The Heterogeneous Capacitated k-Center Problem

Deeparnab Chakrabarty
Microsoft Research India
deeparnab@gmail.com

Ravishankar Krishnaswamy
Microsoft Research India
ravishankar.k@gmail.com

Amit Kumar
Comp. Sci. & Engg., IIT Delhi
amitk@cse.iitd.ac.in

Abstract

The abstract goes here.

1 Introduction

2 Reduction to Max-Min Allocation via Region Growing

In this section, we give a reduction to $Q|k_i|C_{min}$ when we allow logarithmic approximations. The main technique is region growing which was first used in the context of sparsest and multi cut problems [**?**, **?**].

Theorem 2.1. *Given an α-approximation algorithm for $Q|k_i|C_{min}$, for any $\epsilon > 0$ there exists an $(O(\log n/\epsilon), \alpha(1+\epsilon))$-approximate algorithm for the Heterogeneous Cap-k-Center problem.*

Proof. We start with a guess of OPT and the OPT-induced graph G. We assume this is a correct guess. Given G we given an algorithm construct an instance $\mathcal{I}$ of $Q|k_i|C_{min}$. Initially $\mathcal{I}$ is empty. We maintain a set of alive clients C' which is initially C. We maintain a working graph H which is initialized to G and is always a subgraph of G. Given a client j and an integer t, let $N_t(j)$ denote all the clients j' s.t. $d_H(j,j') < 2t$ and $\Gamma_t(j)$ denote all the clients j' with $d_H(j,j') = 2t$.

Till C' is empty, we perform the following operation. Select an arbitrary active client $j \in C'$. Find the smallest t such that $|\Gamma_t(j)| < \varepsilon \cdot |N_t(j)|$. Since for all $s < t$ we have $|N_{s+1}(j)| \geq (1+\epsilon)|N_s(j)|$ which implies $|N_{s+1}(j)| > (1+\epsilon)^s$, we get $t \leq (\ln n)/\epsilon$ where $n = |C'|$. Let us use A_j to denote $N_t(j)$ and B_j to denote $\Gamma_t(j)$. Also let us use F_j to denote all the facilities neighboring any client in A_j. Note that any facility $i \in F_j$ has neighbors in $A_j \cup B_j$. We now add a machine to $\mathcal{I}$ and attribute it to the client j. Abusing notation, we call the machine j as well. We add the cardinality constraint $k_j = |F_j|$. The speed of the machine is set to be $s_j := 1/|A_j|$. Finally, we delete $A_j \cup B_j$ from C' and $A_j \cup B_j \cup F_j$ from H and repeat.

At the end of the above operation, we have an instance $\mathcal{I}$ where the machines and their cardinality constraints have been specified. We are given P jobs where the processing time of the pth job is c_p. This completes the description of the $Q|k_i|C_{min}$ instance.

Claim 2.2. *There exists an allocation where the total processing time of each machine is at least* 1*.*

Proof. In the optimal solution to Heterogeneous Cap-k-Center, a total capacity of $|A_j|$ must be installed among the facilities F_j. We mimic this solution for $\mathcal{I}$, and since the speed of machine j is $1/|A_j|$, the claim follows. □

Claim 2.3. *Given an allocation to $\mathcal{I}$ where each machine gets processing time $\geq \alpha$, we can construct a solution for the Heterogeneous Cap-k-Center problem which is $O(\log n/\epsilon)$-approximate and violates the capacities by at most $(1+\epsilon)/\alpha$-factor.*

Proof. For machine j in $\mathcal{I}$, let S_j be the jobs allocated; we have $\sum_{p \in S_j} c_p \geq \alpha \cdot |A_j|$ and $|S_j| \leq k_j = |F_j|$. We arbitrarily open $|S_j|$ facilities in F_j to which we assign all the clients in $A_j \cup B_j$. Since $|B_j| < \varepsilon |A_j|$, we have the total capacity opened is at least $\alpha/(1+\epsilon)$ times the total number of clients in $A_j \cup B_j$. Furthermore, every client in $A_j \cup B_j$ is at most $O(\log n/\epsilon)$-away from any facility in F_j. This implies the assignment. □

□

3 Preliminaries: LP Relaxations and Supply Polyhedra

- Re-state natural LP for Heterogeneous Cap-k-Center.
- Definition of Supply Polyhedra.
- Technical Conditions – running sums and separation oracle.
- Supply Polyhedra for $Q||C_{min}$.
- Supply Polyhedra for $Q|f_i|C_{min}$.
- No $(1+\epsilon)$-supply polyhedra possible.
- Connection to Integrality Gaps ... how α-appx supply polyhedra imply α-approximation.
- Rounding Assignment for Small Jobs.

4 The Decomposition Theorem

In the previous section, we found non-expanding balls around active clients to build the decomposition. These balls were based on the underlying graph G. We now give a more delicate region growing procedure which uses an LP fractional solution as a guide. Recall the LP relaxation mentioned in Section **??**. We had variables x_{ijp} to denote that a client j gets assigned to location i where a type p facility has been opened, and y_{ip} to denote that a facility of type p has been opened at location i. For the purpose of our rounding algorithm, we assume that we are given a feasible solution (x, y) to this LP relaxation. Our first step is to formalize the two notions described in Section **??**, where we wanted to get a decomposition into a collection of complete neighborhoods (which correspond to $Q|k_i|C_{min}$ instances) and locally roundable pieces. Of course, we will not be able to get such a clean decomposition. Indeed, as we iteratively identify pieces which are either locally roundable or a complete neighborhood, the neighborhood structure of the remaining clients and facilities changes, and what may have been originally a locally roundable neighborhood piece may cease to be one subsequently. Our final algorithm then combines the above two ideas in a dynamic manner: we iteratively decompose the clients and facilities into three parts: the first being clients which are supported in the LP solution by locally-roundable neighborhoods, along with the facilities in these neighborhoods, the second is the clients and facilities which are isolated from each other and hence form the a complete neighborhood, and a third kind of clients (which we refer to as the deleted clients), which are in the proximity of the clients of the first two kinds, and their total demand can be charged to the demands in the first two kinds.

