A Constant Approximation for Min-sum k-Clustering

Aditya Anand* Moses Charikar† Ruiquan Gao† Fabrizio Grandoni‡
Euiwoong Lee§ Amatya Sharma§ Ernest van Wijland¶

November 3, 2026

Abstract

In the classical Min-Sum k-Clustering problem we are given a collection D of n clients with metric distances among them, and a positive integer $k \leq n$. A feasible solution is a partition of D into k subsets, the clusters. The cost of cluster C is the sum of the pairwise distances inside C, and the objective function, that we wish to minimize, is the sum of the cluster costs.

This problem is APX-hard, and it is very well studied in terms of approximation algorithms. The current best approximation factor is $O(\log n)$ [Behsaz, Friggstad, Salavatipour, Sivakumar - ICALP15], improving on the first non-trivial $O(\frac{1}{\varepsilon}\log^{1+\varepsilon} n)$ approximation in [Bartal, Charikar, Raz - STOC01]. Several results for relaxations and special cases of the problem have been developed continuously over the last two decades. A major open problem in the area of approximation algorithms for clustering is whether this problem admits a constant approximation.

In this paper we present the first constant approximation for Min-Sum k-Clustering. Our result is based on two main ingredients: (1) a constant approximation for a Lagrangian relaxation of the problem and (2) a constant bipoint rounding algorithm. Both these ingredients involve highly non-trivial, novel ideas.

*Max Planck Institute for Informatics.
†Stanford University.
‡IDSIA, USI-SUPSI.
§University of Michigan.
¶Université Paris-Cité.

Contents

1 Introduction **1**
- 1.1 Further Related Work . . . 1
- 1.2 Challenges in obtaining a constant approximation . . . 2

2 Overview of Our Approach **4**
- 2.1 Overview of the algorithm . . . 4
- 2.2 The LMP approximation . . . 4
- 2.3 Bipoint rounding . . . 11
- 2.4 Organization of the paper . . . 16

3 Cluster Splitting and Editing **16**
- 3.1 Cluster Splitting . . . 16
- 3.2 Cluster Editing . . . 18

4 The LMP approximation **19**
- 4.1 The dual-fitting algorithm . . . 19
- 4.2 Overview of the analysis . . . 24
- 4.3 Basic properties of opened clusters and dependency forests . . . 26
- 4.4 Dual Payments . . . 30
- 4.5 Dual Feasibility . . . 34
  - 4.5.1 Frozen critical clusters . . . 34
  - 4.5.2 Dual feasibility of available sets: an invariant throughout the algorithm . . . 35
  - 4.5.3 On opening times of the clusters . . . 36
  - 4.5.4 Dual feasibility of any critical cluster . . . 38

5 Bipoint Rounding **42**
- 5.1 Partition the clusters and clients into good and bad . . . 42
- 5.2 An overview of the final solution . . . 43
- 5.3 Preprocessing . . . 43
- 5.4 Opening clusters among the residual clusters . . . 44
- 5.5 Putting things together . . . 49
  - 5.5.1 Bounding the expected cost of the $\mathcal{S}_2$-clusters . . . 50
  - 5.5.2 Bounding the expected cost of each $\mathcal{S}_1$-cluster . . . 55
  - 5.5.3 Proof of Theorem 3 . . . 56

A Omitted Proofs **62**
- A.1 Proof of Theorem 1 . . . 62

1 Introduction

In the classical *min-sum k-clustering* problem, we are given a collection D of n elements, the *clients*, with (finite) metric distances $d(u,v) \geq 0$, $u,v \in D$, among them, and a positive integer $k \leq n$. A feasible solution is a partition $\mathcal{S}$ of D into k subsets, the *clusters*. The (connection) cost[1] of cluster S is the sum of the pairwise distances inside S, namely $\mathsf{cost}(S) := \sum_{i \in S} d(i,S) = \sum_{i \in S} \sum_{j \in S} d(i,j)$. The objective function, that we wish to minimize, is the sum of the cluster costs $\mathsf{cost}(\mathcal{S}) := \sum_{S \in \mathcal{S}} \mathsf{cost}(S)$.

This problem was first defined by Sahni and Gonzalez in 1974 [SG74; SG76], and it is very-well studied in terms of approximation algorithms. Guruswami and Indyk proved it to be APX-hard [GI03]. The current-best lower bound on its approximability is 1.415 due to Cohen-Addad, Karthik and Lee [CKL21]. Bartal, Charikar and Raz [BCR01] gave the first non-trivial (polynomial-time) approximation algorithm for this problem, with approximation factor $O(\frac{1}{\varepsilon} \log^{1+\varepsilon} n)$ for any $\varepsilon > 0$. This result is based on probabilistic embeddings into HSTs. Using a refined version of the same high-level approach, the approximation factor was later improved to $O(\log n)$ by Behsaz, Friggstad, Salavatipour and Sivakumar [BFSS15; BFSS19]. This is also the current-best approximation factor for the general (metric) version of the problem.

A lot of effort has been devoted to the design of improved approximation algorithms for relaxations and special cases of the problem. Bartal el al. [BCR01] developed a bicriteria constant approximation if one is allowed to exceed the target number of clusters by a constant factor. Banerjee, Ostrovsky, and Rabani [BOR21] obtained a different bicriteria constant approximation if one is allowed to exclude a small fraction of the points from the clusters. Earlier, Guttman-Beck and Hassin [GH98] gave a 2-approximation that runs in $n^{O(k)}$ time. A polynomial-time $1+\varepsilon$ approximation for constant k was given by de la Vega, Karpinski, Kenyon, and Rabani [VKKR03]. Czumaj and Sohler [CS10] give a constant approximation that runs in polynomial time for $k = O(\log n / \log\log n)$. Behsaz et al. [BFSS19] describe a constant approximation running in quasi-polynomial time for the case of metrics with bounded doubling dimension (that includes fixed dimension Euclidean spaces). This was improved to a $1+\varepsilon$ approximation (still in quasi-polynomial time) by Naderi, Rezapour, and Salavatipour [NRS24]. Their approach also works for metrics induced by graphs of bounded treewidth and by minor-free graphs (e.g., planar graphs).

Despite all the mentioned research efforts in the last quarter of a century, the following major question in the area of approximation algorithms for clustering problems remains open:

Does Min-Sum k-Clustering admit a constant approximation in polynomial time?

We answer this question in the affirmative.

Theorem 1. *There is a randomized polynomial-time* $(2.16 \times 10^9 + \varepsilon)$*-approximation for the min-sum k-clustering problem.*

Our approach can be extended to a constant-factor approximation for the generalized min-sum (k,z)-clustering problem for any fixed $1 \leq z < \infty$, where the objective becomes $\sum_{S \in \mathcal{S}} \sum_{i,j \in S} d^z(i,j)$. The details will appear in a full version of the paper.

An overview of our approach is given in Section 2.

1.1 Further Related Work

Min-Sum k-Clustering belongs to a well-studied family of k-clustering problems, where the goal is to partition a set of input elements, the clients, with metric distances among them into k subsets.

[1] Sometimes in the literature one considers unordered pairs in the cost. This is irrelevant for our goals.

One of the most famous such problems is k-Means. Here for each cluster we identify a center in a candidate set, and the goal is to minimize the sum of the squared distances of each client to the center of its cluster. The current-best[2] approximation for this problem is $4.9 + \varepsilon$ [ACC+27], improving on a long sequence of previous results [GT08; ANSW20; CCG+25; BGH+26]. The same paper [ACC+27] also provides the current-best $3 + \ln 2 + \varepsilon$ approximation for the Euclidean case, again improving on several previous results [KMN+02; ANSW20; GOR+22; CEMN22; CCG+26; BGH+26]. Better results are known for the low-dimensional Euclidean case [FRS16; CKM16; Coh18].

The k-Median problem is defined similarly to the k-Means one, with the difference that one here considers the distances of the elements to the respective centers, rather than the squared distances. A better-than-2 approximation for this problem was recently obtained [BCF+26b; BCF+26a], improving on earlier results [CCGG98; CGTS02; JV01; CG99; JMM+03; AGK+04; CL12; LS16; BPR+17; GPST23; CGLS23; CGL+25]. The earlier $(2+\varepsilon)$-approximation [CGL+25] also improved the previous best results for the Euclidean case [ANSW20; CEMN22].

Other well-studied problems in this area are Min-Sum k-Radii and Min-Sum k-Diameters. The input and output are the same as in Min-Sum k-Clustering, but the objective function is to minimize the sum of the radii and diameters, resp., of the clusters. For Min-Sum k-Radii the current best approximation factor in polynomial time is $3 + \varepsilon$ [BERW24], improving on [CP04; FJ22]. The problem also admits a QPTAS [GKK+12] and an FPT approximation scheme parameterized by k [GGY26]. By a standard reduction, all the mentioned results can also be obtained for Min-Sum k-Diameters with an extra factor 2 in the approximation (and the same running). In addition to that, an FPT approximation scheme parameterized by k is given in [GGY26].

1.2 Challenges in obtaining a constant approximation

A well-established route to obtaining constant factor approximation algorithms for clustering problems with at most k-clusters is *Lagrangian relaxation*. We briefly outline this approach below (discussed in more detail in Section 2) and discuss two separate ingredients that are needed, neither of which were available for Min-Sum k-Clustering.

Lagrangian relaxation. Instead of insisting on at most k clusters, we relax this constraint and charge a price $f \geq 0$ for each cluster. For a partition $\mathcal{S}$ of D into $|\mathcal{S}|$ clusters, we define the Lagrangian objective

$$\mathsf{cost}_f(\mathcal{S}) = f|\mathcal{S}| + \mathsf{cost}(\mathcal{S})$$

For $f = 0$, the optimal solution puts every point in its own cluster and as f increases, the optimal partition uses fewer and fewer clusters. The Lagrangian objective is easier to design approximation algorithms for because it does not have a hard limit on the number of clusters. The hope is that for suitable choice of f, we can turn the solutions of this relaxed problem into solutions of the original clustering problem with at most k clusters. Jain and Vazirani [JV01] introduced this approach and used it to design a primal-dual approximation algorithm for k-median. Since then, it has become a standard tool in the approximation algorithms toolkit. Two additional ingredients are needed:

The first ingredient is an approximation algorithm for the Lagrangian objective that has a special, stronger guarantee called *Lagrangian Multiplier Preserving (LMP)*. Let OPT denote the cost of the optimal solution with k clusters. For constant $c \geq 1$, an algorithm is a c-LMP approximation if, for every f, it returns a partition $\mathcal{S}$ such that

$$f|\mathcal{S}| + \mathsf{cost}(\mathcal{S}) \leq k \cdot f + c \cdot \mathsf{OPT}$$

[2] In this section we focus on peer-reviewed published papers.

Such an algorithm can exceed the optimum by a factor c on the cluster cost term, but not for the per-cluster price.

Now, we find f (by binary search) such that the algorithm returns a solution $\mathcal{S}_1$ with $k_1 \leq k$ clusters and, at essentially the same price f, another solution $\mathcal{S}_2$ with $k_2 \geq k$ clusters. Express k as a convex combination of k_1 and k_2: $a_1k_1 + a_2k_2 = k$, where $a_1, a_2 \geq 0, a_1 + a_2 = 1$. Averaging the two inequalities we have for $\mathsf{cost}(\mathcal{S}_1)$ and $\mathsf{cost}(\mathcal{S}_2)$, we get

$$a_1\mathsf{cost}(\mathcal{S}_1) + a_2\mathsf{cost}(\mathcal{S}_2) \leq c \cdot \mathsf{OPT}$$

where the LMP guarantee is crucial to eliminate terms involving f. The pair $(\mathcal{S}_1, \mathcal{S}_2)$ with weights a_1, a_2 is called a *bipoint solution*: a blend of two clusterings that uses k clusters on average with cost at most c times optimal.

The second ingredient is *bipoint rounding*: a procedure that converts a bipoint solution into a valid solution for the original clustering problem with at most k clusters, losing only a constant factor in the clustering cost. For k-median, [JV01] provided such a rounding algorithm and this was refined in later works.

This algorithmic recipe of designing an LMP approximation together with a bipoint rounding procedure has become a powerful technique in approximation algorithms. However, designing these ingredients for Min-Sum k-Clustering runs into obstacles.

Obstacle 1: LMP algorithm for min-sum clustering. The LMP guarantees known for facility location and k-median come from the primal-dual method, in which every point raises a payment, and a cluster opens once the payments of its points cover its cost. This works because in those problems each point has its own connection cost, and the cost of a cluster depends only on its opening price and on how far its points are from its center. In min-sum k-clustering there are no centers and no per-point costs: the cost of a cluster is a sum over pairs of its members, so it grows quadratically with the cluster size, and what a point should pay depends on how many other points end up with it.

[BCR01] introduced the balanced k-median objective which takes the usual k-median objective and multiplies each cluster cost by the cluster size

$$\sum_{C \in \mathcal{C}} |C| \cdot \min_{i \in C} \sum_{j \in C} d(i, j)$$

This approximates the min-sum objective to within a factor of 2 and has been useful in designing algorithms for min-sum; however existing primal-dual algorithms have only produced weaker bicriteria guarantees (allowing more than k clusters). Since distances are multiplied by cluster sizes in this objective function, the triangle inequality breaks down and this complicates the analysis task.

Obstacle 2: bipoint rounding for min-sum clustering. Even with a bipoint solution available, it is unclear how to round it. [BFSS19] point this out explicitly: "it seems difficult to combine two solutions obtained from the primal-dual method with $k_1 < k < k_2$ number of centers. If one could overcome this technical difficulty then it could lead to a $O(1)$-approximation" (for min-sum k-clustering and balanced k-median). Intuitively, the difficulty arises with the nature of the min-sum objective function: In k-median, a point whose center is closed can be sent to a nearby open center, and the triangle inequality bounds the increase in that point's cost, independently of what other points do. In min-sum clustering, sending a point into an existing cluster raises the cost of that cluster by the distances to all of its current members, and merging two clusters of sizes t_1 and t_2 can add up to $t_1 \cdot t_2$ pair-distances. To go from the $k_2 > k$ clusters of $\mathcal{S}_2$ down to k clusters,

many clusters must be merged, and the merging cost can far exceed what either $\mathcal{S}_1$ or $\mathcal{S}_2$ pays. The same quadratic dependence on cluster size that complicates the LMP step therefore also defeats the standard rounding arguments.

2 Overview of Our Approach

We start this section by introducing some notation that will be used throughout the paper. We then provide a technical overview of our algorithm and its analysis. For a reference solution OPT, we let $\mathsf{opt} = \mathsf{cost}(\mathsf{OPT})$. W.l.o.g., we can assume $2 \le k < n$, otherwise the problem is trivial to solve optimally. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We use $[l : r]$ to denote the set $\{l, l+1, \ldots, r\}$ for any integers $l \le r$.

2.1 Overview of the algorithm

Similarly as the seminal work of [JV01], we first give a constant-factor Lagrange Multiplier Preserving (LMP) approximation, and then use a bipoint rounding algorithm to convert the LMP approximations to true approximation that loses only a constant factor. More specifically, we prove the following two theorems.

Theorem 2. *There exists a polynomial-time algorithm that takes a min-sum k-clustering instance and an additional $f > 0$ as input and outputs a clustering $\mathcal{S}$ satisfying*

$$f|\mathcal{S}| + \mathsf{cost}(\mathcal{S}) \le k \cdot f + 2 \times 10^7 \cdot \mathsf{opt}.$$

Theorem 3. *There is a polynomial-time randomized algorithm, that, when given a k_1-solution $\mathcal{S}_1$ and a k_2-solution $\mathcal{S}_2$ with $k_1 \le k < k_2$, outputs a feasible k-solution $\mathcal{S}$ of expected cost at most*

$$108 \cdot \left(\frac{k_2 - k}{k_2 - k_1} \cdot \mathsf{cost}(\mathcal{S}_1) + \frac{k - k_1}{k_2 - k_1} \cdot \mathsf{cost}(\mathcal{S}_2) \right) .$$

Theorem 1 follows by a standard combination of the above two theorems, analogously to some algorithms for k-median and k-means. We prove it in Appendix A.1.

2.2 The LMP approximation

As previous works on k-median and k-means, we first develop a Lagrange Multiplier Preserving subroutine for the min-sum k-clustering problem (Theorem 2). We will use a dual fitting framework with a uniform dual growth rate, similar to most primal-dual algorithms in the literature.

LP formulations. The min-sum k-clustering problem can be formulated as an (exponential-size) integer program, where we use y_S to denote whether cluster S is selected in the solution. Then the linear programming (LP) relaxation of the min-sum k-clustering problem can be written as follows:

$$\begin{aligned}
\min \quad & \sum_{S \subseteq D} \mathsf{cost}(S) y_S && && \text{(MinSum-LP)} \\
\text{s.t.} \quad & \sum_{S \subseteq D : i \in S} y_S \ge 1, && \forall i \in D, && (1) \\
& \sum_{S \subseteq D} y_S \le k, && && (2) \\
& y_S \ge 0, && \forall S \subseteq D. && (3)
\end{aligned}$$

As we see earlier in k-median and k-means, we can convert the above LP into a Lagrangian relaxation by moving Equation (2) into the objective function with a Lagrange multiplier $f \geq 0$, and convert the problem into a facility location version with a uniform opening cost f. The Lagrange relaxation of the min-sum k-clustering problem and its dual program can be written as follows:

$$\begin{aligned} \min \quad & \sum_{S \subseteq D} \mathsf{cost}(S) y_S + f \cdot \sum_{S \subseteq D} y_S \\ \text{s.t.} \quad & \sum_{S \subseteq D: i \in S} y_S \geq 1, && \forall i \in D, \\ & y_S \geq 0, && \forall S \subseteq D. \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{i \in D} \alpha_i && \text{(MinSum-Lag-Dual)} \\ \text{s.t.} \quad & \sum_{i \in S} \alpha_i \leq f + \mathsf{cost}(S), && \forall S \subseteq D, \\ & \alpha_i \geq 0, && \forall i \in D. \end{aligned}$$

Preliminaries on dual fitting for clustering. We consider two main families of dual-fitting-based approximation algorithms for clustering in the literature: the *JV-style* ones (inspired by [JV01]) and the *JMS-style* ones (inspired by [JMM+03]). At high-level, our algorithm is more similar to JV-style ones, with some ideas taken from recent work on JMS-style algorithms [CGL+25]. For both families, we maintain a dual variable α_i for each client $i \in D$, as in the Lagrange dual (MinSum-Lag-Dual). All clients have α-value 0 and are active at the beginning of the algorithm. Then, in the dual-growth phase, we grow the dual variables of all active clients until one of the following events happens (written in the language of min-sum k-clustering):

- Some cluster becomes *openable*. In this case, we mark all clients in the cluster as inactive. The definition of openable intuitively ensures that the α-values of clients in an opened cluster C can pay for the connection cost $\mathsf{cost}(C)$ plus the opening cost f, but may vary between different algorithms and will be discussed in more details later. We call clients who become inactive because of this step as *directly connected clients.*

- Some client can pay for its connection cost to an existing cluster using its α-value. A natural adaptation of prior work to the min-sum k-clustering case would lead to the condition $\alpha_j \geq 2d(j, C)$ for client j and cluster C. Indeed, $2d(j, C)$ gives the additional cost caused by adding j to C. In this paper, we will use a tighter condition $\alpha_j \geq \gamma_{IC} \cdot d(j, C)$ for a parameter $\gamma_{IC} > 2$ to create some slack that is helpful in other parts of the algorithm. We call clients who become inactive because of this step as *indirectly connected clients.*

The algorithm terminates when no active client is left. Notice that implementing the above steps in polynomial time might be non-trivial in the min-sum k-clustering case: this is discussed later.

Depending on the algorithm, each client my contribute to opening multiple clusters. In the JV-style ones, each client can only pay for one cluster, and thus the dual fitting algorithm needs an additional pruning phase to decide which cluster each client belongs to in the final solution (i.e., we need to close some clusters opened in the dual-growth phase).

Usually, the analysis of the dual fitting algorithm follows from two properties[3]:

Dual payments. Each cluster C in the solution satisfies $\sum_{i \in C} \alpha_i \geq f + \mathsf{cost}(C)$. Intuitively, this property ensures that we can charge the α-values in each cluster for its opening cost f and connection cost $\mathsf{cost}(C)$.

[3]The original analysis of Jain-Vazirani [JV01] can be fitted into establishing these two properties, while the original analysis uses something different.

Dual feasibility. For any subset $C \subseteq D$, we have $\sum_{i \in C} \alpha_i \leq f + \Gamma \cdot \mathsf{cost}(C)$ for some $\Gamma \geq 1$. Intuitively, this property ensures that α is an approximately feasible solution to the Lagrange dual (MinSum-Lag-Dual)[4].

With these two properties, the proof of Theorem 2 would be straightforward: we take summation over all clusters in our algorithm's solution for the dual payments property, and take summation over all clusters in the optimal solution for the dual feasibility property. Then we can upper bound the cost of our solution by $k \cdot f + \Gamma \cdot \mathsf{opt}$.

Barrier 1: connection cost is not stationary. One clear barrier to extending JV-style and JMS-style algorithms to min-sum k-clustering problem is that the cost of connecting a client to an existing cluster is not stationary. For example, suppose C is an existing cluster and i, j are two clients not in C. Adding i to C increases the cost of the cluster by $2d(i, C)$. However, after we add j into C, the additional cost of adding i to C becomes $2d(i, C \cup \{j\}) = 2d(i, C) + 2d(i, j)$. If active clients i, j can pay for their connection cost to C at the same time, we can only add one of them to C and thus only one of them can become inactive at that time.

This issue could break one key property widely used in the existing analysis framework: if j is inside an opened cluster C, we can upper bound α_i simply by distances related to i, j and clients in C[5]. However, because the connection cost is not stationary, the upper bound should also be related to other clients that are added to C.

New idea 1: a cluster splitting technique. To resolve this issue, we first introduce a new parameter $\gamma_{IC} \geq 2$ (threshold for indirect connection) and change the condition that a client i can pay for its connection cost to a cluster C', which is originally opened as C, to be $\alpha_i \geq \gamma_{IC} d(i, C)$. One benefit of the new condition is that the connection cost on the right-hand side is stationary, i.e., does not change when other clients are added to the cluster. Therefore, by maintaining the original cluster at the time each cluster is opened, we can independently decide whether each client becomes inactive by paying for its connection cost to an existing cluster. Throughout our algorithm, we will refer to an open cluster as the cluster it is originally opened instead of the one after adding all indirectly connected clients who joined later.

This approach however has a major issue: we are no longer able to prove the dual payment property. We resolve this issue by introducing a novel post-processing technique, *cluster-splitting*.

First, one key observation is that if there are no more than $2|C|$ clients added to C after its creation, the sum of these clients' α-values can still upper bound the increment of the cost of the cluster (for a large enough γ_{IC}). Indeed, suppose that C is the original cluster and S is the set of clients added to C later. Then

$$\begin{aligned}
\mathsf{cost}(C \cup S) - \mathsf{cost}(C) &= 2 \sum_{i \in S} d(i, C) + \sum_{i,j \in S} d(i, j) \\
&\leq 2 \sum_{i \in S} d(i, C) + \sum_{i,j \in S} \frac{d(i, C) + d(j, C)}{|C|} && \text{(triangle inequality)} \\
&= \left(2 + \frac{2|S|}{|C|}\right) \cdot \sum_{i \in S} d(i, C) \leq \sum_{i \in S} \alpha_i \ . && (\gamma_{IC} \geq 6)
\end{aligned}$$

[4]We can actually ensure that α/Γ is feasible for f by running the algorithm for $\hat{f} = \Gamma f$. In this way, dividing both side by Γ gives the LP constraints of (MinSum-Lag-Dual).

[5]This is written in the language of min-sum k-clustering. For k-median and k-means, more accurately, it should be i, j and the center of C.

Then, we need to deal with the case where more than $2|C|$ clients are added to C after its creation. A natural guarantee from the dual fitting framework is that: the α-value of any client added later is greater than that of any client in the original cluster (upon creation). If γ_{IC} is large enough, we can show that any additional $2|C|$ clients can satisfy the dual payment property themselves, without any contribution from the clients in the original cluster. The reason is simple: the total cost of a cluster C' involving these $2|C|$ clients is at most $O(1) \cdot \sum_{i \in C'} d(i, C)$ via a similar calculation as in the earlier observation. Therefore, each client $i \in C'$ can contribute at least half of its α-value to the opening cost on average if γ_{IC} is large enough. This inspires us for the novel *cluster splitting* technique: we keep splitting $2|C|$ clients added to the cluster later away from the original cluster to form a new cluster, until there are less than $2|C|$ clients added to the cluster later. We formalize this process in Algorithm 1 and show the dual payment property of the new clusters in Lemma 5.

Barrier 2: No "constant-factor" triangle inequality on the connection cost. Both JV-style and JMS-style algorithms use in their analysis triangle inequality, or at least an approximate version of it. More specifically, they use the fact that the additional connection cost of adding i to a cluster S already containing j is at most the connection cost of j plus the connection cost of i and j in any other cluster C, up to a constant factor. In the language of min-sum k-clustering, suppose that j is in cluster S. Then, directly using the previous analytic framework would require to upper bound $d(i, S)$ by $O(1) \cdot (d(i, C) + d(j, C) + d(j, S))$. However, as already pointed out in Bartal, Charikar, and Raz [BCR01] and Behsaz, Friggstad, Salavatipour, and Sivakumar [BFSS19], this cannot be true when $|S| \gg |C|$, because we need an unbounded factor of $\frac{|S|}{|C|}$ before the terms $d(i, C)$ and $d(j, C)$:

$$d(i, S) \leq \frac{|S|}{|C|}\big(d(i, C) + d(j, C)\big) + d(j, S) \ . \tag{4}$$

This barrier even widely exists in generalizing the existing dual fitting approaches, and has received a lot of discussions in the literature (e.g., [BCR01; BFSS19]). For example, it exposes a clear challenge in designing a different pruning phase for JV-style algorithms.

In more detail, in a natural adaptation of JV-style algorithms, there is an initial dual-growth phase where certain clusters are tentatively opened. A cluster C is defined as openable if we have $\sum_{i \in C} \alpha_i = f + \gamma_{IC} \cdot \mathsf{cost}(C)$. After all clients become inactive, there is a second cluster pruning phase where we close a subset of the clusters. In particular, if the same client contributed to opening two clusters, those clusters are in conflict and one of them must be closed. This way, each client only lies in at most 1 open cluster. The clients whose original cluster was closed due to a conflict, are reassigned to an open cluster causing such closure. To prove the properties of dual payment and feasibility at the same time, the original analysis of [JV01] upper bounds the new connection cost of each client i whose clusters are all closed by $O(1) \cdot \alpha_i$. However, for min-sum k-clustering, when clusters with completely different sizes intersect, it is not clear how we can establish this upper bound by closing some of them.

First, we explain why we cannot close a cluster due to a conflict with another open cluster with a much larger size. Suppose that $|C| \ll |S|$ and S closes C because of $j \in S \cap C$. For clients $i \in C$ whose dual variable $\alpha_i = \Theta(d(i, C))$, we cannot simply upper bound $d(i, S)$ by $O(1) \cdot \alpha_i$ because of Equation (4).

On the other hand, we cannot close multiple clusters due to a conflict with the same cluster of much smaller size. Indeed, suppose that we wish to close S and S' due to the opening of C, where $|S| \approx |S'| \gg |C|$. It is possible that the α-values of clients in $(S \cup S') \cap C$ contribute a large

amount of the opening cost to open C (though it might be $o(f)$), and clients in S and S' have much larger inter-cluster distance than their respective intra-cluster distances. In this case, if we add all clients in S, S' to C, the new connection cost for $i \in S$ would be $d(i, C \cup S \cup S')$, which is greater than $d(i, S') \gg d(i, S)$. If $\alpha_i = \Theta(d(i, S))$, we cannot upper bound the new connection cost of i by $O(1) \cdot \alpha_i$. We shall remark that this barrier still exists with our new cluster splitting technique, because clients in $S \setminus C$ and $S' \cap C$ can fall short of opening a new cluster[6].

It turns out that the same kind of difficulties are encountered by using variants of the dual-fitting algorithm JMS in [JMM+03]

We anticipate that we also need to circumvent this barrier in the bipoint rounding part, that is discussed later (see Subsection 2.3).

New idea 2: dual fitting with dependencies. To circumvent this major barrier to min-sum k-clustering, we augment the existing dual-fitting framework with a new algorithmic ingredient: only clusters whose membership is preserved by a carefully-defined dependency relation are potential openable clusters. With this ingredient, our algorithm looks more similar to JV than JMS, using an open condition of $\sum_{i \in C} \alpha_i \geq f + \gamma_{DC} \mathsf{cost}(C)$ for some parameter $\gamma_{DC} \geq \gamma_{IC}$[7] and has a pruning phase after making every client inactive. On the other hand, because we do not consider any cluster that the dual variables are able to pay for as a candidate openable cluster, it is more challenging to prove the dual feasibility property with this new ingredient. Next, we will discuss some details of implementing this idea and explain how this resolves the earlier challenge in designing a pruning phase. Later, we will discuss our new analytic framework for the dual feasibility property and how some parameters in the algorithm help with the analysis.

More specifically, we consider a directed dependency graph (D, E), and a natural preorder induced by it: for $i, j \in D$, we have $i \preceq j$ if and only if there is a directed path from i to j. We say that membership is preserved with respect to the preorder for S if $i \in S$ and $i \preceq j$ together imply $j \in S$. We introduce two parameters $\gamma_{DC} \gg \gamma_{IC}$ and $r_S \gg 1$, whose choice is clarified later. In the dual-growth phase, we consider a cluster S as openable if the following five conditions hold:

1. Total dual contribution is enough: $\sum_{j \in S} \alpha_j \geq f + \gamma_{DC} \cdot \mathsf{cost}(S)$.
2. Individual dual payment to the connection cost: $\forall i \in S, \alpha_i \geq \gamma_{DC} d(i, S)$.
3. The size of the cluster $|S|$ is small enough: it is at most $\frac{1}{r_S}$ times the size of any previously opened cluster that intersects it.
4. Membership is preserved with respect to the preorder.
5. The sum of α-values of active clients in S can be upper bounded by $O(f)$.

We already mentioned the first two natural properties. The remaining three properties play a role in the analysis that will become clearer later. We anticipate that conditions 2-4 are the key to our novel pruning phase, while conditions 2, 3 and 5 will be significantly used to prove the dual feasibility.

With this definition of openable clusters, we update the dependency graph in the dual-growth phase as follows: let $D_A(S)$ be the set of clients in S that are still active right before S is opened; after opening S, we add a directed edge from every client $i \in S$ to every active client $j \in D_A(S)$.

[6] In the analysis of cluster splitting, we need clients in S have larger α-values than clients in C, which does not hold here.

