, we reduce the running time from $n^{O(1/\epsilon)}$ to $\text{poly}(n, 1/\epsilon)$.

Our island construction follows the spanning tree framework of prior work [29, 28, 9, 17]. Given a fractional solution to the basic LP, a filtering step selects a set of representative clients, and all demands are moved to representatives at a small cost. We then run Kruskal’s algorithm on these representatives to build a Minimum Spanning Tree (MST) and color its edges in black, gray, and white. The resulting *black components* (maximal connected subtrees of black edges) form our

islands. For each black component, we define a family of capacitated clustering instances with penalty, parameterized by the number t of open facilities. We solve these instances approximately and use the resulting lower bounds to generate valid LP constraints. As in previous work, the construction of black components depends on the LP solution. Thus these valid constraints cannot be fixed a priori, necessitating the round-or-cut framework.

Finally, our rounding procedures for both the cardinality- and capacity-violation settings follow a similar framework. We first construct a local fractional solution for each black component by combining the LP solution with our local penalty solutions. Next, we partition the black components into *groups*, each containing $\Theta(1/\epsilon)$ fractional open facilities, ensuring that demand within each group can be satisfied internally with a $(1 + \epsilon)$ -violation. We then formulate an auxiliary LP over these local solutions. The structure of this auxiliary LP guarantees that in an extreme solution, each group contains $O(1)$ strictly fractional facility variables. A $(1 + \epsilon)$ -violation provides just enough slack to round these residual fractional facilities to integers at low cost. The specific constructions of local solutions and group partition schemes vary between the two violation settings and are motivated by previous work.

Organization In Section 2, we give a common algorithm for cardinality-violation and capacity-violation settings. In Section 3, we give the $(1/\epsilon^5)$ -approximation for the capacitated k -median problem with $(1 + \epsilon)k$ open facilities, proving Theorem 1.1. In Section 4, we give the $(1/\epsilon^6)$ -approximation for the capacitated k -median problem with at most k facilities while connecting at most $\lceil (1 + \epsilon)u_i \rceil$ clients to each open facility, proving Theorem 1.2. All missing proofs can be found in Appendix A.

2 Common algorithm for cardinality-violation and capacity-violation settings

In this section, we describe the common procedure in our rounding algorithms for both the cardinality and capacity-violation settings. The following is the basic LP relaxation for both settings:

$$\min \sum_{j \in C} \phi_j \tag{1}$$

$$\sum_{i \in F} x_{i,j} = 1, \quad \forall j \in C \tag{2} \quad \sum_{j \in C} x_{i,j} \leq u_i y_i, \quad \forall i \in F \tag{5}$$

$$\sum_{i \in F} y_i \leq k, \tag{3} \quad \phi_j \geq \sum_{i \in F} d(i, j) x_{i,j}, \quad \forall j \in C \tag{6}$$

$$x_{i,j} \leq y_i, \quad \forall i \in F, j \in C \tag{4} \quad x_{i,j}, y_i \in [0, 1], \quad \forall i \in F, j \in C \tag{7}$$

In the corresponding integer program, $x_{ij} \in \{0, 1\}$ for every facility $i \in F$ and client $j \in C$ indicates whether j is connected to i , $y_i \in \{0, 1\}$ for $i \in F$ indicates whether i is open, and ϕ_j for $j \in C$ denotes the connection cost of the client $j \in C$. The objective (1) is the sum of connection costs over all clients. (2) requires that every j is connected to exactly 1 facility, (3) requires that the number of open facilities is at most k , and (4) says that j can be connected to i only if i is open. (5) is the capacity requirement, and (6) is by the definition of ϕ_j . In the LP relaxation program, we only require x_{ij} 's and y_i 's to be in $[0, 1]$, as stated in (7).

We shall use the “round-or-cut” paradigm [10, 3]. Given a fractional solution (x^*, y^*, ϕ^*) to LP (1), our rounding algorithm either succeeds by returning an integral solution whose cost is at most $O_\epsilon(1) \cdot \sum_{j \in C} \phi_j^*$ (which may violate the cardinality constraint or the capacity constraints by a factor of $(1 + \epsilon)$), or returns a valid constraint of the following form, that is violated by the solution (x^*, y^*, ϕ^*) :

$$s \sum_{i \in F'} y_i + \sum_{j \in C} w_j \phi_j \geq \Phi, \quad (8)$$

where $s \in \mathbb{R}_{\geq 0}$, $F' \subseteq F$, $w_j \in [0, 1]$ for every $j \in C$, and $\Phi \in \mathbb{R}_{\geq 0}$. We can think of that each facility $i \in F'$ has a cost s , and a client $j \in C$ has weight $w_j \in [0, 1]$. The inequality says that the sum of facility cost and the weighted connection cost of clients is at least Φ . When our rounding algorithm returns a constraint of this form, we guarantee its validity. That is, it is satisfied by all integral solutions to the CKM instance. It is well-known that such an algorithm is sufficient to obtain an $O_\epsilon(1)$ -approximation, with $(1 + \epsilon)$ -cardinality or capacity violation.

Furthermore, when the facility opening variables y_i 's are integral ($y_i \in \{0, 1\}$ for every $i \in F$), the optimum client-to-facility assignment reduces to a minimum-cost maximum flow (or matching) problem. As allowing x_{ij} 's to take fractional values does not decrease the cost, our rounding algorithm allows the connections between clients and facilities to be fractional. We assume every $j \in C$ has 1 unit of demand. Our goal is to choose a set of at most k open facilities, and send the demands to the open facilities, so that every open facility i receives at most u_i units of demand. When a $(1 + \epsilon)$ -capacity violation is allowed, an open facility i can take up to $(1 + \epsilon)u_i$ units of demand fractionally. But in an integral solution, we need to increase its capacity to $\lceil (1 + \epsilon)u_i \rceil$.

Fix a solution (x^*, y^*, ϕ^*) to LP (1). For any set $F' \subseteq F$ of facilities and any set $C' \subseteq C$ of clients, we shall let $x^*(F', C') := \sum_{i \in F', j \in C'} x_{i,j}^*$; we simply write $x^*(i, C')$ for $x^*({i}, C')$ and $x^*(F', j)$ for $x^*(F', {j})$. We define $y^*(F') = \sum_{i \in F'} y_i^*$ for any $F' \subseteq F$. This notational convention applies to other variants of y as well.

Throughout the paper, we let $\ell = \Theta(1/\epsilon)$ be an integer with a sufficiently large hidden constant in $\Theta(\cdot)$.

2.1 Representatives R and Initial Moving of Demands

We use the same framework as in [29, 28, 17, 9] to choose a set R of representatives and move all demands to R . We define the average connection cost of each client $j \in C$ as $d_{\text{av}}(j) := \sum_{i \in F} x_{i,j}^* d(i, j)$. The value of the solution (x^*, y^*, ϕ^*) is $\phi^* \geq \sum_{i \in F, j \in C} x_{i,j}^* d(i, j) = \sum_{j \in C} d_{\text{av}}(j)$.

We select representatives using the following greedy procedure: initialize $R = \emptyset$. Choose a client $v \in C$ with the smallest d_{av} and add it to R ; then remove all clients $j \in C$ satisfying $d(j, v) \leq 4d_{\text{av}}(j)$ from C . Note that v itself is also removed. We perform a Voronoi partition on facility set F centered at representative set R : each facility $i \in F$ is assigned to its nearest representative $v \in R$, yielding a partition of F into clusters $\{U_v : v \in R\}$, where $U_v = \{i \in F : v = \arg \min_{v' \in R} d(i, v')\}$. For any subset $V \subseteq R$ of representatives, we use $U(V) := \bigcup_{v \in V} U_v$ to denote the union of Voronoi regions with centers V .

Lemma 2.1. *The following statements hold:*

(2.1a) *for all $v, v' \in R$ and $v \neq v'$, we have $d(v, v') > 4 \max\{d_{\text{av}}(v), d_{\text{av}}(v')\}$;*

(2.1b) for all $j \in C$, there exists $v \in R$, such that $d_{\text{av}}(v) \leq d_{\text{av}}(j)$ and $d(v, j) \leq 4d_{\text{av}}(j)$;

(2.1c) $y^*(U_v) \geq 1/2$ for every $v \in R$;

(2.1d) for any $v \in R, i \in U_v$ and $j \in C$, we have $d(i, v) \leq d(i, j) + 4d_{\text{av}}(j)$.

We move all demands to client representatives. First, for each client $j \in C$ and $i \in F$, we move $x_{i,j}$ units of demand from j to i . Obviously, the moving cost is exactly $\sum_{i \in F, j \in C} x_{i,j}^* d(i, j)$. After this step, each $i \in F$ has $\sum_{j \in C} x_{i,j}^* = x^*(i, C) \leq u_i y_i$ units of demand. Second, for each $v \in R$ and $i \in U_v$, we move the $x^*(i, C)$ units of demand at i to v . Then, all demands are moved to R and a representative $v \in R$ has $\sum_{i \in U_v} x^*(i, C) = x^*(U_v, C)$ units of demand.

Let $g_{j,v}$ be the amount of demand moved from a $j \in C$ to $v \in R$. That is, we define $g_{j,v} = x^*(U_v, j)$. Clearly, for every $j \in C$, we have $\sum_{v \in R} g_{j,v} = x^*(F, j) = 1$.

Let $D_i := \sum_{j \in C} x_{i,j}^* (d(i, j) + d_{\text{av}}(j))$ for each $i \in F$, and $D(F') := \sum_{i \in F'} D_i$ for every $F' \subseteq F$. Notice that $D(F) = \sum_{i \in F} D_i = \sum_{i \in F, j \in C} x_{i,j}^* d(i, j) + \sum_{j \in C} d_{\text{av}}(j) \leq 2 \sum_{j \in C} \phi_j^*$. We can think of D_i as the budget given to i .

Lemma 2.2. For every $v \in R$, we have $\sum_{i \in U_v} x^*(i, C) d(i, v) \leq O(1) \cdot D(U_v)$.