Towards that end, we first define the condition which implies that a subset of facilities S can be rounded integrally while (approximately) preserving the total demand they were serving fractionally.

Definition 1. *A set of facilities $S \subseteq F$ is said to be (a, b)-roundable w.r.t feasible solution (x, y) if*

(a) $\text{diam}_G(S) \leq a$

(b) there exists a rounding $Y_{ip} \in \{0, 1\}$ for all $i \in S, p$ such that

1. $\sum_{q \geq p} \sum_{i \in S} Y_{iq} \leq \lfloor \frac{1}{2} \cdot \sum_{q \geq p} \sum_{i \in S} y_{iq} \rfloor$ *for all p, and*
2. $\sum_{j \in C} d_j \sum_{i \in S, p \in [P]} x_{ijp} \leq b \cdot \sum_{i \in S} \sum_{p \in [t]} c_p Y_{ip}$

The idea behind the above definition should be clear – suppose S is a set of facilities and consider a subset of demands J which are being assigned (fractionally) by the solution (x, y) to S. If S is (a, b)-roundable wrt (x, y), then we can open facilities in S integrally and reassign the demands in J to these integral facilities. By doing so, we shall increase the connection cost of demands in J by an additive factor of a, and then we may violate the capacities of the integrally open facilities by at most a factor b. Note that the rounding in each such piece locally preserves the bounds on the number of centers the integral solution opens as compared to the fractional LP solution – in fact we only use half of the fractionally open facilities[1].

Definition 2. *A subset $S \subseteq F$ of facilities is called a* complete neighborhood *if there exists a client-set $J \subseteq C$ such that $\Gamma(J) \subseteq S$. In this case the subset J is said to be* responsible *for S. Additionally, a complete neighborhood S is said to be an a-complete neighborhood if* $\text{diam}(S) \leq a$.

The idea behind this definition should also be clear – indeed, if we find a complete neighborhood S of facilities with say a set J of clients responsible for it, then we know that the optimal solution must satisfy all the demand in J by suitably opening facilities of sufficient capacity in S. Moreover, if we can find a disjoint collection of such complete neighborhoods, then the overall problem resembles the $Q|k_i|C_{min}$ problem.

As described earlier, our algorithm tries to *partition* the set of facilities into a disjoint collection of sets of small diameter, such that each set is either a *complete neighborhood* of some clients, or is a roundable set with a and b being small constants. However, since the client-facility connections in the LP could be pretty

[1] The factor 1/2 here is only needed for our rounding algorithm. The decomposition theorem can be proved even if we do not allow this factor.

complex, it will not be possible to achieve such a clean partition. Our algorithm therefore may *delete some clients*, and *charge* the total demand of deleted clients to the demands of undeleted nearby clients. This is encapsulated in the following (rather verbose) theorem.

Theorem 4.1. *Given a feasible solution (x, y), there exists a polynomial time algorithm which finds a decomposition with the following properties.*

1. *The client set C is partitioned into three disjoint subsets $C = C_{\mathsf{del}} \cup C_{\mathsf{black}} \cup C_{\mathsf{blue}}$, and the facility set F is partitioned into two families $\mathcal{S} = (S_1, S_2, \ldots, S_K)$ and $\mathcal{T} = (T_1, T_2, \ldots, T_L)$ of mutually disjoint subsets.*
2. *Each T_ℓ is a 4-complete neighborhood with a corresponding set J_ℓ of clients responsible for it, and $C_{\mathsf{black}} = \cup_{\ell=1}^{L} J_\ell$.*
3. *Each $S_k \in \mathcal{S}$ is $(20, 32)$-roundable with respect to (x, y), and moreover, each client in C_{blue} satisfies $\sum_{i \in \mathcal{S}, p} x_{ijp} \geq \frac{1}{2}$.*
4. *For the deleted clients C_{del}, there is a mapping $\phi : C_{\mathsf{del}} \to C_{\mathsf{black}} \cup C_{\mathsf{blue}}$ such that (a) $d(j, \phi(j)) \leq 4$ for all $j \in C_{\mathsf{del}}$, and (b) for all $j \in C_{\mathsf{black}} \cup C_{\mathsf{blue}}$, we have $\sum_{j' \in C_{\mathsf{del}} : \phi(j') = j} d_{j'} \leq 32 \cdot d_j$, i.e., the total demand mapped to j is small.*

Before we prove the theorem, let us interpret each of the conditions above (note that we have abused notation in the third statement by using $\mathcal{S}$ to denote the set of locations in $S_1, \ldots, S_k$). C_{del} is the set of deleted clients. The fourth statement above argues that each such client can be charged to a nearby client, and the total charge to a client does not exceed its own demand by a constant factor. Therefore, given a feasible assignments of clients which are not in C_{del}, we can easily extend this to all the clients with a constant increase in the connection cost and constant factor violation of capacities. The second and the third statements give two different ways of forming partitions as mentioned earlier.

Proof of Theorem 4.1 We prove this theorem by describing an algorithm which constructs these partitions. The algorithm is written formally in Algorithm 1. We we describe it in detail first.

