Fair Range k-Supplier Clustering in Offline and Streaming Models

Meiyun Lu Wei Yue Weihong Wu Zeyu Lin Longkun Guo*
Fuzhou University, P.R. China

Abstract

Recognizing the growing importance of fairness in machine learning, we study fair range k-supplier clustering, which seeks to optimize clustering quality while ensuring fair representation across supplier groups. Given sets of suppliers and clients, with each supplier belonging to one or more demographic or functional groups, the objective is to select at most k suppliers to minimize the maximum distance from any client to its nearest selected supplier, subject to prescribed lower and upper bounds on the number of selected suppliers from each group. For disjoint supplier groups, we develop a polynomial-time 3-approximation algorithm in the offline setting and a $(3+\epsilon)$-approximation algorithm in the one-pass client-first streaming model. We further develop a fixed-parameter tractable (FPT) 3-approximation algorithm for overlapping supplier groups, with running time exponential in the number of clusters and groups. Experiments on real-world and synthetic datasets demonstrate competitive clustering quality and substantial runtime improvements over the baselines.

1 Introduction

Classical clustering algorithms aim to optimize global objectives, such as minimizing total assignment cost or the maximum distance between clients and service centers. However, optimizing these objectives alone may produce solutions in which certain demographic or geographic groups are systematically underrepresented or receive poorer service. Clustering methods under fairness constraints address this problem by incorporating sensitive attributes to ensure more balanced representation of different groups within clusters and reduce structural bias [4, 12].

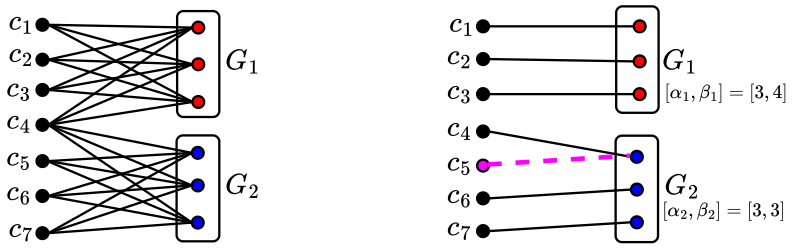
In this paper, we focus on the k-supplier clustering problem, a fundamental optimization model in which a set of clients is assigned to at most k suppliers to minimize the maximum distance between any client and its assigned supplier. This problem arises in numerous practical contexts. For instance, in public service facility location, one must select facilities (e.g., hospitals or emergency centers) from a set of candidate locations to serve a population, while ensuring that different demographic groups are fairly represented in the service [33, 38]. Similarly, in supply chain network design, distribution centers are chosen from potential suppliers to serve geographically distributed demands, while balancing service quality across different supplier groups or regions [21, 28].

The k-supplier problem and its special case, k-center, have been studied extensively from the perspectives of approximation algorithms and computational complexity. Both problems are NP-hard. Moreover, under standard complexity-theoretic assumptions, k-center and k-supplier admit no polynomial-time approximation algorithms with factors better than 2 and 3, respectively [18, 19].

We consider fair range k-supplier, where suppliers are partitioned into groups $F_1, \ldots, F_m$ with lower and upper bounds $[\alpha_l, \beta_l]$. The goal is to select at most k suppliers S satisfying $\alpha_l \leq |S \cap F_l| \leq \beta_l$ for every $l \in [m]$, while minimizing the maximum distance from any client to its nearest selected supplier.

*Corresponding Author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) (b)

Figure 1: An example showing that augmenting a maximum matching can fail. In (a), $\{c_1, \ldots, c_7\}$ denotes the set of clients, and a line between c_i and a supplier set G_ℓ indicates that c_i can be served by a supplier in G_ℓ within radius R^*. In (b), a maximum matching (shown by black solid lines) satisfies the lower bound α_ℓ. However, the unmatched client c_5 can only be matched to a supplier in G_2, violating the upper bound $\beta_2 = 3$.

Although fair k-center has been studied under a broad range of fairness models, including exact representation constraints [2, 10, 16, 23, 26, 43], upper-bound constraints [15], and simultaneous lower- and upper-bound constraints [36], substantially less is known for fair k-supplier variants [7, 13].

Unlike k-center, where centers are chosen from the same set as clients, the k-supplier problem has two sets (clients and suppliers), which leads to asymmetric coverage. This makes it harder to coordinate supplier selection and client coverage under fairness constraints. Prior work [7, 13] studies one-sided fairness constraints using maximum-matching formulations. However, the above asymmetry introduces a fundamental limitation: matching-based methods can typically enforce either the lower-bound constraints or the upper-bound constraints, but not both simultaneously. In particular, a maximum matching that satisfies the lower-bound constraints may fail to be augmentable into a feasible solution that also respects upper bounds. As illustrated in Fig. 1, once facilities are selected to meet lower-bound requirements, the remaining capacity may be insufficient or structurally incompatible with upper-bound constraints, preventing further augmentation.

Modern large-scale applications increasingly require streaming algorithms, where data arrives sequentially and must be processed under strict memory and time constraints. In this setting, maintaining both clustering quality and fairness becomes more challenging, as the full dataset is not available for simultaneous processing. Despite its practical importance, the fair range k-supplier problem has received little attention in streaming models. Motivated by these considerations, we study fair range k-supplier clustering in both offline and streaming settings, aiming to design algorithms with provable approximation guarantees under fairness and resource constraints.

1.1 Related Work

Fair range clustering. Fair range clustering imposes lower and upper bounds on the number of selected centers or facilities from each demographic or operational group. Early work on fair range clustering [23, 26] focused on the fair k-center problem with disjoint demographic groups, while Mahabadi et al. [30] studied fair diversity maximization under pairwise distance-based measures. These works require the number of cluster centers selected from each group to satisfy exact group constraints. Thejaswi et al. [41] studied the variant with intersecting groups, offering complexity results and tight parameterized approximation algorithms for diversity-aware k-median and diversity-aware k-means (with lower bound constraints). They showed that the problem is inapproximable to any multiplicative factor in polynomial time.

Recent work further strengthened the theoretical foundations of fair range clustering. Angelidakis et al. [2] proposed a unified framework combining the lower- and upper-bound representative k-center and private k-center formulations. Dickerson et al. [11] investigated doubly constrained fair clustering, achieving constant-factor approximations under simultaneous group fairness and center diversity constraints. Hotegni et al. [20] provided an efficient constant-factor approximation algorithm for fair range ℓ_p-clustering for all $p \in [1, \infty)$. Zhang et al. [44] developed parameterized approximation schemes for fair range k-median and k-means in Euclidean spaces, achieving an approximation ratio

of $1 + \epsilon$ with fixed-parameter tractable (FPT) running time. More recently, Thejaswi [39] established a tight inapproximability bound for fair k-center, proving that no polynomial-time algorithm can achieve an approximation ratio better than 3 unless $\mathrm{P} = \mathrm{NP}$. This hardness result holds even for two disjoint groups, demonstrating that fairness constraints can fundamentally increase the approximation hardness of the classical k-center problem.

The fair range k-supplier problem further generalizes fair clustering to scenarios where only a subset of points can be selected as cluster centers. For disjoint groups, Chen et al. [7] proposed a polynomial-time 5-approximation algorithm under upper-bound constraints only. Thejaswi et al. [40] extended the work of [41], broadening the framework to incorporate both lower and upper bound constraints, and further generalizing it to k-median, k-means, and k-supplier objectives. They proposed parameterized approximation algorithms, achieving an approximation ratio of 5 for the diversity-aware k-supplier problem under intersecting groups. More recently, Gadekar et al. [13] improved this to a 3-approximation for both disjoint and overlapping supplier groups. Their algorithm runs in polynomial time for disjoint supplier groups under lower-bound constraints. For the more general setting of intersecting supplier groups, including fair range constraints with both lower and upper bounds, they provide an FPT algorithm whose running time has exponential dependence on the number of selected suppliers and groups. Gadekar and Thejaswi [14] studied capacitated fair range clustering with overlapping facility groups, establishing strong inapproximability results even when fairness constraints are trivially satisfiable. These hardness results persist on tree metrics and with logarithmically many groups. For a constant number of groups, they developed polynomial-time approximation algorithms for capacitated fair-range k-median and k-means, along with constant-factor FPT approximation algorithms.

Other fair clustering. An alternative fairness notion enforces balance on the members assigned to each cluster, rather than on the selection of cluster centers. This was first introduced for k-center clustering by [9], requiring each cluster to satisfy equality constraints between demographic groups. It was later generalized to multiple demographic groups and bounded representation constraints in [1, 3, 5, 17]. Beyond demographic group-based definitions, several works study fairness notions not explicitly tied to sensitive attributes. Proportional fairness, which balances clustering outcomes relative to group sizes or individual contributions, has been studied in [8, 34], while individual fairness, which ensures similar individuals are treated similarly, was studied in [31, 35].

Large-scale setting. Several recent works have studied the fair range k-center problem in large-scale settings. Chiplunkar et al. [10] proposed the first constant-factor distributed algorithm and a two-pass streaming algorithm for fair k-center clustering for data summarization. Wu et al. [43] studied the problem in the massively parallel computation model, while Guo et al. [16] designed a one-pass streaming algorithm achieving a $(5+\epsilon)$-approximation under upper-bound group constraints, improving upon the earlier $(7 + \epsilon)$-approximation result of [29]. Building on these studies, Guo et al. [15] focused on the Euclidean setting and introduced a one-pass streaming algorithm with approximation ratio 4.46, improving the previous best ratio of $(5 + \epsilon)$ for this problem in general metrics. Relatedly, fair diversity maximization has also been studied in the streaming setting under exact group constraints [42]. Unlike fair k-center, this line of work aims to maximize pairwise diversity among selected points rather than minimizing a covering radius.

1.2 Our Contributions

The main contributions of this paper are summarized as follows:

- We present the first polynomial-time 3-approximation algorithm for fair range k-supplier clustering with disjoint supplier groups, satisfying both lower and upper bounds on the number of selected suppliers from each group.
- We extend our framework to the one-pass client-first streaming model and obtain a $(3 + \epsilon)$-approximation algorithm. To our knowledge, this is the first streaming algorithm for fair range k-supplier clustering with an approximation ratio arbitrarily close to the optimal factor of 3.
- We develop a minimum-cost maximum-flow (MCMF) formulation to enforce both lower and upper bounds, addressing a limitation of existing matching-based approaches. The total unimodularity of the underlying constraint matrix guarantees an integral optimal solution to the linear program, enabling polynomial-time center selection under both types of constraints.

- We evaluate our algorithms in offline and streaming settings using real-world and synthetic datasets, demonstrating competitive clustering quality and runtime efficiency compared with state-of-the-art methods. Experimental results on synthetic datasets and additional results on real-world datasets are provided in Appendix D.

In addition, for overlapping supplier groups, we develop a 3-approximation algorithm that is fixed-parameter tractable in the number of clusters and groups (see App. C).

2 Preliminaries and Problem Formulation

Let (U, d) be a metric space, where $d : U \times U \to \mathbb{R}_{\geq 0}$ is a distance function satisfying the triangle inequality. An instance of the problem is specified by positive integers k and m, a set of clients $C \subseteq U$, and a set of suppliers $F \subseteq U$. We assume that $U = F \cup C$ and denote $|U| = n$. The set F consists of m groups $\mathcal{G} = \{F_1, F_2, \ldots, F_m\}$ with $\bigcup_{l=1}^{m} F_l = F$. Each group F_l is associated with a fairness range $[\alpha_l, \beta_l]$, where $\alpha_l, \beta_l \in \mathbb{Z}_{\geq 0}$ and $\alpha_l \leq \beta_l$. The number of selected suppliers from F_l must lie within this range. For brevity, let $[m] = \{1, 2, \ldots, m\}$.

A feasible solution consists of a subset of suppliers $S \subseteq F$ with $|S| \leq k$ that satisfies all fairness range constraints. The objective of the fair range problem is to minimize the maximum distance from any client to its nearest selected supplier:

$$\begin{aligned} \min_{S \subseteq F} \quad & \max_{c \in C} d(c, S) \\ \text{s.t.} \quad & \alpha_l \leq |S \cap F_l| \leq \beta_l, \quad \forall l \in [m], \\ & |S| \leq k, \end{aligned}$$

where $d(c, S) = \min_{s \in S} d(c, s)$. Given a radius $R \geq 0$, we say that a supplier $f \in F$ can *serve* a client $c \in C$ if $d(f, c) \leq R$. A set of suppliers $S \subseteq F$ can *serve* a client c within radius R if there exists some $f \in S$ that can serve c. We say that S serves a client set $C' \subseteq C$ within radius R if every client in C' is served by S. We denote the fair range problem with disjoint groups by FR-k-SUP.

Definition 1. (Maximal λ-independent set) We say $\Gamma \subseteq C$ is a maximal λ-independent set of C if and only if it satisfies the following two conditions: (i) for any two distinct points $p, q \in \Gamma$, the distance between them is larger than λ, i.e., $d(p, q) > \lambda$; (ii) for any point $p \in C$, there exists a point $q \in \Gamma$ such that $d(p, q) \leq \lambda$.

3 Approximation Algorithms for FR-k-SUP

In this section, we describe an offline algorithm and its one-pass client-first streaming implementation for FR-k-SUP. We additionally assume that $\beta_l \leq |F_l|$ for every $l \in [m]$ and $\sum_{l=1}^{m} \beta_l \geq k$.

3.1 Offline Algorithm

For ease of exposition, we first describe the offline algorithm assuming that the optimal radius $R^* = \max_{c \in C} d(c, S^*)$ is known, where S^* is an optimal supplier set. This assumption is removed using the standard radius-guessing approach [6, 22], which finds a successful candidate radius $R \leq R^*$ without affecting the approximation guarantee, as detailed in Appendix A.8. Given R^*, the algorithm takes as input a supplier set F, a client set C, an integer k specifying the maximum number of selected suppliers, and fairness-bound vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_{\geq 0}^{m}$. Algorithm 1 presents the offline algorithm, which consists of three stages:

- **Representative client selection stage.** We construct a maximal $2R^*$-independent set $C' \subseteq C$ as the representative client set.
- **Feasible supplier identification stage.** For each selected representative client, we identify all suppliers that can serve it within distance R^*, organized by group membership.
- **Flow-based supplier selection stage.** We construct an MCMF instance to enforce the fair range constraints, and recover a feasible supplier set from the resulting flow.