Proof. By Property (2.1d), we have $d(i, v) \leq d(i, j) + 4d_{\text{av}}(j)$ for every $i \in U_v$ and $j \in C$. Thus, $\sum_{i \in U_v} x^*(i, C) d(i, v) \leq \sum_{i \in U_v, j \in C} x_{i,j}^* (d(i, j) + 4d_{\text{av}}(j)) \leq 4 \sum_{i \in U_v} D_i \leq O(1) D(U_v)$. $\square$

This immediately implies the following lemma, which shows that the initial moving of demands from clients C to representatives R is small.

Lemma 2.3. For every $v \in R$, we have $\sum_{j \in C} g_{j,v} d(j, v) \leq O(1) \cdot D(U_v)$.

Proof. For each $i \in U_v$, by the triangle inequality, $d(j, v) \leq d(j, i) + d(i, v)$. So

$$\begin{aligned} \sum_{j \in C} g_{j,v} d(v, j) &= \sum_{j \in C, i \in U_v} x_{i,j}^* d(v, j) \leq \sum_{j \in C, i \in U_v} x_{i,j}^* (d(j, i) + d(i, v)) \\ &\leq O(1) \cdot D(U_v). \end{aligned}$$

The last inequality follows from Lemma 2.2. $\square$

2.2 Construction of Black Components and Gray Forest Using Kruskal's Algorithm

We now describe the framework of [28, 17] in which we run Kruskal's Algorithm to obtain a minimum spanning tree (MST) for R , and color the edges of the MST using black, gray and white colors. We can assume $k \geq \ell$. Otherwise, the $O(\log k)$ -approximation algorithm for capacitated k -median problem by [2] gives a true approximation ratio of $O(\log k) = O(\log 1/\epsilon)$.

For any $v \in R$, define its weight as $y^*(U_v) = \sum_{i \in U_v} y_i^*$ satisfying $y^*(U_v) \geq \frac{1}{2}$. For any subset $J \subseteq R$, define its weight as $y^*(U(J)) = \sum_{v \in J} y^*(U_v)$. If $y^*(U(J)) \geq \ell$, then J is said to be heavy; otherwise, it is said to be light.

Recall that in Kruskal's algorithm for MST, we maintain the edge set E_{MST} and a partition $\mathcal{P}$ of R . Initially, set $E_{\text{MST}} = \emptyset$ and $\mathcal{P} = \{\{v\} : v \in R\}$. The algorithm processes edges $e \in \binom{R}{2}$ sorted in non-decreasing order of length. If v and v' do not belong to the same connected component in $\mathcal{P}$, we add the edge $e = (v, v')$ to E_{MST} and merge the connected components containing v and v' .

We color the edges in the MST as follows. For any edge $e = (v, v') \in E_{\text{MST}}$, let J_v and $J_{v'}$ be the sets in $\mathcal{P}$ containing v and v' respectively before we add e to MST. $e = (v, v')$ is colored according to the following rule:

- If both J_v and $J_{v'}$ are light, then color e in black, treated as an undirected edge.
- If one of J_v and $J_{v'}$ is light and the other is heavy, then color e in gray, treated as a directed edge from the light set to the heavy set.
- If both J_v and $J_{v'}$ are heavy, then color e in white, treated as an undirected edge.

Consider the forest formed by the black edges of the MST. The vertex set of any tree in the forest is said to be a *black component*, or simply *component*. Let $\mathcal{J}$ be the set of all black components; so, $\mathcal{J}$ forms a partition of R . For every $J \in \mathcal{J}$, we define $L_J := d(J, R \setminus J)$.

Claim 2.4. *For each $J \in \mathcal{J}$ and $v, v' \in J$, each black edge $e = (v, v')$ in J satisfies $d(v, v') \leq L_J$.*

We take the MST, remove all white edges, and contract all black edges. The resulting graph is a forest, which we call the *gray forest* and denote by Υ . We call each tree in the forest a gray tree. We keep the direction of the gray edges, as defined earlier. So, the gray forest is over the set of black components, and each gray tree is a rooted tree with gray edges directed towards the root.

Lemma 2.5. *For each tree τ in the gray forest Υ , the following properties hold:*

- (2.5a) *the root of τ is a heavy black component, and all other vertices of τ are light black components.*
- (2.5b) *let J, J' be two black components in τ such that J' is the parent of J . Then $L_J = d(J, J')$ is the length of the gray edge between J and J' .*
- (2.5c) *for any leaf-to-root path of τ , the L -values of nodes are non-increasing.*

2.3 Capacitated Clustering with Penalty Problem

We now introduce the key contribution discussed in Section 1.2: the use of capacitated clustering with penalty instances to generate valid inequalities on the original capacitated k -median instance.

Given a black component $J \in \mathcal{J}$, and an integer parameter $t \geq 0$, the capacitated clustering with penalty instance (or simply, penalty instance) $\mathcal{I}_J(t)$ is defined as follows. We have demands at J , where each $v \in J$ has $x^*(U_v, C)$ units of demand. The facilities are $U(J) = \bigcup_{v \in J} U_v$, and each facility $i \in U(J)$ has capacity u_i . We are allowed to open at most t facilities.

Crucially, for the instance $\mathcal{I}_J(t)$, we are allowed to not connect all the demands: for every unit-demand that is not connected, we pay a penalty cost of $L_J/2$. Our goal is to open at most t facilities S in $U(J)$, and connect some demands from J to S , so that every $i \in S$ receives at most u_i units of demand. Connecting demands at $v \in J$ to $i \in U(J)$ incurs a per-unit cost of $d(v, i)$. For all the demands that are not connected to S , we pay a per-unit penalty cost of $L_J/2$. The goal of the problem is to minimize the sum of connection and penalty costs.

$O(|J|)$ -Approximate Solution $S_J(t)$ for $\mathcal{I}_J(t)$ As $|J|$ is small, we can afford to use an $O(|J|)$ -approximation for the problem.¹ We first convert the metric d to a tree metric: we take the black edges over J , which gives the minimum spanning tree over J . Then, for every $v \in J$ and $i \in U_v$, we add an edge (v, i) of length $d(v, i)$ to the tree. We let d' be this tree metric over $J \cup U(J)$.

Lemma 2.6. *For every $v \in J$ and $i \in U(J)$, we have $d(v, i) \leq d'(v, i) \leq (2|J| - 1)d(v, i)$.*

Proof. The first inequality is obvious. For the second inequality, $d'(v, v') \leq (|J| - 1)d(v, v')$ holds for every $v, v' \in J$, since there are at most $|J| - 1$ edges in J . For every $v' \in J, i \in U_{v'}$, v' is the closest representative to i . So $d'(i, v) = d(i, v') + d'(v', v) \leq d(i, v) + (|J| - 1)d(v', v) \leq (2|J| - 1)d(i, v)$, where the second inequality follows from the triangle inequality. $\square$

By losing a factor of $(2|J| - 1)$ on the connection cost, we can consider the tree metric d' . Furthermore, we can make the following two assumptions on the solution for the instance $\mathcal{I}_J(t)$:

- First, if some $i \in U(J)$ has $d(i, J) > L_J/2$ (notice that $d(i, J) = d(i, v)$ for the $v \in J$ with $i \in U_v$), then $i \notin S_J(t)$. This can be assumed since such i will have distance more than $L_J/2$ to all demands and thus it would be better to pay the penalty for the demands than to connect them to i .
- Second, either all the capacities of the open facilities are used, or all the demands are connected (that is, there is no penalty). This can be assumed as for every $i \in U(J)$ with $d(i, J) \leq L_J/2$, we have $d(i, v) \leq (|J| - 1)L_J + L_J/2 \leq (2|J| - 1) \cdot L_J/2$. So, instead of paying a penalty of $L_J/2$, we pay a connection cost of at most $(2|J| - 1) \cdot L_J/2$. Notice that the factor of $(2|J| - 1)$ is only lost only on the penalty side, and thus this does not multiply with the $(2|J| - 1)$ -factor we lost for converting d to d' .

As all capacities u_i for $i \in U(J)$ are integers in $[n]$, the second assumption allows us to solve the modified instance precisely using dynamic programming. We root the tree for d' over $J \cup U(J)$ at any root. In a DP state, we consider a sub-tree rooted at some node, the number of facilities we open in the tree, and the total capacity of the open facilities. The value of the state is the cost incurred, under the condition we either use all the capacities, or satisfy all the demands in the sub-tree.

Let $S_J(t)$ be the solution obtained using the dynamic programming algorithm, $o_J(t)$ be the amount of demand that is not connected in the solution (that is, we pay a penalty for $o_J(t)$ units of demand), and $\Psi_J(t)$ be the cost of the solution. As we argued, $S_J(t)$ is a $(2|J| - 1)$ -approximate solution for the instance $\mathcal{I}_J(t)$. So $\Psi_J(t)$ is at most $(2|J| - 1)$ times the optimum cost of $\mathcal{I}_J(t)$.

2.4 Generating Valid Constraint Using $S_J(t)$ for $\mathcal{I}_J(t)$

For a given black component $J \in \mathcal{J}$ and parameter $t \geq 0$, we can use the solution $S_J(t)$ for $\mathcal{I}_J(t)$ to generate a valid constraint of form (8).

¹Alternatively, we could solve the problem exactly with running time exponential in $|J|$. However, it is more important to achieve $\text{poly}(n, 1/\epsilon)$ running time, and we lose approximation factors of $\text{poly}(1/\epsilon)$ anyway. So we use the $O(|J|)$ -approximation in $n^{O(1)}$ -time.

Lemma 2.7. *For any $J \in \mathcal{J}$ and integer $t \geq 0$, the following is a valid constraint for the original CKM instance:*

$$\frac{\Psi_J(t)}{t+1}y(U(J)) + (2|J| - 1) \sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j)) \geq \Psi_J(t). \quad (9)$$

Proof. Consider an integral solution (x, y, ϕ) to LP(1). If $y(U(J)) \geq t + 1$, then the inequality clearly holds as the first term is already at least $\Psi_J(t)$.

So we assume $y(U(J)) \leq t$. Assume towards the contradiction that $\frac{\Psi_J(t)}{t+1}y(U(J)) + (2|J| - 1) \sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j)) < \Psi_J(t)$. In particular, $(2|J| - 1) \sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j)) < \Psi_J(t)$. For every $j \in C$, let i_j be the open facility connected to j in the integral solution. So $x_{i_j, j} = 1$.