4.0.1 Algorithm Description

Our algorithm starts with the collections $\mathcal{S}$ and $\mathcal{T}$, and the clients sets C_{del}, C_{black}, and C_{blue} being empty. Once a facility is assigned into a set in $\mathcal{S}$ or $\mathcal{T}$, it is called an *assigned facility*. Similarly clients are assigned once they are added to $C_{\mathsf{del}} \cup C_{\mathsf{black}} \cup C_{\mathsf{blue}}$. As our algorithm forms these clusters, it changes the connection graph G by deleting all assigned clients and facilities. At any time, we denote the residual graph by H. The neighborhood structure in the residual graph is denoted by $\Gamma_H(\cdot)$, e.g., $\Gamma_H(i)$ denotes the set of neighbors of i in H, and $\Gamma_H(S)$ denotes the neighbors of a set of vertices S in H. Note that since we only delete vertices from the graph over the iterations, $\Gamma_H(S) \subseteq \Gamma_G(S)$ for all sets $S \subseteq V$ of the original set of vertices of G. For each of the partitions $\mathcal{S}$ and $\mathcal{T}$, let $L(\mathcal{S}) = \cup_{1 \leq k \leq K} S_k$ and $L(\mathcal{T}) = \cup_{1 \leq \ell \leq L} T_\ell$ denote the set of all locations in them respectively. Each set S_k (resp. T_ℓ) in the partitions $\mathcal{S}$ (resp. $\mathcal{T}$) will have a *root* facility $i_k \in S_k$ (resp. $i_\ell \in T_\ell$). We use $R(\mathcal{S})$ and $R(\mathcal{T})$ to denote the collection of roots $\cup_{1 \leq k \leq K} \{i_k\}$ and $\cup_{1 \leq \ell \leq L} \{i_\ell\}$ in $\mathcal{S}$ and $\mathcal{T}$ respectively.

A key definition in our algorithm is that of *effective capacity*. For every $i \notin L(\mathcal{S}) \cup L(\mathcal{T})$ and $p \in [P]$ with $y_{ip} > 0$, define

$$c_{\text{eff}}(i, p) := \frac{\sum_{j \in H \cap C} d_j x_{ijp}}{y_{ip}}$$

Recall that C denotes the set of all clients, and therefore, $H \cap C$ is the set of unassigned clients. Since all sets are initially empty, $c_{\text{eff}}(i, p)$ is well defined for all $i \in F, p \in [P]$. Whenever a facility enters $L(\mathcal{S}) \cup L(\mathcal{T})$, we fix its $c_{\text{eff}}(i, p)$ to be what it was at the iteration it entered. Since the set of clients in H only monotonically decreases, the effective capacity can only decrease over time. Each iteration of the algorithm (Line 3) begins by picking the pair $(i^\star, p^\star)$ with the highest effective capacity(Line 5). Starting from $i^\star$, we look at nodes in

H at distance at most 4 from it. Since H is a bipartite graph, we alternate between location vertices and client vertices. We denote J_1 and J_2 as clients which are at distance 1 and 3 from $i^\star$ respectively. Similarly, let A and B be locations at distance 2 and 4 from $i^\star$ respectively(Line 6-Line 9). Note that clients in J_1 are adjacent to $i^\star$ and locations in A in H, and those in J_2 are adjacent to locations in A and B only. Now, let $D(J_1)$ and $D(J_2)$ denote the total demand of J_1 and J_2 respectively. The algorithms branches in several cases:

- $D(J_2) \geq 32D(J_1)$: In this case, the (fractionally opened) facilities in $\{i^\star\} \cup A \cup B$ are servicing a large enough demand, in particular, more than the effective capacity of $i^\star$. This is so because the effective capacity of $i^\star$ is about $D(J_1)$, whereas the total demand serviced by $i^\star \cup A \cup B$ is about $D(J_2)$. Now, notice that the effective capacities of all these facilities is at most that of $i^\star$. Thus, this corresponds to a situation where we have large amount of capacity generated by a set of facilities which are fractionally open, but the *integral* capacity of each of these is small. In this case, we will show that one can replace these fractionally open facilities by integral facilities. Therefore, our algorithm adds the set $S_k := \{i^\star\} \cup A \cup B$ to $\mathcal{S}$ (Line 13). It makes $i^\star$ as the root of S_k. The set S_k is now removed from H. We will also remove A and B from H. In fact, we remove any client which is assigned to a total fraction of more than half to the facilities in $\mathcal{S}$, and add it to the set C_{blue} (Line 17).
- $D(J_2) < 32D(J_1)$: In this case, we would like to delete J_2 and assign them to J_1. Further, we can would like to treat $\{i^\star\} \cup A \cup B$ as a 4-complete neighborhood of J_1 (and then add these facilities as a new set in $\mathcal{T}$). However, the worry is that the neighborhoods A, B, J_1, J_2 are defined with respect to the graph H, and not the graph G. So it is possible that we have deleted some facility, and a clients in J_1 are fractionally assigned to such facilities. Therefore, the algorithm considers two sub-cases: (i) There is some root center $i_r \in R(\mathcal{S})$ close to $i^\star$ (Line 19) – in this case, the algorithm considers the closest such root i_r, and *augments* S_r to $S_r \cup \{i^\star\} \cup A$. As in the above case, we update C_{blue} by adding to it any client which has more than half of it fractional assignment to facilities in $\mathcal{S}$ (in particular, J_1 will get added to this set), (ii) There is no such root – in this case, the set $\{i^\star\} \cup A$ gets added as a new set T_ℓ to $\mathcal{T}$. Further, we add J_1 to C_{black}.