[7] The use of a new γ_{DC} different from γ_{IC} is inspired by the variant of JMS in [CCG+25]. However, perhaps surprisingly, the reason we need this parameter is completely different

This update rule guarantees that: if one client in the current cluster S is part of a later opened cluster, then all clients in $D_A(S)$ will be part of that cluster. This gives us a new intuition about the design of the new pruning phase, assuming that every client becomes inactive when we open a cluster in the dual-growth phase:

- We decide whether to close clusters in reverse order of their opening time.
- If a cluster S does not intersect a later opened cluster S' that is not closed by us, we can safely leave S open; otherwise, we close S. Note that, in the latter case, we can guarantee that $D_A(S) \subseteq S'$, meaning that all clients in $D_A(S)$ are already in that intersecting not-closed cluster S'. No matter which case applies, this ensures that our decision will not leave any client in $D_A(S)$ disconnected after the pruning phase.
- Because the rest of the clients in $S \setminus D_A(S)$ are inside other clusters opened earlier, no matter we open S or not, our later decision for earlier-opened clusters can make sure the rest of clients are also connected to some cluster after the pruning phase.

The remaining issue we need to handle is that some client i becomes inactive because its α-value reaches $\gamma_{IC} \cdot d(i, S)$ for some opened cluster S. Recall that we call them indirectly connected clients. Because of condition 3 of opened cluster, we can show by a simple calculation that α_i could pay for its new connection cost if we re-route it to the cluster S' that closes S. However, we may still run into the same issue we encountered earlier when discussing why we cannot close two large clusters by a small cluster, when there are too many such clients with the same cluster S. To resolve this issue, we do the following: we generalize the cluster splitting technique to a *cluster editing* technique, which enables us to delete several clients from a later opened cluster S' that are not closed, while inserting an amount of $\approx 2|S'|$ indirectly connected clients connected to S into S'. The reason we can do this is: half of the inserted clients' total α-values compensates for the contribution of the removed clients, while the remaining half pays the additional connection cost. In this way, if the number of such indirectly connected clients is too large, we can reopen S, by moving its clients from not-closed intersecting clusters S' back to S. The analysis for this more general cluster editing technique is similar to that for cluster splitting, and we formalize it in Definition 6 and Lemma 7.

One additional issue we have to resolve (for the dual-growth phase) is that there can be an exponential number of clusters, and it is inefficient to verify the conditions for all clusters. In Algorithm 2, we give a polynomial-time algorithm that finds one such openable cluster once there exists a cluster that satisfies a slightly stronger condition 1 (up to a constant factor on the $\gamma_{DC} \cdot \mathsf{cost}(S)$ term), as well as conditions 3 and 4. The algorithm mainly uses the connection between min-sum k-clustering and balanced k-median introduced by [BCR01] and some simple cluster-cleaning argument. We shall remark that the actual implementation Algorithm 3 uses a slightly different and more notation-heavy dependency forest that follows the same intuition and makes the technical proof easier. Condition 3 for the openable clusters is also slightly changed.

New idea 3: a new analysis framework for the dual feasibility. Because our algorithm does not examine whether the dual variables of every set are able to pay for the opening cost plus the connection cost (the key for [JV01]'s analysis of the dual feasibility), we need to resolve this challenge via some new analysis framework. Perhaps surprisingly, our new framework is partly inspired by the analysis of JMS-style algorithm in [CGL+25].

To prove dual feasibility, a simple observation is that we only need to consider all Γ-critical clusters C, i.e., the ones such that every client $i \in C$ has $\alpha_i \geq 2\Gamma d(i, C)$. Our analysis is done by

finding contradictions when assuming that a Γ-critical cluster C violates $\sum_{i\in C}\alpha_i \le f + \Gamma\mathsf{cost}(C)$. The main idea underlying the new analysis framework is that we can erase some clients from C and add some additional clients into it to preserve the membership with respect to the preorder, at some carefully-chosen time point of the dual-growth stage. To make this idea work, we choose the parameters so that $r_S \gg 1$ and $\gamma_{DC} \approx r_S\gamma_{IC}$, and consider a even larger $\Gamma \approx r_S\gamma_{DC}$. The choice of these parameters result in the following observations and proof sketches that lead to a contradiction (where we don't consider $r_S, \gamma_{IC}, \gamma_{DC}$ and Γ as constants in big-O notations):

1. *Observation (Corollary 24):* Because $r_s \gg 1$ and Conditions 2-3 of openable clusters, for any $i \preceq j$, i is close to any open cluster S involving j with respect to α_j, i.e., $\alpha_j = \Omega(\gamma_{DC} d(i,S))$. *Implication:* It is a basic fact used almost everywhere in the proof.

2. *Observation (Subsection 4.5.1):* Because $\gamma_{DC} = \Omega(r_S\gamma_{IC})$, when a cluster S, which intersects C has a size $\le r_S|C|$, is opened, all clients in C become inactive immediately afterwards. *Implication:* Let $\theta = \max_{i\in C}\alpha_i$. We choose the time as $\theta - \varepsilon$ for small enough $\varepsilon > 0$ so that our algorithm does nothing between time $[\theta-\varepsilon, \theta)$. Then at time $\theta - \varepsilon$, no such cluster S has been opened, which will be widely used for the rest of the observations. We will consider time $\theta - \varepsilon$ later on. Analysis of the log-adaptive part of [CGL+25] inspires this observation.

3. *Observation (Lemma 25 and Equation (22)):* Because $\Gamma = \Theta(r_S\gamma_{DC})$ and Conditions 2-3 of openable clusters, for any two clients $i \in C$ and $j \in D$, if $i \preceq j$, then $\alpha_j \ge \Omega(\Gamma) d(j, C)$. *Implication:* If we add j into C, the resulting C violating the dual feasibility $\sum_{i\in C}\alpha_i \le f + \Gamma'\mathsf{cost}(C)$ for a slightly smaller Γ'. This could also hold if we add $O(|C|)$ such client j, using a similar argument we used for cluster splitting.

4. *Observation (Subsection 4.5.3):* We define an open cluster S as well-behaved if $|D_A(S)| = \omega(|S|/r_S)$. We can show that if a client $i \in C$ is close to a well-behaved cluster, we can simply upper bound $\sum_{i\in C}\alpha_i$ by f, because of condition 5 of openable clusters. *Implication:* because C violates the dual feasibility property (our assumption), for every $i \in C$ and $j \in D$ such that $i \preceq j$, j is not opened in any well-behaved cluster.

5. *Observation (Subsection 4.5.2):* We say a set S is available, if its membership is preserved with respect to the preorder (Condition 4 of openable clusters). We can show that at any time point of the dual-growth phase, any available set satisfies the dual feasibility for $4\gamma_{DC}$. *Implication:* To reach a contradiction, we only need to find an available set at some time of the dual-growth phase.

6. *A sketch towards constructing a violating available set (Subsection 4.5.4):* Because of our update rule for the dependency graph, the set $\{j \in D : \exists i \in C, i \preceq j\}$ (the minimal set including C whose membership is preserved) can be partitioned by $D_A(S_1), \ldots, D_A(S_r)$ for opened clusters $S_1, \ldots, S_r$, where we order them in reverse order of their opening time.

   Then, roughly speaking, we can find an available set at time $\theta - \varepsilon$ by: (i) finding the smallest $p \in \{1, \ldots, r\}$ such that $|D_A(S_1)| + \ldots |D_A(S_p)| > 2|C|$ (if no such p exists, we let $p = r$); (ii) let the cluster be $D_A(S_1) \cup \cdots \cup D_A(S_p)$.

   We then can use Observation 3 to prove that all clients in $D_A(S_1) \cup \cdots \cup D_A(S_p)$ has a small distance to C. Note that $|D_A(S_1)| + \cdots + |D_A(S_{p-1})| < 2|C|$. Applying a similar argument used for cluster splitting, we can show that $D_A(S_1) \cup \cdots \cup D_A(S_{p-1}) \cup C$ violates the dual feasibility for a smaller factor $\approx \Gamma'$ (as defined in Observation 2).

The main challenge here is to deal with $D_A(S_p)$, which may be much larger than C. We decompose the additional cost of adding $D_A(S_p)$ to $D_A(S_1) \cup \cdots \cup D_A(S_{p-1}) \cup C$ into two parts: the distance between $D_A(S_1) \cup \ldots D_A(S_{p-1}) \cup C$ and $D_A(S_p)$, and the pairwise distance in $D_A(S_p)$. We first show that the distance between $D_A(S_1) \cup \ldots D_A(S_{p-1}) \cup C$ and $D_A(S_p)$ can be upper bounded by adding some more sophisticated steps in the construction of the openable cluster.[8] Second, because S_p is not a well-behaved cluster by Observation 4, we can upper bound the pairwise distance in $D_A(S_p)$ by $O(1/r_S) \cdot \sum_{i \in D_A(S_p)} \alpha_i$.

Finally, applying a similar argument used for cluster editing, we can prove the available set $D_A(S_1) \cup \cdots \cup D_A(S_p)$, which can be viewed as removing some clients in C from $D_A(S_1) \cup \cdots \cup D_A(S_{p-1}) \cup C$ and then adding $D_A(S_p)$, violates the dual feasibility with an even smaller $\Gamma'' > 4\gamma_{DC}$, contradicting Observation 5.

2.3 Bipoint rounding

In the bipoint rounding problem, we are given two partitions $\mathcal{S}_1$ and $\mathcal{S}_2$ of the same client set D, with k_1 and k_2 clusters, respectively, where $k_1 \leq k < k_2$. Our goal is to obtain a solution with at most k clusters and bounded cost. In particular, we want to prove the following theorem.

Theorem 3. *There is a polynomial-time randomized algorithm, that, when given a k_1-solution $\mathcal{S}_1$ and a k_2-solution $\mathcal{S}_2$ with $k_1 \leq k < k_2$, outputs a feasible k-solution $\mathcal{S}$ of expected cost at most*

$$108 \cdot \left(\frac{k_2 - k}{k_2 - k_1} \cdot \mathsf{cost}(\mathcal{S}_1) + \frac{k - k_1}{k_2 - k_1} \cdot \mathsf{cost}(\mathcal{S}_2) \right) .$$

Let $C_i(j)$ denote the cluster containing j in $\mathcal{S}_i$. Let also $\alpha = \frac{k_2 - k}{k_2 - k_1}$. Then $\alpha k_1 + (1 - \alpha) k_2 = k$. The LMP part of our approach gives a bound on

$$\mathsf{opt}_{\text{bipoint}} = \alpha \mathsf{cost}(\mathcal{S}_1) + (1 - \alpha) \mathsf{cost}(\mathcal{S}_2).$$

We want the expected cost of our solution to be at most a constant times $\mathsf{opt}_{\text{bipoint}}$.

If $\alpha > 1/2$, we can simply return $\mathcal{S}_1$. It has at most k clusters and cost at most $2\mathsf{opt}_{\text{bipoint}}$. We can therefore assume that $\alpha \leq 1/2$. Since $1 - \alpha \geq 1/2$, we have $\mathsf{cost}(\mathcal{S}_2) \leq 2\mathsf{opt}_{\text{bipoint}}$. Thus, we can charge a constant multiple of $\mathsf{cost}(\mathcal{S}_2)$ directly to $\mathsf{opt}_{\text{bipoint}}$. To charge a contribution from $\mathsf{cost}(\mathcal{S}_1)$, we need the factor α.

A first attempt at rounding. Let us first try a rounding similar to the bipoint rounding for k-median [JV01]. We aim to open clusters only from $\mathcal{S}_2$, closing $k_2 - k$ of them. Suppose each cluster $A \in \mathcal{S}_1$ has chosen a cluster $\ell_M(A) \in \mathcal{S}_2$ as its *backup destination*. Choosing these labels is one of the hurdles, and we will build towards this choice. A client of A whose cluster in $\mathcal{S}_2$ closes will use $\ell_M(A)$. For now, we leave the choice of these labels unspecified.

Given the labels, we open clusters as follows.

1. We open every cluster in $\{\ell_M(A) \mid A \in \mathcal{S}_1\}$.
2. Among the remaining clusters of $\mathcal{S}_2$, we choose $k_2 - k$ uniformly at random to close and open the others.

[8] For the sake of presenting an intuition, we skip this technical detail in the overview section. In Subsection 4.5.4, we provide a full version of the process in the paragraph **Constructing an available set at time θ.**.

Let t be the number of distinct clusters opened in the first step. Since $t \le k_1 < k$, at least $k_2 - k$ clusters remain available to close in the second step. The total number of clusters we open is $t + (k_2 - t) - (k_2 - k) = k$.

We assign each client j as follows.

1. If $C_2(j)$ opens, we assign j to it.
2. Otherwise, we *reassign* j to $\ell_M(C_1(j))$.

Every backup destination is open, so this assigns each client to an open cluster.

The key observation is that a cluster of $\mathcal{S}_2$ does not close too often. For any fixed choice of the labels, every backup destination stays open, and every other cluster closes with probability

$$\frac{k_2 - k}{k_2 - t} \le \frac{k_2 - k}{k_2 - k_1} = \alpha. \tag{5}$$

Let us now bound the cost of this assignment. For each opened cluster $S \in \mathcal{S}_2$, the pairs of original clients contribute $\mathsf{cost}(S)$. Summing over all opened clusters gives at most $\mathsf{cost}(\mathcal{S}_2) \le 2\mathsf{OPT}_{\text{bipoint}}$. The new cost comes from reassigned clients.

Let T_i be the i-th cluster in $\mathcal{S}_1$. For each destination $S \in \mathcal{S}_2$, let $Z_i \subseteq T_i \setminus S$ be the clients reassigned from T_i to S, and let $Z = \bigcup_i Z_i$. For $A, B \subseteq D$, let $d(A, B) := \sum_{v \in A} d(v, B)$. If S opens, its final client set is $S \cup Z$, so its additional cost is

$$\begin{aligned} \mathsf{cost}(S \cup Z) - \mathsf{cost}(S) &= 2d(S, Z) + d(Z, Z) \\ &= 2\sum_i d(S, Z_i) + \sum_i d(Z_i, Z_i) + \sum_{i \ne j} d(Z_i, Z_j). \end{aligned} \tag{6}$$

Consider the right-hand side of (6). The first term pairs original clients with arriving clients. The second pairs clients arriving from the same cluster of $\mathcal{S}_1$. The third pairs clients arriving from different clusters of $\mathcal{S}_1$. If S closes, all its sets Z_i are empty, so we can sum these terms over all $S \in \mathcal{S}_2$.

For now, let us set aside the third term. If the labelling were a matching, so that at most one cluster of $\mathcal{S}_1$ chose each $S \in \mathcal{S}_2$, this term would be zero. We will return to what happens when two clusters choose the same destination later.

We first consider $d(S, Z_i)$. Suppose T_i has chosen S as its backup destination. A client $v \in T_i$ is reassigned to S only when $C_2(v)$ is closed. By Equation (5), this happens with probability at most α. Hence,

$$\mathbb{E}[d(S, Z_i)] = \sum_{v \in T_i} d(S, v) \Pr[v \in Z_i] \le \alpha d(S, T_i).$$

Thus, we need to charge $d(S, T_i)$ to distances contributing to $\mathsf{cost}(S)$ and $\mathsf{cost}(T_i)$. Suppose $R_i = S \cap T_i$ is nonempty. For $u \in S$, $v \in T_i$, and $r \in R_i$, the triangle inequality gives $d(u, v) \le d(u, r) + d(r, v)$. Averaging over r and summing over u gives

$$d(S, v) \le \frac{d(S, R_i)}{|R_i|} + \frac{|S|}{|R_i|} d(v, R_i). \tag{7}$$

Summing Equation (7) over $v \in T_i$, we obtain

$$d(S, T_i) \le \frac{|T_i|}{|R_i|} \left(d(S, R_i) + \frac{|S|}{|T_i|} d(T_i, R_i) \right). \tag{8}$$

The distances $d(S, R_i)$ lie within S, and the distances $d(T_i, R_i)$ lie within T_i. However, there are two difficulties in charging this bound to the input costs.

- **Difficulty 1:** The intersection R_i may be much smaller than T_i. The factor $|T_i|/|R_i|$ can therefore be large, multiplying both terms.
- **Difficulty 2:** The destination S may be much larger than the source T_i. Even if we handle the first factor, the second term still contains the potentially large ratio $|S|/|T_i|$.

Idea 1. Choosing backup destinations. To address Difficulty 1, we choose the destination S with probability $|R_i|/|T_i|$. Multiplying the bound in Equation (8) by this probability cancels the factor $|T_i|/|R_i|$.

Independently for each T_i, we sample a uniform client v and choose its cluster $C_2(v) \in \mathcal{S}_2$ as $\ell_M(T_i)$. For a fixed destination $S \in \mathcal{S}_2$, let $p_i = \Pr[\ell_M(T_i) = S] = |R_i|/|T_i|$. A client $v \in T_i$ arrives at S only if T_i chooses S and $C_2(v)$ closes. For every choice of backup destinations, the latter event has probability at most α. Applying this bound and then averaging over the choices gives

$$\Pr[v \in Z_i] \leq \alpha \Pr[\ell_M(T_i) = S] = \alpha p_i. \tag{9}$$

The expectation now includes both the random labelling and the random choice of clusters to close. By linearity of expectation, $\mathbb{E}[d(S, Z_i)] \leq \alpha p_i d(S, T_i)$. Multiplying Equation (8) by p_i cancels the factor $|T_i|/|R_i|$, giving

$$p_i d(S, T_i) \leq d(S, R_i) + \frac{|S|}{|T_i|} d(T_i, R_i). \tag{10}$$

If R_i is empty, then $p_i = 0$, no clients of T_i arrive at S, and both sides of Equation (10) are zero.

The first term in (10) can be charged to the cost of S. In fact, the R_i partition S, so summing Equation (10) over i gives

$$\sum_i p_i d(S, T_i) \leq \mathsf{cost}(S) + \sum_i \frac{|S|}{|T_i|} d(T_i, R_i). \tag{11}$$

The second term uses distances within T_i, but we still need to handle the ratio $|S|/|T_i|$.

Idea 2: Good clients. To address Difficulty 2, we would like the destination S of T_i to have size at most $|T_i|$. For this, we call a client j *good* if $|C_1(j)| \geq |C_2(j)|$, and *bad* otherwise.

The case when every client is good. Suppose first that every client is good. We keep this assumption until the discussion of bad clients below. Any client in R_i belongs to T_i in $\mathcal{S}_1$ and to S in $\mathcal{S}_2$. Thus, $|S| \leq |T_i|$ whenever R_i is nonempty. The second term of Equation (10) is therefore at most $d(T_i, R_i)$, which is part of $\mathsf{cost}(T_i)$. We have now obtained $\mathbb{E}[d(S, Z_i)] \leq \alpha\big(d(S, R_i) + d(T_i, R_i)\big)$.

Algorithm for the all-good case. Combining the two ideas with our first rounding gives the following algorithm when all clients are good.

1. For each $T_i \in \mathcal{S}_1$, we independently sample a uniform client and let $\ell_M(T_i)$ be its cluster in $\mathcal{S}_2$.
2. We open every cluster selected by these labels. Among the remaining clusters of $\mathcal{S}_2$, we choose $k_2 - k$ uniformly to close and open the others.
3. We keep a client j in $C_2(j)$ if it opens. Otherwise, we reassign j to $\ell_M(C_1(j))$.

As shown in the initial counting argument, we open k clusters from $\mathcal{S}_2$, and every client has an open destination.

Let us now finish the cost analysis of this algorithm. We start with the first term in Equation (6), which pairs original clients with arriving clients. The two ideas above bound this term for each destination S. We now sum these bounds over all destinations. Since the R_i partition S, the terms $d(S, R_i)$ sum to $\mathsf{cost}(S)$. For a fixed T_i, its intersections with the clusters of $\mathcal{S}_2$ partition T_i, so

$$\sum_{S \in \mathcal{S}_2} \frac{|S|}{|T_i|} d(T_i, S \cap T_i) \le \sum_{S \in \mathcal{S}_2} d(T_i, S \cap T_i) = \mathsf{cost}(T_i). \tag{12}$$

Using Equations (9) and (11), summing over S, and including the factor two from Equation (6), we obtain the first cost bound

$$2 \sum_{S \in \mathcal{S}_2} \mathbb{E}\left[\sum_i d(S, Z_i)\right] \le 2\alpha \mathsf{cost}(\mathcal{S}_2) + 2\alpha \mathsf{cost}(\mathcal{S}_1). \tag{13}$$

The two ideas have therefore bounded the cost between original clients and arriving clients. The factor α comes from the fact that a client is reassigned only when its original cluster in $\mathcal{S}_2$ closes. However, we are still not done. We have only bounded the first term in Equation (6). We need to bound the other two as well.

First, we bound the cost of arrivals from the same cluster of $\mathcal{S}_1$. For $u, v \in T_i$, both clients arriving requires u to arrive. Thus, Equation (9) gives

$$\Pr[u, v \in Z_i] \le \alpha p_i, \qquad \mathbb{E}[d(Z_i, Z_i)] \le \alpha p_i \mathsf{cost}(T_i). \tag{14}$$

Since $\sum_{S \in \mathcal{S}_2} |S \cap T_i|/|T_i| = 1$, summing over S and i gives

$$\sum_{S \in \mathcal{S}_2} \mathbb{E}\left[\sum_i d(Z_i, Z_i)\right] \le \alpha \mathsf{cost}(\mathcal{S}_1). \tag{15}$$

Next, we bound the cost of arrivals from different clusters of $\mathcal{S}_1$. The bounds in Equations (13) and (15) still hold when several clusters choose the same S. For $u \in T_i$, $v \in T_j$, and $i \neq j$, independence of the labels and the closure bound in Equation (5), applied after the labels are chosen, give

$$\Pr[u \in Z_i,\ v \in Z_j] \le \alpha p_i p_j. \tag{16}$$

For nonempty R_i, R_j, averaging $d(u, v) \le d(u, r) + d(r, v)$ over $r \in R_i$ and summing over u, v gives

$$p_i p_j d(T_i, T_j) \le p_j d(R_i, T_j) + \frac{|R_j|}{|T_i|} d(T_i, R_i). \tag{17}$$

Empty intersections contribute zero. Since the R_i partition S, summing and using Equation (11) gives

$$\mathbb{E}\left[\sum_{i \neq j} d(Z_i, Z_j)\right] \le \alpha \sum_{i \neq j} p_i p_j d(T_i, T_j) \le \alpha \mathsf{cost}(S) + 2\alpha \sum_i \frac{|S|}{|T_i|} d(T_i, R_i).$$

Summing over S and applying Equation (12) gives

$$\sum_{S \in \mathcal{S}_2} \mathbb{E}\left[\sum_{i \neq j} d(Z_i, Z_j)\right] \le \alpha \mathsf{cost}(\mathcal{S}_2) + 2\alpha \mathsf{cost}(\mathcal{S}_1). \tag{18}$$

Adding $\mathsf{cost}(\mathcal{S}_2)$ for the original clients to the three bounds in Equations (13), (15) and (18), we obtain the following guarantee when every client is good and $\alpha \leq 1/2$.

$$\mathbb{E}[\mathsf{cost}(\mathcal{S})] \leq 5\mathsf{opt}_{\text{bipoint}}.$$

Handling Bad clients In the above analysis, we crucially used the property of good clients: whenever a good client moved from T_i to S, we used the fact that $|T_i| \geq |S|$, and this was crucial in bounding the cost of the re-routing by triangle inequality.

If not all clients are good, then this property may not hold, and this poses a significant challenge for us. If a bad client's $\mathcal{S}_2$ cluster is closed, it cannot reroute from the $\mathcal{S}_1$ cluster via our labeling, as the size blow-up $\frac{|S|}{|T_i|}$ can be huge. This is where we depart from the usual k-median algorithms: *we allow opening clusters of $\mathcal{S}_1$ as well*. On the one hand, this seems reasonable, as we can hope to serve bad clients whose $\mathcal{S}_2$ cluster is closed using their $\mathcal{S}_1$ cluster directly, without re-routing via the labeling. But this creates two problems:

- We cannot simply guarantee that, for every bad client, either its $\mathcal{S}_1$ cluster or its $\mathcal{S}_2$ cluster is open.
- We still need every $\mathcal{S}_2$ cluster to be closed with probability only $O(\alpha)$. The earlier cost analysis uses this probability bound, and in particular we can afford contributions from $\mathsf{cost}(\mathcal{S}_1)$ only with an $O(\alpha)$ factor.

Idea 3: A Reverse labelling. To solve the first problem, we introduce a labeling in the other direction as well. For each $S \in \mathcal{S}_2$ containing bad clients, we sample a bad client j uniformly at random from S, and let $\ell_L(S) = C_1(j)$, which gives S a *backup cluster* in $\mathcal{S}_1$. If S closes, we send all its bad clients to $\ell_L(S)$. This need not be the original $\mathcal{S}_1$ cluster of every bad client in S, it is the cluster of the one bad client sampled to define the label. This sampled client lies in both S and $\ell_L(S)$, and this overlap lets us bound the rerouting cost later, just like in the good client case. Notice that this creates a dependency: if there is a bad client j for which $C_2(j)$ is not open, then $\ell_L(C_2(j))$ must be open. We must ensure this while still opening k clusters, and still ensuring that each cluster of $\mathcal{S}_2$ is closed with probability at most $O(\alpha)$.

To achieve this, for every $T \in \mathcal{S}_1$ such that $T = \ell_L(S_2)$ for some $S_2 \in \mathcal{S}_2$, we choose one *representative* cluster from $\ell_L^{-1}(T)$ at random, and call it $M(T)$. Then we obtain pairs $(T, M(T))$, and we will open *exactly one of T or $M(T)$*. Among the non-representative $\mathcal{S}_2$ clusters available to close, we close an $O(\alpha)$ fraction of them. Thus, after the representatives have been chosen, each such cluster closes with probability at most $\frac{k_2-k}{k_2-k_1} = \alpha$.

Finally, it remains to decide for each pair $(T, M(T))$ when to open T and when to open $M(T)$. This decision is made as follows. First, if $T = \ell_L(S_2)$ for some $S_2 \in \mathcal{S}_2$ that is a non-representative which is closed (recall that we randomly close an $O(\alpha)$ fraction of non-representatives) it is clear that we must open T. The bad clients of S_2 need T as their backup.

If no such S_2 exists, we have the choice to open either. This part of our algorithm becomes more technical, as there is a tension between two factors. First, if we open T instead of $M(T)$, we need to worry about keeping the probability that $M(T)$ is closed to be at most $O(\alpha)$. But if we open $M(T)$ instead of T, then the good clients in T need to be re-routed (we note that after our modification in the algorithm for handling bad clients, we can no longer guarantee that $\ell_M(T)$ is open). This is where we use a third way to re-route, which involves neither ℓ_M nor ℓ_L. For this rule, we only count good clients of T whose $\mathcal{S}_2$ clusters were not fixed to stay open before we chose the representatives. If at least a $\frac{1}{3}$ fraction of good clients in a cluster $T \in \mathcal{S}_1$ are *settled*, that is,

their $\mathcal{S}_2$ clusters are already open, then the remaining good clients can re-route to an open cluster $C_2(j)$ through a settled client $j \in T$ without paying too much additional cost , and hence we can open $M(T)$.

When rerouting is needed, we choose one settled good client uniformly at random and send the unsettled good clients to its $\mathcal{S}_2$ cluster. We sample from at least one third of the good clients being counted, so the probability of choosing any particular destination is at most three times its fraction among those clients. This lets us use the same kind of cost bound as before.

If fewer than a $\frac{1}{3}$ fraction are settled, we open T. We apply this process sequentially, ordering the clusters T in increasing order of size, deciding for each T based on these considerations which of T and $M(T)$ is opened. Importantly, this order ensures that when we process T and ask whether a good client of T is settled, we already know whether its $\mathcal{S}_2$ cluster S is open. To see this, note that if S was a non-representative, this was decided in the initial random choice. Otherwise, let $T' = \ell_L(S)$. The good client in $T \cap S$ gives $|S| \leq |T|$, while the bad client sampled to define $\ell_L(S) = T'$ gives $|T'| < |S|$. Hence $|T'| < |S| \leq |T|$, so the pair containing $S = M(T')$ has already been processed.

The final step is to show that this probabilistic process closes every cluster $S_2 \in \mathcal{S}_2$ with probability at most $O(\alpha)$. For non-representative clusters, this follows easily, since we randomly close an $O(\alpha)$ fraction of them. For representative clusters in $\mathcal{S}_2$, this is far from straightforward. However, we are able to show, via a coupling argument, that even for these clusters, the probability of being closed is $O(\alpha)$. This problem turns out to be a generalization of the Erdős–Ko–Rado theorem [EKR61], whose proof we adapt to our more general setting.

2.4 Organization of the paper

The rest of the paper is organized as follows. In Section 3 we describe our cluster splitting and editing techniques, and prove some of their properties that we will need later. In Section 4 we describe our LMP approximation algorithm. Our bipoint rounding procedure is described in Section 5. Some omitted proofs are given in Section A.

3 Cluster Splitting and Editing

In this section, we present our novel cluster splitting and editing techniques for min-sum k-clustering. Throughout the section, we suppose that $\{\alpha_i\}_{i \in D}$ is any dual solution and $f > 0$ is the uniform opening cost of a cluster.

3.1 Cluster Splitting

We first begin with the definition of γ-splittable clusters. We characterize a γ-splittable by two: the core part C_{core} and the outer part C_{out}, corresponding to the cluster immediately after the creation and the set of clients added to the cluster later. Three conditions should then hold: (1) two parts are disjoint; (2) the core part satisfies the dual payment property; (3) the α-value of each client in the outer part can pay for its own connection cost by γ times, and is greater than any of the α-value in the core part.

Definition 4 (γ-splittable clusters)**.** *A sequence of sets $\langle C_{\text{core}}, C_{\text{out}} \rangle$ is said to be a γ-splittable cluster with respect to the dual solution $\{\alpha_i\}_{i \in D}$ if*

1. *$C_{\text{core}} \neq \emptyset$, and C_{core} and C_{out} are disjoint, i.e., $C_{\text{core}} \cap C_{\text{out}} = \emptyset$.*

2. $\sum_{i \in C_{\text{core}}} \alpha_i \geq f + \mathsf{cost}(C_{\text{core}})$.

3. *For each* $i \in C_{\text{out}}$, *we have* $\alpha_i \geq \gamma \sum_{j \in C_{\text{core}}} d(i,j)$ *and* $\alpha_i \geq \max_{j \in C_{\text{core}}} \alpha_j$.

In the following lemma, we prove that Algorithm 1 can split each γ-splitable cluster into parts that satisfy the dual payment property. In the algorithm, we repeatedly split $2|C_{\text{core}}|$ clients from C_{out} to create a new cluster until there are not enough clients to split.

Algorithm 1 (Cluster Splitting).

Initialization: γ-splittable cluster $\langle C_{\text{core}}, C_{\text{out}} \rangle$. Initialize $r = 1$, $C' \leftarrow C_{\text{out}}$.

While $|C'| > 2|C_{\text{core}}|$:

1. we set $r \leftarrow r + 1$ and find an arbitrary subset $C_r \subseteq C'$ of size $|C_r| = 2|C_{\text{core}}|$;
2. we then remove C_r from C' and repeat.