We then construct a solution to the capacitated clustering with penalty instance $\mathcal{I}_J(t)$. We open the set of facilities $i \in U(J)$ with $y_i = 1$. So, we open at most t facilities. For every $v \in J$, and every $j \in C$ with $g_{j,v} > 0$, we consider how to connect the $g_{j,v}$ units of demand moved from j to v at the initial moving phase:

- If $i_j \in U(J)$, we connect the demand to i_j . In this case, the connection cost for the demand is $g_{j,v}d(v, i_j) \leq g_{j,v}(d(v, j) + d(j, i_j)) \leq g_{j,v}(\phi_j + d(v, j))$. The first inequality holds by the triangle inequality, and the second by (6).
- If $i_j \notin U(J)$, we treat the demand as unconnected, and pay a penalty of $g_{j,v}L_J/2$. Notice that $d(v, i_j) \geq L_J/2 = d(J, R \setminus J)/2$ since otherwise i_j would be in $U(J)$. So, the penalty we pay is at most $g_{j,v}d(v, i_j) \leq g_{j,v}(\phi_j + d(v, j))$; again, we used the triangle inequality and (6).

So, the cost of the solution is at most $\sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j)) < \frac{\Psi_J(t)}{(2|J|-1)}$, which contradicts the fact that $S_J(t)$ is a $(2|J| - 1)$ -approximate solution to $\mathcal{I}_J(t)$. $\square$

Notice that the inequality in Lemma 2.7 can be written as an inequality of form (8). We can check if our (x^*, y^*, ϕ^*) satisfies the constraint for every $J \in \mathcal{J}$ and integer $t \in [0, |U(J)|]$. If not, we return a constraint that is violated. So from now on, we assume (x^*, y^*, ϕ^*) satisfies all these constraints. As we shall see later, for our rounding algorithm to succeed, we only need a small subset of these constraints to hold, but we can afford to check all of them.

So far, we have finished the description of the algorithm that is common to both the cardinality violation and capacity violation settings. We describe the two algorithms separately in the next two sections. As we mentioned, the algorithms follow a similar framework: we partition the black components into groups, construct local solutions for black components, and use an auxiliary LP to obtain a fractional solution with only a few fractional facility variables, and round these variables to integral with a $(1 + \epsilon)$ -violation.

3 Algorithm for $(1 + \epsilon)$ -Cardinality-Violation Setting

In this section, we describe the algorithm for the cardinality-violation setting.

3.1 Partition Gray Forest into Groups of Weight $\Theta(\ell)$

We partition each gray tree τ in the gray forest into groups of weight $\Theta(\ell)$, using a bottom-up greedy approach similar to that in [28]. (The only exception is that the root of τ may be a singleton group,

with weight much larger than $\Theta(\ell)$.) Focus on such a tree τ . In each iteration, select the deepest node (black component) J in τ such that the total weight of all descendants of J is at least ℓ ; recall that the weight of a black component J' is defined as $y^*(U(J'))$. As the root of τ has weight at least ℓ , such J exists.

By Property (2.5a), every non-root node in τ has weight less than ℓ . By the choice of J , the total weight of the subtree rooted at any child of J is less than ℓ . If the total weight of all these subtrees is at most ℓ , then we take G to be the set of black components in the whole sub-tree rooted at J and let $\text{root}_G = J$. Otherwise, a greedy procedure can select a subset $\mathcal{J}'$ of children of J such that the total weight of subtrees rooted at $\mathcal{J}'$ is between ℓ and 2ℓ . In this case, we let G contain the black components in these subtrees, and define root_G to be child $J' \in \mathcal{J}'$ with the smallest $L_{J'}$. Notice that in this case G is not connected in τ , as we did not include J in G .

In both cases, we remove G from the tree τ and repeat until τ becomes empty. We define $\mathcal{G}$ as the set of all groups we constructed, over all trees τ in the gray forest. Clearly, $\mathcal{G}$ is a partition of $\mathcal{J}$. The total weight of black components in any G is at least ℓ .

Lemma 3.1. *For every $G \in \mathcal{G}$, every $J \in G$ and $v \in J$, the distance between v and any representative in root_G is at most $O(\ell) \cdot L_J$.*

3.2 Construction of Local Solutions for Black Components $J \in \mathcal{J}$

For every black component $J \in \mathcal{J}$, we define $t_J := \lfloor (1 + 1/\ell)y^*(U(J)) \rfloor$. We shall only use the capacitated clustering with penalty instance $\mathcal{I}_J(t_J)$ for each J . For convenience, we use $\mathcal{I}_J, S_J, o_J$ and Ψ_J for $\mathcal{I}_J(t_J), S_J(t_J), o_J(t_J)$ and $\Psi_J(t_J)$ respectively.

Fix a black component $J \in \mathcal{J}$, and let S_J be the $(2|J| - 1)$ -approximate solution to the instance $\mathcal{I}_J$ with cost Ψ_J . In this solution, we construct a local solution $\bar{y} \in [0, 1]^{U(J)}$ for this J , along with a per-unit cost vector $b \in \mathbb{R}_{\geq 0}^{U(J)}$.

For every $i \in U(J)$, we define $u'_i := \frac{x^*(i, C)}{y_i^*} \leq u_i$ to be the “effective capacity” of i , which is the capacity used by the LP solution (x^*, y^*, ϕ^*) . J is one of the following two types:

- (1) if $o_J \leq \frac{1}{\ell} \cdot x^*(U(J) \setminus S_J, C)$, which is equivalent to $x^*(U(J) \setminus S_J, C) \geq \ell o_J$. In this case, we let $\bar{y}_i = 1$ for every $i \in S_J$, and $\bar{y}_i = \frac{o_J}{x^*(U(J) \setminus S_J, C)} \cdot y_i^* \leq \frac{1}{\ell} \cdot y_i^*$ for every $i \in U(J) \setminus S_J$. That is, $\bar{y}$ is the indicator vector for S_J , plus $\frac{o_J}{x^*(U(J) \setminus S_J, C)} \leq \frac{1}{\ell}$ times y^* restricted to $U(J) \setminus S_J$. We let $S'_J = S_J$ in this case.
- (2) if $o_J > \frac{1}{\ell} \cdot x^*(U(J) \setminus S_J, C)$. We let $S'_J = \{i \in S_J : y_i^* \geq 1 - \frac{1}{\ell}\} \subseteq S_J$. Then, we define $\bar{y}_i = 1$ for every $i \in S'_J$ and $\bar{y}_i = y_i^*$ for every $i \in U(J) \setminus S'_J$. So, $\bar{y}$ is y^* with values for S'_J scaled up to 1.

In both cases we have $S'_J = \{i \in S_J : \bar{y}_i = 1\}$. Then for both cases, we define $b_i = 0$ for every $i \in S'_J$, and $b_i = d(i, J) + L_J$ for every $i \in U(J) \setminus S'_J$.

Lemma 3.2. *The following inequalities hold:*

$$(3.2a) \quad \bar{y}(U(J)) \leq (1 + \frac{2}{\ell})y^*(U(J)).$$

$$(3.2b) \quad \sum_{i \in U(J)} u'_i b_i \bar{y}_i \leq O(1) \cdot D(U(J)) + O(\ell^2) \cdot \Psi_J.$$

Proof. We first prove (3.2a). We consider the two cases (1) and (2) above. For type-(1) component J , we have $\bar{y}(U(J)) \leq |S_J| + \frac{1}{\ell} y^*(U(J) \setminus S_J) \leq (1 + \frac{1}{\ell}) y^*(U(J)) + \frac{1}{\ell} y^*(U(J)) = (1 + \frac{2}{\ell}) y^*(U(J))$. For type-(2) component J , $\bar{y}(U(J)) = |S'_J| + y^*(U(J) \setminus S'_J) \leq (1 + \frac{1}{\ell-1}) y^*(S'_J) + y^*(U(J) \setminus S'_J) = (1 + \frac{1}{\ell-1}) y^*(U(J))$. The first inequality holds by $y_i^* \geq 1 - \frac{1}{\ell}$ for each $i \in S'_J$.

It remains to consider (3.2b). For type-(1) component J , we have

$$\begin{aligned} \sum_{i \in U(J)} u'_i b_i \bar{y}_i &= \frac{o_J}{x^*(U(J) \setminus S_J, C)} \cdot \sum_{i \in U(J) \setminus S_J} u'_i y_i^* (d(i, J) + L_J) \\ &\leq \frac{1}{\ell} \cdot \sum_{i \in U(J) \setminus S_J} u'_i y_i^* d(i, J) + o_J L_J \\ &= \frac{1}{\ell} \sum_{i \in U(J) \setminus S_J} x_{i,C}^* d(i, J) + o_J L_J \leq O(\frac{1}{\ell}) \cdot D(U(J)) + 2 \cdot \Psi_J. \end{aligned}$$

Recall that $u'_i y_i^* = x_{i,C}^*$ for every $i \in U(J) \setminus S_J$. The last inequality follows from Lemma 2.2 and $\Psi_J \geq o_J \cdot \frac{L_J}{2}$.

For type-(2) component J , we have $\sum_{i \in U(J)} u'_i b_i \bar{y}_i = \sum_{i \in U(J) \setminus S'_J} u'_i y_i^* (d(i, J) + L_J) \leq O(1) \cdot D(U(J)) + x^*(U(J) \setminus S'_J, C) \cdot L_J$. We shall bound $x^*(U(J) \setminus S'_J, C)$. As J is of type-(2), we have $o_J > 0$, and $x^*(U(J) \setminus S_J, C) < \ell o_J$. By our assumption on the solution S_J , all capacities in S_J are fully used in the solution S_J for $\mathcal{I}_J$. Therefore, we have

$$\begin{aligned} x^*(U(J) \setminus S_J, C) &\geq o_J + \sum_{i \in S_J \setminus S'_J} (1 - y_i^*) u_i \geq \sum_{i \in S_J \setminus S'_J} (1 - y_i^*) u'_i \geq \frac{1}{\ell - 1} \sum_{i \in S_J \setminus S'_J} y_i^* u'_i \\ &= \frac{1}{\ell - 1} \cdot x^*(S_J \setminus S'_J, C). \end{aligned}$$

So, we have $x^*(S_J \setminus S'_J, C) \leq (\ell - 1) x^*(U(J) \setminus S_J, C) \leq \ell(\ell - 1) o_J$. Overall, we have $x^*(U(J) \setminus S'_J, C) \leq \ell^2 o_J$. So, we have $\sum_{i \in U(J)} u'_i b_i \bar{y}_i \leq O(1) D(U(J)) + \ell^2 o_J L_J \leq O(1) D(U(J)) + O(\ell^2) \cdot \Psi_J$. This finishes the proof of (3.2b) and thus the whole lemma. $\square$

3.3 Rounding Procedure for Each Group $G \in \mathcal{G}$

Our rounding algorithm handles each group $G \in \mathcal{G}$ independently. We focus on a fixed $G \in \mathcal{G}$. Recall that every black component $J \in G$, we defined two vectors $\bar{y} \in [0, 1]^{U(J)}$ and $b \in \mathbb{R}_{\geq 0}^{U(J)}$. We then let $\bar{y}$ and b be the concatenation of all black components $J \in G$. So, the domain of both $\bar{y}$ and b is $U(G) := \bigcup_{J \in G} U(J)$.