Algorithm 1: Offline algorithm for FR-k-SUP

Input: A client set C, a supplier set F, a partition $\mathcal{G} = \{F_1, \ldots, F_m\}$ of F, the maximum number of selected suppliers k, fairness-parameter vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}$, and radius R^*
Output: A subset of suppliers S

`// Stage 1: Selection of representative client set` C'.
1 $C' \leftarrow \{c\}$, where c is an arbitrary client in C;
2 **for** $c \in C$ **do**
3 **if** $d(c, C') > 2R^*$ **then**
4 $C' \leftarrow C' \cup \{c\}$;

`// Stage 2: Identification of feasible supplier set` F_c^l `for each` $c \in C'$ `and each` $l \in [m]$.
5 **for** $c \in C'$ **do**
6 **for** $l \in [m]$ **do**
7 $F_c^l \leftarrow \{\, f \in F_l \mid d(f, c) \leq R^* \,\}$;

`// Stage 3: Selection of suppliers serving the clients via the flow network.`
8 $S \leftarrow$ SELECT-SUPPLIERS$(C', F, \mathcal{G}, \{F_c^l\}_{c \in C', l \in [m]}, \boldsymbol{\alpha}, \boldsymbol{\beta}, k)$; `// see Alg. 2`
9 **return** S.

3.1.1 Representative Client Selection

In this stage, we construct a maximal $2R^*$-independent set $C' \subseteq C$ and use it as the representative client set, as shown in Lines 1–4 of Algorithm 1. Specifically, we initialize C' with an arbitrary client from C and then examine each client $c \in C$ once, adding it to C' if $d(c, C') > 2R^*$. The following lemmas establish the key properties of C'.

Lemma 1. *Let $C' \subseteq C$ be a maximal $2R^*$-independent set. Then (i) $d(c_i, c_j) > 2R^*$ for any two distinct $c_i, c_j \in C'$; (ii) for every $c \in C$, there exists $c' \in C'$ such that $d(c, c') \leq 2R^*$; and (iii) $|C'| \leq k$.*

3.1.2 Feasible Supplier Identification

As shown in Lines 5–7 of Algorithm 1, for each representative client $c \in C'$, we identify the suppliers that can serve c within distance R^*, classified by their group membership. Since the supplier set F is partitioned into m groups $\{F_l\}_{l \in [m]}$, for each group $l \in [m]$ we define $F_c^l = \{f \in F_l : d(f, c) \leq R^*\}$. That is, F_c^l denotes the set of suppliers from group l that are feasible for client c under radius R^*.

3.1.3 Flow-based Supplier Selection

Note that the representative client set C' is not a feasible solution to FR-k-SUP, as it consists of clients rather than suppliers. Therefore, an additional step is required to recover a feasible supplier set based on C'. To this end, we reduce the problem of selecting a set of suppliers that serves all clients in C' within radius R^* and satisfies the fair range constraints to an MCMF instance by constructing a flow network $G = (V, E)$.

With a slight abuse of notation, we use the same symbols $\mathcal{G} = \{F_1, F_2, \ldots, F_m\}$ to denote both the m supplier groups and the corresponding group nodes in the flow network. Specifically, each node F_l in the network represents the supplier group F_l. We further construct a copy of these group nodes, denoted by $\mathcal{G}' = \{F_1', F_2', \ldots, F_m'\}$, where F_l' is the auxiliary node corresponding to F_l for each $l \in [m]$. The vertex set of the network is defined as $V = \{s, t\} \cup \mathcal{G} \cup \mathcal{G}' \cup \bigcup_{c \in C', l \in [m]} F_c^l \cup C'$, where s and t denote the source and sink, respectively.

The directed edges of G are constructed as follows, with their capacity ranges and costs summarized in Table 3 in Appendix A.1.

- For each group $F_l \in \mathcal{G}$, we add an edge (s, F_l) with capacity range $[\alpha_l, \beta_l]$ and cost 0, and an edge (F_l, F_l') with capacity range $[0, \beta_l]$ and cost 0.
- For each $F_l' \in \mathcal{G}'$, we add an edge (F_l', t) with capacity range $[0, \beta_l]$ and unit cost.

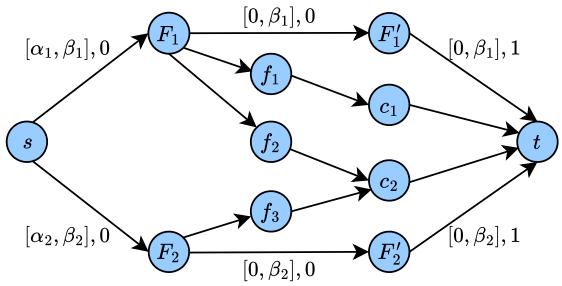


Figure 2: An example of the flow network used in the MCMF formulation. Each group node F_l is connected to the source s with capacity range $[\alpha_l, \beta_l]$ and cost 0, and to its auxiliary node F_l' with capacity range $[0, \beta_l]$ and cost 0. Each auxiliary node F_l' is connected to the sink t with capacity range $[0, \beta_l]$ and unit cost. For each relevant supplier $f \in F_l$, the edge (F_l, f) has capacity range $[0, 1]$ and cost 0. For each client c and supplier f with $d(c, f) \leq R^*$, the edge (f, c) has capacity range $[0, 1]$ and cost 0, while the edge (c, t) has capacity range $[0, 1]$ and cost 0. The source s has supply k.

- For each group $F_l \in \mathcal{G}$ and each distinct supplier $f \in \bigcup_{c \in C'} F_c^l$, we add a single edge (F_l, f) with capacity range $[0, 1]$. For every pair (f, c) such that $c \in C'$ and $f \in F_c^l$, we add an edge (f, c) with capacity range $[0, 1]$. We also add an edge (c, t) with capacity range $[0, 1]$ for each $c \in C'$. All these edges have cost 0.

We set the source supply to k, while the lower capacities on the edges (s, F_l) enforce the lower-bound constraints. Figure 2 illustrates the constructed flow network. Minimum-cost flow admits strongly polynomial-time algorithms [37] and integral optimal solutions under integral capacities and supplies [27], yielding the following result.

Theorem 1. *For the constructed flow network $G = (V, E)$, LP 1, provided in Appendix A.2, formulates the MCMF problem. Whenever this LP is feasible, it admits an integral optimal solution.*

After solving the MCMF instance, let $\boldsymbol{x}$ denote the resulting integral flow. We construct the final supplier set S as follows. For each representative client $c \in C'$, we add to S the unique supplier $f_c^l \in F_c^l$ such that $x_{f_c^l c} = 1$. Then, for each supplier group $F_l \in \mathcal{G}$, if $x_{F_l' t} > 0$, we arbitrarily select $x_{F_l' t}$ additional suppliers from $F_l \setminus S$ and add them to S.

Lemma 2. *Given an optimal radius R^* and a $2R^*$-independent representative client set C', there exists a supplier set $S \subseteq F$ satisfying: (i) $|S| = k$; (ii) $d(c, S) \leq R^*$ for every $c \in C'$; and (iii) $\alpha_l \leq |S \cap F_l| \leq \beta_l$ for every $l \in [m]$, if and only if the constructed flow network G admits an integral feasible flow of value k and cost $k - |C'|$. Such a flow is necessarily a minimum-cost flow of value k.*

Theorem 2. *Algorithm 1 is a 3-approximation algorithm for* FR-k-SUP. *Moreover, the algorithm runs in $O\big((nk)^{4.5} \log^2 n\big)$ time.*

3.2 Client-First Streaming Algorithm

We study FR-k-SUP in the one-pass streaming model. Our goal is to design a streaming algorithm that selects at most k suppliers to serve all clients while satisfying the fairness constraints and minimizing the service radius. A key challenge in the streaming setting is that decisions are irrevocable, and thus the order in which clients and suppliers arrive has a crucial impact on the quality of the solution. In the following, we discuss this problem and present our algorithm to address it.

Limitation of arbitrary arrival orders. Suppose a large number of suppliers arrive early in the stream. The algorithm may be forced to select some suppliers to satisfy the fairness constraints, even though they are arbitrarily far from the clients that arrive later. As a result, such irrelevant suppliers occupy the selection budget, causing the algorithm to discard those suppliers that are close to the clients. Since streaming decisions are irreversible, the algorithm cannot revise these early choices. Consequently, it may satisfy the fairness constraints while failing to serve the clients, leading to an unbounded clustering cost. We illustrate this phenomenon with an example in Figure 3. This example

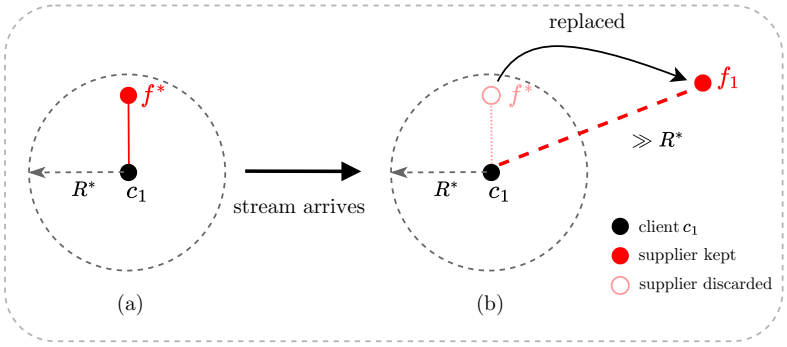


Figure 3: An example showing that suppliers arriving before clients in the streaming setting can lead to an unbounded approximation ratio. Let $k = 1$, $C = \{c_1\}$, and suppose there is a single supplier group ($m = 1$) with $[\alpha_1, \beta_1] = [1, 1]$. In (a), a supplier f^* within distance R^* of c_1 arrives early in the stream and is retained. In (b), another supplier f_1 from the same group arrives later and replaces f^*, which is then discarded. Although the resulting solution satisfies the fairness constraint, f_1 can be arbitrarily far from c_1, yielding an unbounded approximation ratio.

shows that without controlling the arrival order, the approximation guarantee can break down. To avoid this issue, we require that clients arrive before suppliers.

Motivation for the client-first streaming model. We consider large-scale database systems where client and supplier datasets cannot fit entirely into memory, but their processing order can be controlled. In such settings, the system can first stream the clients to construct a compact representative summary and then process the suppliers to select a feasible set satisfying the fair range constraints. Thus, the client-first model captures a controlled data-access order in practical large-scale applications.

3.2.1 Algorithm for Client-First Arrival

We adopt the multi-scale McCutchen–Khuller framework [32], maintaining representative client sets at multiple distance scales. During the client stream, a streaming k-center procedure maintains a representative set C' with $|C'| \leq k$ at each active scale. During the subsequent supplier stream, we maintain a bounded supplier set S_l for each group $l \in [m]$, containing suppliers close to the representatives. After the stream, we apply the offline FR-k-SUP algorithm to each scale using C' and the candidate suppliers in $\bigcup_{l\in[m]} S_l$, and return the best feasible solution across all scales.

Client streaming stage. We use the multi-scale McCutchen–Khuller framework [32]. Fix $\epsilon \in (0, 1]$. We buffer clients until $k + 1$ distinct clients have arrived or the client stream ends. Let B be the buffered set, t_0 the time when buffering ends, and c_1 the first client.

If $|B| \geq 2$, initialize L and U_{t_0} as the minimum and maximum pairwise distances in B, respectively, and introduce the positive scales $\Delta_j = L(1 + \epsilon/2)^j$ for $0 \leq j \leq J_{t_0} = \lceil \log_{1+\epsilon/2}(U_{t_0}/L) \rceil$. At each initial positive scale, set $C'_j = \{c_1\}$ and scan the remaining buffered clients in arrival order, adding c whenever $d(c, C'_j) > \Delta_j$. Discard a scale whenever $|C'_j| > k$. A discarded scale must satisfy $\Delta_j < 2R^*$, since otherwise its representatives would require more than k distinct optimal suppliers. If $|B| \leq k$, buffering has ended with the client stream, so $B = C$; additionally introduce a zero scale $\Delta_z = 0$ with $C'_z = C$. When $|B| = 1$, only the zero scale is maintained.

Before processing each subsequent client c_t, let $\mathcal{A}_{t-1}$ denote the set of active scales. For each $\Delta \in \mathcal{A}_{t-1}$, let $C'_{\Delta,t-1}$ denote the representative client set maintained at scale Δ before processing c_t. Update $U_t = \max\left\{U_{t-1}, \max_{\Delta\in\mathcal{A}_{t-1}} \max_{c'\in C'_{\Delta,t-1}} d(c_t, c')\right\}$, and set $J_t = \lceil \log_{1+\epsilon/2}(U_t/L) \rceil$. Introduce the scales with $J_{t-1} < j \leq J_t$, initializing each with $C'_j = \{c_1\}$. Then process c_t at every active scale using the same insertion and discarding rules. For $|B| \geq 2$, at every time $t \geq t_0$, the number of active instances, including the zero scale when present, is $O(1 + \lceil \log_{1+\epsilon/2}(U_t/L) \rceil)$.

Each active set contains at most k representatives, pairwise separated by distances greater than its scale. The following lemma establishes that every active set covers all processed clients within its

scale and that the largest scale remains active with representative set $\{c_1\}$. Consequently, after the client stream, each remaining C'_j is a maximal Δ_j-independent set.

Lemma 3. *Let $k \geq 1$. Assume $|B| \geq 2$. After initialization and after processing any subsequent client c_t, where $t \geq t_0$, let U_t, J_t, and $\mathcal{A}_t$ denote the current distance bound, maximum scale index, and set of active scales, respectively. Then the following properties hold: (i) $d(c_i, c_1) \leq U_t$ for every processed client c_i; (ii) the maximum scale Δ_{J_t} remains active, with $C'_{\Delta_{J_t},t} = \{c_1\}$; and (iii) $d(c_i, C'_{\Delta,t}) \leq \Delta$ for every active scale $\Delta \in \mathcal{A}_t$ and every processed client c_i.*

Lemma 4. *Let U denote the final value of the maintained distance bound at the end of the client streaming stage. Assume $|B| \geq 2$ and $L \leq 2R^* \leq U$. There exists an active positive scale Δ_j remaining after this stage such that $2R^* \leq \Delta_j < (2+\epsilon)R^*$. Moreover, distinct representatives in C'_j belong to different clusters in an optimal solution.*

After the client streaming phase, every representative client set is further processed by Algorithm 3. When $2R^* \leq U$, the above lemma guarantees that Algorithm 3 returns the desired solution when applied to the candidate set C'_j satisfying $\Delta_{j-1} < 2R^* \leq \Delta_j$. Moreover, when $2R^* > U$, we can also obtain the desired solution by applying Algorithm 3 to C'_J.

Supplier streaming stage. After the client streaming stage, the algorithm processes the supplier stream independently for each remaining active scale Δ_j and its corresponding representative client set C'_j. Thus, each arriving supplier is processed with respect to every active scale, and a separate candidate supplier set is maintained for each pair of scale and supplier group.