Let $C_1 \leftarrow C_{\text{core}} \cup C'$ and return the clusters $C_1, \ldots, C_r$.

Lemma 5. *Suppose that* $\gamma \geq 8$. *There exists a polynomial-time algorithm that for any* γ*-splittable cluster* $\langle C_{\text{core}}, C_{\text{out}} \rangle$, *it partitions* $C = C_{\text{core}} \cup C_{\text{out}}$ *into clusters* $\{C_1, \ldots, C_r\}$ *such that for each* C_j, *we have* $\sum_{i \in C_j} \alpha_i \geq f + \mathsf{cost}(C_j)$. *In particular, we can guarantee that* $C_{\text{core}} \subseteq C_1$.

Proof. The algorithm is as Algorithm 1: First, we initialize $r = 1$ and $C' \leftarrow C_{\text{out}}$. Then, if $|C'| > 2|C_{\text{core}}|$, we split the last $2|C_{\text{core}}|$ clients of C' into a new cluster and we repeat this step until $|C'| \leq 2|C_{\text{core}}|$. Finally, the remaing clients in C' are added to C_{core} to form the first cluster C_1. Suppose that the newly created clusters are $C_2, \ldots, C_r$. We have $|C_i| = 2|C_{\text{core}}|$ for each $i \geq 2$. Because we end our algorithm when $|C'| \leq 2|C_{\text{core}}|$, we have $|C_1| \leq 3|C_{\text{core}}|$.

For C_1, we have

$$\begin{aligned}
\mathsf{cost}(C_1) - \mathsf{cost}(C_{\text{core}}) &= \sum_{j \in C_1 \setminus C_{\text{core}}} \sum_{j' \in C_{\text{core}}} 2d(j,j') + \sum_{j,j' \in C_1 \setminus C_{\text{core}}} d(j,j') \\
&\leq \sum_{j \in C_1 \setminus C_{\text{core}}} 2d(j, C_{\text{core}}) + \sum_{j,j' \in C_1 \setminus C_{\text{core}}} \frac{\sum_{u \in C_{\text{core}}} d(j,u) + d(u,j')}{|C_{\text{core}}|} \\
&= 2 \sum_{j \in C_1 \setminus C_{\text{core}}} d(j, C_{\text{core}}) + \frac{2(|C_1| - |C_{\text{core}}|)}{|C_{\text{core}}|} \cdot \sum_{j \in C_1 \setminus C_{\text{core}}} d(j, C_{\text{core}}) \\
&= \frac{2|C_1|}{|C_{\text{core}}|} \sum_{j \in C_1 \setminus C_{\text{core}}} d(j, C_{\text{core}}) \\
&\leq 6 \sum_{j \in C_1 \setminus C_{\text{core}}} d(j, C_{\text{core}}) \leq \sum_{j \in C_1 \setminus C_{\text{core}}} \alpha_j \ . \qquad (\gamma \geq 6)
\end{aligned}$$

Since we have $\sum_{j \in C_{\text{core}}} \alpha_j \geq f + \mathsf{cost}(C_{\text{core}})$, we have $\sum_{j \in C_1} \alpha_j \geq f + \mathsf{cost}(C_1)$.

Consider any $i \geq 2$. For C_i, we have $\alpha_j \geq \alpha_{j'}$ for each $j' \in C_{\text{core}}$ by the third condition of γ-splittable clusters, as defined in Definition 4. Since we have $|C_i| = 2|C_{\text{core}}|$, we have $\frac{1}{2} \sum_{j \in C_i} \alpha_j \geq$

$\sum_{j \in C_{\text{core}}} \alpha_j$. Then,

$$\begin{aligned}
\sum_{j \in C_i} \alpha_j - \mathsf{cost}(C_i) &\geq \sum_{j \in C_i} \alpha_j - \sum_{j,j' \in C_i} d(j,j') \\
&\geq \sum_{j \in C_i} \alpha_j - \sum_{j,j' \in C_i} \frac{1}{|C_{\text{core}}|} \cdot \sum_{u \in C_{\text{core}}} (d(j,u) + d(u,j')) \\
&\geq \sum_{j \in C_i} \alpha_j - \frac{2|C_i|}{|C_{\text{core}}|} \cdot \sum_{j \in C_i} d(j, C_{\text{core}}) \\
&= \sum_{j \in C_i} (\alpha_j - 4d(j, C_{\text{core}})) \geq \frac{1}{2} \sum_{j \in C_i} \alpha_j \geq f
\end{aligned}$$

The second last inequality follows the fact that $\alpha_j \geq \gamma d(j, C_{\text{core}})$ for each $j \in C_i$ and $\gamma \geq 8$. □

3.2 Cluster Editing

The following definition and lemma together give a generalization of Definition 4 and Lemma 5, which allows us to delete some clients from the core and insert some other clients into the cluster (with weaker than conditions than those for the outer part), while still being able to split the cluster into smaller clusters that maintain the LMP guarantee.

Definition 6 (γ-editable clusters)**.** *A sequence of sets $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}} \rangle$ is said to be a γ-editable cluster with respect to the dual solution $\{\alpha_i\}_{i \in D}$ if*

1. *$\langle C_{\text{core}}, C_{\text{out}} \rangle$ is a γ-splittable cluster.*
2. *$C_{\text{del}} \subseteq C_{\text{core}}$, and $C_{\text{core}}, C_{\text{ins}}, C_{\text{out}}$ are disjoint sets.*
3. *$|C_{\text{ins}}| \leq 2|C_{\text{core}}|$ and $\sum_{i \in C_{\text{ins}}} \alpha_i \geq 2 \sum_{j \in C_{\text{del}}} \alpha_j$.*
4. *For each $i \in C_{\text{ins}}$, we have $\alpha_i \geq \gamma \cdot d(i, C_{\text{core}})$.*

Lemma 7. *Suppose that $\gamma \geq 20$. There exists a polynomial-time algorithm such that for any γ-editable cluster $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}} \rangle$, it partitions $C = (C_{\text{core}} \setminus C_{\text{del}}) \cup C_{\text{ins}} \cup C_{\text{out}}$ into clusters $\{C_1, \ldots, C_r\}$ such that for each C_j, we have $\sum_{i \in C_j} \alpha_i \geq f + \mathsf{cost}(C_j)$. In particular, we can guarantee that $(C_{\text{core}} \setminus C_{\text{del}}) \cup C_{\text{ins}} \subseteq C_1$.*

Proof. We consider the following algorithm: we first apply Algorithm 1 with the following modification, we output $C_1 \leftarrow (C_{\text{core}} \setminus C_{\text{del}}) \cup C' \cup C_{\text{ins}}$. Using Lemma 5, it is straightforward to verify that the clusters $C_2, \ldots, C_r$ satisfy the LMP guarantee. Next, we show that C_1 also satisfies the LMP guarantee.

First, we consider the total α-value of the clients in C_1. Let C' be the set in the final step of Algorithm 1. We have

$$\begin{aligned}
\sum_{i \in C_1} \alpha_i &= \sum_{i \in C_{\text{core}} \setminus C_{\text{del}}} \alpha_i + \sum_{i \in C_{\text{ins}} \cup C'} \alpha_i \\
&= \sum_{i \in C_{\text{core}}} \alpha_i - \sum_{i \in C_{\text{del}}} \alpha_i + \sum_{i \in C_{\text{ins}} \cup C'} \alpha_i \\
&\geq \sum_{i \in C_{\text{core}}} \alpha_i + \frac{1}{2} \sum_{i \in C_{\text{ins}} \cup C'} \alpha_i \,, \qquad (19)
\end{aligned}$$

where the last inequality follows the third bullet of the definition of γ-editable clusters. Let $C'' = C_{\text{ins}} \cup C'$, which gives $C_1 = (C_{\text{core}} \setminus C_{\text{del}}) \cup C''$. Note that $|C''| = |C_{\text{ins}}| + |C'| \le 4|C_{\text{core}}|$ by the third bullet of the definition of γ-editable clusters. The cost of C_1 can be upper bounded as follows:

$$\begin{aligned}
\mathsf{cost}(C_1) &\le \mathsf{cost}(C_{\text{core}}) + 2\sum_{i \in C''} d(i, C_{\text{core}}) + \mathsf{cost}(C'') \\
&\le \mathsf{cost}(C_{\text{core}}) + 2\sum_{i \in C''} d(i, C_{\text{core}}) + \sum_{j,j' \in C''} d(j, j') \\
&\le \mathsf{cost}(C_{\text{core}}) + 2\sum_{i \in C''} d(i, C_{\text{core}}) + \sum_{j,j' \in C''} \frac{d(j, C_{\text{core}}) + d(j', C_{\text{core}})}{|C_{\text{core}}|} \\
&\le \mathsf{cost}(C_{\text{core}}) + \left(2 + \frac{2|C''|}{|C_{\text{core}}|}\right) \sum_{i \in C''} d(i, C_{\text{core}}) \le \mathsf{cost}(C_{\text{core}}) + 10 \sum_{i \in C''} d(i, C_{\text{core}})
\end{aligned}$$

Recall that we have $\alpha_j \ge \gamma d(j, C_{\text{core}})$ for each $j \in C''$ by the first and the fourth bullet of the definition of γ-editable clusters. Because $\gamma \ge 20$, we have $\frac{1}{2}\alpha_j \ge 10 d(j, C_{\text{core}})$ for each $j \in C''$. By combining the definition of γ-splitable clusters, Equation (19) and the above inequality, we have completed the proof. □

4 The LMP approximation

In this section, we present our LMP algorithm for the MinSum clustering problem. Throughout this section, we will use D to denote the set of clients and $f > 0$ to denote the facility opening cost.

Theorem 2. *There exists a polynomial-time algorithm that takes a min-sum k-clustering instance and an additional $f > 0$ as input and outputs a clustering $\mathcal{S}$ satisfying*

$$f|\mathcal{S}| + \mathsf{cost}(\mathcal{S}) \le k \cdot f + 2 \times 10^7 \cdot \mathsf{opt}.$$

4.1 The dual-fitting algorithm

Our algorithm is a primal-dual algorithm that maintains a dual solution $\{\alpha_i\}_{i \in D}$ and the set of opened clusters $\mathcal{S}$ throughout the algorithm. At the beginning of the algorithm, all clients are active and have a dual variable of value 0, and $\mathcal{S}$ is empty. We will increase the dual variables α_i for each active client i at a uniform rate. Upon the time when a cluster S becomes *openable*, which will be defined later, we will open the cluster S (i.e., including it in $\mathcal{S}$) and mark all clients in S inactive. We also make an active client j inactive if its dual variable α_j can pay for the connection cost a fixed constant multiple of $d(j, S)$ for some opened cluster $S \in \mathcal{S}$. Unlike the simple adaptations of JV and JMS, we don't consider all possible clusters $S \subseteq D$ to be openable. Instead, we maintain some dependency over the clients, and only open clusters that are *available* with respect to the dependency.

We will start with a simple version, the client-based dependency graph, so that we can see how we update the dependency and define the available sets. Then, from the update rule, we will derive a more structured version, the cluster-based dependency forest. The structured version will be used in the final algorithm, which will give an efficient implementation of our algorithm and will be easier to analyze.

Client-based dependency graph and a vanilla version of our algorithm. Consider that we have a dependency graph (D, E) over the clients, where D is the set of clients and E is a set of directed edges between the clients. The graph is a directed graph that is updated throughout the algorithm. Intuitively, a directed edge (i, j) in the dependency graph indicates that client j must be in an opened cluster with client i.

With the dependency graph (D, E), we can naturally define a preorder $\mathcal{P}$ over the clients, where $i \preceq_{\mathcal{P}} j$ if there is a directed path from i to j in (D, E). Two clients in the same strongly connected component are equivalent in the preorder $\mathcal{P}$. With the preorder, we can define the notion of *available sets* of clients.

Definition 8 (available sets)**.** *Given a preorder $\mathcal{P}$ over D, we say a set $S \subseteq D$ is* available *with respect to the preorder $\mathcal{P}$ if the membership is preserved under the preorder $\preceq_{\mathcal{P}}$, i.e., if $i \preceq_{\mathcal{P}} j$ and $i \in S$, then $j \in S$.*

We shall remark here that an available set either contains two clients that are equivalent in the preorder or neither of them.

Our update rule is simple as follows: whenever a cluster S is opened, we consider the set of active clients in S right before the cluster S is opened, where we use $D_A(S)$ to denote this set. Then, in the dependency graph, we make all clients in $D_A(S)$ are strongly connected to each other, and make all other clients in S point to them. Mathematically, we add a directed edge (i, j) to E for every pair of clients $i \in S, j \in D_A(S)$. An easy observation here is that two different clients are equivalent in the preorder if and only if they appear in $D_A(S)$ for some opened cluster S.

A more structured cluster-based dependency forest. Inspired by the above discussion, we are going to define a more structured version for the dependency, the cluster-based dependency forest. We shall remark that the forest definition is only valid for algorithms that only open clusters that are available sets. Recall that we use $\mathcal{S}$ to denote the set of all opened clusters. For each $S \in \mathcal{S}$, we contract all clients in $D_A(S)$ into a single node.

More specifically, we consider consider a new graph $(\mathcal{V}, \mathcal{E})$, which is a directed forest. $\mathcal{V}$ consists of an abstract node for each $S \in \mathcal{S}$ and a node for each client that is not in $D_A(S)$ for any $S \in \mathcal{S}$. For simplicity, we define the following notations for each $v \in \mathcal{V}$ for future use:

- **Involved clients:** $D_{\mathcal{V}} : \mathcal{V} \to 2^D$ maps v to $D_A(S)$ if v is the node corresponding to $S \in \mathcal{S}$, and maps v to $\{i\}$ if v is the node corresponding to a client $i \in D$ that is not in $D_A(S)$ for any $S \in \mathcal{S}$. We shall remark here that $D_{\mathcal{V}}(v)$ can be an empty set if S is opened when all clients in S are inactive. Further, we say that i is *involved* in v if $i \in D_{\mathcal{V}}(v)$.
- **Original cluster:** $S_{\mathcal{V}} : \mathcal{V} \to \mathcal{S} \cup \{\perp\}$ maps v to S if v is the node corresponding to $S \in \mathcal{S}$, and maps v to $\perp$ if v is the node corresponding to a client $i \in D$ that is not in $D_A(S)$ for any $S \in \mathcal{S}$.

Then, we define the directed edges $\mathcal{E}$ as follows: at the time each $S \in \mathcal{S}$ is opened, we consider v as the node for S, and we add a directed edge (v', v) to $\mathcal{E}$ for every other $v' \in \mathcal{V}$ such that v' is still a root in the forest and there exists v'' in the tree rooted at v' satisfying $D_{\mathcal{V}}(v'') \cap S \neq \emptyset$. We shall remark that each tree in the resulting forest is an in-tree, meaning that each non-root node has a pointer to its parent node.

Recall the preorder $\mathcal{P}$ we defined via the client-based dependency graph (D, E). According to above discussion, we can construct a slightly differnt preorder $\mathcal{P}(\mathcal{V}, \mathcal{E})$. [9]

[9] The difference occurs (subtly) when $D_A(S) = \emptyset$ for some $S \in \mathcal{S}$. In the earlier preorder induced by the dependency graph, if part of S is opened in another S' later, only part of S is less than $D_A(S')$. However, in the preorder induced by the cluster-based dependency forest, all clients in S are less than $D_A(S')$.

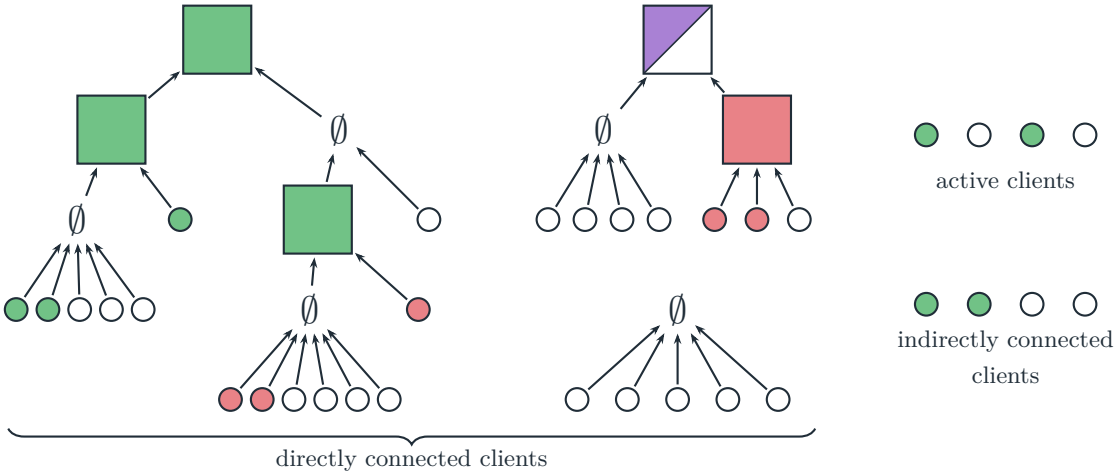


Figure 1: An example of how the sets of involved clients of the nodes appear in the cluster-based dependency forest $(\mathcal{V}, \mathcal{E})$ and partition the set of all clients. The squares correspond to the sets of involved clients of opened clusters, and the circles correspond to clients that are not in $D_A(S)$ for any opened cluster S. Green squares and circles correspond to an available set, while red squares and circles correspond to a non-available set.

Definition 9. *The preorder $\mathcal{P}(\mathcal{V}, \mathcal{E})$ with respect to $(\mathcal{V}, \mathcal{E})$ is defined as follows. For each pair of clients $i, j \in D$, we have $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$ if and only if the node v involving i is equal to or a descendant of the node $v' \in \mathcal{V}$ involving j.*

Remark 10. *According to Definition 8, an available set $S \subseteq D$ with respect to the preorder $\mathcal{P}(\mathcal{V}, \mathcal{E})$ is a union of the involved clients of nodes in an upward-closed subset $\Upsilon \subseteq \mathcal{V}$. Upward-closed means that if $v \in \Upsilon$ and v' is an ancestor of v in the forest, then $v' \in \Upsilon$.*

Under the cluster-based dependency forest $(\mathcal{V}, \mathcal{E})$, we can define the notion of *current root* and *current pointer* for each client, which will be used in our algorithm to determine whether a cluster is openable or not. The current pointer can be viewed as the opened cluster with the minimum size that has a guaranteed small total distance to the client (if we only consider the dependency forest without the underlying metric), as we established in Corollary 24 later.

Definition 11 (current root)**.** *For each client $i \in D$, the* current root $\mathsf{root}(i)$ *is the root of the tree in $(\mathcal{V}, \mathcal{E})$ that contains the node involving i.*

Definition 12 (current pointer)**.** *For each client $i \in D$, we use the* current pointer $Q(i)$ *to denote the original cluster of the current root of i, i.e., $Q(i) = S_{\mathcal{V}}(\mathsf{root}(i))$.*

Definition of openable clusters. In our algorithm, we have three constants $\gamma_{DC}, \gamma_{IC}, r_S > 1$. We define a cluster S to be (γ_{DC}, r_S)-*openable* with respect to the current dual variables $\{\alpha_i\}_{i \in D}$, the current set of opened clusters $\mathcal{S}$ and the dependency forest $(\mathcal{V}, \mathcal{E})$ as follows.

Definition 13 ((γ_{DC}, r_S)-openable clusters)**.** *A cluster S is (γ_{DC}, r_S)-openable with respect to the current dual variables $\{\alpha_i\}_{i \in D}$, the current set of active clients A, the current set of opened clusters $\mathcal{S}$ and the dependency forest $(\mathcal{V}, \mathcal{E})$ if the following conditions hold:*

1. *Total dual contribution is enough: $\sum_{j \in S} \alpha_j \geq f + 4\gamma_{DC} \cdot \mathsf{cost}(S)$.*

2. *The cluster size is small enough: for each $i \in S$, we have either $Q(i) = \perp$ or $r_S \cdot |S| \le |Q(i)|$.*

3. *The cluster is an available set: If $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$ and $i \in S$, then $j \in S$.*

One issue with the above definition is that it is not clear how to find an openable cluster efficiently. We provide a polynomial-time algorithm to find an approximately openable cluster in Algorithm 2. In the algorithm, we first process all singletons with $\alpha_i \ge f$. Then, we guess a center c and a size s, and find an available set S of size at most s that maximizes the summation $\sum_{j\in S}(\alpha_j - s \cdot 4\gamma_{DC} \cdot d(j,c))$, where the term $s \cdot d(j,c)$ can be viewed as the cost of the balanced k-median problem with a center c and size s. This part can be easily implmented via dynamic programming on the cluster-based dependency forest $(\mathcal{V},\mathcal{E})$. If the maximized summation is at least f, we clean the set S so that it has two additional properties (Observation 20 and Lemma 44), and return the cleaned set as an approximately openable set.

We first show that the algorithm is always well-defined by eliminating the possibility that $\mathsf{cost}(S) = 0$ in the first step of the cleaning process (step III.1). We prove the following lemma, which implies that we can always have two clients in S because the first step of Algorithm 2 and the uniform dual increment (shown in Algorithm 3 later) ensure that $\alpha_i < f$ in the rest of the algorithm. Because we assume that d is a metric, we have $\mathsf{cost}(S) > 0$ if $|S| \ge 2$.

Lemma 14. *Suppose S is a cluster found in Algorithm 2 during the enumeration of (s,c). Then, we have $\sum_{j\in S}\alpha_j \ge f + 2\gamma_{DC} \cdot \mathsf{cost}(S)$ throughout the first step of the cleaning process.*

Proof. First, we show that by the end of the enumeration of (s,c) in Algorithm 2, we have $\sum_{j\in S}\alpha_j \ge f + 2\gamma_{DC} \cdot \mathsf{cost}(S)$. Because $|S| \le s$, for any $c \in D$, we have

$$\mathsf{cost}(S) = \sum_{j,j'\in S} d(j,j') \le \sum_{j,j'\in S} d(j,c) + d(c,j') \le 2s \cdot \sum_{j\in S} d(j,c)\ .$$

Therefore, at the beginning of the cleaning process in Algorithm 2, we have $\gamma' \ge 2\gamma_{DC}$.

Next, we show that the first step of the cleaning process in Algorithm 2 does not make γ' smaller. For the first case, if we set $S \leftarrow A \cap S$, we have

$$\mathsf{cost}(A\cap S) = \mathsf{cost}(S) - \sum_{i,j\in S\setminus A} d(i,j) - 2\sum_{i\in S\setminus A}\sum_{j\in S\cap A} d(i,j) \le \mathsf{cost}(S) - \sum_{i\in S\setminus A} d(i,S),\ \text{thus}$$
$$\gamma'\mathsf{cost}(A\cap S) \le \gamma'\mathsf{cost}(S) - \gamma'\sum_{i\in S\setminus A} d(i,S) \le \sum_{j\in S}\alpha_j - f - \gamma'\sum_{i\in S\setminus A} d(i,S) < \sum_{j\in A\cap S}\alpha_j - f\ .$$

Similarly, for the second case, we can show that γ' does not decrease:

$$\mathsf{cost}(S\setminus\{j\}) = \mathsf{cost}(S) - 2\sum_{i\in S} d(i,j),\ \text{thus}$$
$$\gamma'\mathsf{cost}(S\setminus\{j\}) \le \gamma'\mathsf{cost}(S) - 2\gamma' d(j,S) = \sum_{i\in S}\alpha_i - f - 2\gamma' d(j,S) < \sum_{i\in S\setminus\{j\}}\alpha_i - f\ .$$

Hence, after the first cleaning loop terminates, we still have $\gamma' \ge 2\gamma_{DC}$ and thus $\alpha_i \ge 2\gamma_{DC} d(i,S)$ for every $i \in S \cap A$. □

On the other hand, we can show that if there exists a (γ_{DC}, r_S)-openable cluster, then Algorithm 2 will return at least one cluster.

Algorithm 2 (Obtain an approximately openable set).

Input: $\{\alpha_i\}_{i\in D}$, opened clusters $\mathcal{S}$, active clients A, and the dependency forest $(\mathcal{V},\mathcal{E})$.

I. Dealing with singletons. If there exists $i \in A$ such that $\alpha_i \geq f$, we return $\{i\}$.

II. Enumeration. We enumerate every size $s \in [|D|]$ and every potential "center" $c \in D$, and do the following:

- For each tree in the dependency forest $(\mathcal{V},\mathcal{E})$ rooted by the set v, if $S_{\mathcal{V}}(v) \neq \perp$ and $|S_{\mathcal{V}}(v)| < r_S \cdot s$, we ignore the tree. Let D_{ignore} be the union of involved clients of the nodes in the ignored trees.
- We find an available set $S \subseteq D \setminus D_{\text{ignore}}$ with respect $\mathcal{P}(\mathcal{V},\mathcal{E})$ such that $|S| \leq s$ and $\sum_{j\in S}(\alpha_j - s \cdot 4\gamma_{DC} \cdot d(j,c))$ is maximized, via dynamic programming[a].
- If $\sum_{j\in S}(\alpha_j - s \cdot 4\gamma_{DC} \cdot d(j,c)) \geq f$, we end the enumeration with (s,c,S).

III. Cleaning. If no (s,c,S) is found, we return that there is no openable cluster. Otherwise, we do the following cleaning process:

1. Repeat the following:
   - Let $\gamma' = (\sum_{j\in S}\alpha_j - f)/\mathsf{cost}(S)$.
   - If $\sum_{j\in S\setminus A}\alpha_j < \gamma' \cdot \sum_{j\in S\setminus A} d(j,S)$, we set $S \leftarrow A \cap S$.
   - Otherwise, if there exists $j \in S \cap A$ such that $\alpha_j < 2\gamma' d(j,S)$, we remove one such client j from S.
   - If no client is removed in the above two steps, we break the loop.
2. Repeat until there is no client $i \in S$ such that $\alpha_i < \gamma_{DC} d(i,S)$: for such client i, we remove all $j \in S$ such that $j \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} i$ from S.
3. Return the cleaned set S.

If no (s,c,S) is found in the enumeration, we return $\perp$.

[a] The dynamic programming approach is a straightforward knapsack-style DP on trees. We compute the maximized objective $\mathsf{DP}[v,t]$ for the sub-tree rooted at v that contains t clients, using its children's DP values. Finally, we use a knapsack DP over each tree in the forest that are not ignored.

Lemma 15. *Suppose that there exists S is a (γ_{DC}, r_S)-openable cluster with respect to the current dual variables $\{\alpha_i\}_{i\in D}$, the current set of active clients A, the current set of opened clusters $\mathcal{S}$ and the dependency forest $(\mathcal{V},\mathcal{E})$. Then, Algorithm 2 will return a set S.*

Proof. We assume that the set S satisfies all conditions for a (γ_{DC}, r_S)-openable cluster. Let $s = |S|$ and let $c^* \in S$ be the client that minimizes $\sum_{j\in S} d(j,c^*)$. First, because of the second condition, we have $Q(i) = \perp$ or $r_S \cdot s \leq |Q(i)|$ for every $i \in S$. This implies that the current root $\mathsf{root}(i)$'s original cluster $S_{\mathcal{V}}(\mathsf{root}(i))$ is $\perp$ or has size at least $r_S \cdot s$ for every $i \in S$. Therefore, $i \notin D_{\text{ignore}}$ for every $i \in S$ when we enumerate (s,c^*) in Algorithm 2. Because S is an available set by the third condition, S is a candidate set when we enumerate (s,c^*) in Algorithm 2. Second, using a similar

averaging argument connecting balanced k-median and k-minsum [BCR01], we have

$$\mathsf{cost}(S) = \sum_{c \in S} \sum_{j \in S} d(j, c) \geq s \cdot \sum_{j \in S} d(j, c^*) \ .$$

Therefore, according to the first condition, we have $\sum_{j \in S} \alpha_j \geq f + s \cdot \sum_{j \in S} 4\gamma_{DC} d(j, c^*)$. This means that Algorithm 2 will return a set S, at least when we enumerate (s, c^*). □

Description of the dual fitting process. We choose three constants $\gamma_{DC}, \gamma_{IC}, r_S > 1$ for our algorithm. Ideally, we would like to choose the constants such that

$$\gamma_{DC} \approx r_S \gamma_{IC} \gg \gamma_{IC} \gg 1, \quad \text{and} \quad r_S \gg 1.$$

We will give the conditions for these constants in each lemma in our analysis and the exact values of these constants at the end of Subsection 4.2.

As mentioned earlier, we initialize the dual variables α_i to be zero and $\mathcal{S}$ to be empty. We will increase the dual variables α_i for each active client i at a uniform rate until either a set S becomes "openable" with respect to the current dual variables $\{\alpha_i\}_{i \in D}$, or a client j can pay for the connection cost $\gamma_{IC} \cdot d(j, S)$ for some opened cluster $S \in \mathcal{S}$. To find such an "openable" cluster, we employ the polynomial-time algorithm in Algorithm 2 to find an approximately openable cluster. With this efficient implementaion of identifying approximate openable clusters, we conclude the entire dual fitting algorithm in Algorithm 3, where we also implement the process of updating $\mathcal{V}$ and defining $D_{\mathcal{V}}$ and $S_{\mathcal{V}}$. Also see Figure 2 for a demonstration.

Algorithm 3 (Dual Fitting – the forward stage).

Initialization: $A = D$; $\mathcal{S} = \emptyset$; the dependency forest $(\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_i : i \in D\}$ (with $D_{\mathcal{V}}(v_i) = \{i\}$ and $S_{\mathcal{V}}(v_i) = \perp$) and $\mathcal{E} = \emptyset$; and $C^I(i) = \emptyset, \alpha_i = 0$ for every $i \in D$.

While $A \neq \emptyset$, uniformly grow the α_j of every client in A until:

1. Some cluster S is returned by Algorithm 2.

   In this case, we set $\mathcal{S} = \mathcal{S} \cup \{S\}$ and let $D_A(S) = S \cap A$. We remove all v_i from $\mathcal{V}$ for $i \in D_A(S)$, and create a node v in $\mathcal{V}$ such that $D_{\mathcal{V}}(v) = D_A(S)$ and $S_{\mathcal{V}}(v) = S$.

   For each $v' \in \mathcal{V} \setminus \{v\}$ such that v' is still a root in the forest and there exists v'' in the tree rooted at v' satisfying $D_{\mathcal{V}}(v'') \cap S \neq \emptyset$, we add a directed edge (v', v) in $\mathcal{E}$. Remove all clients in S from A.

2. For some client $j \in A$ and $S \in \mathcal{S}$, we have

   $$\alpha_j \geq \gamma_{IC} \cdot d(j, S).$$

   In this case, we remove j from A and set $C^I(j) = S$.

Output: $\mathcal{S}$, $\{\alpha_i\}_{i \in D}$ and $\{C^I(i)\}_{i \in D}$.