Consider the following LP with variables $\{\tilde{y}_i\}_{i \in U(G)}$:

$$\begin{aligned} \min \quad & \sum_{i \in U(G)} u'_i b_i \tilde{y}_i \tag{10} \\ \tilde{y}(U(G)) &\leq \bar{y}(U(G)); \quad \sum_{J \in G} \sum_{i \in U(J) \setminus S'_J} u'_i \tilde{y}_i \geq \sum_{J \in G} \sum_{i \in U(J) \setminus S'_J} u'_i \bar{y}_i; \\ \tilde{y}_i &\in [0, 1], \forall i \in U(G); \quad \tilde{y}_i = 1, \text{ if } \bar{y}_i = 1. \end{aligned}$$

By setting $\tilde{y}_i = \bar{y}_i$, we obtain a solution to LP(10) of value $\sum_{i \in U(G)} \tilde{y}_i u'_i b_i \leq \sum_{i \in U(G)} \bar{y}_i u'_i b_i \leq \sum_{J \in G, i \in U(J)} (x(i, C^*) d(i, J) + 4\ell^2 \cdot \Psi_J)$. Fix on such an optimum vertex-point solution $\tilde{y}$ of LP(10).

Since there are only two non-box-constraints, $\tilde{y}$ has at most two fractional values. We then open all facilities $i \in U(G)$ with $\tilde{y}_i > 0$.

We now reassign demands to these finally opened facilities. We define the root representative of G to be any representative in the root black component of G . The whole process consists of the following two steps.

- **Sending step.** For each black component $J \in G$, we consider the solution S_J for the capacitated clustering with penalty instance $\mathcal{I}_J$. Recall that $S'_J = S_J$ if J is of type-(1) and $S'_J \subseteq S_J$ if J is of type-(2). Then we pre-connect demands at J to S'_J according to the solution S_J . For the demands that are not connected, we send them to the designated root representative of G . Notice that every $i \in S'_J$ has $\tilde{y}_i = \bar{y}_i = 1$, and its full capacity $u_i \geq u'_i \geq x^*(i, C)$ is used in the local solution, unless $\alpha_J = 0$. So, for a $J \in G$, the amount of demands sent from J to the root is at most $x^*(U(J) \setminus S'_J, C)$.
- **Receiving step.** For the demands sent to the root representative in the previous step, we send them to the open facilities in $\bigcup_{J \in G} U(J)$. For every $J \in G$, only facilities in $U(J) \setminus S'_J$ are allowed to receive demand, and such a facility i receives at most $u'_i \tilde{y}_i$ units of demand from the root representative. This is the only requirement for the receiving step.

So, the total amount of demands sent to the root is at most

$$\sum_{J \in G} \sum_{i \in U(J) \setminus S'_J} u'_i \tilde{y}_i \leq \sum_{J \in G} \sum_{i \in U(J) \setminus S'_J} u'_i \tilde{y}_i.$$

Therefore, we have enough capacity to receive the demands sent to the root representative.

Lemma 3.3. *The connection cost incurred by the sending and receiving steps is at most $O(\ell) \cdot D(U(G)) + O(\ell^3) \cdot \sum_{J \in G} \Psi_J$.*

Proof. Consider the sending step for each $J \in G$. The cost of local matching of demands according to the solution S_J is at most Ψ_J . The cost of sending the remaining demands up to the root representative is at most $O(\ell) \cdot L_J \cdot x^*(U_J \setminus S'_J, C)$. The cost of $U(J) \setminus S'_J$ receiving demands from the root representative is at most

$$\sum_{i \in U(J) \setminus S'_J} \tilde{y}_i u'_i (O(\ell) \cdot L_J + d(i, J)) \leq O(\ell) \cdot \sum_{i \in U(J) \setminus S'_J} \tilde{y}_i u'_i b_i.$$

The cost of receiving the demands is at most the above upper bound, plus $\sum_{i \in U_J} \tilde{y}_i u'_i d(i, J)$.

Now we consider all black components $J \in G$. The cost of receiving the demands is at most

$$\begin{aligned} O(\ell) \cdot \sum_{J \in G} \sum_{i \in U(J) \setminus S'_J} \tilde{y}_i u'_i b_i &\leq O(\ell) \cdot \sum_{J \in G} \sum_{i \in U(J) \setminus S'_J} \tilde{y}_i u'_i b_i \\ &\leq O(\ell) \cdot D(U(G)) + O(\ell^3) \cdot \sum_{J \in G} \Psi_J. \end{aligned}$$

Clearly, the upper bound dominates the cost incurred by local matching and sending demands up to the root representative. $\square$

Then, we wrap up the analysis of cost by considering all groups G , as well as the cost incurred by initial moving. The total cost is at most

$$O(\ell)D(F) + O(\ell^3) \cdot \sum_J \Psi_J \leq O(\ell^5) \sum_{j \in C} \phi_j^*.$$

To see this, we just need to bound $\sum_J \Psi_J$. The inequality in Lemma 2.7 holds for (x^*, y^*, ϕ^*) , and every black component J . We add up the inequality over all J . We obtain

$$\begin{aligned} \sum_J \Psi_J &\leq O(\ell^2) \sum_J \sum_{v \in J, j \in C} g_{j,v}(\phi_j^* + d(v, j)) = O(\ell^2) \cdot \left(\sum_{j \in C} \phi_j^* + (\text{cost of initial moving}) \right) \\ &\leq O(\ell^2) \cdot \sum_{j \in C} \phi_j^*. \end{aligned}$$

The equality holds because $\sum_v g_{j,v} = 1$ for every $j \in C$. Therefore, $\sum_J \Psi_J \leq O(\ell^2) \cdot \sum_{j \in C} \phi_j^*$.

The following lemma gives an upper bound on the number of open facilities:

Lemma 3.4. *The total number of open facilities is at most $(1 + \frac{4}{\ell}) \cdot k$.*

Proof. Focus on a fixed $G \in \mathcal{G}$, since the vector $\tilde{y}$ has at most two fractional entries, so $\lfloor \sum_{i \in U(G)} \tilde{y}_i \rfloor + 2 \leq \sum_{i \in U(G)} \tilde{y}_i + 2 \leq (1 + \frac{2}{\ell})y^*(U(G)) + 2$. Since every group has weight at least ℓ , and the groups form a partition of all black components, we have $\ell|\mathcal{G}| \leq \sum_{G \in \mathcal{G}} y_i^*(U(G)) = \sum_{i \in F} y_i^* \leq k$. By summing over all $\tau \in \Upsilon$, $|\mathcal{G}| = \sum_{\tau} |\mathcal{G}_{\tau}| \leq \frac{1}{\ell} \sum_{v \in R} y^*(U_v) = \frac{1}{\ell} \sum_{i \in F} y_i^* \leq \frac{k}{\ell}$. Then summing over all $G \in \mathcal{G}$, the total number of open facilities is at most $(1 + \frac{2}{\ell}) \sum_{G \in \mathcal{G}} y^*(U(G)) + 2|\mathcal{G}| \leq (1 + \frac{2}{\ell})k + \frac{2k}{\ell} = (1 + \frac{4}{\ell})k$. $\square$

With a large enough hidden constant in $\ell = \Theta(1/\epsilon)$, we finish the proof of Theorem 1.1.

4 Algorithm for the Capacity-Violation Setting

This section presents the rounding algorithm for the capacity-violation setting. In contrast to the cardinality-violation setting, we enforce the total number of opened facilities to be at most k , and each open facility is allowed to carry demand up to $(1 + \epsilon)u_i$.

4.1 Converting Gray Forest Υ into A Forest Υ^* of Binary Trees

The way we partition each gray tree into groups is different from the one we used for the cardinality-violation setting. In the capacity violation setting, we need to use the tree structure over the groups, and it is important to make sure that the L_J values of black components in a parent group is at most the L_J values of black components in a child group. Therefore, we use a top-to-bottom approach to construct the groups. Moreover, to bound the number of child groups of each group, we first make each gray tree binary using the left-child-right-sibling method.

We use the same framework as in [17] to convert Υ into a binary forest. For every rooted tree $\tau \in \Upsilon$, we convert it to a tree τ^* with the same set of nodes (black components) as follows. For every non-leaf black component $J \in \tau$, we sort its children in non-decreasing order of L -values, and let the order be $J_1, J_2, \dots, J_a$. Then, we add edges $(J_1, J), (J_2, J_1), (J_3, J_2), \dots, (J_a, J_{a-1})$ to τ^* ,

with the first node of each edge being the child, and the second node being the parent. So, τ^* is a rooted tree with edges directed towards the root and every node in τ^* has at most 2 children.

We call the tree τ^* a *transformed gray tree*. Let Υ^* be the set of all transformed gray trees τ^* , over all gray trees $\tau \in \Upsilon$, and we call Υ^* the *transformed gray forest*.

Lemma 4.1. Υ^* satisfies the following properties:

- (4.1a) for any leaf-to-root path of $\tau^* \in \Upsilon^*$, the L -values of nodes are non-increasing.
- (4.1b) every root component $J \in \mathcal{J}$ of Υ^* has either $y^*(U(J)) \leq 2\ell$ or $|J| = 1$.
- (4.1c) for any non-root component J and its parent J' in Υ^* , we have $\max_{v \in J, v' \in J'} d(v, v') \leq O(\ell) \cdot L_{J'}$.