We describe the supplier-streaming procedure for a fixed active scale Δ and its corresponding representative client set C'. The same procedure is applied independently to every active scale. For each supplier group $l \in [m]$, let S_l denote the set of currently retained group-l suppliers, initialized as $S_l = \emptyset$. When a new supplier $f \in F_l$ arrives, define the temporary candidate set $A_l = S_l \cup \{f\}$. For every representative client $c \in C'$, choose $f_c^l \in \operatorname{argmin}_{g \in A_l} d(c, g)$, and define the protected supplier set $P_l = \{f_c^l \mid c \in C'\}$. Thus, P_l contains, for each representative client in C', one supplier in A_l attaining the minimum distance to that client. If $|A_l| \leq k$, the algorithm retains all suppliers and sets $S_l = A_l$. Otherwise, since $|S_l| \leq k$ before the arrival of f, we have $|A_l| = k+1$. Moreover, $|P_l| \leq |C'| \leq k$, and hence $A_l \setminus P_l \neq \emptyset$. The algorithm selects an arbitrary supplier $f' \in A_l \setminus P_l$ and updates $S_l = A_l \setminus \{f'\}$. Therefore, $|S_l| \leq k$ is maintained throughout the supplier stream, while at least one closest observed group-l supplier to every representative client in C' is retained.

Lemma 5. *Fix an active scale and its representative client set C'. For every representative client $c \in C'$ and every supplier group $l \in [m]$, the final retained set S_l contains at least one closest supplier to c among all suppliers in F_l. Equivalently, $\min_{g \in S_l} d(c, g) = \min_{g \in F_l} d(c, g)$.*

At the end of the supplier stream, for each active scale and each supplier group $l \in [m]$, let S_l denote the final retained supplier set. By construction, $|S_l| \leq k$. Moreover, by Lemma 5, for every representative client $c \in C'$, the set S_l contains at least one closest group-l supplier to c among all suppliers in F_l.

Post-streaming stage. Let $\mathcal{A}$ denote the set of active scales remaining after the client streaming stage. We describe the procedure for a fixed scale $\Delta \in \mathcal{A}$, suppressing the dependence on Δ in the notation. The same procedure is applied independently to every active scale.

Let $\widehat{F} = \bigcup_{l \in [m]} S_l$ be the retained supplier set for this scale. The retention procedure ensures that $|S_l| = \min\{k, |F_l|\}$ for every $l \in [m]$, and hence $|\widehat{F}| \leq mk$. We define the candidate radius set as $\mathcal{D} = \{d(c, f) : c \in C',\ f \in \widehat{F}\}$. Since $|C'| \leq k$, there are at most $|C'|\,|\widehat{F}| \leq mk^2$ distinct candidate radii. To test a candidate radius $\rho \in \mathcal{D}$, we construct a threshold flow network using the fixed representative set C' and the retained supplier groups $\{S_l\}_{l \in [m]}$. We use the same network structure, capacities, and costs as in the offline supplier-selection subroutine, with the original fairness bounds $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. A retained supplier $f \in S_l$ is connected to a representative client $c \in C'$ if and only if $d(c, f) \leq \rho$. The representative set C' remains fixed throughout these radius tests.

Let z_ρ denote the optimal objective value of LP 1 on this threshold network, with $z_\rho = +\infty$ if the LP is infeasible. We call ρ *feasible* if $z_\rho = k - |C'|$. For a feasible radius, Theorem 1 guarantees an integral optimal flow x_ρ. We recover a supplier set $S_\rho \subseteq \widehat{F}$ from x_ρ using the supplier-selection rule of the offline subroutine. As established in Lemma 6, the recovered set satisfies $|S_\rho| = k$, $\alpha_l \leq |S_\rho \cap F_l| \leq \beta_l$ for every $l \in [m]$, and $\max_{c \in C'} d(c, S_\rho) \leq \rho$.

Lemma 6. *For any feasible candidate radius $\rho \in \mathcal{D}$, the post-streaming procedure recovers a supplier set $S \subseteq \widehat{F}$ satisfying $|S| = k$, all fair range constraints, and $\max_{c \in C'} d(c, S) \leq \rho$.*

Lemma 7. *If $\rho \in \mathcal{D}$ is feasible, then every candidate radius $\rho' \in \mathcal{D}$ with $\rho' \geq \rho$ is also feasible.*

By Lemma 7, feasibility is monotone over the candidate radii. We now restore the scale dependence in the notation. For each active scale $\Delta \in \mathcal{A}$, we sort the distinct candidate radii in $\mathcal{D}_\Delta$ and use binary search to find the minimum feasible radius $\rho_\Delta = \min\{\rho \in \mathcal{D}_\Delta \mid z_{\Delta,\rho} = k - |C'_\Delta|\}$, where $z_{\Delta,\rho}$ denotes the optimal objective value of the corresponding threshold LP, with $z_{\Delta,\rho} = +\infty$ if the LP is infeasible. If no feasible radius exists, we discard the scale. Otherwise, let S_Δ denote the supplier set recovered from an integral optimal flow at radius ρ_Δ.

We apply this procedure independently to every active scale and define $\mathcal{A}' = \{\Delta \in \mathcal{A} \mid \mathcal{D}_\Delta \text{ contains a feasible radius}\}$. For every $\Delta \in \mathcal{A}'$, each original client is within distance at most Δ of a representative in C'_Δ. By Lemma 6, every representative in C'_Δ is within distance at most ρ_Δ of a supplier in S_Δ. Thus, the triangle inequality gives $\max_{c \in C} d(c, S_\Delta) \leq \Delta + \rho_\Delta$. The algorithm selects $\Delta^* \in \arg\min_{\Delta \in \mathcal{A}'} (\Delta + \rho_\Delta)$ and returns S_{Δ^*}. The following lemma guarantees that $\mathcal{A}' \neq \emptyset$ and establishes the approximation bound.

Lemma 8. *Let (S^*, R^*) be an optimal solution to the* FR-k-SUP *instance. There exists an active scale $\Delta \in \mathcal{A}'$ whose minimum feasible radius ρ_Δ satisfies $\rho_\Delta \leq R^*$ and $\Delta + \rho_\Delta \leq (3+\epsilon)R^*$.*

Theorem 3. *There is a $(3+\epsilon)$-approximation one-pass client-first streaming algorithm for the* FR-k-SUP *problem. The total space required by Algorithm 3 is $O\Big(\max\Big\{mk(1 + \lceil \log_{1+\epsilon/2}(U/L) \rceil), mk^2\Big\}\Big)$, for any fixed $\epsilon \in (0, 1]$, where L and U denote the lower and upper distance bounds maintained by the multi-scale framework.*

4 Experiments

In this section, we evaluate our algorithms for FR-k-SUP and compare them with state-of-the-art baselines[2]. All experiments are conducted over at least 10 independent runs. Experiments are conducted on a Linux machine with a 32-core Hygon C86 7380 CPU and 256 GiB RAM using Python 3.10. All LP formulations are solved with the Gurobi solver. Baseline implementations are adapted from the publicly available code released by [13]. We evaluate our algorithms on real-world and synthetic datasets. Results on synthetic datasets and additional results on real-world datasets are provided in Appendix D.

Baselines. For instances with disjoint groups, we compare the following algorithms: the standard 3-approximation greedy algorithm for the k-supplier problem [19], denoted by Unfair; our offline algorithm, denoted by Ours (off.); our streaming algorithm, denoted by Ours (str.); the FPT algorithm of [13], denoted by FPT; and the 5-approximation algorithm of [7], denoted by 5-apx. The algorithm of [13] is the closest existing baseline for fair range k-supplier that supports both lower- and upper-bound constraints. We therefore include it as the strongest available fair range comparison.

We use the original 5-approximation algorithm of [7] without modification. Because this algorithm supports only upper-bound constraints, we apply it using the upper bounds of our fair-range instances while ignoring the corresponding lower bounds. We include 5-apx as an upper-bound-only reference and report both its solution cost and any violations of the lower-bound constraints.

4.1 Experimental Analysis

Scalability on real-world datasets. We evaluate our algorithms on nine datasets from the UCI Machine Learning Repository [25], summarized in Table 4 in Appendix D. Datasets are preprocessed using one-hot encoding and min–max normalization. Data points are treated as clients, while a subset is selected as suppliers according to the dataset-specific selection rules. The suppliers are then partitioned into two disjoint groups: randomly for the `Bank` and `Bank-full` datasets, and based on the `sex` attribute for the remaining datasets. Note that the `sex` attribute is used only to determine supplier group membership and is excluded from distance calculations.

Table 1 and Table 2 respectively report the clustering cost and the running time of all the algorithms on real-world datasets for $k = 10$, with $[\alpha_i, \beta_i] = [4, 8]$. For the streaming k-center procedure used

[2]Source code and datasets: `https://github.com/weihong-wu/fair-range-k-supplier-clustering`.

Table 1: Cost comparison under fair range constraints $[\alpha_i, \beta_i] = [4, 8]$ and $k = 10$. For Unfair and 5-apx, values in parentheses indicate the number of runs (out of 10) with lower-bound violations. Bold and underlined values respectively denote the lowest clustering costs with and without full fairness guarantees. "–" indicates a runtime exceeding 100,000s.

Dataset	Unfair	5-apx [7]	Ours (str.)	Ours (off.)	FPT [13]
Heart	4.26(6)	4.26(3)	**4.26**	**4.26**	**4.26**
National-poll	4.32(5)	4.32(4)	**4.54**	**4.54**	4.57
Student-mat	18.59(5)	19.44(2)	19.48	**19.39**	20.64
Student-perf	19.45(6)	19.86(3)	19.87	**19.29**	23.19
Bank	11.59(5)	11.68(4)	12.51	**12.48**	–
Census	14.34(7)	15.38(3)	15.42	**15.23**	–
Credit-card	4.70(5)	4.70(5)	5.84	**5.61**	–
Bank-full	12.81(8)	12.61(9)	13.20	**12.91**	–
Query	2.49(6)	2.49(5)	2.81	**2.69**	–

Table 2: Runtime in seconds (mean $\pm$ std) under fair range constraints $[\alpha_i, \beta_i] = [4, 8]$ and $k = 10$. Bold and underlined values respectively indicate the lowest mean runtimes among methods that enforce all fairness constraints and those that do not. "–" indicates a runtime exceeding 100,000s.

Dataset	Unfair	5-apx [7]	Ours (str.)	Ours (off.)	FPT [13]
Heart	0.00±0.00	0.00±0.00	3.58±0.18	**0.05**±0.01	24.29±0.45
National-poll	0.01±0.01	0.06±0.14	2.73±0.08	**0.03**±0.01	21.54±0.90
Student-mat	0.02±0.00	0.03±0.00	2.92±0.11	**0.08**±0.01	243.63±6.27
Student-perf	0.03±0.00	0.03±0.00	3.51±0.22	**0.13**±0.02	610.86±12.35
Bank	0.19±0.02	0.24±0.03	7.48±1.94	**1.07**±0.09	–
Census	3.57±0.11	2.73±0.00	98.27±9.43	**3.53**±0.43	–
Credit-card	0.63±0.02	0.93±0.06	25.86±6.75	**20.35**±0.98	–
Bank-full	1.88±0.02	2.65±0.11	63.16±5.51	**10.15**±1.39	–
Query	1.77±0.12	2.19±0.08	110.17±12.81	**73.04**±0.95	–

in the client-streaming stage, we set $\epsilon = 0.01$ in all experiments. Noting that the Unfair and 5-apx algorithms may violate the lower-bound constraints, we also report the number of runs with violations. In addition, the results for $k = 20$ with $[\alpha_1, \beta_1] = [\alpha_2, \beta_2] = [8, 16]$ are given in Appendix D.

As shown in Tables 1 and 2, for $k = 10$, the Unfair algorithm achieves the lowest clustering cost except on the `Student-perf` and `Bank-full` datasets, and the shortest runtime except on the `Census` dataset. However, it ignores all fairness constraints and violates the lower-bound constraints in 50%–80% of the runs. In contrast, our offline algorithm satisfies all fairness constraints and returns costs close to those of the Unfair baseline on most datasets, while our streaming method achieves costs close to those of the offline algorithm across all datasets. This trend holds across different values of k (as shown in Table 5 in Appendix D), indicating stable performance as the number of selected suppliers increases. The 5-apx algorithm achieves lower costs than our methods on several datasets, but violates the lower-bound constraints in 20%–90% of the runs. In terms of runtime efficiency, both variants of our algorithm are substantially faster than the FPT baseline, as shown in Table 2. In particular, on the `Bank` dataset, our methods achieve competitive clustering costs, whereas the FPT algorithm fails to complete within the $100{,}000$ s time budget. Our streaming algorithm also handles large-scale datasets containing hundreds of thousands of points with competitive clustering quality.

5 Conclusion

In this paper, we study fair range k-supplier clustering in offline and streaming settings. For disjoint supplier groups, we develop a polynomial-time 3-approximation algorithm and extend it to the one-pass client-first streaming model, obtaining a $(3 + \epsilon)$-approximation while satisfying all lower- and upper-bound fairness constraints. To the best of our knowledge, this is the first constant-factor streaming approximation algorithm for fair range k-supplier clustering. For overlapping supplier groups, we additionally present an FPT 3-approximation algorithm parameterized by the number of clusters and groups. Experiments demonstrate competitive clustering quality and scalability compared with existing methods.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 12271098) and the Key Project of the Natural Science Foundation of Fujian Province (No. 2025J02011).

References

[1] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 267–275, 2019.

[2] Haris Angelidakis, Adam Kurpisz, Leon Sering, and Rico Zenklusen. Fair and fast k-center clustering for data summarization. In *International Conference on Machine Learning*, pages 669–702, 2022.

[3] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Suman K Bera, Syamantak Das, Sainyam Galhotra, and Sagar Sudhir Kale. Fair k-center clustering in MapReduce and streaming settings. In *Proceedings of the ACM Web Conference 2022*, pages 1414–1422, 2022.

[5] Ioana O Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R Schmidt, and Melanie Schmidt. On the cost of essentially fair clusterings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, pages 18:1–18:22, 2019.

[6] Brian Brubach, Darshan Chakrabarti, John P Dickerson, Aravind Srinivasan, and Leonidas Tsepenekas. Fairness, semi-supervised learning, and more: A general framework for clustering with stochastic pairwise constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6822–6830, 2021.

[7] Xianrun Chen, Sai Ji, Chenchen Wu, Yicheng Xu, and Yang Yang. An approximation algorithm for diversity-aware fair k-supplier problem. *Theoretical Computer Science*, 983:114305, 2024.

[8] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. Proportionally fair clustering. In *International Conference on Machine Learning*, pages 1032–1041, 2019.

[9] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. *Advances in Neural Information Processing Systems*, 30, 2017.

[10] Ashish Chiplunkar, Sagar Kale, and Sivaramakrishnan Natarajan Ramamoorthy. How to solve fair k-center in massive data models. In *International Conference on Machine Learning*, pages 1877–1886, 2020.

[11] John Dickerson, Seyed Esmaeili, Jamie H Morgenstern, and Claire Jie Zhang. Doubly constrained fair clustering. *Advances in Neural Information Processing Systems*, 36:13267–13293, 2023.

[12] Seyed Esmaeili, Brian Brubach, Aravind Srinivasan, and John Dickerson. Fair clustering under a bounded cost. *Advances in Neural Information Processing Systems*, 34:14345–14357, 2021.