4.2 Overview of the analysis

Our analysis of Algorithm 3 is based on the dual fitting framework. We first show that the output of Algorithm 3 can be converted into a set of γ_{IC}-editable clusters $\{(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}})\}$ in

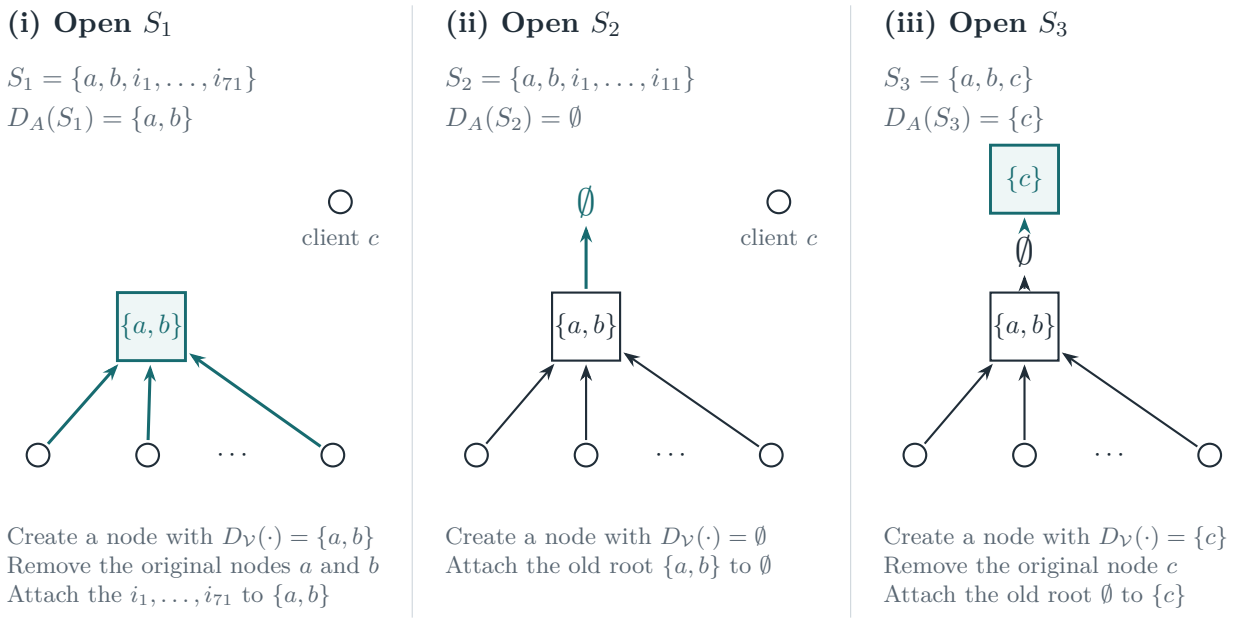


Figure 2: An example of how the dependency forest $(\mathcal{V}, \mathcal{E})$ is updated in Algorithm 3, with the parameter $r_S = 4$ here.

polynomial-time (Lemma 16). Then, we show that the final dual solution $\{\alpha_i^*\}_{i \in D}$ created by Algorithm 3 is feasible with respect to the dual Lagrange LP (MinSum-Lag-Dual), up to a factor of Γ on the connection costs(Lemma 17). Finally, we combine these two results to prove Theorem 2.

Lemma 16 (dual payments)**.** *Assume that $\gamma_{DC} \geq 2\gamma_{IC}, \gamma_{IC} \geq 20$ and $r_S \geq 6$. Given the output of Algorithm 3, there is a polynomial-time algorithm that partitions the clients into a set of γ_{IC}-editable clusters $\mathcal{C} = \{(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}})\}$, i.e.,*

$$\bigsqcup_{\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}} \rangle \in \mathcal{C}} (C_{\text{core}} \setminus C_{\text{del}}) \sqcup C_{\text{ins}} \sqcup C_{\text{out}} = D \ ,$$

with respect to the final dual solution $\{\alpha_i^*\}_{i \in D}$ created by Algorithm 3.

Lemma 17 (Dual feasibility)**.** *Assume that $\Gamma = r_S \gamma_{DC}, \gamma_{DC} = r_S \gamma_{IC}$ and $r_S > 832$. Then, for any $C \subseteq D$, we have $\sum_{i \in C} \alpha_i^* \leq f + \Gamma \cdot \mathsf{cost}(C)$, where $\{\alpha_j^*\}_{j \in D}$ is the final dual solution created by Algorithm 3.*

Proof of Theorem 2. Let $\Gamma = 2 \times 10^7$, $\gamma_{DC} = 2 \times 10^4, \gamma_{IC} = 20$ and $r_S = 1000$. It is easy to verify that these constants satisfy the conditions in Lemmas 16 and 17.

Let us run Algorithm 3 first. By Lemma 16, we can then partition the clients into a set of γ_{IC}-editable clusters $\{(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}})\}$ with respect to the final dual solution $\{\alpha_i^*\}_{i \in D}$ created by Algorithm 3. According to Lemma 7, we can partition the clients in them into smaller clusters C such that each satisfies $\sum_{i \in C} \alpha_i^* \geq f + \mathsf{cost}(C)$. Suppose $\mathcal{S}_{\text{final}}$ is the set of clusters after this partitioning. Then, we have

$$f \cdot |\mathcal{S}_{\text{final}}| + \mathsf{cost}(\mathcal{S}_{\text{final}}) \leq \sum_{i \in D} \alpha_i^* \ .$$

Let OPT be any optimal solution of k-minsum instance. For any cluster $C \in \mathsf{OPT}$, because of Lemma 17, we have $\sum_{i \in C} \alpha_i^* \leq f + \Gamma \cdot \mathsf{cost}(C)$. Therefore, by weak duality, we have

$$\sum_{i \in D} \alpha_i^* \leq k \cdot f + \Gamma \cdot \mathsf{cost}(\mathsf{OPT}) \ .$$

Putting these two inequalities together, we have shown that the output $\mathcal{S}_{\text{final}}$ satisfies the desired approximation guarantee. □

4.3 Basic properties of opened clusters and dependency forests

We will prove several basic properties of the opened clusters and the dependency forest $(\mathcal{V}, \mathcal{E})$, which will be useful in the analysis of Algorithm 3. We consider the dual variables $\{\alpha_i\}_{i \in D}$ throughout the algorithm, which are changing as the algorithm proceeds. For the analysis, we will define the following notitions, which are also changing throughout the algorithm:

- **current time** θ equals to the value of α_i for each $i \in A$.
- **directly connected (DC) clients:** $DC = \cup_{S \in \mathcal{S}} S$ denote the set of inactive clients that is included in some opened cluster.
- **indirectly connected (IC) clients:** $IC = D \setminus (A \uplus DC)$ denote the set of inactive clients that is not included in any opened cluster.

Two straightforward observations are that (i) the original cluster $S_{\mathcal{V}}(v)$ is either $\perp$ or drops by a factor of r_S when moving to its parent in the dependency tree; and that (ii) if $(v', v) \in \mathcal{E}$, the α-value of any client involved in v' is at most α-value of the client in v.

Observation 18. *Throughout the algorithm, for any edge (v', v) in the dependency forest $(\mathcal{V}, \mathcal{E})$, we have $|S_{\mathcal{V}}(v')| \geq r_S \cdot |S_{\mathcal{V}}(v)|$ or $S_{\mathcal{V}}(v') = \perp$.*

Observation 19. *Throughout the algorithm, we have $\alpha_i \leq \alpha_j$ for any $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$.*

Proof. This is straightforward if $i = j$. Suppose that $i \neq j$. The node v invovling i is equal to or is a descendant of the node v' involving j in the dependency forest $(\mathcal{V}, \mathcal{E})$. This implies that v' corresponds to an opened cluster S'. If v corresponds to an opened cluster S, then it is clear that S' is opened no earlier than S, and thus $\alpha_i \leq \alpha_j$. If v is a singleton node that corresponds to a client that does not belong to any opened cluster before S' is opened, then j becomes inactive no earlier than i and thus $\alpha_i \leq \alpha_j$. □

Now we consider one cluster that is returned by Algorithm 2. First, we observe that the steps of the cleanning process ensures that the α-values are large compared to the connection cost and guarantees the conditions 2 and 3 for the returned cluster.

Observation 20. *For every opened cluster $S \in \mathcal{S}$, we have $\alpha_j \geq \gamma_{DC} \cdot d(j, S)$ for each $j \in S$.*

Proof. If $S = \{i\}$ for some $i \in A$, this trivially holds as $d(i, S) = 0$. Otherwise, by the second step of the cleaning process in Algorithm 2, we have $\alpha_j \geq \gamma_{DC} \cdot d(j, S)$ for each $j \in S$ before terminating the algorithm. □

Observation 21. *For every opened cluster $S \in \mathcal{S}$, we have that for each $i \in S$, we have either $Q(i) = \perp$ or $r_S \cdot |S| \leq |Q(i)|$ before opening the new cluster.*

Proof. If $S = \{i\}$ for some $i \in A$, this trivially holds as $Q(i) = \perp$ before the opening. Otherwise, this follows the fact that the returned set has size $\leq s$ in the enumeration of (s, c), and the fact that we ignore all trees in the dependency forest $(\mathcal{V}, \mathcal{E})$ whose original cluster is not $\perp$ and has size $< r_S \cdot s$ in the enumeration of (s, c). □

Observation 22. *For every opened cluster $S \in \mathcal{S}$, we have that S is an available set with respect to the preorder $\mathcal{P}(\mathcal{V}, \mathcal{E})$ before opening the new cluster.*

Proof. If $S = \{i\}$ for some $i \in A$, this trivially holds as $Q(i) = \perp$ before the opening. We consider the case we return some cluster after the enumeration of (s, c). If in the first step of the cleaning process in Algorithm 2, we execute $S \leftarrow A \cap S$, which is an available set because active clients are singletons in $(\mathcal{V}, \mathcal{E})$. Otherwise, we can only remove active clients from S in the first step of the cleaning process, which preserves the property of being an available set. In the second step of the cleaning process, we remove all clients $i \in S$ such that $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$ for some $j \in S$, which also preserves the property of being an available set. □

With these observations, we can present a key lemma for the opened clusters and the forests produced by the algorithm: the distance between any two clients in the same tree in the dependency forest is bounded by the current time θ and the size of its current pointer $|Q(i)|$.

Lemma 23. *Assume that $r_S \geq 6$. Throughout the algorithm, for any two clients i, j such that $\mathsf{root}(i) = \mathsf{root}(j)$ and $Q(i) \neq \perp$, we have $d(i, j) \leq \frac{3\theta}{\gamma_{DC}|Q(i)|}$.*

Proof. First, we note that at time θ, all clients have $\alpha_i \leq \theta$.

Next, we prove this lemma by induction. At the beginning of the algorithm, each tree in $(\mathcal{V}, \mathcal{E})$ contains only one node, and the lemma holds trivially. Suppose that we open a cluster S at time θ and add a new node v to the dependency forest $(\mathcal{V}, \mathcal{E})$. Consider any two clients i, j whose roots are v' and v'' before the opening of S and $v' \neq v''$, but are in the tree rooted at v after the opening of S (this is the only case not covered by the induction hypothesis). This could only happen if two conditions hold at the same time:

- $i \in D_A(S)$ or there exists a directed edge (v', v) in $\mathcal{E}$, and
- $j \in D_A(S)$ or there exists a directed edge (v'', v) in $\mathcal{E}$.

This implies that there exist a client i' whose root is v' and a client j' whose root is v'' before S is opened, such that $i' \in S$ and $j' \in S$. For simplicity, with a slight abuse of notation, we assume that $|\perp| = \infty$. According to our induction hypothesis, we have $d(i, i') \leq \frac{3\theta}{\gamma_{DC}|Q(i)|} = \frac{3\theta}{\gamma_{DC}|S_{\mathcal{V}}(v')|}$ and $d(j, j') \leq \frac{3\theta}{\gamma_{DC}|Q(j)|} = \frac{3\theta}{\gamma_{DC}|S_{\mathcal{V}}(v'')|}$, which holds trivially even when $Q(i) = \perp$ (or $Q(j) = \perp$) because we have $i = i'$ (or, $j = j'$). According to Observation 20, we have $\alpha_{i'} \geq \gamma_{DC} d(i', S)$ and $\alpha_{j'} \geq \gamma_{DC} d(j', S)$. Therefore, because we have we have

$$
\begin{aligned}
d(i, j) &\leq d(i, i') + d(i', j') + d(j', j) \\
&\leq d(i, i') + \frac{d(i', S) + d(j', S)}{|S|} + d(j', j) \\
&\leq \frac{3\theta}{\gamma_{DC}|S_{\mathcal{V}}(v')|} + \frac{\alpha_{i'}}{\gamma_{DC}|S|} + \frac{\alpha_{j'}}{\gamma_{DC}|S|} + \frac{3\theta}{\gamma_{DC}|S_{\mathcal{V}}(v'')|} \\
&\leq \frac{3\theta}{\gamma_{DC} r_S |S|} + \frac{\theta}{\gamma_{DC}|S|} + \frac{\theta}{\gamma_{DC}|S|} + \frac{3\theta}{\gamma_{DC} r_S |S|} \leq \frac{3\theta}{\gamma_{DC}|S|} \ . \qquad \text{(Observation 18)}
\end{aligned}
$$

Hence, we have completed the proof. □

This lemma has many implications: for example, it gives upper bounds for $d(i, Q(i))$ for each $i \in DC$ and $d(i, C)$ by $d(j, C)$ for any $j \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} i$ and any set C with small sizes.

Corollary 24. *Assume that $r_S \geq 6$. For each $i \in D$, we have either $Q(i) = \perp$ or $d(i, Q(i)) \leq \frac{3\theta}{\gamma_{DC}}$.*

Proof. It suffices to consider the case when $Q(i) \neq \perp$. Let $v = \mathsf{root}(i)$. $Q(i)$ is the opened cluster at the time when v is created in the dependency forest $(\mathcal{V}, \mathcal{E})$. First we observe that all clients are involved in at least one node in the dependency forest $(\mathcal{V}, \mathcal{E})$. Therefore, any client in $Q(i)$, either it is in $D_A(Q(i))$ or the tree containing the node involving it (before the creation of v) becomes a subtree of v after the creation of v. Therefore, all clients in $Q(i)$ are in the same tree as i in the dependency forest $(\mathcal{V}, \mathcal{E})$. By Lemma 23, we have $d(i, j) \leq \frac{3\theta}{\gamma_{DC}|Q(i)|}$ for each $j \in Q(i)$. By summing over all $j \in Q(i)$, we have $d(i, Q(i)) \leq \frac{3\theta}{\gamma_{DC}}$. □

Lemma 25. *Assume that $r_S \geq 6$. Suppose that $i \neq j \in D$ are two clients such that $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$. Consider any $r \geq 1$ and any subset $C \subseteq D$ such that $r \cdot |C| \leq |Q(i)|$. Then, for any $\gamma \geq 1$, if $\alpha_i \geq \gamma \cdot d(i, C)$, then we have*

$$\alpha_j \geq \frac{\gamma}{1 + \frac{3\gamma}{r\gamma_{DC}}} \cdot d(j, C) \ .$$

Proof. Because $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$, the node v involving i is equal to or a descendant of node v' involving j in the dependency forest $(\mathcal{V}, \mathcal{E})$, and thus the node v' coresponds to an opened cluster S and $j \in D_A(S)$. In addition, we have $Q(i) = Q(j)$.

Suppose the current time is θ. Consider the time θ' when S is opened. We have $\theta' \leq \theta$ because S has opened at time θ. Because j is active right before S is opened and stops growing after that, we have $\alpha_j = \theta'$ at time θ. Note that $Q(i)$ is the current pointer of i. Because of Observation 18, we have $|Q(i)| \leq |S_{\mathcal{V}}(v')|$ and thus $|C| \leq |S_{\mathcal{V}}(v')|/r$. Applying Lemma 23 at time θ', we have $d(i, j) \leq \frac{3\theta'}{\gamma_{DC}|S_{\mathcal{V}}(v')|} = \frac{3\alpha_j}{\gamma_{DC}|S_{\mathcal{V}}(v')|}$. Then, we have

$$d(j, C) \leq d(i, j) \cdot |C| + d(i, C) \leq \frac{3\alpha_j \cdot |C|}{\gamma_{DC}|S_{\mathcal{V}}(v')|} + \frac{\alpha_i}{\gamma} \leq \left(\frac{3}{r\gamma_{DC}} + \frac{1}{\gamma}\right) \alpha_j \ ,$$

where the last inequality holds because $\alpha_i \leq \alpha_j$ by Observation 19. By dividing $\frac{3}{r\gamma_{DC}} + \frac{1}{\gamma}$ on both sides, we have completed the proof. □

Corollary 26. *Assume that $r_S \geq 6$. For any client $i \neq j \in D$ such that $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$, and any set $C \subseteq D$ such $|C| \leq |Q(j)|/r_S$, if we have $\alpha_j < \gamma_{DC} \cdot d(j, C)$, then we have $\alpha_i < 2\gamma_{DC} \cdot d(i, C)$.*

Now, we are ready to show that the cluster returned by Algorithm 2 is an approximately openable cluster. Our Observations 21 and 22 show the second and third conditions of Definition 13 for a (γ_{DC}, r_S)-openable cluster for the returned cluster. Next, we present the proof for an approximate version of the first condition.

Lemma 27. *Assume that $r_S \geq 6$. For every opened cluster $S \in \mathcal{S}$, we have $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{DC} \cdot \mathsf{cost}(S)$.*

Proof. If the opened cluster S is a singleton $\{i\}$ for some $i \in A$, this trivially holds as $\mathsf{cost}(S) = 0$ and $\alpha_i \geq f$. Next, we assume that S is returned after enumeration of (s, c) in Algorithm 2. Because of Lemma 14, we have $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{DC} \cdot \sum_{j \in S} d(j, S)$ after the first step of the cleaning process in Algorithm 2. Due to Corollary 26 and the definition of the second step of the cleaning process, if i meets the condition in some loop of the second step of the cleaning process and j is removed

from in the same loop, we have $\alpha_j \le 2\gamma_{DC} \cdot d(j,S)$. Suppose that we remove S' from S in one loop of the second step of the cleaning process because of i. Then, we have

$$\sum_{j \in S \setminus S'} \alpha_j = \sum_{j \in S} \alpha_j - \sum_{j \in S'} \alpha_j \ge f + 2\gamma_{DC}\mathsf{cost}(S) - 2\gamma_{DC} \cdot \sum_{j \in S'} d(j,S) \ge f + 2\gamma_{DC}\mathsf{cost}(S \setminus S') \ .$$

Therefore, we have $\sum_{j \in S} \alpha_j \ge f + 2\gamma_{DC}\mathsf{cost}(S)$ after the second step of the cleaning process, and we have completed the proof. □

Corollary 28. *Assume that $r_S \ge 6$. Every opened cluster $S \in \mathcal{S}$ is not empty.*

Corollary 29. *Assume that $r_S \ge 6$. Throughout the algorithm, $|\mathcal{S}| \le |D|(1 + \frac{\log |D|}{\log r_S})$.*

Proof. At the time we open a cluster S, at least one of the following holds:

- One active or indirectly connected (IC) client $i \in S$ becomes directly connected; or
- $|Q(i)|$ for some $i \in DC$ decreases by a factor of r_S.

Note that the first case can happen at most once per client. Because $1 \le |Q(i)| \le |D|$ for each $i \in DC$, the second case can happen at most $\log_{r_S} |D|$ times per client. Therefore, the total number of opened clusters is at most $|D|(1 + \frac{\log |D|}{\log r_S})$. □

As a simple corollary, we can show the following bound for any set S satisfying conditions 2 and 3 for a (γ_{DC}, r_S)-openable cluster. The proof is straightforward by the fact that the algorithm increases the dual variables continuously and keeps opening some (possibly other) cluster when S satisfies $\sum_{j \in S} \alpha_j = f + 4\gamma_{DC} \cdot \mathsf{cost}(S)$ until at least one of the conditions is broken (because of newly opened clsuters), and Corollary 29 that the number of opened clusters is not infinite.

Corollary 30. *Assume that $r_S \ge 6$. Throughout the algorithm, for any $S \subseteq D$ that satisfies conditions 2 and 3 for a (γ_{DC}, r_S)-openable cluster with respect to the current dual variables $\{\alpha_i\}_{i \in D}$, the current set of active clients A, the current set of opened clusters $\mathcal{S}$ and the dependency forest $(\mathcal{V}, \mathcal{E})$, we have $\sum_{j \in S} \alpha_j \le f + 4\gamma_{DC} \cdot \mathsf{cost}(S)$.*

Finally, we show that if a client's α-value becomes inactive because it can pay for a cluster S, then its α-value can also pay for any subsequent opened cluster intersecting S. This property will be crucial for the dual payment part of the analysis.

Lemma 31. *Assume that $r_S \ge 4$ and $\gamma_{DC} \ge 2\gamma_{IC}$. Let $i, j \in D$ be any client. Suppose $S, S' \in \mathcal{S}$ are two opened clusters such that $i \in S \cap S'$ and S is opened earlier than S'. If $C^I(j) = S$, then we have $\alpha_j \ge \gamma_{IC} \cdot d(j, S')$.*

Proof. Since $C^I(j) = S$, we have $\alpha_j \ge \gamma_{IC} \cdot d(j,S)$. Let v, v' be the node in the dependency forest $(\mathcal{V}, \mathcal{E})$ corresponding to S and S', respectively. Note that at the time S' is opened, because $S \cap S' \ne \emptyset$, v becomes a descendant of v' in the dependency forest $(\mathcal{V}, \mathcal{E})$. Therefore, we have $|S'| \le |S|/r_S$ according to Observation 18. Also, note that i becomes inactive no later than S is opened, and j becomes inactive after that, we have $\alpha_j \ge \alpha_i$. Because of Observation 20, we have $\alpha_i \ge \gamma_{DC} \cdot d(i, S')$ and $\alpha_i \ge \gamma_{DC} d(i,S)$. Therefore, we have

$$\begin{aligned} d(j,S') \le d(j,i)|S'| + d(i,S') &\le \frac{(d(j,S) + d(i,S))|S'|}{|S|} + \frac{\alpha_i}{\gamma_{DC}} \\ &\le \frac{\alpha_j + \alpha_i}{\gamma_{IC} r_S} + \frac{\alpha_i}{\gamma_{DC}} \le \frac{2\alpha_j}{4\gamma_{IC}} + \frac{\alpha_j}{2\gamma_{IC}} = \frac{\alpha_j}{\gamma_{IC}} \ . \end{aligned}$$

Hence, we have completed the proof. □

4.4 Dual Payments

Let $(\alpha_i^*)_{i\in D}$ be the final dual variables created by Algorithm 3. In this subsection, we prove Lemma 16. More specifically, we present Algorithm 4 that outputs a set of γ_{IC}-editable clusters (Definition 6) whose total opening and connection cost does not exceed the sum of the final dual variables.

Lemma 16 (dual payments)**.** *Assume that $\gamma_{DC} \geq 2\gamma_{IC}, \gamma_{IC} \geq 20$ and $r_S \geq 6$. Given the output of Algorithm 3, there is a polynomial-time algorithm that partitions the clients into a set of γ_{IC}-editable clusters $\mathcal{C} = \{(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}})\}$, i.e.,*

$$\bigsqcup_{\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle \in \mathcal{C}} (C_{\text{core}} \setminus C_{\text{del}}) \sqcup C_{\text{ins}} \sqcup C_{\text{out}} = D \ ,$$

with respect to the final dual solution $\{\alpha_i^*\}_{i\in D}$ created by Algorithm 3.

The recovery algorithm. In Algorithm 4, we present a polynomial-time algorithm that outputs a set of γ_{IC} editable clusters $\mathcal{C} = \{\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle\}$. The algorithm enumerates each cluster $S \in \mathcal{S}$ in the reverse order that they were added to $\mathcal{S}$ in Algorithm 3. At each step, the algorithm first finds the set U of unsettled clients that are not covered by any $(C_{\text{core}} \setminus C_{\text{del}}) \cup C_{\text{ins}} \cup C_{\text{out}}$, and computed the set IC of unsettled clients that are indirectly connected to S. If $S \subseteq U$, then the algorithm adds a new editable cluster $\langle S, \emptyset, \emptyset, IC\rangle$ to $\mathcal{C}$. Otherwise, we go through all editable clusters whose $C_{\text{core}} \setminus C_{\text{del}}$ intersects with S, and we add the clients in IC to C_{ins} of that editable cluster until it is saturated, which means $|C_{\text{ins}}| = 2|C_{\text{core}}|$. If after the process, there are still clients in IC that are not added to any editable cluster, we add a new editable cluster $\langle S, \emptyset, \emptyset, IC'\rangle$ to $\mathcal{C}$, where IC' is the set of remaining clients in IC, and remove clients $j \in S$ from the existing editable clusters that intersect with S (i.e., add j to C_{del}).

To establish Lemma 16, we first show that the output "editable" clusters of Algorithm 4 always partition the set of clients. We then show that each output "editable" clusters is γ_{IC}-editable, following the original Definition 6. For simplicity, we will use the notation $\tau := \langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle$ to denote the 4-tuple of an editable cluster in the rest of this subsection.

The output partitions the clients. We first show that the output of Algorithm 4 partitions the clients into editable clusters. We start with the following observation, which consists of three direct consequences of the algorithm.

Observation 32. *Throughout Algorithm 4, for any $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle \in \mathcal{C}$,*

- *$C_{\text{out}} \subseteq \{j : C^I(j) = C_{\text{core}}\}$; and*
- *C_{core} and C_{out} are not modified after they are created, and we only add clients to C_{ins} and C_{del}.*

Further, once a client belongs to $C_{\text{core}} \setminus C_{\text{del}}$, C_{ins} or C_{out} of some editable cluster, it will always appear in those sets of some (possibly other) editable cluster.

Now, we are ready to show that the output of Algorithm 4 partitions the clients into editable clusters.

Lemma 33. *After Algorithm 4, sets in $\{C_{\text{core}} \setminus C_{\text{del}}\}_{\tau\in\mathcal{C}} \sqcup \{C_{\text{ins}}\}_{\tau\in\mathcal{C}} \sqcup \{C_{\text{out}}\}_{\tau\in\mathcal{C}}$ partition D.*

Algorithm 4 (Cluster pruning – the backward stage).

Initialization: the set of potential clusters $\mathcal{S}$ and pointers $C^I(\cdot)$. We maintain the set of editable clusters $\mathcal{C} = \{\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle\}$, which is initially empty.

Enumerating $S \in \mathcal{S}$ in the reverse order that they were added to $\mathcal{S}$, we do the following:

1. Compute U, the current set of unsettled clients that are not in any editable cluster, i.e.,
$$U = D \setminus \left(\bigcup_{\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle \in \mathcal{C}} (C_{\text{core}} \setminus C_{\text{del}}) \cup C_{\text{ins}} \cup C_{\text{out}} \right).$$
2. Let $IC = \{j \in U : C^I(j) = S\}$.
3. If $S \subseteq U$, we add a new editable cluster $\langle S, \emptyset, \emptyset, IC\rangle$ to $\mathcal{C}$, and continue to next S.
4. Otherwise, we first initiate $I = IC$. We then enumerate all current editable clusters $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle \in \mathcal{C}$ such that $C_{\text{core}} \setminus C_{\text{del}}$ contains some $i \in S$, and do the following for each of them (in any order):
   - If $|C_{\text{ins}}| + |I| \le 2|C_{\text{core}}|$, we update the current editable cluster to $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}} \cup I, C_{\text{out}}\rangle$, and set $I \leftarrow \emptyset$.
   - Otherwise, we choose $I' \subseteq I$ with $|I'| = 2|C_{\text{core}}| - |C_{\text{ins}}|$, arbitrarily, and update the current editable cluster to
   $$\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}} \cup I', C_{\text{out}}\rangle.$$
   Further, we set $I \leftarrow I \setminus I'$.
5. If after the above step, $I \neq \emptyset$, for every editable cluster $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle \in \mathcal{C}$ such that $C_{\text{core}} \setminus C_{\text{del}}$ contains some $i \in S$, we update it to $\langle C_{\text{core}}, C_{\text{del}} \cup (S \cap C_{\text{core}}), C_{\text{ins}}, C_{\text{out}}\rangle$, and finally, we add a new editable cluster $\langle S, \emptyset, \emptyset, I\rangle$ to $\mathcal{C}$.

Output $\mathcal{C}$.

Proof. We separately show that these sets are disjoint and cover D.

Disjointness. Our disjointness argument is that each client appears at most once in these multi-sets. We show this by induction on the number of iterations of Algorithm 4. Initially, $\mathcal{C} = \emptyset$ and $U = D$, so the statement holds. Suppose that we are now enumerating $S \in \mathcal{S}$ and the statement holds before this iteration. Let $IC = \{j \in U : C^I(j) = S\}$ as defined in the algorithm. Because we only set $C^I(j) = S$ for active clients after S is opened, we have $S \cap IC = \emptyset$.

Note that U is the set of clients that are not in any editable cluster, i.e., for any $i \in U$, there is no $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle \in \mathcal{C}$ such that $i \in C_{\text{core}} \setminus C_{\text{del}}$, $i \in C_{\text{ins}}$ or $i \in C_{\text{out}}$. Our definition of IC implies that $IC \subseteq U$. Therefore, if we add $\langle S, \emptyset, \emptyset, IC\rangle$ to $\mathcal{C}$ in step 3 (executed if $S \subseteq U$), then the statement still holds after this iteration.

Otherwise, in step 4, we only put clients in $I \subseteq IC \subseteq U$ into some C_{ins} of some editable cluster. The statement still holds after this step. Finally, in step 5, we put clients in $S \cap C_{\text{core}}$ into its C_{del} for every editable cluster whose $(C_{\text{core}} \setminus C_{\text{del}}) \cap S \neq \emptyset$, and then create a new editable cluster

$\langle S, \emptyset, \emptyset, I\rangle$. This ensures any $i \in S$ no longer belong to $\{C_{\text{core}} \setminus C_{\text{del}}\} \sqcup \{C_{\text{ins}}\} \sqcup \{C_{\text{out}}\}$ of any existing editable cluster before this step. Therefore, creating a new editable cluster $\langle S, \emptyset, \emptyset, I\rangle$ does not violate the statement.

Covering. Suppose that $\mathcal{S}'$ is the set of clusters $S \in \mathcal{S}$ that are already enumerated by Algorithm 4. We show that, throughout Algorithm 4, clients in the following set are covered by the sets in $\{\{C_{\text{core}} \setminus C_{\text{del}}\} \sqcup \{C_{\text{ins}}\} \sqcup \{C_{\text{out}}\}\}_{\tau \in \mathcal{C}}$:

$$\left(\bigcup_{S \in \mathcal{S}'} D_A(S)\right) \cup \{j \in D : C^I(j) \in \mathcal{S}'\} .$$

This proves that all clients are finally covered because a client i becomes inactive either because some opened cluster S has $D_A(S)$ including it, or because it pays for its connection cost to an opened cluster S and sets $C^I(i)$ to be S.