4.2 Partition $\mathcal{J}$ into Groups using Υ^*

With the transformed gray forest Υ^* , we can now partition the black components $\mathcal{J}$ into groups. We also use the framework of [17]. Let $\mathcal{G} = \emptyset$ initially. For every transformed gray tree τ^* in Υ^* and its root J , we call `construct-groups( $\tau^*, J$ )`, defined in Algorithm 1.

Algorithm 1 `construct-groups( $\tau^*, J$ )`

- 1: $G = \{J\}$
 - 2: **while** $\sum_{J \in G} y^*(U(J)) < \ell$ and G does not contain all descendants of J in τ^* **do**
 - 3: choose an edge (J'', J') in τ^* with the smallest $L_{J''}$ satisfying $J' \in G, J'' \notin G$ (recall that edges are directed towards the root, so J' is the parent of J'')
 - 4: $G \leftarrow G \cup \{J''\}$
 - 5: $\mathcal{G} \leftarrow \mathcal{G} \cup \{G\}$
 - 6: **for** every edge (J'', J') in τ^* such that $J' \in G, J'' \notin G$ **do** $\triangleright J'$ is the parent of J''
 - 7: `construct-groups( $\tau^*, J''$ )`
-

Notice that every group G induces a (weakly) connected component in Υ^* . Thus, every G has a unique root black component. The forest structure naturally defines a parent-child relationship among groups: a group G is the parent of a group G' if the root of G' is a child of some black component in G . Also, the root of any transformed gray tree forms a singleton group, as its weight is at least ℓ . We call this a root group.

We extend the definition of $U(\cdot)$ to sets of black components, and in particular groups. For any set $\mathcal{J}' \subseteq \mathcal{J}$ of components, we define $U(\mathcal{J}') := \bigcup_{J \in \mathcal{J}'} U(J) = \bigcup_{J \in \mathcal{J}', v \in J} U_v$. Let $\text{children}(G)$ be the set of all child-groups of a group $G \in \mathcal{G}$. Let $\text{children}'(G) = \text{children}(G) \cup \{G\}$ if G is a root group, and $\text{children}'(G) = \text{children}(G)$ otherwise.

Lemma 4.2. $\mathcal{G}$ satisfies the following properties:

- (4.2a) if $G \in \mathcal{G}$ is not a root group, then $y^*(U(G)) < 2\ell$;
- (4.2b) if $G \in \mathcal{G}$ is a non-leaf group, then $y^*(U(G)) \geq \ell$;
- (4.2c) any group G has at most $O(\ell)$ child groups;
- (4.2d) let G be a group, $G' \in \text{children}'(G)$, and $J' \in G'$, then $\max_{v' \in J', v \in J \in G} d(v, v') \leq O(\ell^2)L_{J'}$.

4.3 Types of Black Components and Construction of Local Solutions

For every black component $J \in \mathcal{J}$, we define $t_J = \lfloor (1 + \frac{1}{c\ell^2}) y^*(U(J)) \rfloor$, for a sufficiently large absolute constant $c > 0$. If J is a root black component of some gray tree (equivalent, transformed gray tree), and $t_J > \lceil y^*(U(J)) \rceil$, then we update $t_J = \lceil y^*(U(J)) \rceil$. Notice that for a non-root black component J , we have $t_J = \lfloor (1 + \frac{1}{c\ell^2}) y^*(U(J)) \rfloor \leq \lceil y^*(U(J)) \rceil$ as $y^*(U(J)) < \ell$.

Fix a black component $J \in \mathcal{J}$. We consider the solution $S_J(t_J)$ for the penalty instance $\mathcal{I}_J(t_J)$. When $t_J = \lceil y^*(U(J)) \rceil > y^*(U(J))$, we may need to also consider the penalty instance $\mathcal{I}_J(t_J - 1)$. Recall that $o_J(t_J)$ denotes the amount of unconnected demands in the solution $S_J(t_J)$, and $\Psi_J(t_J)$ is the overall cost of the solution $S_J(t_J)$. We assume $\Psi_J(t_J - 1) \geq \Psi_J(t_J)$; otherwise, we can use the solution $S_J(t_J - 1)$ for the instance $\mathcal{I}_J(t_J)$.

A black component $J \in \mathcal{J}$ is one of four types. If J is a root component with $|J| = 1$, then it is of type-(1). Otherwise, its type is:

- (1) if $o_J(t_J) > \frac{1}{\ell} \cdot x^*(U(J), C)$,
- (2) if $o_J(t_J) \leq \frac{1}{\ell} \cdot x^*(U(J), C)$ and $t_J \leq y^*(U(J))$,
- (3) if $o_J(t_J) \leq \frac{1}{\ell} \cdot x^*(U(J), C)$, $t_J > y^*(U(J))$ and $o_J(t_J - 1) > \frac{1}{\ell} \cdot x^*(U(J), C)$, and
- (4) if $o_J(t_J) \leq \frac{1}{\ell} \cdot x^*(U(J), C)$, $t_J > y^*(U(J))$ and $o_J(t_J - 1) \leq \frac{1}{\ell} \cdot x^*(U(J), C)$.

As before, we shall construct local solutions for the 4 types of components respectively. If J is of type-(1), we simply use the fractional solution $y^*|_{U(J)}$. If J is of type-(2), we use the solution $S_J(t_J)$ for the penalty instance $\mathcal{I}_J(t_J)$, with unconnected demands sent to $S_J(t_J)$, using *proportional distribution*: the amount of demand sent to any $i \in S_J(t_J)$ is proportional to the amount of demand it already serves. In this case, we say we *follow* the instance $\mathcal{I}_J(t_J)$ for J . If J is of type-(4), we follow either $\mathcal{I}_J(t_J - 1)$ or $\mathcal{I}_J(t_J)$, depending on how many facilities will be open in $U(J)$ in the end. If J is of type-(3), we follow the instance $\mathcal{I}_J(t_J)$ initially. If in the end we can only open $t_J - 1$ facilities in $U(J)$, we perform a remove operation to decrease the number of open facilities by 1.

4.4 Construction of Final Solution Using Auxiliary Linear Program

With the ideas of local solutions, we form an auxiliary linear program to construct the final integral solution for our CKM instance. Define $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \mathcal{J}^{(3)}$ and $\mathcal{J}^{(4)}$ as the sets of type-(1), (2), (3) and (4) black components respectively. We shall use $\mathcal{J}^{(3,4)}$ to denote $\mathcal{J}^{(3)} \cup \mathcal{J}^{(4)}$; the notations for other subsets of $\{1, 2, 3, 4\}$ are defined analogously. This section mainly focuses on the algorithm, and we defer the analysis of the cost and capacity violation to the next section.

For every $J \in \mathcal{J}^{(1)}$, and $i \in U(J)$, we have a variable $\tilde{y}_i \in [0, 1]$. If $\tilde{y}_i \in \{0, 1\}$, it indicates if i is open or not in the final solution. For every $J \in \mathcal{J}^{(3,4)}$, we have a single variable $\tilde{y}_J \in [t_J - 1, t_J] \ni y^*(U(J))$. If $\tilde{y}_J \in \{t_J - 1, t_J\}$, then it denotes the number of facilities we open in $U(J)$. For black components of $J \in \mathcal{J}^{(2)}$, as we always follow $S_J(t_J)$. So we do not have variables for them.

As in the previous section, we use $u'_i := \frac{x^*(i, C)}{y_i^*} \leq u_i$ as the “effective capacity” of a facility $i \in F$. The auxiliary LP is defined as follows:

$$\min \quad \sum_{J \in \mathcal{J}^{(1)}} \sum_{i \in U(J)} u'_i (d(i, J) + O(\ell^2) L_J) \tilde{y}_i + O(\ell^3) \cdot \sum_{J \in \mathcal{J}^{(3,4)}} \Psi_J(t_J - 1) (t_J - \tilde{y}_J) \quad (11)$$

$$\tilde{y}(U(\mathcal{J}^{(1)} \cap G')) - \lfloor y^*(U(\mathcal{J}^{(1)} \cap G')) \rfloor \geq 0, \quad \forall G' \in \mathcal{G} \quad (12)$$

$$\sum_{G' \in \text{children}'(G), J \in \mathcal{J}^{(3,4)} \cap G'} (t_J - \tilde{y}_J) \leq 1, \quad \forall \text{ non-leaf } G \in \mathcal{G} \quad (13)$$

$$\sum_{J \in \mathcal{J}^{(1)}, i \in U(J)} \tilde{y}_i + \sum_{J \in \mathcal{J}^{(2)}} y^*(U(J)) + \sum_{J \in \mathcal{J}^{(3,4)}} \tilde{y}_J \leq k, \quad (14)$$

$$\sum_{G' \in \text{children}'(G), i \in U(\mathcal{J}^{(1)} \cap G')} (\tilde{y}_i - y_i^*) u_i' \geq 0, \quad \forall \text{ non-leaf } G \in \mathcal{G} \quad (15)$$

$$\tilde{y}_i \in [0, 1], \quad \forall J \in \mathcal{J}^{(1)}, i \in U(J) \quad (16)$$

$$\tilde{y}_J \in [t_J - 1, t_J], \quad \forall J \in \mathcal{J}^{(3,4)} \quad (17)$$

In the objective, the functions in $O(\ell^2)$ and $O(\ell^3)$ notations will be specified later.

Lemma 4.3. *Setting $\tilde{y}_i = y_i^*$ for every $i \in U(J), J \in \mathcal{J}^{(1)}$ and $\tilde{y}_J = y^*(U(J))$ for every $J \in \mathcal{J}^{(3)} \cup \mathcal{J}^{(4)}$ gives a valid solution to LP (11).*

Proof. (12), (15), (16) and (17) hold trivially.