This completes the description of the algorithm. We now show that the decomposition has the desired properties.

4.0.2 Algorithm Analysis

In this section, we analyze the algorithm. At the beginning of each iteration, we want to show that the algorithm maintains the following invariants:

I1. For any facility $i \in L(\mathcal{T})$, $\Gamma_G(i) \cap V(H) = \emptyset$, i.e., $\Gamma_G(i)$ contains no unassigned clients. Note that this holds even w.r.t. all the neighbors according to the original graph G.

I2. Similarly, for any facility $i \in L(\mathcal{S})$ added in Line 23 in Algorithm 1, $\Gamma_G(i)$ contains no unassigned clients.

Note that in I2, we count only those i which get added to $L(\mathcal{S})$ in Line 23, and so do not consider locations getting added in Line 13.

Claim 4.2. *The two invariants hold at the beginning of every iteration of the while loop in Line 3.*

Proof. We show this by induction over the number of iterations t. Clearly, at $t = 1$, $L(\mathcal{T})$ and $L(\mathcal{S})$ are empty, so the invariants hold tautologically. Suppose they hold for iterations upto i. We show that they

Algorithm 1 Rounding algorithm for Theorem 4.1

procedure AlgDecompose(x, y)
  $t \leftarrow 1$; $k \leftarrow 1$; $\ell \leftarrow 1$; $H \leftarrow G$
  while there are no unassigned clients, i.e., $V(H) \cap C \neq \Phi$ **do**
    update $c_{\text{eff}}(\cdot, \cdot)$ for all $i \in V(H) \cap F, p \in [P]$
    $(i^*, p^*) \leftarrow \arg\max_{i \in H, p \in [P]} c_{\text{eff}}(i, p)$ ▷ *pick location and type with largest effective capacity*
    $J_1 \leftarrow \Gamma_H(i^*)$ ▷ J_1 *is the set of unassigned client neighbors*
    $A \leftarrow \Gamma_H(J_1) \setminus \{i^*\}$ ▷ $A \cup \{i^*\}$ *is the set of facility neighbors of* J_1 *in* H
    $J_2 \leftarrow \Gamma_H(A) \setminus J_1$ ▷ $J_1 \cup J_2$ *is the set of unassigned client neighbors of* A
    $B \leftarrow \Gamma_H(A) \setminus \{A \cup i^*\}$ ▷ $A \cup B \cup \{i^*\}$ *is the set of facility neighbors of* $J_1 \cup J_2$ *in* H
    $D(J_1) \leftarrow \sum_{j \in J_1} d(j)$
    $D(J_2) \leftarrow \sum_{j \in J_2} d(j)$
    if $D(J_2) \geq 32D(J_1)$ **then** ▷ $i^* \cup A \cup B$ *will be locally roundable*
      add a new part $S_k := i^* \cup A \cup B$ to $\mathcal{S}$
      define i^* to be the root of S_k, i.e., $R(\mathcal{S}) \leftarrow R(\mathcal{S}) \cup \{i^*\}$
      $H \leftarrow H \setminus (i^* \cup A \cup B)$ ▷ *delete assigned facilities from* H
      for each $j \in H$ s.t $\sum_{i \in \mathcal{S}, p} x_{ijp} \geq \frac{1}{2}$ **do**
        $C_{\mathsf{blue}} \leftarrow C_{\mathsf{blue}} \cup \{j\}$ and $H \leftarrow H \setminus \{j\}$ ▷ *assign clients to* C_{blue}
      $k \leftarrow k + 1$
    else if $D(J_2) < 32D(J_1)$ and $\text{dist}_G(i^*, R(\mathcal{S})) \leq 8$ **then** ▷ i^* *is close to some root in* $R(\mathcal{S})$
      $C_{\mathsf{del}} \leftarrow C_{\mathsf{del}} \cup J_2$ and augment ϕ appropriately ▷ *delete* J_2 *and charge to* J_1
      $H \leftarrow H \setminus J_2$
      let $i_r = \arg\min_{i \in R(\mathcal{S})} \text{dist}_G(i^*, i)$ ▷ i_r *is the nearby root from* $\mathcal{S}$
      $S_r \leftarrow S_r \cup i^* \cup A$ ▷ *add these facilities to* S_r
      for each $j \in H$ s.t $\sum_{i \in \mathcal{S}, p} x_{ijp} \geq \frac{1}{2}$ **do**
        $C_{\mathsf{blue}} \leftarrow C_{\mathsf{blue}} \cup \{j\}$ and $H \leftarrow H \setminus \{j\}$ ▷ *assign clients to* C_{blue}
    else ▷ $i^* \cup A \cup B$ *will be a 4-complete neighborhood of* J_1
      $C_{\mathsf{del}} \leftarrow C_{\mathsf{del}} \cup J_2$ and augment ϕ appropriately ▷ *delete* J_2 *and charge to* J_1
      $H \leftarrow H \setminus J_2$
      add a new part $T_\ell := i^* \cup A$ to $\mathcal{T}$
      $H \leftarrow H \setminus (i^* \cup A)$ ▷ *delete assigned facilities from* H
      $J_\ell \leftarrow J_1$, $C_{\mathsf{black}} \leftarrow C_{\mathsf{black}} \cup J_1$, and $H \leftarrow H \setminus J_1$ ▷ *assign clients to* C_{black}
      $\ell \leftarrow \ell + 1$
    $t \leftarrow t + 1$ ▷ *Iteration Counter*
  return $\mathcal{S}, \mathcal{T}$

also hold at the end of the t^{th} iteration, and hence they hold at the beginning of the $(t+1)^{th}$ iteration, thus completing the proof. To this end, consider the t^{th} iteration.