[13] Ameet Gadekar, Aristides Gionis, and Suhas Thejaswi. Fair clustering for data summarization: Improved approximation algorithms and complexity insights. In *Proceedings of the ACM on Web Conference 2025*, pages 4458–4469, 2025.

[14] Ameet Gadekar and Suhas Thejaswi. Capacitated fair-range clustering: hardness and approximation algorithms. In *Forty-third International Conference on Machine Learning*, 2026.

[15] Longkun Guo, Chaoqi Jia, and Chao Chen. Fair k-center clustering on massive social network data streams. In *Proceedings of the ACM Web Conference 2026*, pages 4689–4700, 2026.

[16] Longkun Guo, Zeyu Lin, Chaoqi Jia, and Chao Chen. Improved streaming algorithm for fair k-center clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36946–36954, 2026.

[17] Elfarouk Harb and Ho Shan Lam. KFC: A scalable approximation algorithm for k-center fair clustering. *Advances in Neural Information Processing Systems*, 33:14509–14519, 2020.

[18] Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[19] Dorit S Hochbaum and David B Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.

[20] Sedjro Salomon Hotegni, Sepideh Mahabadi, and Ali Vakilian. Approximation algorithms for fair range clustering. In *International Conference on Machine Learning*, pages 13270–13284, 2023.

[21] Armin Jabbarzadeh, Behnam Fahimnia, and Stefan Seuring. Dynamic supply chain network design for the supply of blood in disasters: A robust model with real world application. *Transportation Research Part E: Logistics and Transportation Review*, 70:225–244, 2014.

[22] Chaoqi Jia, Longkun Guo, Kewen Liao, Zhigang Lu, Chao Chen, and Jason Xue. Approximation algorithm for constrained k-center clustering: A local search approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36982–36990, 2026.

[23] Matthew Jones, Huy Nguyen, and Thy Nguyen. Fair k-centers via maximum matching. In *International Conference on Machine Learning*, pages 4940–4949, 2020.

[24] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[25] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository. `https://archive.ics.uci.edu`.

[26] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. Fair k-center clustering for data summarization. In *International Conference on Machine Learning*, pages 3448–3457, 2019.

[27] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 6th edition, 2018.

[28] Stef Lemmens, Catherine Decouttere, Nico Vandaele, and Mauro Bernuzzi. A review of integrated supply chain network design models: Key issues for vaccine supply chains. *Chemical Engineering Research and Design*, 109:366–384, 2016.

[29] Zeyu Lin, Longkun Guo, and Chaoqi Jia. Streaming fair k-center clustering over massive dataset with performance guarantee. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 105–117, 2024.

[30] Sepideh Mahabadi and Stojan Trajanovski. Core-sets for fair and diverse data summarization. *Advances in Neural Information Processing Systems*, 36:78987–79011, 2023.

[31] Sepideh Mahabadi and Ali Vakilian. Individual fairness for k-clustering. In *International Conference on Machine Learning*, pages 6586–6596, 2020.

[32] Richard Matthew McCutchen and Samir Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 165–178, 2008.

[33] Donald M McAllister. Equity and efficiency in public facility location. *Geographical Analysis*, 8(1):47–63, 1976.

[34] Evi Micha and Nisarg Shah. Proportionally fair clustering revisited. In *47th International Colloquium on Automata, Languages, and Programming*, pages 85:1–85:16, 2020.

[35] Maryam Negahbani and Deeparnab Chakrabarty. Better algorithms for individually fair k-clustering. *Advances in Neural Information Processing Systems*, 34:13340–13351, 2021.

[36] Huy Lê Nguyen, Thy Nguyen, and Matthew Jones. Fair range k-center. *arXiv preprint arXiv:2207.11337*, 2022.

[37] James B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[38] Jiandong Peng, Yue Liu, Zehan Ruan, and Hong Yang. Study on the optimal allocation of public service facilities from the perspective of living circle—a case study of Xiangyang high-tech zone, China. *Journal of Urban Management*, 12(4):344–359, 2023.

[39] Suhas Thejaswi. On the hardness of approximation of the fair k-center problem. *Information and Computation*, 312:105503, 2026.

[40] Suhas Thejaswi, Ameet Gadekar, Bruno Ordozgoiti, and Aristides Gionis. Diversity-aware clustering: Computational complexity and approximation algorithms. *arXiv preprint arXiv:2401.05502*, 2024.

[41] Suhas Thejaswi, Ameet Gadekar, Bruno Ordozgoiti, and Michal Osadnik. Clustering with fair-center representation: Parameterized approximation algorithms and heuristics. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1749–1759, 2022.

[42] Yanhao Wang, Francesco Fabbri, and Michael Mathioudakis. Streaming algorithms for diversity maximization with fairness constraints. In *2022 IEEE 38th International Conference on Data Engineering*, pages 41–53, 2022.

[43] Xiaoliang Wu, Qilong Feng, Ziyun Huang, Jinhui Xu, and Jianxin Wang. New algorithms for distributed fair k-center clustering: almost accurate as sequential algorithms. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 1938–1946, 2024.

[44] Zhen Zhang, Xiaohong Chen, Limei Liu, Jie Chen, Junyu Huang, and Qilong Feng. Parameterized approximation schemes for fair-range clustering. *Advances in Neural Information Processing Systems*, 37:60192–60211, 2024.

Appendix

A Additional Details and Proofs for the Offline Algorithm

A.1 MCMF Network Capacities and Costs

Table 3: Capacity ranges and costs of the edges in the constructed MCMF network.

Edge	Capacity range	Cost
(s, F_l)	$[\alpha_l, \beta_l]$	0
(F_l, F_l')	$[0, \beta_l]$	0
(F_l', t)	$[0, \beta_l]$	1
(F_l, f)	$[0, 1]$	0
(f, c)	$[0, 1]$	0
(c, t)	$[0, 1]$	0

A.2 Linear Programming Formulation of MCMF

Based on the flow network $G = (V, E)$ constructed in Section 3.1, we formulate the MCMF problem as LP 1:

$$\min \quad \sum_{i\in[m]} x_{F_i't} \tag{1a}$$

$$\text{s.t.} \quad \sum_{u:(u,v)\in E} x_{uv} = \sum_{w:(v,w)\in E} x_{vw}, \quad \forall v \in V \setminus \{s, t\} \tag{1b}$$

$$\sum_{i\in[m]} x_{sF_i} = k \tag{1c}$$

$$\alpha_i \le x_{sF_i} \le \beta_i, \quad \forall i \in [m] \tag{1d}$$

$$x_{F_iF_i'} \le \beta_i, \quad x_{F_i't} \le \beta_i, \quad \forall i \in [m] \tag{1e}$$

$$x_{F_if} \le 1, \quad \forall i \in [m],\ f \in F_i \text{ with } (F_i, f) \in E \tag{1f}$$

$$x_{fc} \le 1, \quad \forall (f, c) \in E,\ f \in F,\ c \in C' \tag{1g}$$

$$x_{ct} \le 1, \quad \forall c \in C' \tag{1h}$$

$$x_{uv} \ge 0, \quad \forall (u, v) \in E. \tag{1i}$$

A.3 Pseudocode for MCMF-Based Supplier Selection

The pseudocode for the MCMF-based supplier selection procedure is presented in Algorithm 2.

A.4 Proof of Lemma 1

Proof. Claims (i) and (ii) follow directly from Definition 1 with $\lambda = 2R^*$.

It remains to show Claim (iii). Let S^* be an optimal solution of radius R^*, with $|S^*| = k$. Assign each representative $c \in C'$ to a supplier $f(c) \in S^*$ that serves c in the optimal solution. We claim that this assignment is injective. Suppose, for contradiction, that two distinct representatives $c, c' \in C'$ are assigned to the same supplier $f \in S^*$. Since the optimal solution has radius R^*, $d(c, f) \le R^*$ and $d(c', f) \le R^*$. By the triangle inequality, $d(c, c') \le d(c, f) + d(f, c') \le 2R^*$, contradicting the fact that C' is $2R^*$-independent, i.e., every two distinct representatives in C' are at distance greater than $2R^*$. Hence, distinct representatives are assigned to distinct suppliers in S^*. Therefore, $|C'| \le |S^*| = k$. □

A.5 Proof of Lemma 2

Proof. For any feasible flow x of value k, flow conservation gives $\text{cost}(x) = \sum_{l\in[m]} x_{F_l't} = k - \sum_{c\in C'} x_{ct} \ge k - |C'|$, where the inequality follows from $x_{ct} \le 1$ for every $c \in C'$.

Algorithm 2: SELECT-SUPPLIERS

Input: A representative client set C', a supplier set F, a partition $\mathcal{G} = \{F_1, \ldots, F_m\}$ of F, candidate supplier sets $F_c^l \subseteq F_l$ for each $c \in C'$ and $l \in [m]$, fairness-parameter vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and an integer k.

Output: A supplier set S.

`// Phase 1: Construction of the MCMF network` $G = (V, E)$. `Treat each group` $F_l \in \mathcal{G}$ `as a node; let` F_l' `denote its copy.`

Initialize $V \leftarrow \{s, t\}$, $E \leftarrow \emptyset$;
for $l \in [m]$ **do**
  $V \leftarrow V \cup \{F_l, F_l'\}$;
  $E \leftarrow E \cup \{(s, F_l), (F_l, F_l'), (F_l', t)\}$;
  Set the capacity–cost pairs of (s, F_l), (F_l, F_l'), and (F_l', t) to $([\alpha_l, \beta_l], 0)$, $([0, \beta_l], 0)$, and $([0, \beta_l], 1)$, respectively;
for $c \in C'$ **do**
  $V \leftarrow V \cup \{c\}$;
  $E \leftarrow E \cup \{(c, t)\}$;
  Set the capacity–cost pair of (c, t) to $([0, 1], 0)$;
  for $l \in [m]$ **do**
    for $f \in F_c^l$ **do**
      $V \leftarrow V \cup \{f\}$;
      if $(F_l, f) \notin E$ **then**
        $E \leftarrow E \cup \{(F_l, f)\}$;
        Set the capacity–cost pair of (F_l, f) to $([0, 1], 0)$;
      $E \leftarrow E \cup \{(f, c)\}$;
      Set the capacity–cost pair of (f, c) to $([0, 1], 0)$;
Set supply at s to k;
`// Phase 2: Flow computation and supplier extraction.`
Compute an integral optimal solution $\boldsymbol{x}$ to LP 1 on $G = (V, E)$;
$S \leftarrow \{f_c^l : c \in C', l \in [m], x_{f_c^l c} = 1\}$;
for $l \in [m]$ **do**
  select arbitrarily $x_{F_l' t}$ suppliers from $F_l \setminus S$ and add them to S;
return S;

($\Rightarrow$) Let S satisfy conditions (i)–(iii). For each $c \in C'$, choose a supplier $\phi(c) \in S$ with $d(c, \phi(c)) \leq R^*$. The mapping ϕ is injective. If two distinct representatives $c, c' \in C'$ shared the same supplier f, then $d(c, c') \leq d(c, f) + d(f, c') \leq 2R^*$, contradicting the $2R^*$-independence of C'.

For each group F_l, let $s_l = |S \cap F_l|$ and $r_l = |\phi(C') \cap F_l|$. Injectivity implies $r_l \leq s_l$. Send s_l units of flow along (s, F_l). For each $c \in C'$ with $\phi(c) \in F_l$, send one unit along $F_l \to \phi(c) \to c \to t$. Route the remaining $s_l - r_l$ units along $F_l \to F_l' \to t$.

All required supplier–client edges exist because $d(c, \phi(c)) \leq R^*$. The fairness constraints on S and the injectivity of ϕ ensure that all capacity constraints are respected. The resulting flow is integral, has value $\sum_l s_l = k$, and has cost $\sum_{l \in [m]} (s_l - r_l) = k - |C'|$. It therefore attains the minimum possible cost.

($\Leftarrow$) Let x be an integral feasible flow of value k and cost $k - |C'|$. The cost identity above implies $\sum_{c \in C'} x_{ct} = |C'|$. Since $0 \leq x_{ct} \leq 1$, we have $x_{ct} = 1$ for every $c \in C'$.

For each group F_l, define $S_{0,l} = \{f \in F_l : (F_l, f) \in E,\ x_{F_l f} = 1\}$, and let $S_0 = \bigcup_{l \in [m]} S_{0,l}$. By integrality and the unit capacities of the supplier selection edges, each representative client receives one unit of flow from a unique supplier, and each supplier serves at most one representative. Consequently, $|S_0| = |C'|$, and every client in C' is served by S_0 within radius R^*.

For each $l \in [m]$, let $q_l = x_{F_l F_l'}$. Flow conservation at F_l gives $|S_{0,l}| + q_l = x_{sF_l}$. Moreover, $q_l \le \beta_l - |S_{0,l}| \le |F_l| - |S_{0,l}|$. Hence, there are enough unselected suppliers in F_l to choose a set $T_l \subseteq F_l \setminus S_{0,l}$ with $|T_l| = q_l$.

Let $S = S_0 \cup \bigcup_{l\in[m]} T_l$. Since the supplier groups are disjoint, for every $l \in [m]$, $|S \cap F_l| = |S_{0,l}| + q_l = x_{sF_l} \in [\alpha_l, \beta_l]$. Furthermore, $|S| = \sum_{l\in[m]} x_{sF_l} = k$. Finally, S contains S_0, so it serves every representative client within radius R^*. Thus, S satisfies conditions (i)–(iii). □

A.6 Proof of Theorem 1

We begin by proving the following lemma, which is instrumental for establishing Theorem 1.

Let A be the node–arc incidence matrix of the constructed directed network, with a -1 at the tail and a $+1$ at the head of each arc. Define the supply–demand vector b by

$$b_v = \begin{cases} -k, & v = s, \\ k, & v = t, \\ 0, & v \in V \setminus \{s, t\}. \end{cases}$$

Thus, the flow-conservation and total-flow constraints can be written as $Ax = b$. The equation at the sink is implied by the source equation and flow conservation at the internal nodes.

Lemma 9. *The node–arc incidence matrix A of the constructed directed network is totally unimodular.*

Proof. By definition, it suffices to show that every square submatrix B of A satisfies $\det(B) \in \{0, \pm 1\}$. We prove this statement by induction on the order of B.

For a 1×1 submatrix, the claim follows immediately because every entry of A belongs to $\{0, \pm 1\}$. Now let $p \ge 2$, and assume that the claim holds for every $(p-1) \times (p-1)$ submatrix of A. Consider an arbitrary $p \times p$ submatrix B.

Since A is a node–arc incidence matrix, each column contains at most one $+1$ and at most one -1, with all remaining entries equal to zero. The same property holds for every column of B. We distinguish the following three cases.

Case 1: B contains a zero column. In this case, $\det(B) = 0$.