Now consider (13). If J is of type-(3) or (4), then $t_J = \lfloor (1 + \frac{1}{c\ell^2})y^*(U(J)) \rfloor = \lceil y^*(U(J)) \rceil > y^*(U(J))$. The equality holds as $y^*(U(J)) < 2\ell$ and thus $\frac{1}{c\ell^2} \cdot y^*(U(J)) < 1$. Therefore, $t_J \leq (1 + \frac{1}{c\ell^2})y^*(U(J))$. Therefore

$$0 < t_J - y^*(U(J)) \leq \frac{1}{c\ell^2} \cdot y^*(U(J)) < 1.$$

Summing up the inequality over all $J \in \mathcal{J}^{(3,4)} \cap G'$ for a fixed $G' \in \text{children}(G)$, we have

$$\sum_{J \in \mathcal{J}^{(3,4)} \cap G'} (t_J - y^*(U(J))) \leq \frac{1}{c\ell^2} \cdot \sum_{J \in \mathcal{J}^{(3,4)} \cap G'} y^*(U(J)) \leq \frac{2}{c\ell}.$$

As G has at most $O(\ell)$ children, we have

$$\sum_{G' \in \text{children}'(G), J \in \mathcal{J}^{(3,4)} \cap G'} (t_J - y^*(U(J))) \leq 1,$$

when c is large enough. Therefore, (13) holds.

For every $J \in \mathcal{J}^{(2)}$, we have $t_J = \lfloor y^*(U(J)) \rfloor \leq y^*(U(J))$ facilities. Therefore, setting $\tilde{y}$ as in the lemma, we have $\sum_{J \in \mathcal{J}^{(1)}} \tilde{y}(U(J)) + \sum_{J \in \mathcal{J}^{(2)}} t_J + \sum_{J \in \mathcal{J}^{(3,4)}} \tilde{y}_J \leq \sum_{J \in \mathcal{J}} y^*(U(J)) = y^*(F) \leq k$. Thus, (14) holds. $\square$

We solve the LP to obtain a vertex solution $\tilde{y}$. From now on, we fix this $\tilde{y}$.

Lemma 4.4. *The following statements hold:*

1. *there are at most three facilities $i \in \bigcup_{G' \in \text{children}'(G)} U(\mathcal{J}^{(1)} \cap G')$ with $\tilde{y}_i \in (0, 1)$, for every non-leaf group $G \in \mathcal{G}$.*
2. *at most one black component $J \in \mathcal{J}^{(3,4)}$ has $\tilde{y}_J \in (t_J - 1, t_J)$.*

Proof. Consider the first statement. We fix a non-leaf group $G \in \mathcal{G}$. We say i is a strictly fractional facility if $\tilde{y}_i \in (0, 1)$ (and $\tilde{y}_i$ exists). We say $G' \in \text{children}'(G)$ contains a strictly fractional facility i , if additionally $i \in \mathcal{J}^{(1)} \cap U(G')$.

Suppose some group $G' \in \text{children}'(G)$ contains at least 3 strictly fractional facilities $i_1, i_2, i_3 \in U(\mathcal{J}^{(1)} \cap G')$. Notice that there are at most three tight constraints involving one of the three variables $\tilde{y}_{i_1}, \tilde{y}_{i_2}, \tilde{y}_{i_3}$: (12) for G' , (14) and (15) for G . The coefficient matrix between the three constraints and three variables is $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ u'_{i_1} & u'_{i_2} & u'_{i_3} \end{pmatrix}$. The rank of the matrix is less than 3, contradicting that $\tilde{y}$ is a vertex solution.

Now suppose there are two different groups in $\text{children}'(G)$, each of which contains at least 2 strictly fractional facilities. There are at least 4 of these facilities, but the rank of the coefficient matrix defined as above is at most 3. Again a contradiction.

Now suppose there are three groups in $\text{children}'(G)$, each of which contains exactly 1 strictly fractional facility. Notice that (12) is not tight for G', G'' and G''' . There are at most 2 tight constraints involving the one of the fractional variables. Again, $\tilde{y}$ would not be a vertex solution.

Finally, if there are three groups in $\text{children}'(G)$, two of which contain exactly 1 strictly fractional facility, and one of which contains exactly 2 strictly fractional facilities, using the above method one can argue that the rank of the coefficient matrix is at most 3. Therefore, we conclude that there are at most 3 strictly fractional facilities over all groups in $\text{children}'(G)$.

Consider the second statement for a fixed G . Similarly, suppose there are at least 2 such components J_1 and J_2 involved in the same constraint from (13). Then (13) and (14) might be tight for G . The coefficient matrix is $\begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix}$, whose rank is 1. This contradicts that $\tilde{y}$ is a vertex solution. $\square$

We shall maintain a set W of open facilities, a flow from R to W . Every $v \in R$ sends $x^*(U_v, C) = \sum_{i \in U_v} u'_i y_i^*$ units of demand (this is the demand received by v during the initial moving procedure in Section 2.1). At the beginning, W may have size more than k . We shall use the “remove” procedure to remove facilities from W to guarantee the upper bound.

We start with $W = \emptyset$, and add facilities as follows:

- For every $J \in \mathcal{J}^{(1)}$, we add $\{i \in U(J) : \tilde{y}_i > 0\}$ to W . For every non-leaf group $G \in \mathcal{G}$, we send the demands at $\bigcup_{G' \in \text{children}'(G), J \in \mathcal{J}^{(1)} \cap G'} J$ to $W \cap \bigcup_{G' \in \text{children}'(G)} U(\mathcal{J}^{(1)} \cap G')$, where every i receives $u'_i \tilde{y}_i$ units of demand. The total demand and the total capacity are respectively

$$\sum_{G' \in \text{children}'(G)} \sum_{J \in \mathcal{J}^{(1)} \cap G'} \sum_{i \in U(J)} u'_i y_i^* \quad \text{and} \quad \sum_{G' \in \text{children}'(G)} \sum_{J \in \mathcal{J}^{(1)} \cap G'} \sum_{i \in U(J)} u'_i \tilde{y}_i.$$

By (15), they are equal.

- For $J \in \mathcal{J}^{(2)}$, we add $S_J(t_J)$ to W . We satisfy the demands at J using $S_J(t_J)$, by following the instance $\mathcal{I}_J(t_J)$. To address a small technicality issue, for every $G' \in \mathcal{G}$, we arbitrarily choose $\lfloor \sum_{J \in \mathcal{J}^{(2)} \cap G'} (y^*(U(J)) - t_J) \rfloor$ facilities in $\bigcup_{J \in \mathcal{J}^{(2)} \cap G'} (U(J) \setminus S_J(t_J))$, and add them to W . They do not serve any demand and we call them “dummy facilities”.
- For $J \in \mathcal{J}^{(3)}$, we add $S_J(t_J)$ to W . We satisfy the demands at J using $S_J(t_J)$, by following the instance $\mathcal{I}_J(t_J)$.

- For $J \in \mathcal{J}^{(4)}$, we add $S_J(\lceil \tilde{y}_J \rceil)$ to W . Similarly, we satisfy the demands at J using $S_J(\lceil \tilde{y}_J \rceil)$, by following the instance $\mathcal{I}_J(\lceil \tilde{y}_J \rceil)$.

The size of W may be more than k for two reasons. First, some type-(1) component J may have some facility $i \in U(J)$ with $\tilde{y}_i \in (0, 1)$ and we included it in W . Second, some type-(3) component may have $\tilde{y}_J = t_J - 1$ but we added $S_J(t_J)$ to W . To reduce the size of W , we run Algorithm 2.

Algorithm 2 reduce the size of W to at most k

- 1: **for** every black component $J \in \mathcal{J}$ respecting the top-to-bottom order **do**
 - 2: let G be the parent group of the group containing J , or $G = \{J\}$ if J is a root group
 - 3: **if** J is of type-(1) **then**
 - 4: let a be the number of facilities $i \in W \cap U(J)$ with $\tilde{y}_i < 1$
 - 5: repeat a times: $\text{remove}(G, J)$
 - 6: **else if** J is of type-(3) and $\tilde{y}_J = t_J - 1$ **then**
 - 7: $\text{remove}(G, J)$
-

Algorithm 3 $\text{remove}(G, J)$

- 1: let i^* be the facility in $W \cap U(G \cup \{J\})$ serving the least demand
 - 2: remove i^* from W
 - 3: resend the demands sent to i^* to $W \cap U(G)$, split proportionally to the amount of demands they already serve
-

We first argue that the algorithm is well-defined. The initial W we constructed has $|W \cap U(G')| \geq y^*(U(G')) - O(1)$ for every group G' . This holds due to (12), (13), which implies $\sum_{J \in \mathcal{J}^{(3,4)} \cap G'} (t_J - \tilde{y}_J) \leq 1$ for every group G' , and that we added dummy facilities for type-(2) components. In particular, for every non-leaf group $G \in \mathcal{G}$, we have $|W \cap U(G)| \geq \Omega(\ell)$. By the first statement of Lemma 4.4, for any non-leaf $G \in \mathcal{G}$, the number of times we call $\text{remove}(G, J)$ over all type-(1) components J at most 3. By (13), the number of times we call $\text{remove}(G, J)$ over all type-(3) components J at most 1. Therefore, the facility i^* can always be chosen. Moreover, at the end of the algorithm, we still have $|W \cap U(G)| \geq \Omega(\ell)$.

The initial $|W|$ exceeds k by at most the number of strictly fractional type-(1) facilities, plus the number of type-(3) components J with $\tilde{y}_J = t_J - 1$. This holds due to the second statement of Lemma 4.4. As we removed this number of facilities, we have $|W| \leq k$ in the end.

4.5 Analysis of Cost and Capacity Violation

We analyze the cost and capacity violation of our solution.

Cost of Initial Flow Before Removing Operations We first analyze the cost of the flow sent from v to the initial W , before we apply the remove operations.