To prove this invariant, we use induction on the size of $\mathcal{S}'$. The base case is when $\mathcal{S}' = \emptyset$, in which case the set is empty and the statement holds trivially. For the induction step, suppose the statement holds for some $\mathcal{S}'$ and consider we are enumerating a new cluster $S \in \mathcal{S} \setminus \mathcal{S}'$.

If $S \subseteq U$, then we add $\langle S, \emptyset, \emptyset, IC\rangle$ to $\mathcal{C}$ in step 3, and the statement still holds after this iteration. Otherwise, we proceed to step 4 and step 5. If after step 4, $I \neq \emptyset$, we have added all clients in $IC \setminus I$ into some editable cluster and create a new cluster $\langle S, \emptyset, \emptyset, I\rangle$, which implies that all clients in $D_A(S) \cup IC$ are covered by some editable cluster. Next, we consider the case $I = \emptyset$ after step 4.

Suppose that v is the node corresponding to the cluster S in the dependency forest. Consider any $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle \in \mathcal{C}$ whose $C_{\text{core}} \setminus C_{\text{del}}$ contains some $i \in S$. Note that C_{core} is some opened cluster $S' \in \mathcal{S}'$ that is opened later than S. Therefore, S' is an available set at the time it is opened in Algorithm 3. Since $S' \cap S \neq \emptyset$, S' contains some clients in the sub-tree rooted at v in the dependency forest, because at the time S is opened in Algorithm 3, we add all trees whose nodes involve a client in S to be descendants of v. Hence, at least one client in S' appears in v or some descendant of v. Recall Remark 10 that S' must be the union of the involved clients of a upward-closed set in the dependency forest. Therefore, $D_A(S) \subseteq S'$, and at the time we create the editable cluster whose $C_{\text{core}} = S'$, all clients in $D_A(S)$ are already covered by it. Due to Observation 32, all clients in $D_A(S)$ are covered by some editable cluster created so far. Further, $I = \emptyset$ after step 4 means that all clients in IC are also covered by some editable cluster created so far. □

The output clusters are γ_{IC}-editable. For convenience, we restated the our definition of γ_{IC}-editable clusters in Definition 6: a sequence of sets $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}\rangle$ is said to be a γ_{IC}-editable cluster with respect to the dual solution $\{\alpha_i\}_{i \in D}$ if

1. $\langle C_{\text{core}}, C_{\text{out}}\rangle$ is a γ_{IC}-splittable cluster.
2. $C_{\text{del}} \subseteq C_{\text{core}}$, and $C_{\text{core}}, C_{\text{ins}}, C_{\text{out}}$ are disjoint sets.
3. $|C_{\text{ins}}| \leq 2|C_{\text{core}}|$ and $\sum_{i \in C_{\text{ins}}} \alpha_i \geq 2 \sum_{j \in C_{\text{del}}} \alpha_j$.
4. For each $i \in C_{\text{ins}}$, we have $\alpha_i \geq \gamma_{IC} \cdot d(i, C_{\text{core}})$.

We show that the output clusters of Algorithm 4 satisfy the definition. First of all, we show that the $(C_{\text{core}}, C_{\text{out}})$ pair of any output cluster is γ_{IC}-splitable (Definition 4, restated below):

1. $C_{\text{core}} \neq \emptyset$, and C_{core} and C_{out} are disjoint, i.e., $C_{\text{core}} \cap C_{\text{out}} = \emptyset$.
2. $\sum_{i \in C_{\text{core}}} \alpha_i \geq f + \mathsf{cost}(C_{\text{core}})$.

3. For each $i \in C_{\text{out}}$, we have $\alpha_i \geq \gamma_{IC} \sum_{j \in C_{\text{core}}} d(i,j)$ and $\alpha_i \geq \max_{j \in C_{\text{core}}} \alpha_j$.

Lemma 34. *For any $(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}) \in \mathcal{C}$, $(C_{\text{core}}, C_{\text{out}})$ is γ_{IC}-splitable.*

Proof. As we have shown in Lemma 33, the first condition of Definition 4 is satisfied. The second condition of Definition 4 follows from Lemma 27. The third condition of Definition 4 follows from the fact that we increase the dual variables uniformly, the fact that $C^I(j) = C_{\text{core}}$ for any $j \in C_{\text{out}}$ after C_{core} is opened and our condition of setting $C^I(j)$ to be C_{core} in Algorithm 3. □

For the second condition of Definition 6, it is clear that $C_{\text{del}} \subseteq C_{\text{core}}$ according to the rule we modify C_{del} in Algorithm 4 (only in step 5), and Lemma 33 shows that $C_{\text{core}}, C_{\text{ins}}, C_{\text{out}}$ are disjoint.

Next, we show the third condition of Definition 6 is satisfied by the output clusters of Algorithm 4 (as shown in Corollary 36). Before establishing the condition, we give the following characterization of the output clusters of Algorithm 4. As a result, the third condition follows from the characterization.

Lemma 35. *For any $(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}) \in \mathcal{C}$, at least one of the following properties is satisfied:*

- $|C_{\text{ins}}| \leq 2|C_{\text{core}}|$ *and* $C_{\text{del}} = \emptyset$*;*
- $|C_{\text{ins}}| = 2|C_{\text{core}}|$*,* $C_{\text{del}} \neq \emptyset$ *and* $\min_{j \in C_{\text{ins}}} \alpha^*_j \geq \max_{j \in C_{\text{del}}} \alpha^*_j$.

Proof. Note that in Algorithm 4, we only modify C_{del} when all "editable" clusters whose $C_{\text{core}}C_{\text{del}}$ intersects the current set S has C_{ins} of size $2|C_{\text{core}}|$, and the "editable" clusters whose C_{del} is modified are those whose $C_{\text{core}} \setminus C_{\text{del}}$ intersects the current set S. Therefore, if $|C_{\text{ins}}| < 2|C_{\text{core}}|$, then $C_{\text{del}} = \emptyset$. Note that our rule of updating C_{ins} in step 4 of Algorithm 4 ensures that $|C_{\text{ins}}| \leq 2|C_{\text{core}}|$. Next, we consider the case when $|C_{\text{ins}}| = 2|C_{\text{core}}|$ and $C_{\text{del}} \neq \emptyset$.

Suppose that when we enumerate the set S^*, we modify C_{ins} to be of size $2|C_{\text{core}}|$. Let θ^* be the time S^* is opened in Algorithm 3. According to Algorithm 4, we enumerate clusters $S \in \mathcal{S}$ in the reverse order that they were added to $\mathcal{S}$, and we only include add j whose appears in a cluster $S \in \mathcal{S}$ that is opened no later than S^* into C_{del}. Therefore, for any $j \in C_{\text{del}}$, we have $\alpha^*_j \leq \theta^*$. On the other hand, all clients i in C_{ins} are added to the editable cluster before or when we enumerate S^*, and thus their $C^I(i)$ is opened no earlier than S^*. Therefore, we have $\alpha^*_i \geq \theta^*$ for any $i \in C_{\text{ins}}$. Therefore, we have the desired property that $\min_{i \in C_{\text{ins}}} \alpha^*_i \geq \max_{j \in C_{\text{del}}} \alpha^*_j$. □

Corollary 36. *For any $(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}) \in \mathcal{C}$, we have $|C_{\text{ins}}| \leq 2|C_{\text{core}}|$ and $\sum_{j \in C_{\text{ins}}} \alpha^*_j \geq 2 \sum_{j \in C_{\text{del}}} \alpha^*_j$.*

The fourth condition of Definition 6 can be viewed as a direct corollary of Lemma 31. For the sake of completeness, we provide a short proof below.

Lemma 37. *Assume that $r_S \geq 4$ and $\gamma_{DC} \geq 2\gamma_{IC}$. For any $(C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}}) \in \mathcal{C}$ and any $i \in C_{\text{ins}}$, we have $\alpha^*_i \geq \gamma_{IC} d(i, C_{\text{core}})$.*

Proof. Suppose that $i \in C_{\text{ins}}$ is added to the editable cluster $\langle C_{\text{core}}, C_{\text{del}}, C_{\text{ins}}, C_{\text{out}} \rangle$ when we enumerate $S \in \mathcal{S}$ in Algorithm 4. This could only happen if $C^I(i) = S$ and $S \cap C_{\text{core}} \neq \emptyset$. Note that C_{core} is another opened cluster $S' \in \mathcal{S}$ opened later than S because of Algorithm 4 and Observation 32. Therefore, we can apply Lemma 31 to show that $\alpha^*_i \geq \gamma_{IC} d(i, S') = \gamma_{IC} d(i, C_{\text{core}})$. □

4.5 Dual Feasibility

Let $(\alpha_i^*)_{i \in D}$ be the final dual variables created by Algorithm 3. We show that there exists some constant Γ (depending on good choices of $\gamma_{DC}, \gamma_{IC}, r_S$) such that, for any $C \subseteq D$, we have $\sum_{i \in C} \alpha_i^* \leq f + \Gamma \cdot \mathsf{cost}(C)$.

Lemma 17 (Dual feasibility)**.** *Assume that $\Gamma = r_S \gamma_{DC}, \gamma_{DC} = r_S \gamma_{IC}$ and $r_S > 832$. Then, for any $C \subseteq D$, we have $\sum_{i \in C} \alpha_i^* \leq f + \Gamma \cdot \mathsf{cost}(C)$, where $\{\alpha_j^*\}_{j \in D}$ is the final dual solution created by Algorithm 3.*

To establish the dual feasibility (Lemma 17), we define a cluster to be *critical* if it satisfies the condition $\alpha_j^* > 2\Gamma \cdot d(j, C)$ for every $j \in C$. According to the observation below, it suffices to show dual feasibility for any critical cluster C.

Definition 38. *A cluster C is Γ-*critical *if for every $i \in C$, we have $\alpha_i^* > 2\Gamma \cdot d(i, C)$ for every $i \in C$.*

Observation 39. *Suppose we have the following guarantee: for any Γ-critical cluster C, we have $\sum_{i \in C} \alpha_i^* \leq f + \Gamma \cdot \mathsf{cost}(C)$. Then, we can show that for any C, we have $\sum_{i \in C} \alpha_i^* \leq f + \Gamma \cdot \mathsf{cost}(C)$.*

Proof. We can prove this observation by induction on the size of C. The base case is when $|C| = 1$, which results directly from the guarantee because $d(i, C) = 0$ for the only $i \in C$. Next, we consider any C and assume that our inductive hypothesis holds for any C' such that $|C'| < |C|$.

Suppose that there exists an $i \in C$ such that $\alpha_i^* \leq 2\Gamma d(i, C)$. (If there is no such i, then C is critical and we are done by the guarantee.) Because of our inductive hypothesis, we have $\sum_{j \in C \setminus \{i\}} \alpha_j^* \leq f + \Gamma \cdot \mathsf{cost}(C \setminus \{i\})$. Then, we have

$$\begin{aligned}
\sum_{j \in C} \alpha_j^* &= \sum_{j \in C \setminus \{i\}} \alpha_j^* + \alpha_i^* \\
&\leq f + \Gamma \cdot \mathsf{cost}(C \setminus \{i\}) + 2\Gamma d(i, C) && \text{(inductive hypothesis)} \\
&= f + \Gamma \cdot \mathsf{cost}(C)
\end{aligned}$$

Hence, we complete the inductive step and finish the proof. □

4.5.1 Frozen critical clusters

In this part, we show that if a critical cluster C has a client i such that the current point $Q(i)$ has a relatively small size compared to C, then all clients in C become inactive immediately (Corollary 41). We will say that cluster C is *frozen* by $Q(i)$. More generally, we first prove the following lemma, where any client $i \in C$ can freeze any other client $j \in C$ if α_i^* is relatively large compared to $d(i, C)$ and $Q(i)$ has a relatively small size compared to C.

Lemma 40. *Suppose that $\Gamma \geq 4\gamma_{IC}$ is a constant. Assume that $\gamma_{DC} \geq 6\gamma_{IC}$ and $r_S \geq 6$. Consider two clients i, j and time θ. Let C be any set of clients such that $i, j \in C$ and $\Gamma d(i, C) \leq \alpha_i^* \leq \theta$. If $Q(i) \neq \perp$ and $|Q(i)| \leq \frac{\Gamma}{4\gamma_{IC}} \cdot |C|$ at time θ, then we have $\alpha_j^* \leq \max\{\theta, \Gamma d(j, C)\}$.*

Proof. Because $Q(i)$ has been opened at time θ, we have $\alpha_i^* \leq \theta$ and $3\theta \geq \gamma_{DC} \cdot d(i, Q(i))$ (Corollary 24). Next, we assume that $\alpha_j^* > \theta$ and $\alpha_j^* > \Gamma d(j, C)$, and we will show that this leads to

a contradiction. Now, we fix S to be the cluster $Q(i)$ at time θ. Then, at any time $\theta' \in (\theta, \alpha_j^*)$, because $S \in \mathcal{S}$, we have

$$\begin{aligned}
\alpha_j^* &\le \gamma_{IC} d(j,S) \le \gamma_{IC}(d(i,S) + |S| \cdot d(i,j)) \\
&\le \gamma_{IC}(d(i,S) + |S| \cdot (d(i,C) + d(j,C))/|C|) \\
&< \gamma_{IC}\left(d(i,S) + \frac{\alpha_i^* + \alpha_j^*}{\Gamma} \cdot \frac{|S|}{|C|}\right) \\
&< \gamma_{IC}\left(d(i,S) + \frac{\alpha_j^*}{2\gamma_{IC}}\right) . && (\alpha_i^* \le \theta < \alpha_j^*)
\end{aligned}$$

This implies that

$$\frac{1}{2}\alpha_j^* < \gamma_{IC} d(i,S) \le \frac{3\gamma_{IC}}{\gamma_{DC}} \cdot \theta < \frac{3\gamma_{IC}}{\gamma_{DC}} \cdot \theta' .$$

Because of the assumption that $\gamma_{DC} \ge 6\gamma_{IC}$, we have that $\alpha_j^* < \theta'$, which is a contradiction. □

Corollary 41. *Suppose that $\Gamma \ge 4\gamma_{IC}$ is a constant. Assume that $\gamma_{DC} \ge 6\gamma_{IC}, r_S \ge 6$, and C is a Γ-critical cluster. If there exists a client $i \in C$ and time θ such that $Q(i) \neq \perp$ and $|Q(i)| \le \frac{\Gamma}{4\gamma_{IC}} \cdot |C|$ at time θ, then we have $\alpha_j^* \le \theta$ for every $j \in C$.*

4.5.2 Dual feasibility of available sets: an invariant throughout the algorithm

In this part, we first present a proof of the dual feasibility for available sets[10]. This will be used as a key lemma for Lemma 17. The first observation is that the set of available sets is monotone in time, i.e., available sets at a later time are also available at an earlier time.

Observation 42. *Let $\theta < \theta'$ be two time points in the algorithm and S be any subset of D. If S is available at time θ', then S is also available at time θ.*

Now, we present the dual feasibility of available sets. The proof sketch is as follows. Suppose that there exists an available set S at time θ violating the dual feasibility condition, we first clean S, θ so that S is available and at least one client in S is active at time $\theta - \varepsilon$ for sufficiently small $\varepsilon > 0$. Then, according to Lemma 40, we can show that all clients i in S with large α-values must satisfy $Q(i) = \perp$ or $|Q(i)| > r_s \cdot |S|$ at time $\theta - \varepsilon$ (the second condition of Definition 13). Therefore, we can further clean S by removing all clients i that does not satisfy the second condition of Definition 13, while still violating the dual feasibility condition. This finally leads to a contradiction to Corollary 30.

Lemma 43. *Assume that $\gamma_{DC} \ge \gamma_{IC} r_S$ and $r_S \ge 6$. Throughout the algorithm, for any available set S, we have*

$$\sum_{i \in S} \alpha_i \le f + 4\gamma_{DC} \cdot \mathsf{cost}(S) . \tag{20}$$

Proof. In this proof, we use $\alpha_i(\theta)$ to denote the α-value of client i at time θ in the algorithm. For the sake of contradiction, suppose that there exists an available set S such that $\sum_{i \in S} \alpha_i(\theta) > f + 4\gamma_{DC} \cdot \mathsf{cost}(S)$ at some time θ.

We first clean S and θ, maintaining the following invariants: (i) S is available at time $\theta - \varepsilon$ and (ii) $\sum_{i \in S} \alpha_i(\theta - \varepsilon) > f + 4\gamma_{DC} \cdot \mathsf{cost}(S)$, both for sufficnently small $\varepsilon > 0$. All invariants hold for the initial S and θ by our assumption. The cleaning is repeating the following steps.

[10] Available sets and Γ-critical sets are two incomparable concepts. Hence, this is not considered as a special case of the later Lemma 47

1. Let $\theta' = \max_{i \in S} \alpha_i(\theta)$. We set $\theta \leftarrow \theta'$ and don't modify S in this step. *Analysis:* This ensures that S remains available at time $\theta' - \varepsilon$ because of $\theta' \le \theta$ and Observation 42 for sufficiently small $\varepsilon > 0$. In addition, we have $\alpha_i(\theta) = \alpha_i(\theta')$ for every $i \in S$. Because $\varepsilon > 0$ is sufficently small, we still have $\sum_{i \in S} \alpha_i(\theta' - \varepsilon) > f + 4\gamma_{DC} \cdot \mathsf{cost}(S)$ at time $\theta' - \varepsilon$.

2. We find a client i with the maximum $\alpha_i(\theta)$ (breaking ties arbitrarily). If $\alpha_i(\theta) \le 8\gamma_{DC} \cdot d(i, S)$, we remove i from S and repeat these two steps. Otherwise, we stop. We don't modify θ in this step. *Analysis:* Consider i is picked in this step. Because step 1 ensures that $\alpha_i(\theta) = \theta$, we have $\theta - \varepsilon < \theta \le \alpha_i^*$. Therefore, the picked client i is still active at time $\theta - \varepsilon$. Therefore, after removing i, S is still available at time $\theta - \varepsilon$. Further the removal clearly ensures that $\sum_{i \in S} \alpha_i(\theta - \varepsilon) > f + 4\gamma_{DC} \cdot \mathsf{cost}(S)$ at time $\theta - \varepsilon$.

Let i^* be the picked client in step 2 of the last iteration of the cleaning. After the cleaning, we have $\alpha_{i^*}(\theta - \varepsilon) > 8\gamma_{DC} d(i^*, S)$ for a client $i^* \in S$ that is still active right before time θ, for sufficiently small $\varepsilon > 0$.

In the rest of the proof, we consider time $\theta - \varepsilon$, where $\varepsilon > 0$ is sufficiently small so that all previous invariants hold and the algorithm does not open any cluster or make any client inactive during time $[\theta - \varepsilon, \theta)$[11]. Let $(A, \mathcal{S}, \mathcal{V}, \mathcal{E})$ be the state of the algorithm at time $\theta - \varepsilon$. Let $Q(\cdot)$ be the pointers induced by this state. For simplicity, we use α_i to denote $\alpha_i(\theta - \varepsilon)$ for every $i \in D$ in the rest of the proof.

First, we show that at time $\theta - \varepsilon$, we have $Q(j) = \perp$ or $|Q(j)| > r_S \cdot |S|$ for every $j \in S$ such that $\alpha_j \ge 4\gamma_{DC} d(j, S)$ by contradiction. Suppose that $j \in S$ satisfies $Q(j) \neq \perp$, $|Q(j)| \le r_S \cdot |S|$ and $\alpha_j \ge 4\gamma_{DC} d(j, S)$ at time $\theta - \varepsilon$. Because $\frac{4\gamma_{DC}}{4\gamma_{IC}} \ge r_S$, according to Lemma 40, we have $\alpha_{i^*}^* \le \max\{\theta - \varepsilon, 4\gamma_{DC} d(i^*, S)\}$, which contradicts $\alpha_{i^*}^* > 8\gamma_{DC} d(i^*, S)$ and $\alpha_{i^*}^* \ge \theta$.

Second, we consider $S' = S \setminus T$, where $T = \{j \in S : Q(j) \neq \perp, |Q(j)| \le r_S \cdot |S|\}$. It is clear that S' satisfies the second condition of Definition 13 at time $\theta - \varepsilon$. Also, if $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$ for some $i, j \in S'$, then we have $Q(i) = Q(j)$. It is impossible to have $i, j \in S$ so that $i \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} j$, and i remains in S' but j does not. Therefore, S' is still available at time $\theta - \varepsilon$ and satisfies the third condition of Definition 13. According to Corollary 30, we have $\sum_{i \in S'} \alpha_i \le f + 4\gamma_{DC} \cdot \mathsf{cost}(S')$ at time $\theta - \varepsilon$. However, because of our earlier discussion, we have $\alpha_j < 4\gamma_{DC} d(j, S)$ for every $j \in T$ at time $\theta - \varepsilon$, implying that

$$\begin{aligned}\sum_{i \in S'} \alpha_i = \sum_{i \in S} \alpha_i - \sum_{i \in T} \alpha_i &> f + 4\gamma_{DC} \cdot \mathsf{cost}(S) - \sum_{i \in T} 4\gamma_{DC} d(i, S) \\ &\ge f + 4\gamma_{DC} \cdot \mathsf{cost}(S') \ ,\end{aligned}$$

where we find a contradiction. □

4.5.3 On opening times of the clusters

In this part, we show that the time when S is opened is at most in the order of $f/|D_A(S)|$ (Corollary 45). This further implies that for sufficiently large Γ, for any Γ-critical cluster C, if a client i whose point $Q(i)$ has a large proportion being active clients at the time it is opened, then we can simply upper bound the sum of α_j^* for all $j \in C$ by f (Lemma 46).

Lemma 44. *For each $S \in \mathcal{S}$, we have $\sum_{j \in D_A(S)} \alpha_j^* \le 2f$.*

[11] This is possible because each client can only become inactive once and we have shown in Corollary 29 that the number of opened clusters is bounded. There are only finitely many such times.

Proof. The proof is clear if S is a singleton returned by the first step of Algorithm 2. Next, we consider S is returned after the third step of Algorithm 2.

Let T be a cluster that is found by the enumeration of (s, c) in Algorithm 2 and A be the set of active clients right before S is opened. Let T' be the cluster after the first step of the cleaning process. It is clear that we terminate the first step of the cleaning process when $\sum_{j\in T'\setminus A}\alpha_j \geq \gamma'\sum_{j\in T'\setminus A} d(j,T')$ and $\forall j \in T'\cap A, \alpha_j \geq 2\gamma' d(j,T')$ for $\gamma' = (\sum_{j\in T'}\alpha_j - f)/\mathsf{cost}(T')$.

$$\begin{aligned}\sum_{j\in T'\cap A}\alpha_j = \sum_{j\in T'}\alpha_j - \sum_{j\in T'\setminus A}\alpha_j &= f + \gamma'\cdot\mathsf{cost}(T') - \sum_{j\in T'\setminus A}\alpha_j \\ &\leq f + \gamma'\cdot\mathsf{cost}(T') - \gamma'\sum_{j\in T'\setminus A} d(j,T') \\ &= f + \gamma'\cdot\sum_{j\in T'\cap A} d(j,T') \leq f + \frac{1}{2}\sum_{j\in T'\cap A}\alpha_j\ ,\end{aligned}$$

and thus we have completed the proof by rearranging the terms in the above inequality. □

Corollary 45. *Suppose that $S \in \mathcal{S}$ is opened at time θ. Then, we have $D_A(S) = \emptyset$ or $\theta \leq 2f/|D_A(S)|$.*

Lemma 46. *Let $\Gamma \geq r_S\gamma_{IC}$ be a constant. Assume that $\gamma_{DC} \geq 6\gamma_{IC}$ and $r_S \geq 6$. Consider any Γ-critical cluster C. If there exists a client $i \in C$ at some time θ such that $Q(i) \neq \perp$, $|Q(i)| \geq r_S\cdot|C|$ and $|D_A(Q(i))| > 4|Q(i)|/r_S$. Then, we have $\sum_{i\in C}\alpha_i^* \leq f$.*

Proof. Let S be the clsuter $Q(i)$ at time θ and let θ' be the time when S is opened. We clearly has $\theta' \leq \theta$ because $Q(i)$ has been opened at time θ. Because we have $|D_A(S)| > 4|S|/(r_S)$, according to Corollary 45, we have

$$\theta' \leq \frac{2f}{|D_A(S)|} < \frac{r_S f}{2|S|} \leq \frac{f}{2|C|}\ . \tag{21}$$

Because C is critical, we have $\alpha_i^* > 2\Gamma d(i,C)$. Because i is inactive (as $Q(i) \neq\perp$) at time θ', we have $\alpha_i^* \leq \theta'$ and thus $2\Gamma d(i,C) < \theta'$. According to Corollary 24 and our assumption on r_S, we have $d(i,S) \leq \frac{3\theta'}{\gamma_{DC}}$. For any $j \in C$, because of the triangle inequality, we have $d(j,S) \leq |S|\cdot d(j,i) + d(i,S)$. Therefore, we have

$$\begin{aligned}\sum_{j\in C} d(j,S) &\leq \sum_{j\in C}(|S|\cdot d(j,i) + d(i,S)) \\ &\leq |S|\cdot d(i,C) + |C|\cdot d(i,S) \\ &< \left(\frac{|S|}{2\Gamma} + \frac{3|C|}{\gamma_{DC}}\right)\cdot\theta'\end{aligned}$$

Now, we discuss two cases for each $j \in C$: (i) j is active right at the end of time θ', or (ii) j is inactive at the end of time θ'. In case (i), we have $\alpha_j^* \leq \max(\theta', \gamma_{IC}d(j,S))$ because S has been opened at time θ' and j is still active right before time θ'. In case (ii), we simply have $\alpha_j^* \leq \theta'$.

Putting these two cases together, we have $\alpha_j^* \le \theta' + \gamma_{IC} d(j,S)$ for every $j \in C$. Therefore, we have

$$\begin{aligned}\sum_{j\in C} \alpha_j^* &\le |C| \cdot \theta' + \gamma_{IC}\left(\frac{|S|}{2\Gamma} + \frac{3|C|}{\gamma_{DC}}\right)\cdot \theta' \\ &\le \frac{f}{2} + \frac{r_S \gamma_{IC} f}{4\Gamma} + \frac{3\gamma_{IC} f}{2\gamma_{DC}} && \text{(Equation (21))} \\ &\le \frac{f}{2} + \frac{f}{4} + \frac{f}{4} = f && \text{(our assumptions about } \Gamma, \gamma_{DC} \text{ and } r_S)\end{aligned}$$

Therefore, we have completed the proof. □

4.5.4 Dual feasibility of any critical cluster

We are going to show the following lemma. Together with Observation 39, it proves Lemma 17.

Lemma 47. *Assume that $\Gamma = r_S\gamma_{DC}, \gamma_{DC} = r_S\gamma_{IC}$ and $r_S > 832$. For any Γ-critical set C, we have $\sum_{i\in C} \alpha_i^* \le f + \Gamma \cdot \mathsf{cost}(C)$.*

For the sake of contradiction, suppose that there exists a Γ-critical set C such that $\sum_{i\in C} \alpha_i^* > f + \Gamma \cdot \mathsf{cost}(C)$. Without loss of generality, we assume that $C = [s]$ for some integer s and we have $\alpha_1^* \ge \cdots \ge \alpha_s^*$.

Let θ be $\max_{i\in C} \alpha_i^* - \varepsilon$ for some sufficiently small $\varepsilon > 0$ so that

- no cluster is opened and no client becomes inactive at time θ;
- the cluster C is still Γ-critical at time θ; and
- at time θ, we have $\sum_{i\in C} \alpha_i > f + \Gamma \cdot \mathsf{cost}(C)$.

In addition, because we have $\alpha_i^* \le f$ for every $i \in C$ and thus $\theta < f$.

For the rest of this part, we will use α_i to denote the α-value of client i at time θ in the algorithm. We assume that v_i is node involving client i at time θ. We will use $Q(i)$ to denote the current point of each client $i \in [s]$ at time θ.

By our definition, there exists one client $i \in C$ such that $\alpha_i^* > \theta$. If there exists a client $j \in C$ such that $Q(j) \neq \perp$ and $|Q(j)| \le r_S \cdot |C|$, because of Corollary 41 and because our assumption ensures $\Gamma \ge 4 r_S \gamma_{IC}$, we would $\alpha_i^* \le \theta$ for every $i \in C$, which contradicts $\alpha_i^* > \theta$ for some $i \in C$. Therefore, we have $Q(j) = \perp$ or $|Q(j)| > r_S \cdot |C|$ for every $j \in C$ at time θ.

Let $\mathcal{V}_{\text{anc}}$ be the set of nodes in the dependency forest that are equal to or ancestors of some node involving the clients in C, and has a non-empty set of involved clients, i.e.,

$$\mathcal{V}_{\text{anc}} = \{v \in \mathcal{V} : D_{\mathcal{V}}(v) \neq \emptyset \text{ and } \exists i \in [s], v \text{ is equal to or an ancestor of } v_i \text{ in } (\mathcal{V}, \mathcal{E})\}.$$

This definition ensures that for any $v \in \mathcal{V}_{\text{anc}}$ and any $i \in D_{\mathcal{V}}(v)$, there exists $j \in C$ such that $j \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} i$. Consider any $v \in \mathcal{V}_{\text{anc}}$. If $S_{\mathcal{V}}(v) \neq \perp$ and $|D_{\mathcal{V}}(v)| > 4|S_{\mathcal{V}}(v)|/r_S$, applying Lemma 46 at the time $S_{\mathcal{V}}(v)$ is opened, we have $\sum_{i\in C} \alpha_i^* \le f$, contradicting our assumption that $\sum_{i\in C} \alpha_i^* > f + \Gamma \cdot \mathsf{cost}(C)$. Therefore, we have either $S_{\mathcal{V}}(v) = \perp$ or $|D_{\mathcal{V}}(v)| \le 4|S_{\mathcal{V}}(v)|/r_S$ for every $v \in \mathcal{V}_{\text{anc}}$.