Case 2: B has no zero column, and some column contains exactly one nonzero entry. Suppose that this entry is $B_{ij} \in \{+1, -1\}$. Let B' be the $(p-1) \times (p-1)$ matrix obtained by deleting row i and column j from B. Expanding the determinant along column j gives $\det(B) = (-1)^{i+j} B_{ij} \det(B')$. Since B' is a square submatrix of A, the induction hypothesis yields $\det(B') \in \{0, \pm 1\}$. Together with $B_{ij} \in \{+1, -1\}$, this implies $\det(B) \in \{0, \pm 1\}$.

Case 3: Every column of B contains exactly two nonzero entries. By the incidence-matrix property, these two entries must be one $+1$ and one -1. Hence, the entries in each column sum to zero, so $\mathbf{1}^\mathsf{T} B = \mathbf{0}^\mathsf{T}$, where $\mathbf{1}$ is the all-ones vector in $\mathbb{R}^p$. Thus, the rows of B are linearly dependent, and $\det(B) = 0$.

These three cases exhaust all possibilities. Therefore, every $p \times p$ submatrix of A has determinant in $\{0, \pm 1\}$. By induction, the claim holds for every square submatrix of A, proving that A is totally unimodular. □

Proof of Theorem 1. Let ℓ and u be the lower- and upper-capacity vectors specified in LP 1. In particular, $\ell_{sF_i} = \alpha_i$ for every $i \in [m]$, and all other lower capacities are zero. Since k and the fairness bounds are integers, ℓ, u, and b are integral. Thus, the feasible region of LP 1 is $P = \{x \in \mathbb{R}^E : Ax = b,\ \ell \le x \le u\}$.

Writing the bounds as $x \le u$ and $-x \le -\ell$, the matrix collecting the coefficients of the equality and bound constraints is

$$M = \begin{bmatrix} A \\ I \\ -I \end{bmatrix}.$$

By Lemma 9, A is totally unimodular. We show that M is also totally unimodular by induction on the order of its square submatrices.

Every 1×1 submatrix of M has determinant in $\{0, \pm 1\}$. Now let $p \geq 2$, assume the claim holds for all square submatrices of order $p-1$, and consider an arbitrary $p \times p$ submatrix B of M. We distinguish three cases.

Case 1: All rows of B are inherited from A. Then B is a square submatrix of A. Since A is totally unimodular, $\det(B) \in \{0, \pm 1\}$.

Case 2: Some row of B inherited from I or $-I$ is zero. In this case, $\det(B) = 0$.

Case 3: B contains a row inherited from I or $-I$, and none of these rows is zero. Choose one such row. It contains exactly one nonzero entry, equal to $+1$ or -1. Expanding the determinant along this row gives $\det(B) = \pm \det(B')$, where B' is a $(p-1) \times (p-1)$ submatrix of M. By the induction hypothesis, $\det(B') \in \{0, \pm 1\}$, and hence $\det(B) \in \{0, \pm 1\}$.

These cases exhaust all possibilities. Therefore, M is totally unimodular.

Let x^* be any vertex of P. The equality constraints and active bound constraints at x^* have rank $|E|$. Otherwise, there would exist a nonzero direction along which sufficiently small perturbations of x^* in both directions remain feasible, contradicting the fact that x^* is a vertex.

We can therefore select $|E|$ linearly independent active constraints and write them as $Bx^* = r$, where B is a nonsingular $|E| \times |E|$ submatrix of M. The entries of r are drawn from b, u, and $-\ell$, so r is integral. Since M is totally unimodular and B is nonsingular, $\det(B) \in \{+1, -1\}$.

For each coordinate j, Cramer's rule gives $x^*_j = \frac{\det(B_j)}{\det(B)}$, where B_j is obtained from B by replacing column j with r. The matrix B_j is integral, so $\det(B_j)$ is an integer. As $\det(B) = \pm 1$, every coordinate of x^* is integral. Thus, every vertex of P is integral.

Finally, all lower and upper capacities are finite, so P is bounded. Whenever P is nonempty, the linear objective attains its minimum at a vertex of P. Since every such vertex is integral, LP 1 admits an integral optimal solution whenever it is feasible. □

A.7 Proof of Theorem 2

Proof. Suppose that the guessed radius equals the optimal radius R^*. We show that the algorithm outputs a feasible solution with maximum distance at most $3R^*$.

We first establish feasibility. Let S^* be an optimal supplier set. Under the assumptions on the disjoint groups and their upper bounds, S^* can be augmented to size k without increasing its service radius. We may therefore assume that $|S^*| = k$. The set S^* satisfies all fair range constraints and serves every representative client in C' within radius R^*. By Lemma 2, the constructed network admits an integral feasible flow of value k and cost $k - |C'|$, which is minimum. Consequently, an integral minimum-cost flow also has cost $k - |C'|$. The supplier-selection procedure therefore recovers a set S with $|S| = k$ satisfying all fair range constraints and serving every representative client within radius R^*.

We next bound the service cost. By Lemma 1, for every client $c \in C$, there exists a representative $c' \in C'$ such that $d(c, c') \leq 2R^*$. Moreover, by construction of the flow solution, each $c' \in C'$ is assigned to a selected supplier $f \in S$ with $d(c', f) \leq R^*$. By the triangle inequality, $d(c, f) \leq d(c, c') + d(c', f) \leq 2R^* + R^* = 3R^*$.

Therefore, the algorithm outputs a feasible solution with maximum distance at most $3R^*$.

We finally analyze the running time. Consider one feasibility check for a fixed candidate radius R. Algorithm 1 first constructs the representative client set C' in $O(kn)$ time. Since $|C'| \leq k$, identifying the sets $F^l_{c'}$ of suppliers within distance R for all $c' \in C'$ and $l \in [m]$ takes $O(k|F|) \leq O(kn)$ time. The corresponding directed graph G contains $O(n)$ vertices and $O(nk)$ edges and can also be constructed in $O(nk)$ time.

The MCMF instance is solved through its LP formulation. Each edge of the flow network corresponds to one LP variable, and hence the LP has $N_{\mathrm{LP}} = O(nk)$ variables. The flow-conservation, capacity, and fairness constraints contain $O(nk)$ nonzero coefficients in total. All coefficients in the constraint

matrix belong to $\{-1, 0, 1\}$, while the capacities, supplies, and fairness bounds are integers of magnitude at most k. Under the standard sparse binary encoding, each nonzero coefficient is stored together with its row and column indices. The indices require $O(\log(nk))$ bits, and each capacity, supply, or fairness bound requires $O(\log k)$ bits. Therefore, the total binary encoding length of the LP is $\mathcal{L} = O\big(nk(\log(nk) + \log k)\big)$. Since $k \le n$, we have $\log(nk) = O(\log n)$ and $\log k = O(\log n)$, and hence $\mathcal{L} = O(nk \log n)$.

Karmarkar's polynomial-time LP algorithm [24] requires $O\big(N_{\mathrm{LP}}^{3.5}\mathcal{L}\big)$ arithmetic operations. Substituting $N_{\mathrm{LP}} = O(nk)$ and $\mathcal{L} = O(nk \log n)$, the LP for one candidate radius can be solved in $O\big((nk)^{3.5} \cdot nk \log n\big) = O\big((nk)^{4.5} \log n\big)$ arithmetic operations. This dominates the $O(kn)$ time required to construct C', identify the feasible suppliers, build the flow network, and recover the selected supplier set. Therefore, one feasibility check requires $O\big((nk)^{4.5} \log n\big)$ arithmetic operations.

The candidate radii are drawn from the finite set of client–supplier distances, whose size is at most $|C||F| \le n^2$. Therefore, a binary search over these candidate radii requires $O(\log(|C||F|)) = O(\log(n^2)) = O(2 \log n) = O(\log n)$ feasibility checks. Consequently, the overall running time is $O\big((nk)^{4.5} \log n\big) \cdot O(\log(n^2)) = O\big((nk)^{4.5} \log^2 n\big)$ arithmetic operations. □

A.8 Radius Guessing for the Offline Algorithm

By combining Algorithms 1 and 2 with the standard radius-guessing framework and the preceding analysis, we obtain the same 3-approximation guarantee without assuming prior knowledge of R^*. For completeness, we provide the argument below.

We first observe that the optimal radius belongs to a finite set of client–supplier distances.

Lemma 10. *Let $\Psi = \{d(c, f) \mid c \in C,\ f \in F\}$ be the set of distinct client–supplier distances. Then $|\Psi| \le |C||F|$ and $R^* \in \Psi$. In particular, the optimal radius is attained by some client–supplier pair.*

Proof. Let $S^* \subseteq F$ be an optimal supplier set. By definition, $R^* = \max_{c \in C} d(c, S^*) = \max_{c \in C} \min_{f \in S^*} d(c, f)$. Since C and S^* are finite, there exists a client $c^* \in C$ attaining the outer maximum and a supplier $f^* \in S^*$ attaining the corresponding inner minimum. Therefore, $R^* = d(c^*, S^*) = d(c^*, f^*)$. Because $c^* \in C$ and $f^* \in S^* \subseteq F$, we have $d(c^*, f^*) \in \Psi$. Hence, $R^* \in \Psi$. □

Let $R_1 < R_2 < \cdots < R_q$ be the distinct candidate radii in Ψ, where $q \le |C||F|$. For a candidate radius R, Algorithm 1 constructs a maximal $2R$-independent representative client set $C'_R \subseteq C$. If $|C'_R| > k$, the test returns 0. Otherwise, Algorithm 2 constructs the corresponding flow network using radius R. Let z_R denote the optimal objective value of LP 1 on this network, with $z_R = +\infty$ if the LP is infeasible. We define

$$\textsc{RadiusTest}(R) = \begin{cases} 1, & \text{if } |C'_R| \le k \text{ and } z_R = k - |C'_R|, \\ 0, & \text{otherwise.} \end{cases}$$

By flow conservation, every feasible flow x of value k satisfies $\mathrm{cost}(x) = \sum_{l \in [m]} x_{F'_l t} = k - \sum_{c \in C'_R} x_{ct} \ge k - |C'_R|$. Equality holds if and only if $x_{ct} = 1$ for every $c \in C'_R$. Since the flow LP admits an integral optimum whenever it is feasible, a successful radius test yields an integral flow saturating all representative-client edges.

Whenever $\textsc{RadiusTest}(R) = 1$, Lemma 2 guarantees a supplier set satisfying the fair range constraints such that every representative client $c' \in C'_R$ is assigned to a selected supplier within distance R. Moreover, by the maximality of C'_R, every client $c \in C$ is within distance $2R$ of some representative $c' \in C'_R$. Consequently, the triangle inequality implies that every client is served within distance at most $2R + R = 3R$.

The following lemma establishes the one-sided feasibility property needed for the radius search.

Lemma 11. *For every candidate radius $R_j \ge R^*$, $\textsc{RadiusTest}(R_j) = 1$.*

Proof. Fix a candidate radius $R_j \ge R^*$. Since $\beta_l \le |F_l|$ for every $l \in [m]$ and $\sum_{l \in [m]} \beta_l \ge k$, any feasible supplier set of size less than k can be augmented to size k without increasing its service radius. Hence, we fix an optimal supplier set S^* with $|S^*| = k$.

Let C'_j be the maximal $2R_j$-independent client set constructed by Algorithm 1. For each representative client $c' \in C'_j$, choose a supplier $f(c') \in S^*$ satisfying $d(c', f(c')) \leq R^* \leq R_j$. The mapping $c' \mapsto f(c')$ is injective. Indeed, if two distinct representatives c'_1, c'_2 were assigned to the same supplier f, then $d(c'_1, c'_2) \leq d(c'_1, f) + d(f, c'_2) \leq 2R^* \leq 2R_j$, contradicting their $2R_j$-independence. Therefore, $|C'_j| \leq |S^*| = k$.

For each group F_l, let $k_l = |S^* \cap F_l|$ and $r_l = |\{c' \in C'_j : f(c') \in F_l\}|$. Injectivity implies $r_l \leq k_l$. Send k_l units of flow along (s, F_l). For each $c' \in C'_j$ with $f(c') \in F_l$, send one unit along $F_l \to f(c') \to c' \to t$. These edges exist because $d(c', f(c')) \leq R_j$. Route the remaining $k_l - r_l$ units along $F_l \to F'_l \to t$.

Since S^* satisfies the fair range constraints, $\alpha_l \leq k_l \leq \beta_l$ for every $l \in [m]$. Together with injectivity, these inequalities ensure that all capacity and flow-conservation constraints are satisfied. The resulting integral flow has value $\sum_l k_l = k$ and cost $\sum_{l \in [m]} (k_l - r_l) = k - |C'_j|$. Every feasible flow of value k has cost at least $k - |C'_j|$, since each edge (c', t) has capacity one. Consequently, $z_{R_j} = k - |C'_j|$, and hence $\textsc{RadiusTest}(R_j) = 1$. □

Let j^* be the index such that $R_{j^*} = R^*$, which exists by Lemma 10. Define $T(j) = \textsc{RadiusTest}(R_j)$. Lemma 11 implies that $T(j) = 1$ for every $j \geq j^*$. Notice that this property does not require $T(j)$ to be monotone for $j < j^*$. Nevertheless, it is sufficient for the following binary-search procedure.

We apply the following binary-search procedure to the ordered candidate radii $R_1 < R_2 < \cdots < R_q$. Initialize $low = 1$ and $high = q$. While $low \leq high$, let $j = \lfloor (low + high)/2 \rfloor$ and evaluate $T(j) = \textsc{RadiusTest}(R_j)$. If $T(j) = 1$, record $\widehat{j} = j$ together with the corresponding supplier set and continue searching to the left by setting $high = j - 1$. If $T(j) = 0$, continue searching to the right by setting $low = j + 1$. When the search terminates, return the supplier set recorded for $R_{\widehat{j}}$.

Lemma 12. *Suppose that there exists an index j^* such that $T(j) = 1$ for every $j \geq j^*$. Consider the binary-search procedure that records the current index and continues searching to the left when $T(j) = 1$, and searches to the right when $T(j) = 0$. The procedure returns an index $\widehat{j} \leq j^*$ such that $T(\widehat{j}) = 1$.*

Proof. Consider an iteration in which the midpoint of the current search interval is j.

If $T(j) = 0$, then necessarily $j < j^*$, since every index at least j^* is feasible. Therefore, discarding the indices up to and including j does not discard j^*.

If $T(j) = 1$, the algorithm records j as a feasible index and continues searching to its left. If $j \leq j^*$, the recorded index already satisfies the desired conclusion. If $j > j^*$, then j^* remains in the left search interval.

Thus, throughout the search, either a feasible index no larger than j^* has already been recorded, or the current search interval still contains j^*. Since $T(j^*) = 1$, the procedure eventually returns a feasible index $\widehat{j} \leq j^*$. □

Applying Lemma 12, we obtain a candidate radius $R = R_{\widehat{j}} \leq R_{j^*} = R^*$ such that $\textsc{RadiusTest}(R) = 1$. The supplier set recorded during this successful radius test satisfies all fair range constraints and serves every client within distance at most $3R \leq 3R^*$. Therefore, Algorithms 1 and 2 retain their 3-approximation guarantee without prior knowledge of R^*.