Focus on the cost incurred by type-(1) black components. For a non-leaf group $G \in \mathcal{G}$, the cost of sending the demands from $\bigcup_{G' \in \text{children}'(G), J \in \mathcal{J}^{(1)} \cap G'} J$ to $W \cap \bigcup_{G' \in \text{children}'(G)} U(\mathcal{J}^{(1)} \cap G')$ can be bounded as follows. We take an arbitrary representative v at the root component of G , as the transfer point. First we send all the demands to v , and then we dispatch the demands from v to

$W \cap \bigcup_{G' \in \text{children}'(G)} U(\mathcal{J}^{(1)} \cap G')$. The cost incurred in the two steps is at most

$$\sum_{G' \in \text{children}'(G), J \in \mathcal{J}^{(1)} \cap G', i \in U(J)} \left(y_i^* u'_i \cdot O(\ell^2) L_J + \tilde{y}_i u'_i (d(i, J) + O(\ell^2) L_J) \right).$$

Therefore, the cost of the initial flow incurred by type-(1) black components is

$$M_1 := \sum_{J \in \mathcal{J}^{(1)}, i \in U(J)} \left(y_i^* u'_i \cdot O(\ell^2) L_J + \tilde{y}_i u'_i (d(i, J) + O(\ell^2) L_J) \right).$$

The cost of initial flow incurred by type-(2), (3) and (4) black components is at most

$$\begin{aligned} M_{234} &:= \sum_{J \in \mathcal{J}^{(2,3)}} (\Psi_J(t_J) + O(\ell) \cdot o_J(t_J) L_J) + \sum_{J \in \mathcal{J}^{(4)}} (\Psi_J(\lceil \tilde{y}_J \rceil) + O(\ell) \cdot o_J(\lceil \tilde{y}_J \rceil) L_J) \\ &\leq O(\ell) \cdot \sum_{J \in \mathcal{J}^{(2,3,4)}} \Psi_J(t_J) + \sum_{J \in \mathcal{J}^{(4)}} (\Psi_J(t_J - 1)(t_J - \tilde{y}_J)) \end{aligned}$$

The cost of the initial flow is at most $M_1 + M_{234}$.

Cost due to Removing Operations Now we analyze the cost incurred by removing operations. During the analysis, we only bound the cost of the flows between representatives. To be more specific, suppose we resend the flow sent to some i to another facility i' , where $i \in U_v$ and $i' \in U_{v'}$. Then we only consider the cost of sending the flow from v to v' . In our analysis, all the flow sent to i are sent through v . That is, the cost of sending a unit flow from some representative v'' to i is calculated as $d(v'', v) + d(v, i)$. This applies to the solutions for the penalty instances $S_J(t)$'s. Therefore, to resend the flow sent to i , we actually undo a portion of the flow sent from v to i . Second, due to the proportional distribution, we can charge the cost of sending the flow from v' to i' using the existing flow sent to i' . We will call the cost of sending the flow from v to v' the R -to- R cost, and the cost of sending the flow from v' to i' the R -to- F cost.

Notice that as we handle the black components J in Algorithm 2 in a top-to-bottom fashion, no facilities have been removed from $W \cap U(J)$ before we handle J . Then, we prove the following lemma:

Lemma 4.5. *remove(G, J) incurs an R -to- R cost of at most $O(\ell^3) \Psi_J(t_J)$ if J is of type-(1), and $O(\ell^3) \Psi_J(t_J - 1)$ if J is of type-(3).*

Proof. Consider the case where J is a type-(1) component. If $J = \{v\}$ is a singleton root component, then $G = \{J\}$. Also, $i^* \in U_v$. So, the R -to- R cost of resending flow from i^* to the new $W \cap U_v$ is 0.

In all other cases, we have $o_J(t_J) \geq \frac{1}{\ell} x^*(U(J), C)$, implying $\Psi_J(t_J) \geq \frac{1}{\ell} x^*(U(J), C) L_J / 2$. The R -to- R cost of resending the flow to $W \cap U(G)$ is at most $x^*(U(J), C) \cdot O(\ell^2) L_J$ by Lemma (4.2d) and the demand of i^* is at most $x^*(U(J), C)$. So the cost is at most $O(\ell^3) \Psi_J(t_J)$.

The case where J is type-(3) component is proved similarly except that we use $o_J(t_J - 1) \geq \frac{1}{\ell} x^*(U(J), C)$. $\square$

The remove operations incur a R -to- R cost of

$$O(\ell^3) \left(\sum_{J \in \mathcal{J}^{(1)}} \Psi_J(t_J) + \sum_{J \in \mathcal{J}^{(3)}; \tilde{y}_J = t_J - 1} \Psi_J(t_J - 1) \right)$$

$$\leq O(\ell^3) \left(\sum_{J \in \mathcal{J}^{(1)}} \Psi_J(t_J) + \sum_{J \in \mathcal{J}^{(3)}} (t_J - \tilde{y}_J) \Psi_J(t_J - 1) \right)$$

So, the final cost without counting the R -to- F cost of the removing operations is at most

$$\begin{aligned} & M_1 + O(\ell^3) \cdot \sum_{J \in \mathcal{J}} \Psi_J(t_J) + O(\ell^3) \left(\sum_{J \in \mathcal{J}^{(3,4)}} (t_J - \tilde{y}_J) \Psi_J(t_J - 1) \right) \\ &= \sum_{J \in \mathcal{J}^{(1)}, i \in U(J)} \left(y_i^* u_i' \cdot O(\ell^2) L_J + \tilde{y}_i u_i'(d(i, J) + O(\ell^2) L_J) \right) + O(\ell^3) \cdot \sum_{J \in \mathcal{J}} \Psi_J(t_J) \\ &\quad + O(\ell^3) \left(\sum_{J \in \mathcal{J}^{(3,4)}} (t_J - \tilde{y}_J) \Psi_J(t_J - 1) \right) \\ &\leq \sum_{J \in \mathcal{J}^{(1)}, i \in U(J)} \left(y_i^* u_i' \cdot O(\ell^2) L_J + y_i^* u_i'(d(i, J) + O(\ell^2) L_J) \right) + O(\ell^3) \cdot \sum_{J \in \mathcal{J}} \Psi_J(t_J) \\ &\quad + O(\ell^3) \left(\sum_{J \in \mathcal{J}^{(3,4)}} (t_J - y^*(U(J))) \Psi_J(t_J - 1) \right) \\ &\leq \sum_{J \in \mathcal{J}^{(1)}, i \in U(J)} \left(y_i^* u_i' d(i, J) \right) + O(\ell^3) \cdot \sum_{J \in \mathcal{J}} \Psi_J(t_J) + O(\ell^3) \left(\sum_{J \in \mathcal{J}^{(3,4)}} (t_J - y^*(U(J))) \Psi_J(t_J - 1) \right) \\ &\leq O(1) \sum_{j \in C} \phi_j^* + O(\ell^6) \sum_{J \in \mathcal{J}, v \in J, j \in C} g_{j,v}(\phi_j^* + d(v, j)) \leq O(\ell^6) \sum_{j \in C} \phi_j^*. \end{aligned}$$

To see the second-to-last inequality, we bound $\Psi_J(t_J)$ for $J \in \mathcal{J}$ and $(t_J - y^*(U(J))) \Psi_J(t_J - 1)$ for $J \in \mathcal{J}^{(3,4)}$ respectively. By inequality (9), we have

$$\frac{t+1-y^*(U(J))}{t+1} \Psi_J(t) \leq (2|J|-1) \sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j))$$

Recall that $t_J = \lfloor (1 + \frac{1}{c\ell^2}) y^*(U(J)) \rfloor$. We have $\frac{y^*(U(J))}{t_J+1} \leq 1/(1 + \frac{1}{c\ell^2})$, implying that $\frac{t_J+1-y^*(U(J))}{t_J+1} \geq \Omega(1/\ell^2)$. Therefore, we have $\Psi_J(t_J) \leq O(\ell^3) \sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j))$.

Consider the case $J \in \mathcal{J}^{(3,4)}$, where we have $t_J > y^*(U(J))$.

$$\begin{aligned} (t_J - y^*(U(J))) \Psi_J(t_J - 1) &= t_J \cdot \frac{t_J - y^*(U(J))}{t_J} \Psi_J(t_J - 1) \leq t_J (2|J| - 1) \sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j)) \\ &\leq O(\ell^2) \sum_{v \in J, j \in C} g_{j,v}(\phi_j + d(v, j)), \end{aligned}$$

where we used that $t_J \leq O(\ell)$.

Then we argue about the capacity violation of the solution, along with the R -to- F cost of the removing operations. In the initial flow, every facility $i \in W$ receives at most $(1 + O(1/\ell)) u_i$ units of flow. As we argued, $|W \cap U(G)| \geq \Omega(\ell)$ always holds. When we send some flow to $W \cap U(G)$

for some non-leaf group $G \in \mathcal{G}$, we guarantee that the amount of flow sent is at most the least flow received by any facility in $W \cap U(G)$. Also we apply the operation for any G at most $O(1)$ times. Therefore, the operations will increase the flow received by any i by a factor of at most $1 + O(1/\ell)$. This also implies that the R -to- F cost incurred by the removing operations is negligible.

Notice that the initial moving step in Section 2.1, which moves demands from J to R has cost $O(1) \cdot \sum_{j \in C} \phi_j^*$. So, the overall moving cost of our solution is at most $O(\ell^6) \cdot \sum_{j \in C} \phi_j^*$. This finishes the proof of Theorem 1.2.

References

- [1] Karen I. Aardal, Pieter L. van den Berg, Dion Gijswijt, and Shanfei Li. Approximation algorithms for hard capacitated k -facility location problems. *European Journal of Operational Research*, 242(2):358–368, 2015.
- [2] Marek Adamczyk, Jaroslaw Byrka, Jan Marcinkowski, Syed Mohammad Meesum, and Michal Włodarczyk. Constant-factor FPT approximation for capacitated k -median. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144, pages 1:1–1:14, 2019.
- [3] Hyung-Chan An, Mohit Singh, and Ola Svensson. Lp-based algorithms for capacitated facility location. *SIAM Journal on Computing*, 46(1):272–306, 2017.
- [4] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k -median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
- [5] Jaroslaw Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim 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 2015*, pages 722–736, 2015.
- [6] Jaroslaw Byrka, Yuhao Guo, Yang Hu, Shi Li, Chengzhang Wan, and Zaixuan Wang. k -clustering via iterative randomized rounding. In *Proceedings of the 67th Annual Symposium on Foundations of Computer Science, FOCS 2026*, 2026.
- [7] Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k -median, and positive correlation in budgeted optimization. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 737–756, 2015.
- [8] Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k -median and positive correlation in budgeted optimization. *ACM Transactions on Algorithms*, 13(2):23:1–23:31, 2017.
- [9] Jaroslaw Byrka, Bartosz Rybicki, and Sumedha Uniyal. An approximation algorithm for uniform capacitated k -median problem with $1 + \epsilon$ capacity violation. In *Integer Programming and Combinatorial Optimization-18th International Conference, IPCO 2016*, volume 9682, pages 262–274, 2016.