We first show that I1 continues to hold at the end of this iteration. Note that we only need to check if I1 holds for any new facilities added to $L(\mathcal{T})$ in this iteration, which only happens in Line 29. In this case, consider any facility $i \in T_\ell$, the set of facilities added to $L(\mathcal{T})$, and consider the neighborhood $\Gamma_G(i)$: in this set, some clients are already in $C_{\mathsf{blue}} \cup C_{\mathsf{black}} \cup C_{\mathsf{del}}$ in which case they would have been deleted from H in earlier iterations. By definition, the remaining clients belong to $J_1 \cup J_2$, since $J_1 \cup J_2$ contains all remaining neighbors of $i^* \cup A$. But clients in J_1 are added to C_{black}, and those in J_2 are added to C_{del}, hence i would have no clients as neighbors in H at the end of this iteration. Applying this to all $i \in T_\ell$ completes the proof.

We now show that I2 continues to hold at the end of this iteration. Similar to the above proof, note that we only need to check if I2 holds for any new facilities added to $L(\mathcal{S})$ in Line 23. In this case, consider any facility $i \in (\{i^\star\} \cup A)$, the set of facilities added to $L(\mathcal{S})$, and consider the neighborhood $\Gamma_G(i)$: in this neighborhood, some clients are already in $C_{\mathsf{blue}} \cup C_{\mathsf{black}} \cup C_{\mathsf{del}}$ in which case they would have been deleted from H in earlier iterations. By definition, the remaining clients belong to $J_1 \cup J_2$, since $J_1 \cup J_2$ contains all remaining neighbors of $\{i^\star\} \cup A$. But clients in J_2 are added to C_{del}, and we now show that all clients in J_1 would be colored blue in Line 25, hence showing that i would have no clients as neighbors in H at the end of this iteration. Indeed, consider any client $j \in J_1$: by definition, it was in H at the beginning of this iteration and so by invariant I1, there are no edges in H between j and any location $i' \in L(\mathcal{T})$. So all neighbors in $\Gamma_G(j)$ which have already been deleted belong to $L(\mathcal{S})$. Moreover, $\{i^*\} \cup A$ includes all remaining neighbors of j. Hence, for any such j, we know that $\sum_{i \in L(\mathcal{S}),p} x_{ijp} = 1$, and so it would be added to C_{blue} in Line 25. □

We now show that the deleted clients, C_{del} can be charged to C_{blue} and C_{black}.

Claim 4.3. *In Lines 20 and 27, ϕ can be augmented so that the Property (4) of Theorem 4.1 is satisfied.*

Proof. Note that whenever the algorithm executes either of lines 20 and 27, it must be that $D(J_2) \leq 32D(J_1)$. So indeed we are justified in deleting the clients in J_2 and "charging" them to J_1. More precisely, we augment the mapping function $\phi : J_2 \to J_1$ such that for all $j \in J_1, \sum_{k:\phi(k)=j} d_k \leq 32 d_j$. Again, this is possible since $D(J_2) < 32D(J_1)$. Also note for any $j \in J_2$ and $k \in J_1$, we have $d(j,k) \leq 4$. Furthermore, as in the proof of Claim 4.2, our algorithm makes sure that the client set J_1 gets added to C_{blue} or C_{black} (in lines 25 and 31). Therefore, these clients in J_1 will never be images of ϕ again, thus completing the proof. □

We will now show that the sets $\{T_\ell\}$ in the family $\mathcal{T}$ form 4-complete neighborhoods supported by the corresponding client-sets $\{J_\ell\}$.

Lemma 4.4. *Consider an iteration when a new set T_ℓ is added to $\mathcal{T}$ in line 29. The set T_ℓ is a 4-complete neighborhood supported by the set of clients J_ℓ (defined in line 31).*

Proof. Firstly, the diameter of the new set is at most 4, since every $i \in T_\ell$ is 2 hops from the i^* facility identified in line 5. To complete the proof, we show that T_ℓ is supported by the set J_ℓ defined in line 31, which is same as J_1. We establish this by showing that $\Gamma_G(J_1) \subseteq T_\ell$ (recall definition 2).

To this end, consider a client $j \in J_1$. At the beginning of this iteration, j is client in H. We claim that at the beginning of this iteration, $\Gamma_H(j) = \Gamma_G(j)$ (i.e., no neighboring facility has already been assigned in earlier iterations). Indeed, suppose not, and let i be some facility which is present in $\Gamma_G(j)$ but not in $\Gamma_H(j)$. We first observe that i cannot be in $L(\mathcal{T})$ as that would violate invariant I1 at the beginning of this iteration — (i,j) would form the violated pair. Similarly, we note that i cannot be added to $\mathcal{S}$ in line 13 in an earlier iteration — because then the distance between i^* and $R(\mathcal{S})$ would be at most 6 (via the path $i^* \to j \to i \to R(\mathcal{S})$), so this is a contradiction to the fact that the algorithm is in the branch executing line 29. Finally, we note that i cannot be added to $\mathcal{S}$ in line 23 in an earlier iteration, as that would violate invariant I2 at the beginning of this iteration — again (i,j) would form the violated pair. So we can conclude that $\Gamma_H(j) = \Gamma_G(j)$ and thus that the entire neighborhood of j is contained in $\{i^*\} \cup A$ which is added to T_ℓ in this iteration. Repeating this argument for all j shows that $\Gamma_G(J_1) \subseteq T_\ell$. □

We now turn our attention to proving that the sets in $\mathcal{S}$ are locally roundable. Toward this end, we begin with the following useful claim.