For each candidate radius R, Algorithms 1 and 2 implement $\textsc{RadiusTest}(R)$ under the assumption that R is given. To remove the assumption that R^* is known, we invoke this radius-testing procedure within an outer binary search over the $q \leq |C||F| \leq n^2$ candidate radii. Each binary-search iteration reduces the current index interval by at least a factor of two. Hence, the outer search performs at most $\lceil \log_2 q \rceil + 1 = O(\log(|C||F|)) = O(\log(n^2)) = O(\log n)$ calls to $\textsc{RadiusTest}$.

B Missing Pseudocode and Proofs for the Streaming Algorithm

The pseudocode for our streaming algorithm is presented in Algorithm 3.

Algorithm 3: Streaming algorithm for FR-k-SUP at a fixed active scale Δ

Input: Representative client set C' with $|C'| \le k$, supplier stream F partitioned into $\mathcal{G} = \{F_1, \dots, F_m\}$, parameter k, and fair range parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$.

Output: A feasible supplier set S_Δ and a feasible radius ρ_Δ, or FAILURE if no candidate radius is feasible.

`// Phase 1: Supplier streaming.`

Initialize $S_l \leftarrow \emptyset$ for every $l \in [m]$;

for *each incoming supplier* $f \in F_l$ **do**

  $A_l \leftarrow S_l \cup \{f\}$;

  for *each* $c \in C'$ **do**

    Choose $f_c^l \in \arg\min_{g \in A_l} d(c, g)$;

  $P_l \leftarrow \{f_c^l \mid c \in C'\}$;

  if $|A_l| \le k$ **then**

    $S_l \leftarrow A_l$;

  else

    Choose an arbitrary supplier $f' \in A_l \setminus P_l$;

    $S_l \leftarrow A_l \setminus \{f'\}$;

`// Phase 2: Post-streaming computation.`

$\widehat{F} \leftarrow \bigcup_{l \in [m]} S_l$;

$\mathcal{D} \leftarrow \{d(c, f) : c \in C',\ f \in \widehat{F}\}$;

Sort the distinct radii in $\mathcal{D}$;

$\rho_\Delta \leftarrow +\infty$;

`// Binary search for the minimum feasible radius.`

for *each candidate radius* $\rho \in \mathcal{D}$ *examined by binary search* **do**

  Construct G_ρ using the network structure, capacities, and costs in Algorithm 2, with the fixed client set C', retained supplier groups $\{S_l\}_{l \in [m]}$, and fairness bounds $\boldsymbol{\alpha}, \boldsymbol{\beta}$; include (f, c) if and only if $d(c, f) \le \rho$;

  $z_\rho \leftarrow$ the optimal objective value of LP 1 on G_ρ, with $z_\rho = +\infty$ if the LP is infeasible;

  if $z_\rho = k - |C'|$ **then**

    Mark ρ as feasible;

    if $\rho < \rho_\Delta$ **then**

      Obtain an integral optimal flow x_ρ using Theorem 1;

      $\rho_\Delta \leftarrow \rho$, $\widehat{x} \leftarrow x_\rho$;

  else

    Mark ρ as infeasible;

if $\rho_\Delta = +\infty$ **then**

  return FAILURE;

Recover $S_\Delta \subseteq \widehat{F}$ from $\widehat{x}$ using the supplier-selection rule in Algorithm 2, with $\{S_l\}_{l \in [m]}$ as the supplier groups;

return (S_Δ, ρ_Δ);

B.1 Proof of Lemma 3

Proof. Assume $|B| \ge 2$. We prove the three properties by induction on $t \ge t_0$.

At $t = t_0$, initialization sets U_{t_0} to the maximum pairwise distance in B. Every client processed so far belongs to B, so $d(c_i, c_1) \le U_{t_0}$, proving (i). Since $\Delta_{J_{t_0}} \ge U_{t_0}$, the largest positive scale retains only c_1 and remains active because $k \ge 1$, proving (ii). The greedy construction ensures that every active positive scale covers B within its scale. If the additional zero scale is present, its representative set is $B = C$, so it covers all clients within distance zero. Thus, (iii) also holds initially.

For the inductive step, suppose the properties hold after processing $t-1$ clients. By (ii), the largest active scale contains c_1, so the update of U_t includes $d(c_t, c_1)$. Consequently, $d(c_t, c_1) \le U_t$. Since $U_t \ge U_{t-1}$, all earlier clients also satisfy $d(c_i, c_1) \le U_t$, proving (i).

For (ii), if $J_t = J_{t-1}$, the largest scale already has representative set $\{c_1\}$. Since $d(c_t, c_1) \le U_t \le \Delta_{J_t}$, the new client is not added. If $J_t > J_{t-1}$, the new largest scale is initialized with $\{c_1\}$, and the same inequality prevents c_t from being added. Hence, the largest scale remains active and contains exactly c_1.

Finally, each newly introduced scale satisfies $\Delta_j > \Delta_{J_{t-1}} \ge U_{t-1}$. By (i) at time $t-1$, initializing it with $\{c_1\}$ covers all previously processed clients. Existing active scales preserve their previous coverage. At each scale, the greedy update either adds c_t or ensures that its distance to the representative set is at most that scale. Discarding a scale does not affect the others. Therefore, every scale remaining active covers all t processed clients, proving (iii) and completing the induction. □

B.2 Proof of Lemma 4

Proof. Let J denote the final maximum positive scale index. Since $\Delta_J \ge U \ge 2R^*$, the index $j = \min\{h \in \{0, \dots, J\} : \Delta_h \ge 2R^*\}$ is well defined. If $j = 0$, then $2R^* \le \Delta_0 = L \le 2R^*$, so $\Delta_0 = 2R^*$. Otherwise, the minimality of j gives $\Delta_{j-1} < 2R^*$, and hence $\Delta_j = (1+\epsilon/2)\Delta_{j-1} < (2+\epsilon)R^*$. Since $L > 0$ implies $R^* > 0$, both cases yield $2R^* \le \Delta_j < (2+\epsilon)R^*$.

Fix an optimal supplier set S^* with $|S^*| \le k$, and assign each client to a supplier in S^* within distance R^*. The scale Δ_j is introduced either during initialization or when the scale range expands. At any time after its introduction, including immediately after an insertion, its representatives are pairwise separated by distances greater than Δ_j.

If two distinct representatives a, b were assigned to the same optimal supplier f, then $d(a,b) \le d(a,f) + d(f,b) \le 2R^* \le \Delta_j$, contradicting their separation. Thus, their assignment to optimal suppliers is injective, so the representative set always has size at most $|S^*| \le k$. The scale is therefore never discarded.

Consequently, Δ_j remains active, and distinct representatives in its final set C'_j belong to different optimal clusters. □

B.3 Proof of Lemma 5

Proof. Fix a representative client $c \in C'$ and a supplier group $l \in [m]$. During the supplier stream, whenever a new supplier $f \in F_l$ arrives, the algorithm forms $A_l = S_l \cup \{f\}$ and chooses $f_c^l \in \arg\min_{g \in A_l} d(c,g)$. The supplier f_c^l is included in the protected set P_l.

If $|A_l| \le k$, all suppliers in A_l are retained. Otherwise, the algorithm removes a supplier only from $A_l \setminus P_l$. Hence, the closest supplier to c among the candidates considered up to that point is never discarded.

Applying this argument throughout the supplier stream, after all suppliers in F_l have been processed, the final set S_l contains at least one supplier f_c^l satisfying $d(c, f_c^l) = \min_{g \in F_l} d(c,g)$. Therefore, $\min_{g \in S_l} d(c,g) = \min_{g \in F_l} d(c,g)$. Since $c \in C'$ and $l \in [m]$ were arbitrary, the claim holds for every representative client and every supplier group. □

B.4 Proof of Lemma 6

Proof. Fix a feasible candidate radius ρ and choose an integral optimal flow x. By flow conservation, $k - |C'| = z_\rho = k - \sum_{c \in C'} x_{ct}$. Since $0 \le x_{ct} \le 1$ for every $c \in C'$, we obtain $x_{ct} = 1$ for every representative client. By integrality and the unit capacities of the supplier selection edges, these flows identify a set $S_0 \subseteq \widehat{F}$ of $|C'|$ distinct suppliers serving all representatives within radius ρ.

For each group $l \in [m]$, write $n_l = x_{sF_l}$ and $q_l = x_{F_l F'_l}$. Flow conservation gives $n_l = |S_0 \cap F_l| + q_l$. By construction, the retained supplier set satisfies $|S_l| = \min\{k, |F_l|\}$. Since $n_l \le k$, $n_l \le \beta_l$, and $\beta_l \le |F_l|$, we have $n_l \le \min\{k, \beta_l\} \le |S_l|$. Consequently, $q_l \le |S_l| - |S_0 \cap F_l|$. We can therefore add q_l distinct suppliers from $S_l \setminus S_0$ for every group l.

Let S be the resulting supplier set. The supplier groups are disjoint, so $|S \cap F_l| = n_l \in [\alpha_l, \beta_l]$ for every $l \in [m]$, and $|S| = \sum_{l\in[m]} n_l = k$. Since $S_0 \subseteq S$, every representative client is served within radius ρ. □

B.5 Proof of Lemma 7

Proof. Suppose that $\rho \in \mathcal{D}$ is feasible, and let $\rho' \in \mathcal{D}$ satisfy $\rho' \geq \rho$. Every edge present at radius ρ remains present at radius ρ', with the same capacity and cost. Thus, an optimal integral flow at radius ρ, extended by zero flow on newly introduced edges, is feasible at radius ρ'. Therefore, $z_{\rho'} \leq z_\rho = k - |C'|$. On the other hand, every feasible flow of value k has cost at least $k - |C'|$. Hence, $z_{\rho'} = k - |C'|$, so ρ' is also feasible. □

B.6 Proof of Lemma 8

Proof. Since the supplier groups are disjoint, $\beta_l \leq |F_l|$ for every $l \in [m]$, and $\sum_{l\in[m]} \beta_l \geq k$, any feasible supplier set of size less than k can be augmented to size k without increasing its service cost. We may therefore assume that $|S^*| = k$. For each group $l \in [m]$, define $k_l^* = |S^* \cap F_l|$. Thus, the group counts satisfy $\alpha_l \leq k_l^* \leq \beta_l$ for all $l \in [m]$, and their sum is $\sum_{l\in[m]} k_l^* = k$.

We first consider the case where $|C| = 1$, or where $|C| \geq 2$ and $2R^* < L$. In the latter case, $|C| \leq k$. Otherwise, two of the first $k + 1$ distinct clients would share an optimal supplier, implying $L \leq 2R^*$. Thus, the zero scale is present in both situations, with $C'_{\mathrm{z}} = C$.

For each $c \in C$, choose an optimal serving supplier $f^*(c) \in S^*$ and a retained nearest supplier f_c from the same group. By Lemma 5, $d(c, f_c) \leq R^*$. Both assignments are injective. This is immediate for a singleton client set; otherwise, distinct clients have distance at least $L > 2R^*$ and cannot share a supplier within radius R^*.

Set $R = \max_{c\in C} d(c, f_c)$. Then $R \in \mathcal{D}_{\mathrm{z}}$ and $R \leq R^*$. For each group l, let q_l count the clients assigned to that group. Injectivity gives $q_l \leq k_l^*$. Route one unit through each assigned supplier–client pair and route the remaining $k_l^* - q_l$ units through the auxiliary path of group l. This yields an integral feasible flow of value k and cost $k - |C|$. Hence, R is feasible and $\Delta_{\mathrm{z}} + \rho_{\Delta_{\mathrm{z}}} = \rho_{\Delta_{\mathrm{z}}} \leq R^*$.

Otherwise, $|C| \geq 2$ and $L \leq 2R^*$. We distinguish two cases.

Case 1: $2R^* \leq U$**.** By Lemma 4 and the geometric spacing of the scales, there exists an active scale Δ_j satisfying $2R^* \leq \Delta_j < (2 + \epsilon)R^*$. Consider the representative set C'_j maintained at this scale. Any two distinct representatives $u, v \in C'_j$ satisfy $d(u, v) > \Delta_j \geq 2R^*$.

For each $c \in C'_j$, choose a supplier $f^*(c) \in S^*$ satisfying $d(c, f^*(c)) \leq R^*$. The mapping $c \mapsto f^*(c)$ is injective. Indeed, if two distinct representatives u, v were assigned to the same supplier f, then $d(u, v) \leq d(u, f) + d(f, v) \leq 2R^*$, contradicting their separation.

Let $l(c)$ denote the group containing $f^*(c)$. By Lemma 5, the retained set $S_{l(c)}$ contains a closest supplier to c among all suppliers in $F_{l(c)}$. Choose $f_c \in \arg\min_{g\in S_{l(c)}} d(c, g)$. Then $d(c, f_c) = \min_{g\in F_{l(c)}} d(c, g) \leq d(c, f^*(c)) \leq R^*$. The mapping $c \mapsto f_c$ is also injective. Otherwise, two distinct representatives assigned to the same retained supplier would be at distance at most $2R^*$, again contradicting their separation. Define $R = \max_{c\in C'_j} d(c, f_c)$. Since this maximum is attained by a representative–supplier pair in $C'_j \times \widehat{F}$, we have $R \in \mathcal{D}_j$. Moreover, $R \leq R^*$.

We now construct an integral flow showing that R is feasible. For each group $l \in [m]$, let $Q_l = \{f_c : c \in C'_j,\ l(c) = l\}$ and $q_l = |Q_l|$. The injectivity of the optimal assignment $c \mapsto f^*(c)$ implies that $q_l \leq k_l^*$. Furthermore, the retention procedure guarantees $|S_l| = \min\{k, |F_l|\} \geq k_l^*$, so the retained suppliers in each group are sufficient to realize the group count k_l^*.

For each representative $c \in C'_j$, send one unit of flow along $s \to F_{l(c)} \to f_c \to c \to t$. For each group $l \in [m]$, send the remaining $k_l^* - q_l$ units along $s \to F_l \to F'_l \to t$. Thus, the source arc (s, F_l) carries exactly k_l^* units, satisfying its lower and upper bounds. The auxiliary arcs carry $k_l^* - q_l \leq k_l^* \leq \beta_l$ units. The representative assignments respect all unit capacities because the suppliers f_c are distinct. Finally, every supplier–client arc used by this construction is present in the threshold network, since $d(c, f_c) \leq R$.