- [10] Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia 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, 2000.
- [11] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: Deterministic approximation algorithms for group steiner trees and k -median. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, STOC 1998*, pages 114–123, 1998.
- [12] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k -median problem (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, STOC 1999*, pages 1–10, 1999.
- [13] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k -median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.
- [14] Julia Chuzhoy and Yuval Rabani. Approximating k -median with non-uniform capacities. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005*, pages 952–958, 2005.
- [15] Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, Chris Schwiegelshohn, and Ola Svensson. A $(2 + \epsilon)$ -approximation algorithm for metric k -median. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025*, pages 615–624, 2025.
- [16] Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. An improved local search algorithm for k -median. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 1556–1612, 2022.
- [17] H. Gökalp Demirci and Shi Li. Constant approximation for capacitated k -median with $(1 + \epsilon)$ -capacity violation. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55, pages 73:1–73:14, 2016.
- [18] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC 2003*, pages 448–455, 2003.
- [19] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.
- [20] Kishen N. Gowda, Thomas W. Pensyl, Aravind Srinivasan, and Khoa Trinh. Improved bi-point rounding algorithms and a golden barrier for k -median. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 987–1011, 2023.
- [21] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

- [22] Dorit S Hochbaum and David B Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.
- [23] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.
- [24] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, STOC 2002*, pages 731–740, 2002.
- [25] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k -means clustering. *Computational Geometry*, 28(2-3):89–112, 2004.
- [26] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.
- [27] Shanfei Li. An improved approximation algorithm for the hard uniform capacitated k -median problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014*, volume 28, pages 325–338, 2014.
- [28] Shi Li. Approximating capacitated k -median with $1 + \epsilon$ open facilities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 786–796, 2016.
- [29] Shi Li. On uniform capacitated k -median beyond the natural lp relaxation. *ACM Transactions on Algorithms*, 13(2):1–18, 2017.
- [30] Shi Li and Ola Svensson. Approximating k -median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547, 2016.

A Omitted proofs

Lemma 2.1. *The following statements hold:*

- (2.1a) *for all $v, v' \in R$ and $v \neq v'$, we have $d(v, v') > 4 \max\{d_{av}(v), d_{av}(v')\}$;*
- (2.1b) *for all $j \in C$, there exists $v \in R$, such that $d_{av}(v) \leq d_{av}(j)$ and $d(v, j) \leq 4d_{av}(j)$;*
- (2.1c) *$y^*(U_v) \geq 1/2$ for every $v \in R$;*
- (2.1d) *for any $v \in R, i \in U_v$ and $j \in C$, we have $d(i, v) \leq d(i, j) + 4d_{av}(j)$.*

Proof. For Property (2.1a), let $v, v' \in R$ be distinct, and assume $d_{av}(v) \leq d_{av}(v')$. When v is added to R , the algorithm removes every client j with $d(v, j) \leq 4d_{av}(j)$. Since $v' \in R$, it was not removed in that iteration. Hence $d(v, v') > 4d_{av}(v')$. For Property (2.1b), fix any client $j \in C$, and consider the iteration in which j is removed from C . Then consider Property (2.1c). By (2.1a), we have $B := \{i \in F : d(i, v) \leq 2d_{av}(v)\} \subseteq U_v$. Since $d_{av}(v) = \sum_{i \in F} x_{i,v}^* d(i, v)$ and $\sum_{i \in F} x_{i,v}^* = 1$, we have $d_{av}(v) \geq (1 - x_{B,v}^*)2d_{av}(v)$, implying $y_{U_v}^* \geq y_B^* \geq x_{B,v}^* \geq 1/2$, due to Constraint (4).

Then we consider Property (2.1d). By (2.1b), there is a client $v' \in R$ such that $d_{av}(v') \leq d_{av}(j)$ and $d(v', j) \leq 4d_{av}(j)$. Since $d(i, v) \leq d(i, v')$ as $v' \in R$ and i was added to U_v , we have $d(i, v) \leq d(i, v') \leq d(i, j) + d(j, v') \leq d(i, j) + 4d_{av}(j)$. $\square$

Lemma 3.1. *For every $G \in \mathcal{G}$, every $J \in G$ and $v \in J$, the distance between v and any representative in root_G is at most $O(\ell) \cdot L_J$.*

Proof. For each $G = (P, E_T) \in \tau$, and let $\tilde{G}$ be the tree obtained by expanding each node of G back to its corresponding black component. First consider the case where the root r of G satisfies $r \neq r_\tau$. Since r is not the root of τ , by Lemma (2.5a) its weight is less than ℓ . By construction, the total weight of $P \setminus \{r\}$ is at most 2ℓ . Thus the total weight of P is at most 3ℓ . By Lemma (2.1c), each original representative v satisfies $y(U_v) \geq 1/2$. Therefore the number of nodes in $\tilde{G}$ is at most $\frac{3\ell}{1/2} = 6\ell$. Next consider the case $r = r_\tau$. the root node r_τ either has weight at most 2ℓ , or its corresponding black component J_{r_τ} is a singleton. In either case, the number of nodes after expansion can be uniformly bounded by $\frac{2\ell+2\ell}{1/2} = 8\ell$. $\square$

Lemma 4.1. Υ^* satisfies the following properties:

(4.1a) for any leaf-to-root path of $\tau^* \in \Upsilon^*$, the L -values of nodes are non-increasing.

(4.1b) every root component $J \in \mathcal{J}$ of Υ^* has either $y^*(U(J)) \leq 2\ell$ or $|J| = 1$.

(4.1c) for any non-root component J and its parent J' in Υ^* , we have $\max_{v \in J, v' \in J'} d(v, v') \leq O(\ell) \cdot L_J$.

Proof. For Property (4.1a), for any child-parent edge (J, J') in Υ^* , we consider two cases: if J' is the parent of J in the original gray tree. By Lemma (2.5c), the leaf-to-root monotonicity of L -values for the original gray tree, we have $L_{J'} \leq L_J$. If J and J' are adjacent siblings in the original tree, with J' ordered before J . Due to the sorting rule, $L_{J'} \leq L_J$. Hence the L -values are non-increasing along every leaf-to-root path.

For Property (4.1b), suppose a black component J contains multiple representatives, that is $|J| > 1$. Then it must be formed by merging at least two black components through the last black edge. Let the two child components before this final merge be J_v and $J_{v'}$, satisfying $y^*(U(J_v)) < \ell$ and $y^*(U(J_{v'})) < \ell$. After merging, we have $J = J_v \cup J_{v'}$, and the total weight satisfies $y^*(U(J)) = y^*(U(J_v)) + y^*(U(J_{v'})) \leq 2\ell$.

For Property (4.1c), let J'' be the parent of J in the original gray forest Υ . By the construction of Υ^* , either $J' = J''$, or J' is the sibling of J immediately preceding it in the non-decreasing order of L -values. Hence $L_{J''} \leq L_J$ and $L_{J'} \leq L_J$. For every $J \in \mathcal{J}$, $|J| \leq O(\ell)$, and by Claim 2.4, we have $\text{diam}(J) \leq O(\ell)L_J$. If $J' = J''$, then the original gray edge between J and J' has length L_J . Thus, for every $v \in J$ and $v' \in J'$, $d(v, v') \leq \text{diam}(J) + L_J + \text{diam}(J') = O(\ell)L_J$. Otherwise, both J and J' are children of J'' in Υ . The gray edges from J and J' to J'' have lengths L_J and $L_{J'}$, respectively. Hence, $d(v, v') \leq \text{diam}(J) + L_J + \text{diam}(J'') + L_{J'} + \text{diam}(J') = O(\ell)L_J$. Therefore, $\max_{v \in J, v' \in J'} d(v, v') \leq O(\ell)L_J$. $\square$

Lemma 4.2. $\mathcal{G}$ satisfies the following properties:

(4.2a) if $G \in \mathcal{G}$ is not a root group, then $y^*(U(G)) < 2\ell$;

(4.2b) if $G \in \mathcal{G}$ is a non-leaf group, then $y^*(U(G)) \geq \ell$;

(4.2c) any group G has at most $O(\ell)$ child groups;

(4.2d) let G be a group, $G' \in \text{children}'(G)$, and $J' \in G'$, then $\max_{v' \in J', v \in J \in G} d(v, v') \leq O(\ell^2)L_{J'}$.

Proof. For Property (4.2a), any non-root component J has $y^*(U(J)) < \ell$. Thus, if G is not a root group, the terminating condition constructing G implies that G had total weight less than ℓ right before the last black component was added to it. Then we have $\sum_{J \in G} y^*(U(J)) < 2\ell$.

For Property (4.2b), when we construct the group G from the tree τ^* by Algorithm 1, the terminating condition is that G contain all descendants of J in τ^* or $\sum_{J \in G} y^*(U(J)) \geq \ell$. Thus, if G is not a leaf-group, then the condition that G contains all descendants of J in τ^* does not hold. So we have $\sum_{J \in G} y^*(U(J)) \geq \ell$.

For Property (4.2c), since the forest Υ^* is binary and every group $G \in \mathcal{G}$ contains at most $O(\ell)$ components, we have that every group G contains at most $O(\ell)$ children.

Finally for Property (4.2d), in the transformed gray tree Υ^* , the L -values are non-increasing along every leaf-to-root path. By Algorithm 1, we select the child component with the smallest L -value add G each time. Let G' be a child group of G and let J'_0 be the root component of G' . Then $L_J \leq L_{J'_0}$ for every $J \in G$. On the other hand, every component $J' \in G'$ is a descendant of J'_0 . By the monotonicity of L -values along root-to-leaf paths, $L_{J'_0} \leq L_{J'}$ for all $J' \in G'$. Therefore, $\max_{J \in G} L_J \leq L_{J'_0} \leq \min_{J' \in G'} L_{J'}$. The property then holds by Claim 2.4, Lemma (4.1a), (4.1c) and that each group has at most $O(\ell)$ components. $\square$