Constructing an available set at time θ. We consider the following process to construct another cluster C^*: we initiate an empty set $C^* \leftarrow \emptyset$, a final node $v^* = \perp$ and let $\Gamma' = \Gamma/42$ be a constant. We enumerate $v \in \mathcal{V}_{\text{anc}}$ in the decreasing order of α_i for $i \in D_{\mathcal{V}}(v)$ (which is the same independent of the choice of i), breaking ties by putting an ancestor prior to a descendant and then arbitrarily:

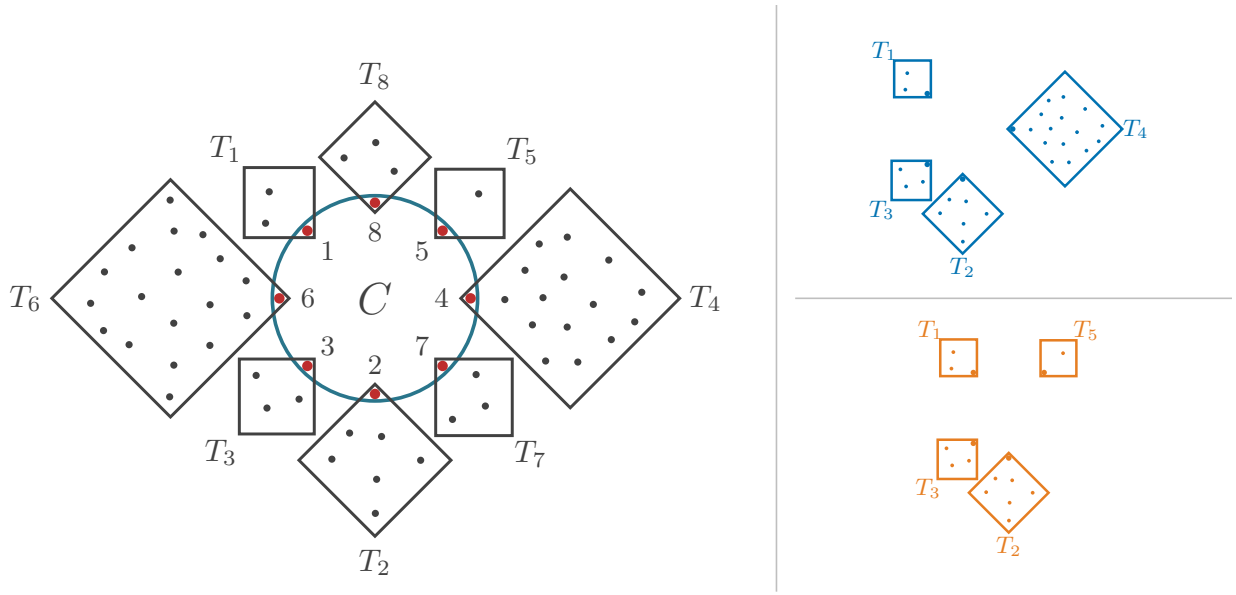


Figure 3: An example of constructing an available set from a Γ-critical cluster C. On the left, C consists of eight clients, labeled $1, \ldots, 8$ in nonincreasing order of their α-values, and T_i is the set of clients equivalent to client i. If $\alpha_4 \geq \frac{\Gamma}{42} \cdot d(4, T_1 \cup T_2 \cup T_3)$, the top-right panel gives the corresponding available set; otherwise, the bottom-right panel gives the corresponding available set.

1. If an ancestor v' of v is already enumerated but we skipped it, we skip v and continue the enumeration.
2. If, for any client $i \in [s]$ whose node v_i is equal to or a descendant of v in the dependency forest, we have $\alpha_i < \Gamma' d(i, C^*)$, then we skip v and continue the enumeration.
3. If $|C^*| + |D_{\mathcal{V}}(v)| \leq 2|C|$, we set $C^* \leftarrow C^* \cup D_{\mathcal{V}}(v)$. Otherwise, we record $v^* = v$ and stop the enumeration.

Finally, if we finish the enumeration with $v^* \neq \perp$, we return $C^* \cup D_{\mathcal{V}}(v^*)$, and otherwise we return C^*. In either case, the returned set is the union of $D_{\mathcal{V}}(v)$ for v in some subset $\mathcal{V}' \subseteq \mathcal{V}$ that is upward closed. (The set $\mathcal{V}'$ consists of all enumerated nodes v that are not skipped, v^* if $v^* \neq \perp$, and all ancestors of these nodes whose set of involved clients is empty.) Therefore, the returned set is available at time θ according to Remark 10.

The available set violates the dual feasibility (Lemma 43). First, we consider any $i \in C^*$. Due to our definition, there exists a client $j \in C$ such that $j \preceq_{\mathcal{P}(\mathcal{V},\mathcal{E})} i$. Since C is Γ-critical, we hav $\alpha_j > 2\Gamma d(j, C)$. If $j = i$, then we have $\alpha_i > 2\Gamma d(i, C)$. Otherwise, $Q(i) = Q(j) \neq \perp$ and $|Q(i)| > r_S \cdot |C|$ at time θ. According to Lemma 25, we have

$$\alpha_i \geq \frac{2\Gamma}{1 + \frac{6\Gamma}{r_S \gamma_{DC}}} \cdot d(i, C) = \frac{2\Gamma}{7} \cdot d(i, C). \tag{22}$$

Note that we can upper bound $\mathsf{cost}(C^* \setminus C)$ as follows:

$$\mathsf{cost}(C^* \setminus C) = \sum_{i,j \in C^* \setminus C} d(i,j) \leq \sum_{i,j \in C^* \setminus C} \frac{d(i,C) + d(j,C)}{|C|} \leq \frac{2|C^* \setminus C|}{|C|} \sum_{i \in C^* \setminus C} d(i,C) \leq 4 \sum_{i \in C^* \setminus C} d(i,C) \ ,$$

where the last inequality follows our algorithm always guarantees $|C^*| \leq 2|C|$. Furthermore, we can upper bound $\mathsf{cost}(C \cup C^*)$ as follows:

$$\mathsf{cost}(C \cup C^*) = \mathsf{cost}(C) + 2 \sum_{i \in C^* \setminus C} d(i,C) + \mathsf{cost}(C^* \setminus C) \leq \mathsf{cost}(C) + 6 \sum_{i \in C^* \setminus C} d(i,C) \ .$$

According to our assumption that $\sum_{i \in C} \alpha_i > f + \Gamma \cdot \mathsf{cost}(C)$, we have

$$\sum_{i \in C \cup C^*} \alpha_i > f + \Gamma \cdot \mathsf{cost}(C) + \frac{2\Gamma}{7} \sum_{i \in C^* \setminus C} d(i,C) > f + \frac{\Gamma}{21} \cdot \mathsf{cost}(C \cup C^*) \ . \tag{23}$$

Furthermore, we have

$$\begin{aligned} \sum_{i \in C \cup C^*} \alpha_i &> f + \Gamma \cdot \mathsf{cost}(C) + \frac{\Gamma}{7} \sum_{i \in C^* \setminus C} d(i,C) + \frac{1}{2} \sum_{i \in C^* \setminus C} \alpha_i \\ &\geq f + \frac{\Gamma}{42} \cdot \mathsf{cost}(C \cup C^*) + \frac{1}{2} \sum_{i \in C^* \setminus C} \alpha_i \ . \end{aligned} \tag{24}$$

We first consider the case where $\upsilon^* = \perp$ and the available set we construct is C^*. In this case, we finish the enumeration of all nodes in $\mathcal{V}_{\mathrm{anc}}$, during which we skip several nodes υ because of the first two steps in our enumeration process. Consider any $i \in C \setminus C^*$. It is clear that we skip the enumeration of υ_i because of the first two steps of our process. Let υ be the topmost node in the dependency forest among υ_i's ancestors and υ_i itself such that we skip the enumeration of υ in either of the first two steps of our process. It is clear that we skip the enumeration of υ because of step 2 of our process because no ancestor of υ is skipped. Because of the condition in step 2 and the fact that we only add new clients into C^*, we have $\alpha_i < \Gamma' d(i, C^*) \leq \frac{\Gamma}{21} d(i, C \cup C^*)$. By removing $C \setminus C^*$ from $C \cup C^*$, because of Equation (23), we have $\sum_{i \in C^*} \alpha_i > f + \frac{\Gamma}{21} \cdot \mathsf{cost}(C^*)$, which violates the dual feasibility of C^* according to Lemma 43 and our assumption that $\Gamma = r_S \gamma_{DC}$ and $r_S \geq 832$.

Next, we consider the case where $\upsilon^* \neq \perp$ and the available set we construct is $C^* \cup D_{\mathcal{V}}(\upsilon^*)$. Because C^* is the union of $D_{\mathcal{V}}(\upsilon)$ for some υ enumerated before υ^*, we have $C^* \cap D_{\mathcal{V}}(\upsilon^*) = \emptyset$. According to the definition of step 2 of our process, there exists a client i such that υ_i is equal to or a descendant of υ^* such that $\alpha_i \geq \frac{\Gamma}{42} d(i, C^*)$. If υ^* is a singleton corresponding to the client i, we have $\alpha_i \geq \frac{\Gamma}{42} d(i, C^*)$. Otherwise, we have $Q(i) \neq \perp$ and $|Q(i)| \geq r_S \cdot |C|$ at time θ. Again, applying Lemma 25 and using the fact that $|C^*| \leq 2|C| \leq 2|Q(i)|/r_S$, for any $j \in D_{\mathcal{V}}(\upsilon^*)$, we have

$$\alpha_j \geq \frac{\Gamma/42}{1 + \frac{\Gamma/14}{r_S \gamma_{DC}/2}} \cdot d(j, C^*) = \frac{\Gamma}{48} \cdot d(j, C^*) \ .$$

We upper bound $\mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*))$ as follows:

$$\begin{aligned}\mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*)) &= \mathsf{cost}(C^*) + \mathsf{cost}(D_{\mathcal{V}}(v^*)) + 2\sum_{i \in D_{\mathcal{V}}(v^*)} d(i, C^*) \\ &\le \mathsf{cost}(C^*) + \mathsf{cost}(D_{\mathcal{V}}(v^*)) + \frac{96}{\Gamma}\sum_{i \in D_{\mathcal{V}}(v^*)} \alpha_i \ .\end{aligned}$$

Consider $\mathsf{cost}(D_{\mathcal{V}}(v^*))$. If v^* is a singleton corresponding to a client i, we have $\mathsf{cost}(D_{\mathcal{V}}(v^*)) = 0$, and thus $\mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*)) \le \mathsf{cost}(C^*) + \frac{96}{\Gamma}\sum_{i \in D_{\mathcal{V}}(v^*)} \alpha_i$. Otherwise, because of our earlier discussion showing that $|D_{\mathcal{V}}(v^*)| \le 4|S_{\mathcal{V}}(v^*)|/r_S$, we have

$$\begin{aligned}\mathsf{cost}(D_{\mathcal{V}}(v^*)) = \sum_{i,j \in D_{\mathcal{V}}(v^*)} d(i,j) &\le \sum_{i,j \in D_{\mathcal{V}}(v^*)} \frac{d(i, S_{\mathcal{V}}(v^*)) + d(j, S_{\mathcal{V}}(v^*))}{|S_{\mathcal{V}}(v^*)|} \\ &\le \frac{2|D_{\mathcal{V}}(v^*)|}{|S_{\mathcal{V}}(v^*)|}\sum_{i \in D_{\mathcal{V}}(v^*)} d(i, S_{\mathcal{V}}(v^*)) \le \frac{8}{r_S}\sum_{i \in D_{\mathcal{V}}(v^*)} d(i, S_{\mathcal{V}}(v^*)) \ .\end{aligned}$$

Because $S_{\mathcal{V}}(v^*)$ is an opened cluster, according to Observation 20, we have $\alpha_j \ge \gamma_{DC} d(j, S_{\mathcal{V}}(v^*))$ for every $j \in D_{\mathcal{V}}(v^*)$. Therefore, we have

$$\mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*)) \le \mathsf{cost}(C^*) + \left(\frac{8}{r_S \gamma_{DC}} + \frac{96}{\Gamma}\right)\sum_{i \in D_{\mathcal{V}}(v^*)} \alpha_i = \mathsf{cost}(C^*) + \frac{104}{\Gamma}\sum_{i \in D_{\mathcal{V}}(v^*)} \alpha_i \ . \quad (25)$$

Let θ^* be the time at which we create the node v. Let C' denote the set of clients $i \in C$ such that v_i has not been enumerated (if a node is skipped, we consider it as enumerated). This implies that $C' \cap C^* = \emptyset$ and $C' \cap D_{\mathcal{V}}(v^*) = \emptyset$. Further, we let $C'' = D_{\mathcal{V}}(v^*) \cap C$. For each $i \in C$ such that $i \notin C' \cup C^* \cup D_{\mathcal{V}}(v^*)$, we have enumerated the node v_i but we don't include $D_{\mathcal{V}}(v_i)$ into C^* because of the first two steps of our process. This implies that $\alpha_i < \frac{\Gamma}{42} \cdot d(i, C^*)$. Combined with Equation (24), we have

$$\begin{aligned}\sum_{i \in C' \cup C'' \cup C^*} \alpha_i &= \sum_{i \in C \cup C^*} \alpha_i - \sum_{i \in C \setminus (C' \cup C'' \cup C^*)} \alpha_i \\ &\ge f + \frac{\Gamma}{42} \cdot \mathsf{cost}(C \cup C^*) + \frac{1}{2}\sum_{i \in C^* \setminus C} \alpha_i - \frac{\Gamma}{42}\sum_{i \in C \setminus (C' \cup C'' \cup C^*)} d(i, C^*) \\ &\ge f + \frac{\Gamma}{42} \cdot \mathsf{cost}(C' \cup C'' \cup C^*) + \frac{1}{2}\sum_{i \in C^* \setminus C} \alpha_i \ .\end{aligned}$$

Further, we note that

$$\begin{aligned}\sum_{i \in C^* \cup D_{\mathcal{V}}(v^*)} \alpha_i &= \sum_{i \in C' \cup C'' \cup C^*} \alpha_i + \sum_{i \in D_{\mathcal{V}}(v^*) \setminus C''} \alpha_i - \sum_{i \in C'} \alpha_i \\ &\ge f + \frac{\Gamma}{42} \cdot \mathsf{cost}(C' \cup C'' \cup C^*) + \frac{1}{2}\sum_{i \in C^* \setminus C} \alpha_i + \sum_{i \in D_{\mathcal{V}}(v^*)} \alpha_i - \sum_{i \in C' \cup C''} \alpha_i \\ &\ge f + \frac{\Gamma}{42} \cdot \mathsf{cost}(C' \cup C'' \cup C^*) + \frac{\Gamma}{208}\left(\mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*)) - \mathsf{cost}(C^*)\right) \\ &\quad + \frac{1}{2}\sum_{i \in (C^* \setminus C) \cup D_{\mathcal{V}}(v^*)} \alpha_i - \sum_{i \in C' \cup C''} \alpha_i \qquad \text{(Equation (25))} \\ &\ge f + \frac{\Gamma}{208} \cdot \mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*)) + \frac{1}{2}\sum_{i \in (C^* \setminus C) \cup D_{\mathcal{V}}(v^*)} \alpha_i - \sum_{i \in C' \cup C''} \alpha_i \ .\end{aligned}$$

Note that we enumerate v in decreasing order of the α-value of the clients in $D_{\mathcal{V}}(v)$ in the dependency forest. We have $\forall i \in C^* \cup D_{\mathcal{V}}(v^*), \forall j \in C' \cup C'', \alpha_i \geq \alpha_j$. Further, note that we terminate the enumeration at v^* because $|C^*| + |D_{\mathcal{V}}(v^*)| > 2|C|$. Therefore, we have

$$\begin{aligned}
|(C^* \setminus C) \cup D_{\mathcal{V}}(v^*)| &= |C^* \setminus C| + |D_{\mathcal{V}}(v^*)| \\
&= |C^*| - |C \cap C^*| + |D_{\mathcal{V}}(v^*)| \\
&\geq |C^*| - (|C| - |C'| - |C''|) + |D_{\mathcal{V}}(v^*)| \qquad (C^* \cap (C' \cup D_{\mathcal{V}}(v^*)) = \emptyset) \\
&\geq 2|C| - (|C| - |C'| - |C''|) \\
&= |C| + |C'| + |C''| \geq 2(|C'| + |C''|) \ ,
\end{aligned}$$

and thus $\frac{1}{2}\sum_{i \in (C^* \setminus C) \cup D_{\mathcal{V}}(v^*)} \alpha_i - \sum_{i \in C' \cup C''} \alpha_i \geq 0$. Since we choose $\Gamma = r_S \gamma_{DC}$ and $r_S > 832$,

$$\sum_{i \in C^* \cup D_{\mathcal{V}}(v^*)} \alpha_i \geq f + \frac{\Gamma}{208} \cdot \mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*)) > f + 4\gamma_{DC} \cdot \mathsf{cost}(C^* \cup D_{\mathcal{V}}(v^*)) \ .$$

The last inequality holds because we can't have $C^* \cup D_{\mathcal{V}}(v^*)$ as a singleton to satisfy the first inequality above (as $\alpha_i < f$) and the metric space ensures that the distance between any two distinct points is positive. We find a violation of the dual feasibility of $C^* \cup D_{\mathcal{V}}(v^*)$ according to Lemma 43.

5 Bipoint Rounding

In this section, we present our bipoint rounding algorithm. Let us define a k'-solution as a partition $\mathcal{S} = \{S_1, \ldots, S_{k'}\}$ of the point set D with k' subsets, which we call clusters.

Theorem 3. *There is a polynomial-time randomized algorithm, that, when given a k_1-solution $\mathcal{S}_1$ and a k_2-solution $\mathcal{S}_2$ with $k_1 \leq k < k_2$, outputs a feasible k-solution $\mathcal{S}$ of expected cost at most*

$$108 \cdot \left(\frac{k_2 - k}{k_2 - k_1} \cdot \mathsf{cost}(\mathcal{S}_1) + \frac{k - k_1}{k_2 - k_1} \cdot \mathsf{cost}(\mathcal{S}_2) \right) \ .$$

Notation. For simplicity, we define the distance $d(i, S) = \sum_{j \in S} d(i, j)$ and $d(S, T) = \sum_{i \in S} d(i, T)$ for any subsets $S, T \subseteq D$. The cost $\mathsf{cost}(C)$ of any cluster $C \subseteq D$ is defined as $\mathsf{cost}(C) = d(C, C)$. We let $C_i(j)$ be the cluster in $\mathcal{S}_i$ containing client $j \in D$, and set $s_i(j) = |C_i(j)|$. We let $\alpha = \frac{k_2 - k}{k_2 - k_1}$. W.l.o.g. we can assume that $\alpha \leq 1/2$, since otherwise $\mathcal{S}_1$ is already a 2-approximate solution for the bipoint objective with $|\mathcal{S}_1| = k_1 \leq k$; splitting it into exactly k nonempty clusters cannot increase the cost.

5.1 Partition the clusters and clients into good and bad

We partition the clusters and clients into good and bad based on the sizes of their clusters in the two solutions. For each client $j \in D$, we compare $s_1(j)$ and $s_2(j)$. If $s_1(j) \geq s_2(j)$, we say that j is a *good* client; otherwise, it is a *bad* client. We say that a cluster S is good if all clients in S are good, and bad otherwise. We use $\mathcal{S}_{1,G}, \mathcal{S}_{2,G}$ to denote the set of good clusters in $\mathcal{S}_1$ and $\mathcal{S}_2$, and $\mathcal{S}_{1,B}, \mathcal{S}_{2,B}$ to denote the set of bad clusters in $\mathcal{S}_1$ and $\mathcal{S}_2$. We use $\mathsf{Good}(S)$ and $\mathsf{Bad}(S)$ to denote the set of good and bad clients in a cluster S, respectively. On a high level, it is useful to think that we can reroute good clients from their $\mathcal{S}_1$ cluster to their $\mathcal{S}_2$ cluster cheaply, and we can reroute bad clients from their $\mathcal{S}_2$ to their $\mathcal{S}_1$ cluster cheaply.

5.2 An overview of the final solution

In our algorithm, we consider the following general scheme to produce k-solutions. Some steps are not fully defined yet, and we will define them later in this section.

- We open k' clusters in $\mathcal{S}_2$ (denoted by $\mathcal{S}_{2,O}$) and $k-k'$ clusters in $\mathcal{S}_1$ (denoted by $\mathcal{S}_{1,O}$) for some k' to be determined later. If a client appears in two opened clusters (i.e., one in $\mathcal{S}_1$ and one in $\mathcal{S}_2$), we assign the client to the cluster in $\mathcal{S}_2$.
- For each good client $j \in D$ not appearing in any opened cluster, we assign j to some opened cluster $C_2(j')$, where $j' \in \mathsf{Good}(C_1(j))$. We will use $A(j)$ to denote the assignment.
- For each bad client $j \in D$ not appearing in any opened cluster, we assign j to some opened cluster $C_1(j')$, where $j' \in \mathsf{Bad}(C_2(j))$[12]. We will use $A(j)$ to denote the assignment.
- For each opened cluster $S \in \mathcal{S}_2$, we include the cluster $S \cup \{j : A(j) = S\}$ in the final solution. Let D_2 be the union of all opened clusters in $\mathcal{S}_2$. For each opened cluster $S \in \mathcal{S}_1$, we include the cluster $(S \setminus D_2) \cup \{j : A(j) = S\}$ in the final solution.

To analyze our algorithm, we will show the following key arguments.

- Every $\mathcal{S}_2$ cluster opens with significant probability. Concretely, for any fixed cluster $S_2 \in \mathcal{S}_2$, the probability that S_2 is not opened is at most $O(\alpha)$.
- For each good client i and each cluster $S_2 \in \mathcal{S}_2$, the probability that i is reassigned to S_2 by the matching rule is at most $O(\alpha) \cdot \frac{|\mathsf{Good}(C_1(i)) \cap S_2|}{|\mathsf{Good}(C_1(i))|}$. For the residual rule, conditioning on ℓ_M, ℓ_L, an analogous bound holds with $\mathsf{Good}(C_1(i))$ replaced by $\mathsf{Good}(C_1(i)) \cap D^{(\mathrm{res})}$ whenever this set is nonempty, where $D^{(\mathrm{res})}$ is the set of residual clients defined below. In words, a good client is likely to go to S_2 if a large fraction of the relevant good clients in its $\mathcal{S}_1$ cluster intersect S_2. This extends to each pair of good clients i, j.
- Likewise, for each bad client i and each cluster $S_1 \in \mathcal{S}_1$, the probability that i is reassigned to S_1 is at most $O(\alpha) \cdot \frac{|\mathsf{Bad}(C_2(i)) \cap S_1|}{|\mathsf{Bad}(C_2(i))|}$. This also similarly extends to each pair of bad clients i, j.

In the next section, we start by defining two maps, one from $\mathcal{S}_1$ to $\mathcal{S}_2$ and the other from $\mathcal{S}_{2,B}$ to $\mathcal{S}_1$, which decide where the clients from that cluster will be rerouted if the cluster is closed. To obtain the probability guarantees discussed above, we choose these maps in a natural way by uniform sampling.

5.3 Preprocessing

We construct a matching labeling $\ell_M : \mathcal{S}_1 \to \mathcal{S}_2 \cup \{\perp\}$ that labels an $\mathcal{S}_2$-cluster for each cluster in $\mathcal{S}_1$ with at least a good client, and a linking labeling $\ell_L : \mathcal{S}_{2,B} \to \mathcal{S}_1$ for each bad cluster in $\mathcal{S}_2$. All samples used to define the two labelings are mutually independent.

- For each cluster $S_1 \in \mathcal{S}_1$ with $\mathsf{Good}(S_1) \neq \emptyset$, we sample a uniformly random good client $j \sim \mathsf{Good}(S_1)$ and label $\ell_M(S_1) = C_2(j)$. For other clusters $S_1 \in \mathcal{S}_1$ with $\mathsf{Good}(S_1) = \emptyset$, we simply define $\ell_M(S_1) = \perp$.

[12] In both cases, we will make sure that such clusters $C_1(j')$ and $C_2(j')$ are indeed open for some valid j'

- For each cluster $S_2 \in \mathcal{S}_{2,B}$, we sample a uniformly random bad client $j \sim \mathsf{Bad}(S_2)$ and label $\ell_L(S_2) = C_1(j)$. In particular, we will call there is a link from S_2 to $C_1(j)$.

We present our deterministic Algorithm 5 that pre-processes the two solutions given the random labeling, so that at the end of this pre-processing, there is a partition: $(\mathcal{S}_1^{(\text{match})}, \mathcal{S}_2^{(\text{match})})$ and $(\mathcal{S}_1^{(\text{res})}, \mathcal{S}_2^{(\text{res})})$ of the clusters in $(\mathcal{S}_1, \mathcal{S}_2)$ so that

- $\mathcal{S}_1^{(\text{match})} \sqcup \mathcal{S}_1^{(\text{res})} = \mathcal{S}_1$ and $\mathcal{S}_2^{(\text{match})} \sqcup \mathcal{S}_2^{(\text{res})} = \mathcal{S}_2$.
- $\mathcal{S}_2^{(\text{match})}$ is the image of $\mathcal{S}_1^{(\text{match})}$ with respect to the matching labeling ℓ_M. That is, $\mathcal{S}_2^{(\text{match})} = \{\ell_M(S_1) : S_1 \in \mathcal{S}_1^{(\text{match})} \text{ and } \ell_M(S_1) \neq \perp\}$.
- Each cluster in $\mathcal{S}_1^{(\text{res})}$ has at least one link from $\mathcal{S}_2^{(\text{res})}$. That is, for each $S_1 \in \mathcal{S}_1^{(\text{res})}$, there exists $S_2 \in \mathcal{S}_2^{(\text{res})}$ such that $\ell_L(S_2) = S_1$.

Algorithm 5 (Phase 1: Pre-processing).

Input: k_1- and k_2-solutions $\mathcal{S}_1, \mathcal{S}_2$; matching and linking labelings ℓ_M, ℓ_L.
Output: The partition $\mathcal{S}_1^{(\text{match})}, \mathcal{S}_2^{(\text{match})}, \mathcal{S}_1^{(\text{res})}, \mathcal{S}_2^{(\text{res})}$.

Let $\mathcal{S}_1^{(\text{match})} = \emptyset$, $\mathcal{S}_2^{(\text{match})} = \emptyset$, $\mathcal{S}_1^{(\text{res})} = \mathcal{S}_1$, and $\mathcal{S}_2^{(\text{res})} = \mathcal{S}_2$.

While there exists $S_1 \in \mathcal{S}_1^{(\text{res})}$ such that $\nexists S_2 \in \mathcal{S}_2^{(\text{res})}$ with $\ell_L(S_2) = S_1$,

- Let S_1 be the cluster. We remove S_1 from $\mathcal{S}_1^{(\text{res})}$ and add it to $\mathcal{S}_1^{(\text{match})}$.
- If $\ell_M(S_1) \neq \perp$, we remove $\ell_M(S_1)$ from $\mathcal{S}_2^{(\text{res})}$ if it is still there, and add it to $\mathcal{S}_2^{(\text{match})}$.

We shall remark the following simple observation about the sizes of the sets in the partitions.

Observation 48. *We have* $|\mathcal{S}_1^{(\text{match})}| \geq |\mathcal{S}_2^{(\text{match})}|$ *and* $|\mathcal{S}_2^{(\text{res})}| - |\mathcal{S}_1^{(\text{res})}| \geq k_2 - k_1$.

In our final solution, we will open all clusters in $\mathcal{S}_2^{(\text{match})}$. After the following Phase 2, all good clients in $\mathcal{S}_1^{(\text{match})}$ who are not included in an opened cluster will be assigned to some cluster in $\mathcal{S}_2^{(\text{match})}$. We will formalize this assignment later in Algorithm 7.

5.4 Opening clusters among the residual clusters

In this part, we consider opening $k - |\mathcal{S}_2^{(\text{match})}|$ clusters among the residual clusters $\mathcal{S}_1^{(\text{res})}$ and $\mathcal{S}_2^{(\text{res})}$. We only consider clients whose $\mathcal{S}_2$-cluster is in $\mathcal{S}_2^{(\text{res})}$, since all other clients are already assigned to some opened cluster in $\mathcal{S}_2^{(\text{match})}$. Let $D^{(\text{res})}$ be the set of residual clients whose $\mathcal{S}_2$-cluster is in $\mathcal{S}_2^{(\text{res})}$, i.e.,

$$D^{(\text{res})} = \bigcup_{S_2 \in \mathcal{S}_2^{(\text{res})}} S_2 \ . \tag{26}$$

We will then decide the assignment for all clients in

$$\left(\bigcup_{S_1 \in \mathcal{S}_1^{(\text{res})}} \mathsf{Good}(S_1) \cap D^{(\text{res})}\right) \cup \left(\bigcup_{S_2 \in \mathcal{S}_2^{(\text{res})}} \mathsf{Bad}(S_2)\right).$$

This includes all the bad clients in $D^{(\text{res})}$ and all the good clients in $D^{(\text{res})}$ whose $\mathcal{S}_1$-cluster remains in $\mathcal{S}_1^{(\text{res})}$. We will deal with the remaining good clients in $D^{(\text{res})}$ in a later phase (Algorithm 7).

In the following Algorithm 6, we present a randomized algorithm that opens $k - |\mathcal{S}_2^{(\text{match})}|$ clusters among the residual clusters $\mathcal{S}_1^{(\text{res})}$ and $\mathcal{S}_2^{(\text{res})}$.

- In the algorithm, we first match each $S_1 \in \mathcal{S}_1^{(\text{res})}$ to a cluster $M(S_1) \in \mathcal{S}_2^{(\text{res})}$ with a link to S_1. For each matched pair $(S_1, M(S_1))$ we will open exactly one of these two clusters.
- Let $\mathcal{S}_2'$ be the unsampled $\mathcal{S}_2$-clusters in $\mathcal{S}_2^{(\text{res})}$. We randomly sample $|\mathcal{S}_2'| - (k_2 - k)$ clusters from un-matched clusters in $\mathcal{S}_2^{(\text{res})}$, and open all of them. The reason for choosing this number is that we have "temporarily opened" all sampled clusters and we want the total number of opened clusters to k.
- Finally, we go through all clusters in $\mathcal{S}_1^{(\text{res})}$ and decide whether to open S_1 or $M(S_1)$ based on the number of good clients in S_1 that are already settled in some opened cluster in $\mathcal{S}_2^{(\text{res})}$.

Intuitively, the rationale behind the final step is as follows. Re-routing good clients via the labels is only possible using ℓ_M, not ℓ_L. But $\ell_M(S_1)$ is not guaranteed to open for $S_1 \in \mathcal{S}_1^{(\text{res})}$. Thus if we do not open an $\mathcal{S}_1$ cluster S_1 and open $M(S_1)$ instead, we need to re-route the S_1 good clients in a different way: and this is accomplished by rerouting to another good client's $\mathcal{S}_2$ cluster. One may wonder as to why we should open $M(S_1)$ at all, as always opening S_1 may avoid such re-routing. The reason is that eventually our algorithm needs the key lemma that for any $\mathcal{S}_2$ cluster, the probability that it is not open is at most $O(\alpha)$. So we still need to open $M(S_1)$ at times to satisfy this condition.

As discussed above, to analyze the algorithm, the key lemma is the following lower bound for the probability that each $S_2 \in \mathcal{S}_2^{(\text{res})}$ is opened. We will use this lemma later to analyze the expected cost imposed on every cluster in the original solution $\mathcal{S}_1$ and $\mathcal{S}_2$.

Lemma 49. *Conditioning on ℓ_M, ℓ_L, for every $S_2 \in \mathcal{S}_2^{(\text{res})}$, the probability that S_2 is opened is at least $1 - 3\alpha$. That is, $\forall S_2 \in \mathcal{S}_2^{(\text{res})}, \Pr[S_2 \in \mathcal{S}_{2,O}^{(\text{res})} | \ell_M, \ell_L] \geq 1 - 3\alpha$.*

To prove the lemma, we consider a random process for N random binary variables $X_1, \ldots, X_N$, where each X_i will be an indicator of whether a cluster is opened. $X_i = 1$ means that the corresponding subset is closed, and $X_i = 0$ means that it is open. We now abstractly state and prove the lemma that we need, whose process exactly mimics Algorithm 6.