Consequently, the constructed flow is integral and feasible, with total value $\sum_{l\in[m]} k_l^* = k$ and cost $\sum_{l\in[m]}(k_l^* - q_l) = k - |C'_j|$. On the other hand, every feasible flow x of value k in this network satisfies $\text{cost}(x) = k - \sum_{c\in C'_j} x_{ct} \geq k - |C'_j|$, because $x_{ct} \leq 1$ for every representative. The constructed flow therefore attains the minimum possible cost, giving $z_{\Delta_j,R} = k - |C'_j|$. Hence, R is feasible.

Since ρ_{Δ_j} is the minimum feasible candidate radius at this scale, $\rho_{\Delta_j} \leq R \leq R^*$. It follows that $\Delta_j + \rho_{\Delta_j} < (2+\epsilon)R^* + R^* = (3+\epsilon)R^*$.

Case 2: $2R^* > U$**.** Consider the largest active scale $\Delta_J = L(1+\varepsilon/2)^J$. By Lemma 3(ii), this scale remains active and its representative set is $C'_J = \{c_1\}$. Moreover, by Lemma 3(i) and $\Delta_J \geq U$, every client is within distance Δ_J of c_1. Thus, C'_J contains exactly one representative, which we denote by c_0.

Choose $f^*(c_0) \in S^*$ satisfying $d(c_0, f^*(c_0)) \leq R^*$, and let l_0 be its supplier group. By Lemma 5, there exists $f_0 \in S_{l_0}$ such that $d(c_0, f_0) \leq d(c_0, f^*(c_0)) \leq R^*$. Set $R = d(c_0, f_0)$. Then $R \in \mathcal{D}_J$ and $R \leq R^*$.

To show that R is feasible, send one unit of flow along $s \to F_{l_0} \to f_0 \to c_0 \to t$. Send $k_{l_0}^* - 1$ additional units through the auxiliary path of group l_0, and send k_l^* units through the auxiliary path of every other group l. This construction is feasible because $k_{l_0}^* \geq 1$ and $\alpha_l \leq k_l^* \leq \beta_l$ for every group. It yields an integral flow of value k and cost $(k_{l_0}^* - 1) + \sum_{l\neq l_0} k_l^* = k - 1 = k - |C'_J|$. As in Case 1, this cost matches the lower bound for every feasible flow of value k. Therefore, $z_{\Delta_J,R} = k - |C'_J|$, so R is feasible and $\rho_{\Delta_J} \leq R \leq R^*$.

By the definition of J, $\Delta_J < (1+\epsilon/2)U$. Since $U < 2R^*$ in this case, we obtain $\Delta_J + \rho_{\Delta_J} < (1+\epsilon/2)U + R^* < (3+\epsilon)R^*$.

Thus, in either case, there exists an active scale Δ with a feasible candidate radius. Its minimum feasible radius satisfies $\rho_\Delta \leq R^*$ and $\Delta + \rho_\Delta < (3+\epsilon)R^*$. □

B.7 Proof of Theorem 3

Proof. We first prove the approximation guarantee. Let (S^*, R^*) be an optimal solution to the FR-k-SUP instance. By Lemma 8, among the active scales remaining after the client streaming stage, there exists a scale Δ such that the corresponding minimum feasible candidate radius satisfies $\rho_\Delta \leq R^*$ and $\Delta + \rho_\Delta \leq (3+\epsilon)R^*$. Let Δ^* be the scale selected by the algorithm. Since the algorithm enumerates all active scales admitting a feasible candidate radius and selects one minimizing $\Delta + \rho_\Delta$, we have $\Delta^* + \rho_{\Delta^*} \leq \Delta + \rho_\Delta \leq (3+\epsilon)R^*$.

Let C'_{Δ^*} be the representative client set associated with Δ^*. By Lemma 3(iii), for every client $c \in C$, there exists a representative $c' \in C'_{\Delta^*}$ such that $d(c, c') \leq \Delta^*$. Moreover, by Lemma 6, there exists a supplier $f \in S_{\Delta^*}$ such that $d(c', f) \leq \rho_{\Delta^*}$. Therefore, by the triangle inequality, $d(c, f) \leq d(c, c') + d(c', f) \leq \Delta^* + \rho_{\Delta^*} \leq (3+\epsilon)R^*$. Since S_{Δ^*} also satisfies all fair range constraints, the algorithm achieves a $(3+\epsilon)$-approximation.

We next analyze the storage required by the streaming summaries and post-streaming data structures, excluding the LP solver's additional workspace. Let $Q = 1$ when $|C| = 1$, and $Q = O(1 + \lceil \log_{1+\epsilon/2}(U/L) \rceil)$ otherwise. In either case, Q bounds the number of scale instances maintained at any time.

The initialization buffer stores at most $k+1$ distinct clients and requires $O(k)$ space. For each active scale, the algorithm maintains at most k representative clients and at most k suppliers from each of the m groups, requiring $O(mk)$ space. The temporary candidate and protected sets require only $O(k)$ additional space when processing an incoming supplier at a fixed scale. Thus, the streaming stage uses $O(Qmk)$ space.

In the post-streaming stage, fix an active scale. Since $|C'| \leq k$ and $|\widehat{F}| \leq mk$, the candidate radius set contains at most mk^2 values and requires $O(mk^2)$ space. For each candidate radius, the threshold network has $O(mk)$ nodes and $O(mk^2)$ edges. Its sparse LP representation and an associated flow each require $O(mk^2)$ space. The recorded best flow also has at most $O(mk^2)$ entries.

Since active scales and radius tests are processed sequentially, threshold instances and their solutions do not accumulate across tests. The post-streaming data structures therefore require $O(mk^2)$ space in addition to the stored summaries. Combining these bounds, the required storage is $O\Big(\max\Big\{mk(1+\lceil\log_{1+\epsilon/2}(U/L)\rceil), mk^2\Big\}\Big)$. □

B.8 Time Complexity of the Streaming Algorithm

Theorem 4. *Let $n = |C \cup F|$ and $Q = O(1+\lceil\log_{1+\epsilon/2}(U/L)\rceil)$. The complete client-first streaming algorithm, which applies Algorithm 3 to each active scale, has total running time $O\big(Q\left[k^2 n + (mk^2)^{4.5}\log^2(2mk)\right]\big)$, under the arithmetic-operation model used in the offline analysis.*

Proof. At most Q scales are introduced during the client stream. We analyze the three stages separately.

In the client streaming stage, processing a client at a fixed scale requires computing its distance to at most k representatives, taking $O(k)$ time. The update of U uses the same distance computations. Including initialization, the client streaming stage therefore takes $O(Qk|C|)$ time.

In the supplier streaming stage, for each incoming supplier $f \in F_l$ and each active scale, the algorithm computes $f_c^l \in \arg\min_{g\in A_l} d(c,g)$ for every $c \in C'$. Since $|C'| \le k$ and $|A_l| \le k+1$, these computations take $O(k^2)$ time. Constructing the protected set and updating S_l do not exceed this bound. Thus, the supplier streaming stage takes $O(Qk^2|F|)$ time.

For the post-streaming stage, fix an active scale. Since $|C'| \le k$ and $|\widehat{F}| \le mk$, there are at most mk^2 candidate radii. For each candidate radius, the threshold network has $O(mk)$ nodes and $O(mk^2)$ edges. Its LP therefore has $N_{\mathrm{LP}} = O(mk^2)$ variables and $O(mk^2)$ nonzero coefficients.

Without loss of generality, we cap each fairness upper bound at k. For feasible instances, the lower bounds are also at most k. Since all constraint coefficients belong to $\{-1,0,1\}$, the sparse binary encoding length is $\mathcal{L} = O(mk^2\log(2mk))$. Using the same bound for Karmarkar's algorithm [24] as in the offline analysis, one LP can be solved in $O(N_{\mathrm{LP}}^{3.5}\mathcal{L}) = O\big((mk^2)^{4.5}\log(2mk)\big)$ arithmetic operations.

By Lemma 7, feasibility is monotone over the candidate radii. Binary search therefore uses $O(\log(2mk))$ radius tests. Consequently, the post-streaming computation takes $O((mk^2)^{4.5}\log^2(2mk))$ time per active scale. Constructing and sorting the candidate radii, building the threshold networks, and recovering the supplier set are dominated by this bound.

Combining all stages gives $O\big(Q\left[k|C| + k^2|F| + (mk^2)^{4.5}\log^2(2mk)\right]\big)$. Since $k \ge 1$ and $|C| + |F| \le 2n$, this yields the claimed bound. □

C Extension to Intersecting Supplier Groups

Although our main results focus on disjoint supplier groups, we extend our approach to the intersecting-group setting, where a supplier may belong to multiple groups and contribute simultaneously to their fairness constraints. Consequently, the MCMF formulation for disjoint groups cannot be directly applied. Gadekar et al. [13] established a tight 3-approximation for this setting using an FPT algorithm. We present an alternative dynamic programming approach that also achieves a 3-approximation.

The key idea is to construct disjoint representative supplier sets and employ two-stage dynamic programming to enforce representative-client coverage and fair range constraints. For a fixed candidate radius R, we first construct a maximal $2R$-independent representative client set $C' = \{c_1,\dots,c_r\}$. For each $c_i \in C'$, define $G_i = \{f \in F : d(f,c_i) \le R\}$. Since the representative clients are pairwise separated by distances greater than $2R$, the sets $G_1,\dots,G_r$ are pairwise disjoint by the triangle inequality. Thus, covering all representative clients requires selecting at least one supplier from each G_i. Let $H = F \setminus \bigcup_{i=1}^{r} G_i$ denote the remaining suppliers. If $r > k$ or any G_i is empty, the current radius is infeasible.

To handle overlapping group memberships, we represent each supplier $f \in F$ by a binary vector $w_f \in \{0,1\}^m$, where $(w_f)_j = 1$ if and only if $f \in F_j$. For any selected supplier set $S \subseteq F$, define its group-count vector as $w(S) = \sum_{f \in S} w_f$, where $w(S)_j = |S \cap F_j|$ denotes the number of selected suppliers belonging to group F_j. Hence, the fair range constraints can be expressed as $\boldsymbol{\alpha} \leq w(S) \leq \boldsymbol{\beta}$, where the inequalities hold componentwise.

We define the group-count state space as $\mathcal{Y} = \{y \in \mathbb{Z}^m_{\geq 0} : 0 \leq y_j \leq \beta_j,\ \forall j \in [m]\}$. Since at most k suppliers can be selected, we may assume $\beta_j \leq k$ without loss of generality. Thus, $|\mathcal{Y}| \leq (k+1)^m$. For each group-count vector y, the dynamic program stores the minimum number of selected suppliers needed to achieve y, rather than explicitly maintaining all corresponding supplier subsets. This is sufficient because, at the same processing stage and with the same coverage status, two partial solutions with identical group-count vectors have the same remaining fairness requirements. Retaining the one with fewer selected suppliers preserves at least as much capacity under the cardinality bound k.

In the first dynamic programming stage, we process the representative supplier sets $G_1, \ldots, G_r$ sequentially. Let $D_i[y]$ denote the minimum number of suppliers selected from the first i sets such that each G_h, $h \in [i]$, contains at least one selected supplier and the resulting group-count vector equals y. We initialize $D_0[\mathbf{0}] = 0$ and $D_0[y] = +\infty$ for all other states.

When processing $G_i = \{f_{i,1}, \ldots, f_{i,s_i}\}$, we define an auxiliary state $A^{(b)}_{i,q}[y]$ as the minimum number of suppliers selected from the first $i-1$ representative supplier sets and the first q suppliers of G_i, such that each of the first $i-1$ sets contains at least one selected supplier, the group-count vector equals y, and $b \in \{0,1\}$ indicates whether at least one supplier has been selected from G_i. If no such selection exists, the state value is $+\infty$. Initially, $A^{(0)}_{i,0}[y] = D_{i-1}[y]$ and $A^{(1)}_{i,0}[y] = +\infty$.

For each supplier $f \in G_i$, the algorithm either skips f or selects it, with the latter increasing the group-count vector by w_f and the supplier count by one. Transitions violating any upper bound β_j or the cardinality limit k are discarded. After processing all suppliers in G_i, we set $D_i[y] = A^{(1)}_{i,s_i}[y]$, retaining only states in which at least one supplier has been selected from G_i. This ensures that the corresponding representative client c_i is covered within distance R. After all representative supplier sets have been processed, $D_r[y]$ records the minimum number of selected suppliers needed to achieve group-count vector y while covering all representative clients and satisfying the upper-bound constraints.

In the second stage, we process the remaining suppliers $H = \{h_1, \ldots, h_t\}$ to satisfy any unmet lower-bound requirements. Let $E_q[y]$ denote the minimum number of selected suppliers needed to achieve group-count vector y after processing the first q suppliers in H, while preserving coverage of all representative clients. We initialize $E_0[y] = D_r[y]$. For each $h_q \in H$, the transition is $E_q[y] = \min\left\{E_{q-1}[y], E_{q-1}[y - w_{h_q}] + 1\right\}$, where the second term is considered only if $y - w_{h_q} \in \mathcal{Y}$ and the resulting supplier count does not exceed k. Since all representative clients have already been covered, no additional coverage indicator is required in this stage. The algorithm returns a supplier set if there exists $y \in \mathcal{Y}$ such that $E_t[y] \leq k$ and $\alpha_j \leq y_j \leq \beta_j$ for every $j \in [m]$.

The corresponding supplier set is recovered by backtracking through the dynamic programming states. Otherwise, the current radius R is infeasible.

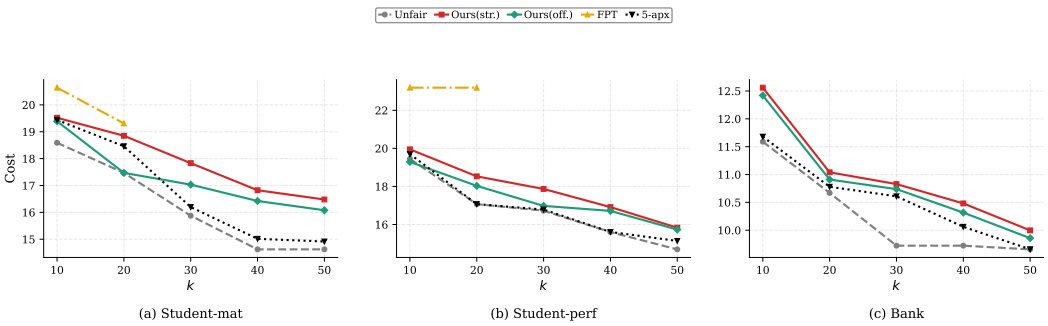
Theorem 5. *For intersecting supplier groups, the proposed dynamic programming algorithm achieves a 3-approximation for the fair range k-supplier problem in $O\left(\left(n^2 + nm\, 2^{m \log_2(k+1)}\right) \log n\right)$ time, where $n = |C| + |F|$.*

Proof. Let R^* denote the optimal radius, and let S^* be an optimal supplier set. For any candidate radius $R \geq R^*$, each representative client $c_i \in C'$ can be served by some supplier in $S^* \cap G_i$. Since S^* satisfies the cardinality and fair range constraints, a feasible supplier selection exists for the representative sets at radius R.