Claim 4.5. *Consider an iteration when a new set S_k is added to $\mathcal{S}$ in line 13. We then have that $\sum_{j\in C}\sum_{i\in S_k}\sum_{p\in[P]} d_j x_{ijp} \geq 16 \cdot \max_{i\in S_k, p\in[P]} c_{\text{eff}}(i,p)$*

Proof. By the definition of $(i^\star, p^\star)$ (line 5), we know that $\max_{i\in S_k,p\in[P]} c_{\text{eff}}(i,p) = c_{\text{eff}}(i^*,p^*)$ in the iteration when S_k was added to $\mathcal{S}$. Furthermore, by definition,

$$c_{\text{eff}}(i^*,p^*) = \sum_{j\in H} d_j \frac{x_{i^*jp^*}}{y_{i^*p^*}} \leq \sum_{j\in\Gamma_H(i^*)} d_j = D(J_1)$$

The inequality follows because we know (a) that $x_{ijp^*} > 0$ only for $j \in \Gamma_H(i^*)$ (i.e., only clients which are neighboring i^* can be serviced by i^*), and (b) that $x_{i^*jp^*} \leq y_{i^*p^*}$ (using inequality (**??**)). Now note that for any $j \in J_2$, since $j \in H$, we have that $\sum_{i\in L(\mathcal{S}),p\in[P]} x_{ijp} < 1/2$ (otherwise it would have been added to C_{blue} in an earlier iteration and deleted from H), and so $\sum_{i\in S_k,p\in[P]} x_{ijp} \geq 1/2$ (because S_k includes all the neighbors of j not already in $L(\mathcal{S})$, and j has no edge to any facility in $L(\mathcal{T})$ even in the original graph G by invariant I1, so the fractional demand from j to vertices in $L(\mathcal{T})$ is 0). Therefore,

$$D(J_2) = \sum_{j\in J_2} d_j \leq 2\sum_{j\in J_2} d_j \left(\sum_{i\in S_k,p\in[P]} x_{ijp}\right)$$

Since $D(J_2) \geq 32D(J_1) \geq 32c_{\text{eff}}(i^*,p^*)$, the claim follows. □

Claim 4.6. *At the end of the algorithm, for all $j \in C_{\mathsf{blue}}$, $\sum_{i\in\mathcal{S},p\in[P]} x_{ijp} \geq \frac{1}{2}$.*

Proof. This follows because we only add clients to C_{blue} when their fractional allocation to $\mathcal{S}$ exceeds $\frac{1}{2}$. □

Lemma 4.7. *Each set $S_k \in \mathcal{S}$ is a $(20, 32)$-roundable set.*

Proof. (Diameter) We claim that $\text{diam}(S_k) \leq 20$ for every $S_k \in \mathcal{S}$. We show by induction that for each $S_k \in \mathcal{S}$, $\text{dist}_G(i, i_k) \leq 10$ for every $i \in S_k$, where i_k is the root of S_k. When we add a new set S to $\mathcal{S}$ (in Line 13), S is the set $(i^\star \cup A \cup B)$ (here, we are using the notation in Algorithm 1). Clearly, $\text{dist}_G(i, i^\star) \leq 4$ for any $i \in S$. Now, consider the case when we augment an existing set in $\mathcal{S}$ (as in Line 23). Again, using the notation in the algorithm, suppose $i_k = \arg\min_{i\in R(\mathcal{S})} \text{dist}_G(i, i^\star)$, and let i_k be the root of $S_k \in \mathcal{S}$. Then, $\text{dist}_G(i^\star, i_k) \leq 8$. Since $\text{dist}_G(i^\star, i') = 2$ for any $i' \in A$, we see that $\text{dist}_G(i_k, i') \leq 10$ for any $i' \in A$. So the desired claim follows by induction.

(Roundability) We now show that there is a rounding of y^{int}_{ip} values for $i \in S_k$ such that

1. $\sum_{q\geq p}\sum_{i\in S_k} y^{\mathsf{int}}_{iq} \leq \lfloor \frac{1}{2}\sum_{q\geq p}\sum_{i\in S} y_{iq} \rfloor$, and
2. $\sum_{j\in C} d_j \sum_{i\in S_k,p\in[P]} x_{ijp} \leq b \cdot \sum_{i\in S}\sum_{p\in[t]} c_p y^{\mathsf{int}}_{ip}$

To do so, we first claim

Claim 4.8. $\sum_{j\in C}\sum_{i\in S_k}\sum_{p\in[P]} d_j x_{ijp} \geq 16 \max_{i\in S_k,p\in[P]} c_{\text{eff}}(i,p)$.