Lemma 50. *Consider N random binary variables $X_1, \ldots, X_N$. Consider integers $N' \in \{0\} \cup [N-1]$ and $M \in [N - N']$. Consider partitioning the suffix $[N'+1 : N]$ into M disjoint non-empty intervals $B_1, \ldots, B_M$, ordered from left to right. Let $l_i = \min B_i$ for each $i \in [M]$. Let $K \in [N - M]$ be any integer. For any $i \in [M]$, we have a non-negative weight w_{it} for each $t < l_i$.*

We consider the following random process to define the values of $X_1, \ldots, X_N$:

- *For each group B_i, we independently sample a representative $r_i \in B_i$ uniformly at random.*

Algorithm 6 (Phase 2: Opening clusters among the residual clusters).

Input: The residual clusters after the pre-processing $\mathcal{S}_1^{(\text{res})}, \mathcal{S}_2^{(\text{res})}$, residual clients $D^{(\text{res})}$, and linking labelings ℓ_L.

Output: Opened cluster $\mathcal{S}_{1,O}^{(\text{res})}$ and $\mathcal{S}_{2,O}^{(\text{res})}$, and the assignments A'.

All sampling steps use fresh independent randomness. The algorithm is 4-step as follows:

1. Compute $D^{(\text{res})}$ according to Equation (26).
2. For each $S_1 \in \mathcal{S}_1^{(\text{res})}$, we independently sample one cluster $M(S_1)$ uniformly from $\{S_2 \in \mathcal{S}_2^{(\text{res})} : \ell_L(S_2) = S_1\}$. Let $\mathcal{S}_2' = \mathcal{S}_2^{(\text{res})} \setminus \{M(S_1) : S_1 \in \mathcal{S}_1^{(\text{res})}\}$.
3. Let $\mathcal{S}_{1,O}^{(\text{res})} = \mathcal{S}_{2,O}^{(\text{res})} = \emptyset$. Sample $|\mathcal{S}_2'| - (k_2 - k)$ clusters from $\mathcal{S}_2'$ uniformly at random and add them to $\mathcal{S}_{2,O}^{(\text{res})}$.
4. We process each $S_1 \in \mathcal{S}_1^{(\text{res})}$ in increasing order of their size $|S_1|$ (breaking ties arbitrarily):
   - Let $G = \{i \in S_1 \cap D^{(\text{res})} : s_1(i) \geq s_2(i)\}$ be the set of good residual clients in S_1.
   - Let D_{settled} be the union of clusters in $\mathcal{S}_{2,O}^{(\text{res})}$.
   - If $|G \cap D_{\mathsf{settled}}| < \frac{1}{3}|G|$, or there exists $S_2 \in \mathcal{S}_2' \setminus \mathcal{S}_{2,O}^{(\text{res})}$ such that $\ell_L(S_2) = S_1$, we open S_1 (i.e., add it to $\mathcal{S}_{1,O}^{(\text{res})}$). For every $S_2 \in \mathcal{S}_2^{(\text{res})}$ such that $\ell_L(S_2) = S_1$, and every $i \in \mathsf{Bad}(S_2)$, we set $A'(i) = S_1$.
   - Otherwise, we open $M(S_1)$ (i.e., add it to $\mathcal{S}_{2,O}^{(\text{res})}$). If $G \neq \emptyset$, we uniformly choose a client $i \in G \cap D_{\mathsf{settled}}$, and let $A'(j) = C_2(i)$ for every $j \in G \setminus D_{\mathsf{settled}}$.

- *Among the $N - M$ non-representatives, we sample K of them uniformly at random and set their values to 1. We set all other non-representatives to 0.*
- *For each $i \in [M]$, in increasing order, we determine the value of X_{r_i} as follows: if there exists $j \in B_i \setminus \{r_i\}$ such that $X_j = 1$, we set $X_{r_i} = 1$; otherwise, we set $X_{r_i} = \mathbb{1}(\sum_{t<l_i} w_{it} X_t > \frac{2}{3}\sum_{t<l_i} w_{it})$.*

Then, we have $\mathbb{E}[X_j] \leq 3 \cdot \frac{K}{N-M}$. [13]

With this lemma, we are now ready to prove Lemma 49.

Proof of Lemma 49. Fix ℓ_M, ℓ_L. Let $N = |\mathcal{S}_2^{(\text{res})}|$, $M = |\mathcal{S}_1^{(\text{res})}|$ and $K = k_2 - k$. Because of Observation 48, we have $N - M \geq k_2 - k_1$. Let N' be the number of $S_2 \in \mathcal{S}_2^{(\text{res})}$ that doesn't have a

[13] AI disclosure: GPT-6 Astra gave the proof of a special case of this lemma: $|B_1| = \cdots = |B_M| = 1$, where it suggested a connection between the random process and the Erdos-Ko-Rado theorem [EKR61]. The proof of this lemma is then extended by the authors, combining the proofs of the simpler version and the Erdos-Ko-Rado theorem. **A short proof for the special case:** assume $N' \geq 2K$, as holds in our application since $\alpha \leq 1/2$. We use $X_i(S)$ to denote the value of X_i if S is the subset of $[N']$ sampled in the second step. We can show that $X_i(S) + X_i(T) \leq 1$ for any $S \cap T = \emptyset$. By Erdos-Ko-Rado, the number of S satisfying $X_i(S) = 1$ is bounded by $\binom{N'-1}{K-1} = \binom{N-M-1}{K-1}$, and thus $\Pr[X_i = 1] \leq \frac{K}{N-M}$.

link to any $S_1 \in \mathcal{S}_1^{(\mathrm{res})}$. If $N' = N$, then each $S_2 \in \mathcal{S}_2^{(\mathrm{res})}$ is opened with probability $1 - \frac{k_2-k}{N}$. Since $|\mathcal{S}_1^{(\mathrm{res})}| = M = 0$ in this case, we have $N \geq k_2 - k_1$, and thus $\Pr[S_2 \in \mathcal{S}_{2,O}^{(\mathrm{res})}] \geq 1 - \frac{k_2-k}{k_2-k_1} = 1 - \alpha$. Next, we focus on the case where $N' < N$.

Consider we arrange the M clusters in $\mathcal{S}_1^{(\mathrm{res})}$ in increasing order of their size, and we will use $S_{1,i}$ to denote the i-th cluster in the order.

We also arrange the N clusters in $\mathcal{S}_2^{(\mathrm{res})}$ in the following sequence: first we put the N' clusters that don't have a link to any $S_1 \in \mathcal{S}_1^{(\mathrm{res})}$; then, we enumerate each cluster $S_1 \in \mathcal{S}_1^{(\mathrm{res})}$ in the above order, and append all clusters $S_2 \in \mathcal{S}_2^{(\mathrm{res})}$ that are linked to it to the end of the sequence. We will use $S_{2,i}$ to denote the i-th cluster in the order. Then, we consider X_i to be the indicator variable of whether $S_{2,i}$ is not opened in Algorithm 6. For each $i \in [M]$, we let B_i to be the set of indices of $\mathcal{S}_2$-clusters that are linked to $\mathcal{S}_1$-cluster i. According to our definition, $B_1, \ldots, B_M$ are disjoint intervals of a suffix of $[N]$. Let $l_i = \min B_i$ for each $i \in [M]$.

With this index, we consider w_{it} to be the number of good residual clients in the intersection of $S_{1,i}$ and $S_{2,t}$, i.e., $w_{it} = |\mathsf{Good}(S_{1,i}) \cap D^{(\mathrm{res})} \cap S_{2,t}|$. Note that here we define w_{it} for every index t, but now we will show that $w_{it} \neq 0$ only if $t < l_i$.

Consider any $i \in [M]$ and any $t \geq l_i$. For the sake of contradiction, suppose that $w_{it} > 0$, i.e., there exists a good client $\ell \in S_{1,i} \cap S_{2,t}$. Because $t \geq l_i$, we have that $S_{2,t}$ is linked to $S_{1,j}$ for some $j \geq i$. Since client ℓ is good, we have $|S_{2,t}| \leq |S_{1,i}|$. Note that $S_{2,t}$ is linked to $S_{1,j}$, implying that there is a bad client in the intersection of $S_{2,t}$ and $S_{1,j}$, and thus $|S_{2,t}| > |S_{1,j}|$. Thus, we have $|S_{1,j}| < |S_{2,t}| \leq |S_{1,i}|$, which contradicts the fact that $j \geq i$ and the order of the $\mathcal{S}_1$-clusters. Therefore, we have $w_{it} = 0$ for any $i \in [M]$ and any $t \geq l_i$. Thus all clusters containing good residual clients of $S_{1,i}$ have their opening status determined before $S_{1,i}$ is processed. In Algorithm 6, when we process $S_{1,i}$, we therefore have $|G \cap D_{\mathsf{settled}}| = \sum_{t \in [N]} w_{it}(1 - X_t)$. Further, we can rewrite the condition $|G \cap D_{\mathsf{settled}}| < \frac{1}{3}|G|$ in Algorithm 6 as $\sum_{t<l_i} w_{it}(1-X_t) < \frac{1}{3}\sum_{t<l_i} w_{it}$, which is equivalent to $\sum_{t<l_i} w_{it}X_t > \frac{2}{3}\sum_{t<l_i} w_{it}$.

Finally, we can apply Lemma 50 to conclude that for each $i \in [N]$, $\mathbb{E}[X_i] \leq 3 \cdot \frac{K}{N-M} \leq 3 \cdot \frac{k_2-k}{k_2-k_1} = 3\alpha$. □

Now, we prove Lemma 50.

Proof. Without loss of generality, we can assume that $N - M > 3K$, since otherwise the lemma is trivial. Let $L = \lfloor \frac{N-M}{K} \rfloor$. Note that it suffices to prove the probability that $X_j = 1$ is at most $2/L$, because

$$\mathbb{E}[X_j] \leq 2 \left\lfloor \frac{N-M}{K} \right\rfloor^{-1} \leq 2 \cdot \frac{K}{N-M-K} < 2 \cdot \frac{K}{2(N-M)/3} = 3 \cdot \frac{K}{N-M} .$$

To prove this, we consider a coupling process. Concretely, we will construct L copies $X_i^{(\ell)}$, $\ell \in [L]$, for every variable X_i, where each $X_i^{(\ell)}$ has the same distribution as X_i, but all the L sequences are generated in a coupled (dependent) way. We will then show that every $X_i^{(\ell)}$ is non-zero for at most 2 values of ℓ. If this were to be true, then we would be done, since for every $j \in [N]$,

$$\mathbb{E}\left[\sum_{\ell=1}^{L} X_j^{(\ell)}\right] = L\mathbb{E}[X_j] \leq 2$$

and hence $\mathbb{E}[X_j] \leq \frac{2}{L}$ as desired. We now define the coupling process as follows.

- First, roughly speaking, we will decide the outcome of the initial non-representative sampling step block-wise. Concretely, we decide how many X_j are set to 1 from each block B_i, and also decide the value X_j exactly for every $j \in [N']$. We need to set $X_j = 1$ for K of the $N - M$ non-representatives j. There are L copies of each variable: to generate them all together in a coupled way, we first generate a string by giving $|B_i| - 1$ (identical) tokens for block i and one token for each of the initial N' variables.
- Next, we do a uniformly random shuffle on these tokens to obtain a permutation of this string.
- To determine the value of $X_j^{(\ell)}$ for $j \in [N']$, we will look at the indices $[(\ell-1)K, \ell K)$ of this string, and set it to 1 if j's token appears there, and otherwise set it to zero.
- Next, for each block B_i, in increasing order of i, we define $q_i^{(\ell)}$ to be the number of tokens of B_i that appears in the sub-string with indices $[(\ell-1)K, \ell K)$. Here $q_i^{(\ell)}$ is intended to be the number of non-representatives sampled in block i. For each $\ell \in [L]$ and each $i \in [M]$, we consider $s_i^{(\ell)}$ as follows:
  - if $q_i^{(\ell)} > 0$, we set $s_i^{(\ell)}$ to be $q_i^{(\ell)} + 1$;
  - otherwise, we set $s_i^{(\ell)}$ to be $\mathbb{1}(\sum_{t<l_i} w_{it} X_t^{(\ell)} > \frac{2}{3} \sum_{t<l_i} w_{it})$.

  Now, $s_i^{(\ell)}$ is the exact number of X_j we will set to 1 for $j \in B_i$. We now describe how to select the elements themselves inside each block.
- Consider B_i. List its $|B_i|$ elements in a uniformly random order, independently of the token shuffle and the orders for other blocks: suppose this order is $b_{i,1}, b_{i,2} \dots b_{i,|B_i|}$. We concatenate this string with itself repeatedly to obtain an infinite string. The first $s_i^{(1)}$ elements are the ones set to 1 in the first run, the next $s_i^{(2)}$ elements are the ones set to 1 in the next run, and so on. This decides the variables $X_j^{(\ell)}$ for every $\ell \in [L]$. Intuitively, we try to spread the selected number of variables from B_i uniformly across the elements of B_i.

First, it is clear that for each $\ell \in [L]$, the random variables $X_1^{(\ell)}, \dots, X_N^{(\ell)}$ are distributed as $X_1, \dots, X_N$. By definition, $q_i^{(\ell)}$ follows the distribution of the number of sampled non-representatives in B_i, and then, $X_i^{(\ell)}$ follows the distribution of X_i for each $i \in [N']$. Conditional on $q_i^{(\ell)} = q > 0$, the original process sets a uniformly random $(q+1)$-subset of B_i to 1; if $q = 0$, it sets either no element or a uniformly random singleton to 1, according to the earlier-block threshold. By the last bullet, the distribution of the selected $s_i^{(\ell)}$ consecutive positions in the shuffled sequence is same as a uniformly random $s_i^{(\ell)}$-sized subset of B_i, even conditioned on all counts, prefix outcomes, and the random choices of elements in all previous blocks across all copies. This conditioning fixes the segment length and its starting position, while the permutation of B_i remains independent and uniform.

Let us now show by induction on $j \in [N]$, that for every j, $X_j^{(\ell)}$ is non-zero for at most two indices in $[L]$ to complete the proof.

The base case $j \in [N']$ is simple, because $X_j^{(\ell)} = 1$ for at most one $\ell \in [L]$. Now consider any $i \in [M]$ and any $j \in B_i$. It is clear that $\sum_{\ell \in [L]} q_i^{(\ell)} \le |B_i| - 1$, because there are $|B_i| - 1$ tokens from B_i. Therefore, we have

$$\sum_{\ell \in [L] : q_i^{(\ell)} > 0} s_i^{(\ell)} \le 2|B_i| - 2 \ .$$

Because our induction hypothesis ensures that $\sum_{\ell\in[L]} X_t^{(\ell)} \le 2$ for each $t < l_i$ (which holds trivially when $l_i = 1$), we have $\sum_{\ell\in[L]} \sum_{t<l_i} w_{it} X_t^{(\ell)} \le 2 \sum_{t<l_i} w_{it}$. Therefore, there exist at most two $\ell \in [L]$ such that $\sum_{t<l_i} w_{it} X_t^{(\ell)} > \frac{2}{3} \sum_{t<l_i} w_{it}$. This implies that

$$\sum_{\ell\in[L]:q_i^{(\ell)}=0} s_i^{(\ell)} \le 2 \ .$$

In total, we have $\sum_{\ell\in[L]} s_i^{(\ell)} \le 2|B_i|$. According to the definition of $X_j^{(\ell)}$ in our process, the distribution inside each block is balanced: hence we have $\sum_{\ell\in[L]} X_j^{(\ell)} \le 2$ for each $j \in B_i$. □

5.5 Putting things together

In this part, we present the final Phase 3 of our algorithm, Algorithm 7, which completes the client assignments.

Algorithm 7 (Phase 3: Client Assignments).

Input: Partition $\mathcal{S}_1^{(\text{match})}, \mathcal{S}_2^{(\text{match})}, \mathcal{S}_1^{(\text{res})}, \mathcal{S}_2^{(\text{res})}$, opened clusters $\mathcal{S}_{1,O}^{(\text{res})}, \mathcal{S}_{2,O}^{(\text{res})}$, matching labeling ℓ_M, and partial assignments A'.

Output: Opened cluster $\mathcal{S}_{1,O}$ and $\mathcal{S}_{2,O}$, and the assignments A.

Let $\mathcal{S}_{1,O} = \mathcal{S}_{1,O}^{(\text{res})}$ and $\mathcal{S}_{2,O} = \mathcal{S}_{2,O}^{(\text{res})} \cup \mathcal{S}_2^{(\text{match})}$. Let D_{settled} be the union of clusters in $\mathcal{S}_{1,O}$ and $\mathcal{S}_{2,O}$. We decide the assignment A for each client $j \notin D_{\mathsf{settled}}$ as follows:

- For each bad client $j \notin D_{\mathsf{settled}}$, we set $A(j) = A'(j)$.
- For each good client $j \notin D_{\mathsf{settled}}$,
  - if $C_1(j) \in \mathcal{S}_1^{(\text{res})}$, we set $A(j) = A'(j)$;
  - if $C_1(j) \in \mathcal{S}_1^{(\text{match})}$, we set $A(j) = \ell_M(C_1(j))$.

We open the following k clusters:

- For each $S_2 \in \mathcal{S}_{2,O}$, we open $S_2 \cup \{j \notin D_{\mathsf{settled}} : A(j) = S_2\}$.
- Let $D_2 = \cup_{S_2\in\mathcal{S}_{2,O}} S_2$. For each $S_1 \in \mathcal{S}_{1,O}$, we open $(S_1 \setminus D_2) \cup \{j \notin D_{\mathsf{settled}} : A(j) = S_1\}$.

An easy observation is that each $S_2 \in \mathcal{S}_2$ is opened with probability at least $1 - 3\alpha$.

Observation 51. *For each $S_2 \in \mathcal{S}_2$, we have* $\Pr[S_2 \in \mathcal{S}_{2,O} | \ell_M, \ell_L] \ge 1 - 3\alpha$.

Proof. Follows directly from Lemma 49 and the fact that all clusters in $\mathcal{S}_2^{(\text{match})}$ are opened. □

To upper bound the total expected cost of the final solution, we will consider two parts separately.

- The cost imposed on each cluster $S_2 \in \mathcal{S}_2$, which includes the cost of its original clients and the additional cost of assigning other good clients to it.

- The cost imposed on each cluster $S_1 \in \mathcal{S}_1$, which includes the cost of the original clients it retains and the additional cost of assigning other bad clients to it.

For the first type of cost, we only need to consider upper bounding the expected additional cost of assigning clients to each $S_2 \in \mathcal{S}_2$, because we can always upper bound the cost of the original clients in the opened $\mathcal{S}_2$-clusters by $\mathsf{cost}(\mathcal{S}_2)$.

5.5.1 Bounding the expected cost of the $\mathcal{S}_2$-clusters

Note that for each $S_2 \in \mathcal{S}_2$, it can be opened in two ways: either it is opened as part of $\mathcal{S}_2^{(\text{match})}$, or it is opened as part of $\mathcal{S}_2^{(\text{res})}$ in Algorithm 6. Because of the linearity of expectation, we will upper bound the expected additional cost of assigning other good clients to each $S_2 \in \mathcal{S}_2$ separately for two different ways.

Case 1: as part of $\mathcal{S}_2^{(\text{match})}$. For a good client j to be reassigned to S_2, the following conditions must hold: (1) $\ell_M(C_1(j)) = S_2$; (2) $C_2(j)$ not opened. Because of Observation 51, the probability that all these conditions are satisfied is at most

$$\frac{|\mathsf{Good}(C_1(j)) \cap S_2|}{|\mathsf{Good}(C_1(j))|} \cdot 3\alpha \ . \tag{27}$$

This bound applies also to that two clients j, j' satisfying $C_1(j) = C_1(j')$ are both reassigned to S_2.

Consider two good clients j, j' such that $C_1(j) \neq C_1(j')$. If j, j' are both reassigned to S_2, then we have $\ell_M(C_1(j)) = \ell_M(C_1(j')) = S_2$, and $C_2(j)$ is not opened. Because of Observation 51, the probability that all these conditions are satisfied is at most

$$\frac{|\mathsf{Good}(C_1(j)) \cap S_2|}{|\mathsf{Good}(C_1(j))|} \cdot \frac{|\mathsf{Good}(C_1(j')) \cap S_2|}{|\mathsf{Good}(C_1(j'))|} \cdot 3\alpha \ . \tag{28}$$

For these bounds, we first condition on all matching and linking labels and apply Observation 51 to one necessary closure event. We then average over the independent matching labels.

The additional cost comes from pairs between original and reassigned clients, and pairs between reassigned clients. This is why we bounded the probability of reassigning one client and of reassigning two clients together. The next two lemmas turn these probability bounds into cost bounds. We will also use them in Case 2 and in the analysis of the $\mathcal{S}_1$-clusters.

Lemma 52. *Consider $S \subseteq D$ and disjoint nonempty sets $T_1, \dots, T_m \subseteq D$. Let $R_i = S \cap T_i$ and $p_i = |R_i|/|T_i|$. Let $Z_i \subseteq T_i$ be random sets of clients assigned to S. Suppose that, for some $a, b \geq 0$,*

$$\begin{aligned} \Pr[u \in Z_i] &\leq a p_i && \text{for every } u \in T_i, \\ \Pr[u \in Z_i,\ v \in Z_j] &\leq b p_i p_j && \text{for every } i \neq j,\ u \in T_i,\ v \in T_j. \end{aligned}$$

Then,

$$\begin{aligned} \mathbb{E}\left[2\sum_i d(S, Z_i) + \sum_{i,j} d(Z_i, Z_j)\right] &\leq (2a+b)\mathsf{cost}(S) \\ &\quad + (2a+2b)\sum_i \frac{|S|}{|T_i|} d(R_i, T_i) + a\sum_i p_i \mathsf{cost}(T_i). \end{aligned} \tag{29}$$

In particular, this bounds the expected additional cost of $S \cup \bigcup_i Z_i$.

Proof. If $R_i = \emptyset$, the single-client bound implies that $Z_i = \emptyset$ almost surely, so we omit such indices. First, let us bound the first term. By averaging the triangle inequality over all R_i we get:

$$p_i d(S, T_i) \leq \frac{|R_i|}{|T_i|} \sum_{u \in S, v \in T_i} \frac{d(u, R_i) + d(R_i, v)}{|R_i|} = d(S, R_i) + \frac{|S|}{|T_i|} d(R_i, T_i).$$

Using this, linearity of expectation and probability bound in the statement,

$$\begin{aligned} \mathbb{E}\left[\sum_i d(S, Z_i)\right] &= \sum_i \sum_{u \in S} \sum_{v \in T_i} d(u, v) \Pr[v \in Z_i] \\ &\leq \sum_i \sum_{u \in S} \sum_{v \in T_i} d(u, v)\, a p_i \\ &= a \sum_i p_i d(S, T_i) \\ &\leq a \sum_i d(S, R_i) + a \sum_i \frac{|S|}{|T_i|} d(R_i, T_i) \\ &\leq a\mathsf{cost}(S) + a \sum_i \frac{|S|}{|T_i|} d(R_i, T_i). \end{aligned}$$

The last inequality follows since R_i are disjoint subsets of S.

Next, lets bound the term corresponding to $i \neq j$. By averaging the triangle inequality over R_i and R_j, we get

$$\begin{aligned} p_i p_j d(T_i, T_j) &\leq \frac{1}{|T_i||T_j|} \sum_{u \in T_i, v \in T_j} (|R_j| d(u, R_i) + d(R_i, R_j) + |R_i| d(R_j, v)) \\ &= d(R_i, R_j) + \frac{|R_j|}{|T_i|} d(R_i, T_i) + \frac{|R_i|}{|T_j|} d(R_j, T_j). \end{aligned}$$

Using this, linearity of expectation and the joint probability bound in the statement,

$$\begin{aligned}
\mathbb{E}\left[\sum_{i\neq j} d(Z_i, Z_j)\right] &= \sum_{i\neq j}\sum_{u\in T_i}\sum_{v\in T_j} d(u,v)\Pr[u\in Z_i,\ v\in Z_j] \\
&\le b\sum_{i\neq j} p_i p_j d(T_i, T_j) \\
&\le b\sum_{i\neq j} d(R_i, R_j) + b\sum_{i\neq j}\left(\frac{|R_j|}{|T_i|}d(R_i,T_i) + \frac{|R_i|}{|T_j|}d(R_j,T_j)\right) \\
&= b\sum_{i\neq j} d(R_i, R_j) + 2b\sum_i \frac{\sum_{j\neq i}|R_j|}{|T_i|}d(R_i,T_i) \\
&\le b\mathsf{cost}(S) + 2b\sum_i \frac{|S|}{|T_i|}d(R_i,T_i).
\end{aligned}$$

The last equality follows by interchanging i and j in one of the two sums. The last inequality follows since the R_i are disjoint subsets of S, so $\sum_{i\neq j} d(R_i,R_j) \le \mathsf{cost}(S)$ and $\sum_{j\neq i}|R_j| \le |S|$.

Finally, we bound the term corresponding to $i = j$. For every $u, v \in T_i$, the probability bound in the statement gives $\Pr[u\in Z_i,\ v\in Z_i] \le \Pr[u\in Z_i] \le ap_i$. Using this and linearity of expectation,

$$\begin{aligned}
\mathbb{E}\left[\sum_i d(Z_i, Z_i)\right] &= \sum_i\sum_{u\in T_i}\sum_{v\in T_i} d(u,v)\Pr[u\in Z_i,\ v\in Z_i] \\
&\le a\sum_i p_i \sum_{u\in T_i}\sum_{v\in T_i} d(u,v) \\
&= a\sum_i p_i \mathsf{cost}(T_i).
\end{aligned}$$

Multiplying the first bound by two and adding these two bounds proves the lemma. □

The first lemma bounds the additional cost at one destination cluster. We now sum this bound over the clusters in $\mathcal{B}$. For each source set T_i, the intersections $S \cap T_i$ partition T_i. This lets us combine the distance sums. The size condition in the next lemma then lets us bound the contribution of each source using the cost of its parent cluster P_i.

Lemma 53. *Let $T_i \subseteq P_i$ be nonempty subsets of disjoint clusters $P_1, \ldots, P_m$, and let $\mathcal{B}$ be a partition of a set containing every T_i. Suppose that $|S| \le |P_i|$ whenever $S \in \mathcal{B}$ and $S \cap T_i \neq \emptyset$. For each $S \in \mathcal{B}$, let $Z_i(S) \subseteq T_i$ satisfy the probability bounds in Lemma 52 with the same constants $a, b \ge 0$, and $p_i(S) = |S\cap T_i|/|T_i|$. Then,*

$$\sum_{S\in\mathcal{B}} \mathbb{E}\left[2\sum_i d(S, Z_i(S)) + \sum_{i,j} d(Z_i(S), Z_j(S))\right] \le (2a+b)\mathsf{cost}(\mathcal{B}) + (3a+2b)\sum_i \mathsf{cost}(P_i). \quad (30)$$

Proof. First, for any nonempty $T \subseteq P$ and any $z \in P \setminus T$, the triangle inequality gives

$$\mathsf{cost}(T) = \sum_{u,v\in T} d(u,v) \le \sum_{u,v\in T}\big(d(u,z) + d(z,v)\big) = 2|T|d(z,T).$$

Summing over $z \in P \setminus T$, we get

$$\mathsf{cost}(P) \geq \mathsf{cost}(T) + 2d(T, P \setminus T) \geq \mathsf{cost}(T) + \frac{|P| - |T|}{|T|}\mathsf{cost}(T) = \frac{|P|}{|T|}\mathsf{cost}(T). \tag{31}$$

For each fixed i, the sets $S \cap T_i$, over $S \in \mathcal{B}$, partition T_i. Using the fact that $|S| \leq |P_i|$ and Equation (31),

$$\sum_{S \in \mathcal{B}} \frac{|S|}{|T_i|} d(S \cap T_i, T_i) \leq \frac{|P_i|}{|T_i|} \sum_{S \in \mathcal{B}} d(S \cap T_i, T_i) = \frac{|P_i|}{|T_i|}\mathsf{cost}(T_i) \leq \mathsf{cost}(P_i).$$

Also, since $\sum_{S \in \mathcal{B}} p_i(S) = 1$, we have

$$\sum_{S \in \mathcal{B}} p_i(S)\mathsf{cost}(T_i) = \mathsf{cost}(T_i) \leq \mathsf{cost}(P_i).$$

Applying Lemma 52 to each $S \in \mathcal{B}$ and summing, we therefore get

$$\begin{aligned}
&\sum_{S \in \mathcal{B}} \mathbb{E}\left[2\sum_i d(S, Z_i(S)) + \sum_{i,j} d(Z_i(S), Z_j(S))\right] \\
&\quad \leq (2a+b)\sum_{S \in \mathcal{B}} \mathsf{cost}(S) + (2a+2b)\sum_i \sum_{S \in \mathcal{B}} \frac{|S|}{|T_i|} d(S \cap T_i, T_i) \\
&\qquad + a\sum_i \sum_{S \in \mathcal{B}} p_i(S)\mathsf{cost}(T_i) \\
&\quad \leq (2a+b)\mathsf{cost}(\mathcal{B}) + (2a+2b)\sum_i \mathsf{cost}(P_i) + a\sum_i \mathsf{cost}(P_i) \\
&\quad = (2a+b)\mathsf{cost}(\mathcal{B}) + (3a+2b)\sum_i \mathsf{cost}(P_i).
\end{aligned}$$

This concludes the proof of the lemma. □

Claim 54. *Let Δ_M be the total additional cost in Case 1. Then,*

$$\mathbb{E}[\Delta_M] \leq 15\alpha\mathsf{cost}(\mathcal{S}_1) + 9\alpha\mathsf{cost}(\mathcal{S}_2).$$

Proof. We now apply Lemma 53 to Case 1. For each $P_i \in \mathcal{S}_1$ take $T_i = \mathsf{Good}(P_i)$, omitting empty sets, and take the partition to be $\mathcal{B} = \mathcal{S}_2$. Then $Z_i(S)$ becomes the clients in T_i actually reassigned to S by the matching rule in Phase 3, with $Z_i(S) = \emptyset$ unless $P_i \in \mathcal{S}_1^{(\text{match})}$ and $S \in \mathcal{S}_2^{(\text{match})}$. If $S \cap T_i \neq \emptyset$, goodness gives $|S| \leq |P_i|$, so the size condition holds. Taking $a, b = 3\alpha$, Lemma 53 gives us total additional cost as $15\alpha\mathsf{cost}(\mathcal{S}_1) + 9\alpha\mathsf{cost}(\mathcal{S}_2)$. □

Case 2: as part of $\mathcal{S}_2^{(\mathrm{res})}$. Here, we choose the destination by sampling a settled good client. The settled clients make up at least one third of the good residual clients in the source cluster. Thus, the probability of choosing a destination is at most three times its fraction of all good residual clients.