The first dynamic programming stage considers all valid selections from the representative supplier sets while ensuring that each set contains at least one selected supplier. The second stage considers selections from the remaining suppliers. Since the dynamic program retains the minimum supplier count for every group-count vector and coverage status, it finds a feasible solution whenever one exists. Therefore, the radius test succeeds for every $R \geq R^*$.

Table 4: Summary of real-world datasets used in the experiments. n: number of points; d: number of features; $(|F_1|, |F_2|)$: supplier group sizes; Supplier criteria: attribute-based rules for selecting suppliers.

Dataset	n	d	$(\lvert F_1\rvert, \lvert F_2\rvert)$	Supplier criteria
Heart	303	13	(43,44)	age ≤ 50
National-poll	714	15	(24,28)	race = "Black"
Student-mat	395	59	(126,147)	guardian = "mother"
Student-perf	649	59	(208,247)	guardian = "mother"
Bank	4521	52	(795,1511)	education = "secondary"
Census	48842	106	(1645,3040)	race = "Black"
Credit-card	30000	24	(8739,4920)	married = "True"
Bank-full	45211	52	(15179,8023)	education = "secondary"
Query	200000	7	(48574,31454)	random sampling



(a) Student-mat (b) Student-perf (c) Bank

Figure 4: Comparison of clustering costs achieved by different algorithms on the Student-mat, Student-perf, and Bank datasets with varying k.

By applying binary search over the candidate client–supplier distances, we obtain a successful radius $\widehat{R} \leq R^*$. Let S be the corresponding supplier set. Since C' is maximal $2\widehat{R}$-independent, every client $c \in C$ has a representative $c_i \in C'$ satisfying $d(c, c_i) \leq 2\widehat{R}$. Moreover, the algorithm selects a supplier $f_i \in S \cap G_i$ with $d(c_i, f_i) \leq \widehat{R}$. By the triangle inequality, $d(c, S) \leq d(c, c_i) + d(c_i, f_i) \leq 3\widehat{R} \leq 3R^*$. The dynamic programming feasibility conditions also guarantee $|S| \leq k$ and $\alpha_j \leq |S \cap F_j| \leq \beta_j$ for every $j \in [m]$. Hence, the algorithm achieves a 3-approximation.

For the running time, the number of group-count states is at most $2^{m \log_2(k+1)}$. Each supplier is processed exactly once across the two dynamic programming stages, and each state transition takes $O(m)$ time to check group-count feasibility. Thus, the dynamic programming procedure requires $O(nm\, 2^{m \log_2(k+1)})$ time per candidate radius. Constructing the representative sets takes $O(kn)$ time, and sorting the candidate distances requires $O(n^2 \log n)$ time. Combining these costs with $O(\log n)$ radius tests yields $O\left(\left(n^2 + nm\, 2^{m \log_2(k+1)}\right) \log n\right)$. □

D Additional Experimental Results

Evaluation metrics. Following prior work on fair clustering [13], we evaluate all methods using standard metrics: cost and runtime.

Additional scalability analysis on real-world datasets. Figure 4 reports the cost achieved by different algorithms on the `Student-mat`, `Student-perf`, and `Bank` datasets under varying k. For different values of k, we proportionally scale the fairness constraints to maintain comparable fairness requirements across different solution sizes. Specifically, for $k = z \cdot 10$, we set $[\alpha_1, \beta_1] = [\alpha_2, \beta_2] = [4z, 8z]$, where $z \in \mathbb{Z}_{>0}$. Overall, the cost of all algorithms consistently decreases as k increases.

Table 5: Cost comparison under fair range constraints $[\alpha_1, \beta_1] = [\alpha_2, \beta_2] = [8, 16]$ and $k = 20$. Underlined values indicate the lowest clustering cost between Unfair and 5-apx, while bold values indicate the lowest clustering cost among Ours (str.), Ours (off.), and FPT. For Unfair and 5-apx, values in parentheses indicate the number of runs (out of 10) with lower-bound violations. "–" indicates that the experiment exceeded the maximum allowed runtime of 100,000 s.

Dataset	Unfair	5-apx [7]	Ours (str.)	Ours (off.)	FPT [13]
Heart	4.26(5)	4.26(4)	**4.26**	**4.26**	**4.26**
National-poll	4.32(5)	4.32(3)	**4.32**	**4.32**	**4.32**
Student-mat	17.47(7)	18.46(2)	18.85	**17.50**	19.31
Student-perf	17.07(5)	17.07(3)	18.31	**18.09**	23.19
Bank	10.67(6)	10.78(7)	11.08	**10.92**	–
Census	13.09(9)	14.82(5)	14.60	**13.45**	–
Credit-card	4.70(5)	4.70(2)	5.10	**5.06**	–
Bank-full	11.21(7)	11.59(6)	11.54	**11.40**	–
Query	2.49(5)	2.49(5)	2.65	**2.50**	–

Table 6: Runtime in seconds (mean ± std) under fair range constraints $[\alpha_1, \beta_1] = [\alpha_2, \beta_2] = [8, 16]$ and $k = 20$. Underlined values indicate the lowest mean runtime between Unfair and 5-apx, while bold values indicate the lowest mean runtime among Ours (str.), Ours (off.), and FPT. "–" indicates that the experiment exceeded the maximum allowed runtime of 100,000 s.

Dataset	Unfair	5-apx [7]	Ours (str.)	Ours (off.)	FPT [13]
Heart	0.00±0.00	0.01±0.00	10.29±0.12	**0.05**±0.01	261.66±6.86
National-poll	0.04±0.00	0.04±0.00	9.92±0.71	**0.03**±0.01	348.74±8.97
Student-mat	0.07±0.02	0.07±0.00	6.72±0.19	**0.09**±0.01	3046.07±203.93
Student-perf	0.10±0.00	0.12±0.01	7.00±0.36	**0.14**±0.02	7602.17±301.66
Bank	0.67±0.03	0.71±0.03	18.34±0.60	**1.29**±0.10	–
Census	14.15±0.65	3.06±0.03	190.54±6.31	**4.36**±0.42	–
Credit-card	2.41±0.04	3.45±0.14	79.23±3.86	**21.04**±0.78	–
Bank-full	7.24±0.13	10.21±0.19	161.28±9.50	**13.65**±0.59	–
Query	6.40±0.31	8.02±0.12	129.76±6.70	**115.47**±14.25	–

The computational complexity of the FPT algorithm grows exponentially with respect to mk, making it impractical for larger parameter settings. Consequently, on both `Student-mat` and `Student-perf` datasets, FPT fails to return results when $k > 20$ within a reasonable running time budget.

Compared with the baselines, our methods Ours (str.) and Ours (off.) do not always achieve the minimum cost. This is mainly because our approaches explicitly enforce both upper-bound and lower-bound fairness constraints during optimization, which inevitably restricts the feasible solution space. In contrast, several baseline methods primarily optimize the objective value without strictly satisfying these fairness constraints, thereby obtaining lower costs at the expense of fairness guarantees. Nevertheless, our methods still achieve competitive performance across all datasets while maintaining strict constraint satisfaction and good scalability.

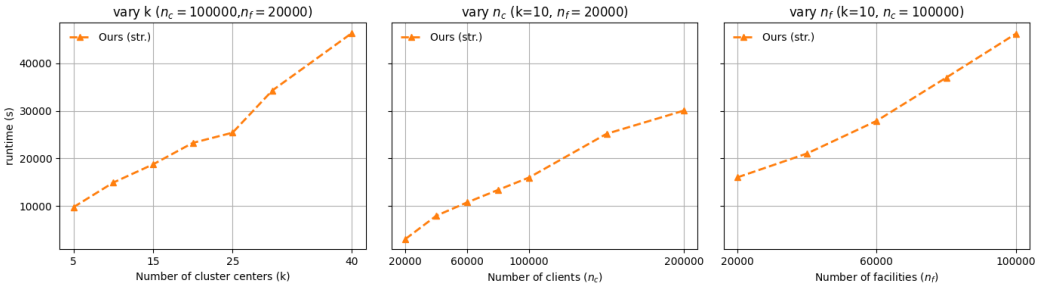


Figure 5: Running time on synthetic datasets under different parameter settings.

Table 7: Effect of k on synthetic instances with overlapping supplier groups, where $n = 1000$ and $m = 4$. For each group, the bounds are set to $\alpha = \lceil k/m \rceil$ and $\beta = k$. Bold values indicate the lower cost or mean runtime between Ours (FPT) and FPT [13].

	Ours (FPT)		FPT [13]		Unfair	
k	Cost	Time (s)	Cost	Time (s)	Cost	Time (s)
5	**0.3061**	**1.74**±0.41	0.3065	18.42±1.68	0.2963	0.01±0.00
6	**0.3007**	**3.82**±0.91	0.3009	180.35±7.45	0.2952	0.01±0.00
7	**0.2979**	**4.61**±0.34	0.2985	258.20±9.59	0.2841	0.02±0.00
8	**0.2819**	**10.44**±2.16	**0.2819**	1458.69±25.72	0.2794	0.02±0.00

Table 8: Scalability with respect to n on synthetic instances with overlapping supplier groups, where $k = 5$, $m = 4$, and each group has lower and upper bounds $\alpha = 3$ and $\beta = 5$, respectively. Bold values indicate the lower cost or mean runtime between Ours (FPT) and FPT [13].

	Ours (FPT)		FPT [13]		Unfair	
n	Cost	Time (s)	Cost	Time (s)	Cost	Time (s)
1,000	**0.284**	**1.22**±0.15	**0.284**	20.73±1.07	0.268	0.012±0.00
2,000	**0.312**	**1.64**±0.10	0.314	32.22±5.82	0.289	0.016±0.01
5,000	**0.332**	**1.95**±0.27	**0.332**	55.07±13.64	0.293	0.027±0.02
7,000	**0.333**	**2.13**±0.24	**0.333**	83.89±15.31	0.304	0.036±0.02
10,000	**0.336**	**2.52**±0.32	0.337	122.80±10.32	0.315	0.053±0.03

Scalability on synthetic datasets. In addition to real-world datasets, we conduct experiments on synthetic instances to study the scalability of our algorithms. Since there are no existing streaming algorithms for k-supplier under the fair range constraints, we report results only for our streaming algorithm in these scalability experiments.

For the simulated experiments, data points are sampled from a mixture of Gaussian distributions in $\mathbb{R}^{10}$. We generate synthetic datasets under multiple configurations of problem parameters. For each instance, reference points are generated uniformly at random. Client points and supplier points are then sampled around these points with different variances, following the same generation rule across all configurations. When varying the number of clients or suppliers, we directly generate datasets with the target sizes, instead of subsampling from a single dataset.

We generate a collection of synthetic datasets and evaluate performance under varying problem scales. Specifically, we examine the impact of three key parameters: the number of selected suppliers k, the number of clients n_c, and the size of the supplier set n_f. In each experiment, one parameter is varied while the others are held fixed to isolate its effect on running time. To study the effect of k, we fix $n_c = 100{,}000$ and $n_f = 20{,}000$, and vary k in the range $[5, 40]$. To evaluate scalability with respect to the number of clients, we fix $k = 10$ and $n_f = 20{,}000$, and vary n_c from 20,000 to 200,000. Similarly, we vary the supplier set size n_f from 20,000 to 100,000 while keeping $k = 10$ and $n_c = 100{,}000$ fixed. Fig. 5 shows that the running time increases steadily with k, n_c, and n_f. This trend is expected, since larger values of these parameters lead to increased candidate radius evaluations, distance computations, and feasibility checks. Importantly, the growth is smooth, indicating good scalability of our approach.

Scalability for intersecting groups. We evaluate the algorithms on synthetic instances with overlapping supplier groups. We compare our FPT algorithm with the overlapping-group FPT algorithm of [13] and the standard unfair 3-approximation algorithm [19] for k-supplier. We generate synthetic instances with overlapping supplier groups, with equal numbers of clients and suppliers, i.e., $n_c = n_f = n/2$, and all points have dimension $d = 5$. We set the number of groups to $m = 4$, and each group contains $2n_f/m$ uniformly sampled suppliers, resulting in overlapping groups.

Running times are reported as mean $\pm$ standard deviation over ten independent runs on the same dataset for each configuration. Table 7 evaluates the effect of varying k, while Table 8 evaluates scalability with respect to the instance size n. Our FPT algorithm is substantially faster than FPT [13] across all tested values of n, while exhibiting much slower runtime growth as the instance size increases.

E Limitations and Future Work

Limitations. Our polynomial-time algorithm applies to disjoint supplier groups. For overlapping groups, the dynamic program maintains up to $(k+1)^m$ group-count states, which can limit scalability as the number of groups or the selection budget increases. Second, our fairness guarantees concern lower and upper bounds on supplier representation and do not cover other fairness notions, such as proportional or individual fairness. Third, our streaming algorithm assumes a client-first arrival order, which limits its applicability when clients and suppliers arrive in an interleaved manner and their processing order cannot be controlled.

Future work. One direction is to improve the time and space efficiency of the overlapping-group algorithm by reducing the state space or exploiting structure in group intersections. Another direction is to combine supplier representation constraints with other fairness requirements and characterize the resulting feasibility and approximation trade-offs. Developing memory-efficient streaming algorithms for interleaved client–supplier arrivals, with provable approximation and fairness guarantees, is also an important direction. Finally, for disjoint supplier groups, it would be valuable to investigate how geometric structure, such as low-dimensional Euclidean metrics, can be exploited to improve computational efficiency and approximation guarantees.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope, including the proposed algorithm and its theoretical guarantees. These claims are well supported by the analysis and experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, we discuss limitations of our model in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide a detailed and formal description of our model in Section 2, and provide formal theorem statements and proofs for each claim that we make.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in the appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions of the datasets, sampling procedures, and algorithm implementations to facilitate the reproduction of our experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide our code as part of the supplementary material, and the datasets we use are openly available.

Guidelines:

- The answer [N/A] means that the paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Yes, we outline all the parameters of our experiments in detail.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in the appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All experiments are conducted over at least 10 runs. Means and standard deviations are reported in the tables, and percentage-based metrics are presented in the experimental analysis, providing appropriate information about variability and statistical reliability.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar rather than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides sufficient information on the computer resources, including the CPU type, memory, software environment, and solver, to support reproducibility. The reported running times further provide insight into the computational cost.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms to the NeurIPS Code of Ethics. Our results are mainly theoretical and contain no ethics violations, and our experiments are conducted on commonly used, publicly available datasets, which raise no ethical concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Our work is a theoretical study on fair clustering algorithms with potential applications in resource allocation and data analysis. We do not anticipate direct negative societal impacts beyond general considerations for algorithmic decision-making systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not involve the release of data or models that pose a risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, we properly cite all datasets and code used in our experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were only used for editing and formatting purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.