Proof. When the set S_k is first formed (Line 13), this follows from Claim 4.5. Let (i_k, p_k) be the pair with the highest effective capacity which was selected in the iteration when S_k was formed. Then i_k is the root of p_k, and $c_{\text{eff}}(i_k, p_k) = \max_{i\in S_k,p\in[P]} c_{\text{eff}}(i,p)$. Henceforth, $c_{\text{eff}}(i_k, p_k)$ does not change. Further, $c_{\text{eff}}(i_k, p_k) \geq c_{\text{eff}}(i,p)$ for any $i \in H, p \in [P]$. Since $c_{\text{eff}}(i,p)$ can never increase as the algorithm progresses, this inequality will hold from this iteration onwards. Therefore, even when we add more facilities to S_k later in the algorithm (line 23), the RHS of the inequality in the statement of the Claim does not change. Observe that the LHS can only increase since the set S_k can grow during the algorithm. □

Now we are armed to prove the roundability property. Define $A_u := \{(i,u), i \in S_k, p \in [P] : c_{\text{eff}}(i,p) \in [2^u, 2^{u+1})\}$ and let $\max_{i \in S_k, p \in [P]} c_{\text{eff}}(i,p) \in [2^U, 2^{U+1})$. Define $\alpha_u := \sum_{(i,p) \in A_u} y_{i,p}$. Clearly, $|A_u| \geq \alpha_u$. Therefore, we can choose a subset F_u of size $\lfloor \alpha_u/2 \rfloor$ from A_u. For each u and for each $i \in F_u$, set $y^{\text{int}}_{i2^u} = 1$. For every other (i,p), set $y^{\text{int}}_{ip} = 0$. We claim that y^{int} satisfies the two conditions of the roundability property.

We check Condition 1 first. Let $p \in [P]$ and let s be the index such that $2^s < p \leq 2^{s+1}$. Then

$$\begin{aligned}
\sum_{q \geq p} \sum_{i \in S_k} y^{\text{int}}_{iq} &= \sum_{u \geq s+1} \sum_{i \in S_k} y^{\text{int}}_{i2^u} = \sum_{u \geq s+1} \lfloor \alpha_u/2 \rfloor \leq \lfloor \frac{1}{2} \cdot \sum_{u \geq s+1} \alpha_u \rfloor \\
&= \lfloor \frac{1}{2} \sum_{u \geq s+1} \sum_{(i,q) \in A_u} y_{iq} \rfloor \leq \lfloor \frac{1}{2} \cdot \sum_{q : c_q \geq 2^{s+1}} \sum_{i \in S_k} y_{iq} \rfloor \\
&\leq \lfloor \frac{1}{2} \cdot \sum_{q \geq p} \sum_{i \in S} y_{iq} \rfloor,
\end{aligned}$$

where we have used the fact that $c_q \geq c_{\text{eff}}(i,q)$ for any i, q. Also note that a similar argument yields $\sum_{i \in S, p \in [P]} c_p y^{\text{int}}_{i,p} = \sum_{u=0}^{U} 2^u \lfloor \alpha_u/2 \rfloor$.

We now need to prove condition 2 is satisfied. Call the parameter q *good* if $\alpha_q \geq 2$ and bad otherwise. Note that if q is good, then $\alpha_u \leq 4\lfloor \alpha_u/2 \rfloor$. Let D denote the total fractional demand assigned to S_k, i.e., $\sum_{j \in C} d_j \sum_{i \in S_k, p \in [P]} x_{ijp}$. Claim 4.8 shows that $D \geq 16 \cdot 2^U$. From the definition of $c_{\text{eff}}(\cdot)$, we get

$$\begin{aligned}
D := \sum_{j \in C} d_j \sum_{i \in S_k, p \in [P]} x_{ijp} &\leq \sum_{i \in S_k, p \in [P]} c_{\text{eff}}(i,p) y_{i,p} \leq 2 \sum_{u=0}^{U} 2^{u+1} \sum_{(i,p) \in A_u} y_{ip} \\
&\leq 8 \sum_{u\text{: bad}} 2^u + 16 \sum_{u\text{: good}} 2^{u-2} \alpha_u \leq 8 \cdot 2^U + 16 \sum_{u\text{:good}} 2^u \lfloor \alpha_u/2 \rfloor \\
&\leq D/2 + 16 \sum_{i \in S, p \in [P]} c_p y^{\text{int}}_{ip}
\end{aligned}$$

Therefore, we get $D \leq 32 \sum_{i \in S, p \in [P]} y^{\text{int}}_{ip}$. Therefore, S_k has the $(20, 32)$-roundability property. □

This completes the proof of Theorem 4.1.

5 LP-Based Reduction to Max-Min Allocation

Ellipsoid, Round and Cut,

6 Approximate Supply Polyhedra for $Q||C_{min}$

Let the instance $\mathcal{I}$ of $Q||C_{min}$ have m machines M with demands $D_1 \geq \cdots \geq D_m$ and n types of jobs J with capacities $c_1 \geq \cdots \geq c_n$. A supply vector $(s_1, \ldots, s_n)$ indicates the number of jobs of each type available; a supply vector is feasible if together they can satisfy all the demands. We wish to find a convex set/polyhedra which captures all the feasible supply vectors. In particular, any feasible supply vector should be in the set, and given any (integer) supply vector in the set there should be an allocation which satisfies the demands to an α-factor.

A feasible supply vector $(s_1, \ldots, s_n)$ must lie in the following polytope.

$$\begin{aligned}
\mathcal{P}_{\mathsf{ass}} = \{(s_1, \ldots, s_n) : & \\
\forall j \in J, \quad & \textstyle\sum_{i \in M} z_{ij} \leq s_j && (1)\\
\forall i \in M, \quad & \textstyle\sum_{j \in C} z_{ij} \min(c_j, D_i) \geq D_i && (2)\\
\forall i \in M, j \in J, \quad & z_{ij} \geq 0\} && (3)
\end{aligned}$$

Not all integral $(s_1, \ldots, s_n) \in \mathcal{P}_{\mathsf{ass}}$ need be feasible; but the following theorem shows given such a supply vector, there exists an assignment satisfying the demands up to a factor 2.