Claim 55. *Let Δ_R be the total additional cost in Case 2. Then,*

$$\mathbb{E}[\Delta_R] \le 81\alpha\mathsf{cost}(\mathcal{S}_1) + 45\alpha\mathsf{cost}(\mathcal{S}_2).$$

Proof. In this case, we condition on ℓ_M and ℓ_L. For a good client j to be reassigned to S_2, the following conditions must hold: (1) $C_2(j)$ not opened; (2) At least $1/3$ of $\mathsf{Good}(C_1(j)) \cap D^{(\mathrm{res})}$ are in an opened $\mathcal{S}_2$-cluster; (3) One client in S_2 is sampled among the clients in $\mathsf{Good}(C_1(j)) \cap D^{(\mathrm{res})}$ who are in an opened $\mathcal{S}_2$-cluster. The probability that all these conditions are satisfied is at most

$$3\alpha \cdot 1 \cdot \frac{|\mathsf{Good}(C_1(j)) \cap S_2|}{|\mathsf{Good}(C_1(j)) \cap D^{(\mathrm{res})}|/3} = 9\alpha \cdot \frac{|\mathsf{Good}(C_1(j)) \cap S_2|}{|\mathsf{Good}(C_1(j)) \cap D^{(\mathrm{res})}|}\ . \tag{32}$$

The same bound applies also to that two clients j, j' satisfying $C_1(j) = C_1(j')$ are both reassigned to S_2.

For each $P \in \mathcal{S}_1^{(\mathrm{res})}$, let $T_P = \mathsf{Good}(P) \cap D^{(\mathrm{res})}$, omitting empty sets. For $S_2 \in \mathcal{S}_2^{(\mathrm{res})}$, define $p_P(S_2) = \frac{|T_P \cap S_2|}{|T_P|}$, and let $Z_P(S_2)$ consist of the clients from T_P actually reassigned to S_2. From the probability bounds above we have $\Pr[u \in Z_P(S_2) \mid \ell_M, \ell_L] \le 9\alpha p_P(S_2)$.

Fix also the representatives and the initial sample. These choices determine all opening decisions and all sets of settled clients, since the samples used for reassignment do not affect opening decisions.

Whenever a sample is needed for P, it is chosen uniformly from a set of at least $|T_P|/3$ settled clients. Thus, its probability of lying in S_2 is at most $3p_P(S_2)$.

For clients from different source clusters, this gives one factor of three for each sample. For distinct P, Q, the samples used for reassignment are independent under this conditioning. Thus, for $u \in T_P$ and $v \in T_Q$, the conditional probability that both samples select S_2 is at most $9p_P(S_2)p_Q(S_2)$. Both clients being reassigned also requires $C_2(u)$ to be closed. Thus, averaging over these opening choices and applying Observation 51, we have

$$\Pr[u \in Z_P(S_2),\ v \in Z_Q(S_2) \mid \ell_M, \ell_L] \le 27\alpha p_P(S_2)p_Q(S_2).$$

We can now apply Lemma 53, conditioned on ℓ_M, ℓ_L, with $P \in \mathcal{S}_1^{(\mathrm{res})}$, source sets T_P, partition $\mathcal{B} = \mathcal{S}_2^{(\mathrm{res})}$, and $a = 9\alpha$, $b = 27\alpha$. If $S_2 \cap T_P \neq \emptyset$, goodness gives $|S_2| \le |P|$, so the size condition holds. We get

$$\begin{aligned} &\mathbb{E}[\Delta_R \mid \ell_M, \ell_L] \le 81\alpha\mathsf{cost}(\mathcal{S}_1^{(\mathrm{res})}) + 45\alpha\mathsf{cost}(\mathcal{S}_2^{(\mathrm{res})}) \\ \implies &\mathbb{E}[\Delta_R] \le 81\alpha\mathsf{cost}(\mathcal{S}_1) + 45\alpha\mathsf{cost}(\mathcal{S}_2) \end{aligned}$$

Averaging over the matching and linking labels gives the claim. □

We can now add the two case bounds.

Claim 56. *For each $S_2 \in \mathcal{S}_2$, let $F(S_2)$ be its final cluster, with $F(S_2) = \emptyset$ if it is closed. Then,*

$$\sum_{S_2 \in \mathcal{S}_2} \mathbb{E}[\mathsf{cost}(F(S_2))] \leq 96\alpha \mathsf{cost}(\mathcal{S}_1) + (1 + 54\alpha)\mathsf{cost}(\mathcal{S}_2). \tag{33}$$

Proof. In every outcome, matched $\mathcal{S}_2$-clusters receive reassigned clients only through the matching rule, and residual $\mathcal{S}_2$-clusters receive reassigned clients only through the residual rule. Hence, there are no additional pair terms between the two cases.

The original clients of the opened $\mathcal{S}_2$-clusters contribute at most $\mathsf{cost}(\mathcal{S}_2)$. Adding this bound and Claims 54 and 55 proves the claim. □

5.5.2 Bounding the expected cost of each $\mathcal{S}_1$-cluster

An opened $\mathcal{S}_1$ cluster has two kinds of clients, (1) its original clients that it retains, and (2) bad clients reassigned to it. The lemma bounds both contributions for each $S_1 \in \mathcal{S}_1$. An original client remains assigned here only if its $\mathcal{S}_2$-cluster is closed. This lets us bound the expected cost of the retained clients by $3\alpha\mathsf{cost}(S_1)$. For the reassigned bad clients, we use the same two merging lemmas as before.

Lemma 57. *For each $S_1 \in \mathcal{S}_1$, let $F(S_1)$ be its final cluster ($F(S_1) = \emptyset$ if it is not opened). For every $S_2 \in \mathcal{S}_{2,B}$, define $T_{S_2} = \mathsf{Bad}(S_2)$ and $p_{S_2}(S_1) = |S_1 \cap T_{S_2}|/|T_{S_2}|$. Then,*

$$\mathbb{E}[\mathsf{cost}(F(S_1))] \leq 12\alpha\mathsf{cost}(S_1) + 12\alpha \sum_{S_2 \in \mathcal{S}_{2,B}} \frac{|S_1|}{|T_{S_2}|} d(S_1 \cap T_{S_2}, T_{S_2}) + 3\alpha \sum_{S_2 \in \mathcal{S}_{2,B}} p_{S_2}(S_1)\mathsf{cost}(T_{S_2}).$$

Consequently,

$$\sum_{S_1 \in \mathcal{S}_1} \mathbb{E}[\mathsf{cost}(F(S_1))] \leq 12\alpha\mathsf{cost}(\mathcal{S}_1) + 15\alpha\mathsf{cost}(\mathcal{S}_2). \tag{34}$$

Proof. Lets first bound the cost within the original clients of S_1 that remain assigned to it. Lets define D_2 be the union of (clients of) all opened $\mathcal{S}_2$-clusters, and let $U_{S_1} = S_1 \setminus D_2$ if S_1 is opened, and $U_{S_1} = \emptyset$ otherwise. Thus, in the end the original clients retained by S_1 are exactly U_{S_1}. For any $u, v \in S_1$, the event $u, v \in U_{S_1}$ implies that their original cluster $C_2(u)$ and $C_2(v)$ resp. are closed. Averaging Observation 51 over the labels gives

$$\mathbb{E}[\mathsf{cost}(U_{S_1})] = \sum_{u,v \in S_1} d(u, v) \Pr[u, v \in U_{S_1}] \leq 3\alpha\mathsf{cost}(S_1). \tag{35}$$

The last inequality follows by the fact that $\Pr[u, v \in U_{S_1}] \leq \Pr[u \in U_{S_1}] \leq 3\alpha$

Now, let us bound the cost of the bad clients that are reassigned to S_1 because of Phase 2. Let $Z_{S_2}(S_1) \subseteq T_{S_2}$ consist of the bad clients actually reassigned from S_2 to S_1. Such an assignment requires both $\ell_L(S_2) = S_1$ and S_2 closed. Conditioning on all labels, using Observation 51, and then averaging gives $\Pr[u \in Z_{S_2}(S_1)] \leq 3\alpha p_{S_2}(S_1)$. For distinct S_2, S_2', the linking labels are independent, thus we have $\Pr[u \in Z_{S_2}(S_1),\ v \in Z_{S_2'}(S_1)] \leq 3\alpha p_{S_2}(S_1) p_{S_2'}(S_1)$.

The final cluster is $F(S_1) = U_{S_1} \cup \bigcup_{S_2} Z_{S_2}(S_1)$. Since $U_{S_1} \subseteq S_1$, we have that

$$\mathsf{cost}(F(S_1)) \leq \mathsf{cost}(U_{S_1}) + 2\sum_{S_2} d(S_1, Z_{S_2}(S_1)) + \sum_{S_2, S_2'} d(Z_{S_2}(S_1), Z_{S_2'}(S_1)).$$

Taking expectations, and using Lemma 52, Equation (35), with $a = b = 3\alpha$, we get:

$$\begin{aligned}
&\mathbb{E}[\mathsf{cost}(F(S_1))] \\
&\leq \mathbb{E}[\mathsf{cost}(U_{S_1})] + \mathbb{E}\left[2\sum_{S_2} d(S_1, Z_{S_2}(S_1)) + \sum_{S_2, S_2'} d(Z_{S_2}(S_1), Z_{S_2'}(S_1))\right] \\
&\leq 3\alpha\mathsf{cost}(S_1) + 9\alpha\mathsf{cost}(S_1) \\
&\quad + 12\alpha \sum_{S_2 \in \mathcal{S}_{2,B}} \frac{|S_1|}{|T_{S_2}|} d(S_1 \cap T_{S_2}, T_{S_2}) \\
&\quad + 3\alpha \sum_{S_2 \in \mathcal{S}_{2,B}} p_{S_2}(S_1)\mathsf{cost}(T_{S_2})
\end{aligned}$$

This proves the first bound.

Now, to conclude the second bound, we bound the last two terms in the expression above. If $S_1 \cap T_{S_2} \neq \emptyset$, a bad client in the intersection gives $|S_1| < |S_2|$, as required by Lemma 53. By summing over all $S_1 \in \mathcal{S}_1$, and applying Lemma 53, with $\mathcal{B} = \mathcal{S}_1, P_{S_2} = S_2, T_{S_2} = \mathsf{Bad}(S_2), a = b = 3\alpha$, we get:

$$\begin{aligned}
&\sum_{S_1 \in \mathcal{S}_1} \mathbb{E}\left[2\sum_{S_2} d(S_1, Z_{S_2}(S_1)) + \sum_{S_2, S_2'} d(Z_{S_2}(S_1), Z_{S_2'}(S_1))\right] \\
&\quad \leq 9\alpha\mathsf{cost}(\mathcal{S}_1) + 15\alpha \sum_{S_2 \in \mathcal{S}_{2,B}} \mathsf{cost}(S_2) \\
&\quad \leq 9\alpha\mathsf{cost}(\mathcal{S}_1) + 15\alpha\mathsf{cost}(\mathcal{S}_2).
\end{aligned}$$

Adding the retained-client bound $3\alpha\mathsf{cost}(\mathcal{S}_1)$ from Equation (35) gives Equation (34). This concludes the proof. □

5.5.3 Proof of Theorem 3

If $\alpha > 1/2$, we return $\mathcal{S}_1$ after splitting it into exactly k nonempty clusters. This is possible since $k \leq |D|$, and splitting cannot increase the cost. This is already a 2-approximation since $\mathsf{cost}(\mathcal{S}_1) \leq 2\left(\alpha\mathsf{cost}(\mathcal{S}_1) + (1-\alpha)\mathsf{cost}(\mathcal{S}_2)\right)$.

Thus, we can assume $\alpha \leq 1/2$.

First, we need to show that the algorithm opens only at most k clusters. In fact, the following claim proves that it opens exactly k.

Claim 58. *The algorithm (Phases 1 and 2) opens exactly k clusters.*

Proof. Let $N = |\mathcal{S}_2^{(\mathrm{res})}|, M = |\mathcal{S}_1^{(\mathrm{res})}|, K = k_2 - k$. Every residual $\mathcal{S}_1$-cluster has an incoming residual link, so its representative exists. The representatives are distinct because each linking label has a unique target. Hence, $|\mathcal{S}_2'| = N - M$. By Observation 48, $N - M \geq k_2 - k_1 > K$, so the initial sample of $N - M - K$ non-representatives is well-defined. Phase 2 opens these $N - M - K$ non-representatives and exactly one cluster from each of the M pairs $(S_1, M(S_1))$. Together with the matched $\mathcal{S}_2$-clusters, the number of opened clusters is therefore $|\mathcal{S}_2^{(\mathrm{match})}| + (N - M - K) + M = k_2 - K = k$. □

Claim 59. *The final clusters form a partition of D into exactly k nonempty clusters.*

Proof. We first show that Phase 3 assigns every client to an open cluster. We consider two cases, depending on whether the client is good or bad.

For a bad client c that is not already contained in an opened cluster, let $S_2 = C_2(c)$ be its original cluster in $\mathcal{S}_2$. This implies that S_2 is a closed residual cluster by Phase 2. Observe that its linking target $S_1 = \ell_L(S_2)$ must also be in $\mathcal{S}_1^{\text{res}}$, since preprocessing could not have removed S_1 while it has an incoming link from S_2. Now, there are two cases. If $S_2 = M(S_1)$, then since exactly one of S_2, S_1 must be open, we know S_1 is open. Otherwise, the closed non-representative S_2 forces S_1 to open. In any case, this implies that the bad client has an opened destination to be assigned.

Now, consider a good client c not already contained in an open cluster. If $C_1(c) \in \mathcal{S}_1^{\text{match}}$, its matching label is non empty and belongs to $\mathcal{S}_2^{\text{match}}$, so its destination is opened. Otherwise, $C_1(c)$ is residual and was not opened. When this cluster was processed, at least one third of its good residual clients were already in opened $\mathcal{S}_2$-clusters. Since, this set of good residual clients contains c, it is non empty. The sampled client therefore exists and provides an opened destination for c to go to.

The priority given to opened $\mathcal{S}_2$-clusters makes the final clusters disjoint, and the assignments above cover every client in D. Each opened $\mathcal{S}_2$-cluster retains its original clients. If S_1 opens, its representative $M(S_1)$ closes, so the bad client defining their link belongs to $S_1 \setminus D_2$. Thus, every opened cluster is nonempty. By the preceding claim, there are exactly k such clusters, so the output is a feasible k-solution. □

Claim 60. *The expected cost of solution is* $\mathbb{E}[\mathsf{cost}(\mathcal{S})] \le 108\left(\alpha\mathsf{cost}(\mathcal{S}_1) + (1-\alpha)\mathsf{cost}(\mathcal{S}_2)\right).$

Proof. Add the bounds for $\mathcal{S}_2$ and $\mathcal{S}_1$ from Equations (33) and (34) respectively, we get:

$$\mathbb{E}[\mathsf{cost}(\mathcal{S})] \le 108\alpha\mathsf{cost}(\mathcal{S}_1) + (1+69\alpha)\mathsf{cost}(\mathcal{S}_2) \le 108\left(\alpha\mathsf{cost}(\mathcal{S}_1) + (1-\alpha)\mathsf{cost}(\mathcal{S}_2)\right).$$

The last inequality follows from the fact that $\alpha \le 1/2$. Thus, we get an approximation factor of 108. □

Acknowledgment and AI Usage

The authors developed all algorithms and analysis except for the proof of Lemma 50 (see the disclosure after the lemma for more details), and then used GPT-6 Astra to verify and polish all the writeup. All figures in this paper are generated by GPT-6 Astra under the guidance of the authors.

At the early stage of the development for bipoint rounding (after we have found the constant-factor LMP approximation), several authors tried to one-shot an algorithm using GPT-5.6 Sol and GPT-6 Astra, but did not successfully find one. In hindsight, all the candidate algorithms generated (which work for some special cases) are completely different from the one independently proposed by the authors.

References

[ACC+27] Aditya Anand, Moses Charikar, Vincent Cohen-Addad, Ruiquan Gao, Fabrizio Grandoni, Euiwoong Lee, Amatya Sharma, and Ernest van Wijland. "Spectral Dual Fitting for k-Means". In: *Proceedings of the 2027 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. To appear. 2027. URL: https://arxiv.org/abs/2607.14654.

[AGK+04] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. "Local Search Heuristics for k-Median and Facility Location Problems". In: *SIAM J. Comput.* 33.3 (2004), pp. 544–562. DOI: 10.1137/S0097539702416402. URL: https://doi.org/10.1137/S0097539702416402.

[ANSW20] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. "Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms". In: *SIAM J. Comput.* 49.4 (2020). DOI: 10.1137/18M1171321. URL: https://doi.org/10.1137/18M1171321.

[BCF+26a] Rares-Darius Buhai, Vincent Cohen-Addad, Henry Fleischmann, Anupam Gupta, Antoni Jubés Monforte, Madhusudhan Reddy Pittu, Miltiadis Stouras, Ola Svensson, and Lukas Vogl. *A Better-than-2 Approximation for k-Median Part II: Stable Instances.* 2026. arXiv: 2610.17172 [cs.DS]. URL: https://arxiv.org/abs/2610.17172.

[BCF+26b] Rares-Darius Buhai, Vincent Cohen-Addad, Henry Fleischmann, Anupam Gupta, Antoni Jubés Monforte, Madhusudhan Reddy Pittu, Miltiadis Stouras, Ola Svensson, and Lukas Vogl. *A Better-than-2 Approximation for k-Median Part I: The Power of General Bids.* 2026. arXiv: 2610.18144 [cs.DS]. URL: https://arxiv.org/abs/2610.18144.

[BCR01] Yair Bartal, Moses Charikar, and Danny Raz. "Approximating min-sum k-clustering in metric spaces". In: *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece.* ACM, 2001, pp. 11–20. DOI: 10.1145/380752.380754. URL: https://doi.org/10.1145/380752.380754.

[BERW24] Moritz Buchem, Katja Ettmayr, Hugo K. K. Rosado, and Andreas Wiese. "A $(3+\varepsilon)$-approximation algorithm for the minimum sum of radii problem with outliers and extensions for generalized lower bounds". In: *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 2024, pp. 1738–1765. DOI: 10.1137/1.9781611977912.69. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611977912.69. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611977912.69.

[BFSS15] Babak Behsaz, Zachary Friggstad, Mohammad R. Salavatipour, and Rohit Sivakumar. "Approximation Algorithms for Min-Sum k-Clustering and Balanced k-Median". In: *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I.* Vol. 9134. Springer, 2015, pp. 116–128. DOI: 10.1007/978-3-662-47672-7_10. URL: https://doi.org/10.1007/978-3-662-47672-7%5C_10.

[BFSS19] Babak Behsaz, Zachary Friggstad, Mohammad R. Salavatipour, and Rohit Sivakumar. "Approximation Algorithms for Min-Sum k-Clustering and Balanced k-Median". In: *Algorithmica* 81.3 (2019), pp. 1006–1030. DOI: 10.1007/S00453-018-0454-1. URL: https://doi.org/10.1007/s00453-018-0454-1.

[BGH+26] Jaroslaw Byrka, Yuhao Guo, Yang Hu, Shi Li, Chengzhang Wan, and Zaixuan Wang. "k-Clustering via Iterative Randomized Rounding". In: *Proceedings of the 67th IEEE Symposium on Foundations of Computer Science, FOCS 2026.* To appear. 2026.

[BOR21] Sandip Banerjee, Rafail Ostrovsky, and Yuval Rabani. "Min-Sum Clustering (With Outliers)". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, University of Washington, Seattle, Washington, USA (Virtual Conference), August 16-18, 2021.* Vol. 207. Schloss

Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 16:1–16:16. DOI: 10.4230/LIPICS.APPROX/RANDOM.2021.16. URL: https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2021.16.

[BPR+17] 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: *ACM Trans. Algorithms* 13.2 (2017), 23:1–23:31. DOI: 10.1145/2981561. URL: https://doi.org/10.1145/2981561.

[CCG+25] Moses Charikar, Vincent Cohen-Addad, Ruiquan Gao, Fabrizio Grandoni, Euiwoong Lee, and Ernest van Wijland. "An Improved Greedy Approximation for (Metric) k-Means". In: *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*. IEEE, 2025, pp. 233–240. DOI: 10.1109/FOCS63196.2025.00016. URL: https://doi.org/10.1109/FOCS63196.2025.00016.

[CCG+26] Moses Charikar, Vincent Cohen-Addad, Ruiquan Gao, Fabrizio Grandoni, Euiwoong Lee, and Ernest van Wijland. "A (4+)-Approximation for Euclidean k-Means via Non-monotone Dual-Fitting". In: *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*. Salt Lake City, UT, USA: Association for Computing Machinery, 2026, pp. 1881–1891. ISBN: 9798400725364. DOI: 10.1145/3798129.3800894. URL: https://doi.org/10.1145/3798129.3800894.

[CCGG98] 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, Dallas, Texas, USA, May 23-26, 1998*. ACM, 1998, pp. 114–123. DOI: 10.1145/276698.276719. URL: https://doi.org/10.1145/276698.276719.

[CEMN22] Vincent Cohen-Addad, Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. "Improved approximations for Euclidean k-means and k-median, via nested quasi-independent sets". In: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. ACM, 2022, pp. 1621–1628. DOI: 10.1145/3519935.3520011. URL: https://doi.org/10.1145/3519935.3520011.

[CG99] Moses Charikar and Sudipto Guha. "Improved Combinatorial Algorithms for the Facility Location and k-Median Problems". In: *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1999, pp. 378–388. DOI: 10.1109/SFFCS.1999.814609.

[CGL+25] Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, Chris Schwiegelshohn, and Ola Svensson. "A $(2+\varepsilon)$-Approximation Algorithm for Metric k-Median". In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*. ACM, 2025, pp. 615–624. DOI: 10.1145/3717823.3718299. URL: https://doi.org/10.1145/3717823.3718299.

[CGLS23] Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, and Chris Schwiegelshohn. "Breaching the 2 LMP Approximation Barrier for Facility Location with Applications to k-Median". In: *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. SIAM, 2023, pp. 940–986. DOI: 10.1137/1.9781611977554.CH37. URL: https://doi.org/10.1137/1.9781611977554.ch37.

[CGTS02] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. "A Constant-Factor Approximation Algorithm for the k-Median Problem". In: *J. Comput. Syst. Sci.* 65.1 (2002), pp. 129–149.

[CKL21] Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. "On Approximability of Clustering Problems Without Candidate Centers". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. SIAM, 2021, pp. 2635–2648. DOI: 10.1137/1.9781611976465.156. URL: https://doi.org/10.1137/1.9781611976465.156.

[CKM16] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. "Local Search Yields Approximation Schemes for k-Means and k-Median in Euclidean and Minor-Free Metrics". In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, Hyatt Regency, New Brunswick, New Jersey, USA, October 9-11, 2016*. IEEE Computer Society, 2016, pp. 353–364. DOI: 10.1109/FOCS.2016.46. URL: https://doi.org/10.1109/FOCS.2016.46.

[CL12] Moses Charikar and Shi Li. "A Dependent LP-Rounding Approach for the k-Median Problem". In: *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*. Vol. 7391. Springer, 2012, pp. 194–205.

[Coh18] Vincent Cohen-Addad. "A Fast Approximation Scheme for Low-Dimensional k-Means". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. SIAM, 2018, pp. 430–440. DOI: 10.1137/1.9781611975031.29. URL: https://doi.org/10.1137/1.9781611975031.29.

[CP04] Moses Charikar and Rina Panigrahy. "Clustering to minimize the sum of cluster diameters". In: *Journal of Computer and System Sciences* 68.2 (2004). Special Issue on STOC 2001, pp. 417–441. ISSN: 0022-0000. DOI: https://doi.org/10.1016/j.jcss.2003.07.014. URL: https://www.sciencedirect.com/science/article/pii/S0022000003001399.

[CS10] Artur Czumaj and Christian Sohler. "Small Space Representations for Metric Min-sum k-Clustering and Their Applications". In: *Theory Comput. Syst.* 46.3 (2010), pp. 416–442. DOI: 10.1007/S00224-009-9235-1. URL: https://doi.org/10.1007/s00224-009-9235-1.

[EKR61] P. Erdős, Chao Ko, and R. Rado. "Intersection theorems for systems of finite sets". In: *The Quarterly Journal of Mathematics* 12.1 (1961), pp. 313–320. DOI: 10.1093/qmath/12.1.313.

[FJ22] Zachary Friggstad and Mahya Jamshidian. "Improved Polynomial-Time Approximations for Clustering with Minimum Sum of Radii or Diameters". In: *30th Annual European Symposium on Algorithms (ESA 2022)*. Vol. 244. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 56:1–56:14. ISBN: 978-3-95977-247-1. DOI: 10.4230/LIPIcs.ESA.2022.56. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESA.2022.56.

[FRS16] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. "Local Search Yields a PTAS for k-Means in Doubling Metrics". In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, Hyatt Regency, New Brunswick,*

New Jersey, USA, October 9-11, 2016. IEEE Computer Society, 2016, pp. 365–374. DOI: 10.1109/FOCS.2016.47. URL: https://doi.org/10.1109/FOCS.2016.47.

[GGY26] Fabrizio Grandoni, Anupam Gupta, and Jatin Yadav. "FPT Approximation Schemes for Min-Sum Radii and Min-Sum Diameters Clustering". In: *2026 IEEE Annual Symposium on Foundations of Computer Science (FOCS)*. To appear. 2026.

[GH98] Nili Guttmann-Beck and Refael Hassin. "Approximation algorithms for min-sum p-clustering". In: *Discrete Applied Mathematics* 89.1–3 (1998), pp. 125–142. DOI: 10.1016/S0166-218X(98)00100-0.

[GI03] Venkatesan Guruswami and Piotr Indyk. "Embeddings and non-approximability of geometric problems". In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*. ACM/SIAM, 2003, pp. 537–538. URL: http://dl.acm.org/citation.cfm?id=644108.644198.

[GKK+12] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi Varadarajan. "On Clustering to Minimize the Sum of Radii". In: *SIAM Journal on Computing* 41.1 (2012), pp. 47–60. DOI: 10.1137/100798144. eprint: https://doi.org/10.1137/100798144. URL: https://doi.org/10.1137/100798144.

[GOR+22] Fabrizio Grandoni, Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Rakesh Venkat. "A refined approximation for Euclidean k-means". In: *Inf. Process. Lett.* 176 (2022), p. 106251. DOI: 10.1016/J.IPL.2022.106251. URL: https://doi.org/10.1016/j.ipl.2022.106251.

[GPST23] 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, Florence, Italy, January 22-25, 2023*. SIAM, 2023, pp. 987–1011. DOI: 10.1137/1.9781611977554.CH38. URL: https://doi.org/10.1137/1.9781611977554.ch38.

[GT08] Anupam Gupta and Kanat Tangwongsan. "Simpler Analyses of Local Search Algorithms for Facility Location". In: *CoRR* abs/0809.2554 (2008). arXiv: 0809.2554. URL: http://arxiv.org/abs/0809.2554.

[JMM+03] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. "Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP". In: *J. ACM* 50.6 (2003), pp. 795–824. DOI: 10.1145/950620.950621. URL: https://doi.org/10.1145/950620.950621.

[JV01] Kamal Jain and Vijay V Vazirani. "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation". In: *Journal of the ACM (JACM)* 48.2 (2001), pp. 274–296.

[KMN+02] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. "An Efficient k-Means Clustering Algorithm: Analysis and Implementation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24.7 (2002), pp. 881–892.

[LS16] Shi Li and Ola Svensson. "Approximating k-Median via Pseudo-Approximation". In: *SIAM J. Comput.* 45.2 (2016), pp. 530–547. DOI: 10.1137/130938645. URL: https://doi.org/10.1137/130938645.

[NRS24] Ismail Naderi, Mohsen Rezapour, and Mohammad R. Salavatipour. "Approximation schemes for Min-Sum k-Clustering". In: *Discret. Optim.* 54 (2024), p. 100860. DOI: 10.1016/J.DISOPT.2024.100860. URL: https://doi.org/10.1016/j.disopt.2024.100860.

[SG74] Sartaj Sahni and Teofilo F. Gonzalez. "P-Complete Problems and Approximate Solutions". In: *15th Annual Symposium on Switching and Automata Theory, New Orleans, Louisiana, USA, October 14-16, 1974*. IEEE Computer Society, 1974, pp. 28–32. DOI: 10.1109/SWAT.1974.22. URL: https://doi.org/10.1109/SWAT.1974.22.

[SG76] Sartaj Sahni and Teofilo F. Gonzalez. "P-Complete Approximation Problems". In: *J. ACM* 23.3 (1976), pp. 555–565. DOI: 10.1145/321958.321975. URL: https://doi.org/10.1145/321958.321975.

[VKKR03] Wenceslas Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. "Approximation schemes for clustering problems". In: *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*. ACM, 2003, pp. 50–58. DOI: 10.1145/780542.780550. URL: https://doi.org/10.1145/780542.780550.

A Omitted Proofs

A.1 Proof of Theorem 1

Consider a given instance of the problem, and let w be the smallest positive distance and W be the largest distance. If $\mathsf{opt} = 0$, the problem is trivial to solve optimally, hence next assume $\mathsf{opt} \geq w > 0$. Observe also that $\mathsf{opt} \leq n(n-1)W$. Recall hat $2 \leq k < n$.

We start with a solution $\mathcal{S}_1$ opening a single cluster and a solution $\mathcal{S}_2$ opening n clusters, one per client. We associate to $\mathcal{S}_1$ a value $f_1 = n^2W$ and to $\mathcal{S}_2$ a value $f_2 = 0$. Notice that the pairs $(\mathcal{S}_i, f_i)$ satisfy the cost condition in the claim of Theorem 2 and furthermore $k_1 \leq k < k_2$ and $f_2 < f_1$. We next apply recursively the following binary search algorithm. Consider the current pairs $(\mathcal{S}_i, f_i)$, $i \in \{1, 2\}$, and let $k_i = |\mathcal{S}_i|$. We keep the invariant $k_1 \leq k < k_2$ and $f_2 < f_1$. We apply the algorithm from Theorem 2 with value $f = \frac{f_1+f_2}{2}$, hence obtaining a solution $\mathcal{S}'$. The obtained pair $(\mathcal{S}', \frac{f_1+f_2}{2})$ replaces either $(\mathcal{S}_1, f_1)$ or $(\mathcal{S}_2, f_2)$, so as to maintain the invariants $k_1 \leq k < k_2$ and $f_2 < f_1$. The process halts when $f_1 - f_2 \leq \frac{\varepsilon \cdot w}{n}$. For the final two solutions we have

$$\frac{k_2 - k}{k_2 - k_1}\mathsf{cost}(\mathcal{S}_1) + \frac{k - k_1}{k_2 - k_1}\mathsf{cost}(\mathcal{S}_2) \leq \varepsilon w + 2 \times 10^7 \cdot \mathsf{opt} \leq (2 \times 10^7 + \varepsilon)\mathsf{opt}.$$

The claim follows by Theorem 3.