Theorem 6.1. *Given $(s_1, \ldots, s_n) \in \mathcal{P}_{\mathsf{ass}}$, there is an of assignment ϕ of the s_j jobs of capacity c_j to the machines such that for all $i \in M$, $\sum_{j:\phi(j)=i} c_j \geq D_i/2$.*

Proof. For simplicity, given the supply vector, abusing notation let J denote the multiset of jobs where job j appears s_j times. We know that the LP(10)-(12) is feasible with the s_j replaced by 1. Let $N = \sum_j s_j$.

The algorithm is a very simple greedy algorithm which doesn't look at the LP solution. Order the jobs (with multiplicities) in decreasing order of capacities $c_1 \geq c_2 \geq \cdots \geq c_N$, and order the machines in decreasing order of D_i's, that is, $D_1 \geq D_2 \geq \ldots \geq D_m$. Starting with machine $i = 1$ and job $j = 1$, assign jobs j to i if the total capacity filled in machine i is $< D_i/2$ and move to the next job. Otherwise, call machine i happy and move to the next machine. Obviously, if all machines are happy at the end we have found our assignment.

The non-trivial part is to prove that if some machine is unhappy, then the LP(10)-(12) is infeasible (with s_j replaced by 1). To do so, we take the Farkas dual of the LP; the following LP is feasible iff LP(10)-(12) is infeasible.

$$\begin{aligned}
& \textstyle\sum_{i=1}^{m} \beta_i D_i > \sum_{j=1}^{n} \alpha_j && (4)\\
\forall i \in M, j \in J \quad & \beta_i \min(c_j, D_i) \leq \alpha_j && (5)\\
\forall i \in M, \quad & \beta_i \geq 0 && (6)
\end{aligned}$$

Suppose machine $i^\star$ is the first machine which is unhappy. Let $S_1, \ldots, S_{i^\star - 1}$ be the jobs assigned to machines 1 to $(i^\star - 1)$ and $S_i^\star$ be the remainder of jobs. We have $\sum_{j \in S_i^\star} c_j < D_i^\star/2$. We also have for all $1 \leq i \leq i^\star$, $\sum_{j \in S_i} \min(c_j, D_i) \leq D_i$. We now describe a feasible solution to (4)-(6).

Given the assignment S_i's, call a machine i *overloaded* if S_i contains a single jobs j_i with $c_{j_i} \geq D_i$. We let $\beta_1 = 1$. For $1 \leq i < i^\star$, we have the following three-pronged rule

- If $i+1$ is not overloaded, $\beta_{i+1} = \beta_i$.
- If $i+1$ is overloaded, and so is i, then $\beta_{i+1} = \beta_i \cdot D_i/D_{i+1}$.
- If $i+1$ is overloaded but i is not, then $\beta_{i+1} = \beta_i \cdot c_{j_{i+1}}/D_{i+1}$, where j_{i+1} is the job assigned to $i+1$.

For any job j assigned to machine i, we set $\alpha_j = \beta_i \min(c_j, D_i)$. Since for any S_i, we have $\sum_{j \in S_i} \min(c_j, D_i) \leq D_i$ and $\sum_{j \in S_i^\star} c_j < D_i^\star/2$, the given (α, β) solution satisfies (4). We now prove that it satisfies (5). From the construction of the β's the following claims follow.

Claim 6.2. $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_m$.

Claim 6.3. $\beta_1 D_1 \geq \beta_2 D_2 \geq \cdots \geq \beta_m D_m$.

Proof. The only non-obvious case is if $i+1$ is overloaded but i is not: in this case $\beta_{i+1} D_{i+1} = \beta_i c_{j_{i+1}}$. But since i is not overloaded, let j be some job assigned to i with $c_j \leq D_i$. By the greedy rule, $c_j \geq c_{j_{i+1}}$, and so $\beta_{i+1} D_{i+1} \leq \beta_i D_i$. □

Now fix a job j and let i be the machine it is assigned to. Note (5) holds for (i, j) and we need to show (5) holds for all (i', j) too. I don't see any more glamorous way than case analysis.

Case 1: $c_j \leq D_i$**.** In this case $\alpha_j = \beta_i c_j$ and i is not overloaded. Let $i' < i$. Then we have $\beta_{i'} \min(c_j, D_{i'}) \leq \beta_{i'} c_j \leq \beta_i c_j$, where the last inequality follows from Claim 6.2.

Now let $i' > i$. If $c_j \leq D_{i'}$, then none of the machines from i to i' can be overloaded. Therefore, $\beta_{i'} = \beta_i$, and so $\beta_{i'} c_j = \beta_i c_j = \alpha_j$. So, we may assume $c_j > D_{i'}$ and we need to upper bound $\beta_{i'} D_{i'}$. Let $i'' > i$ be the first machine which is overloaded with job j'' say. By Claim 6.3, we have $\beta_{i'} D_{i'} \leq \beta_{i''} D_{i''}$. Now note that $\beta_{i''} D_{i''} = \beta_{i''-1} c_{j''} = \beta_i c_{j''} \leq \beta_i c_j = \alpha_j$ where the second equality follows since none of the machines from i to $i''-1$ were overloaded.

Case 2: $c_j > D_i$**.** In this case $\alpha_j = \beta_i D_i$ and i is overloaded. Let $i' > i$. Then, $\beta_{i'} \min(c_j, D_{i'}) = \beta_{i'} D_{i'} \leq \beta_i D_i$ where the last inequality follows from Claim 6.3.

Let $i' < i$. Let $i' \leq i'' < i$ be the smallest entry such that $c_j > D_{i''}$. Note that all machines from i'' to i must be overloaded implying $\beta_{i''} D_{i''} = \beta_i D_i$. Since $c_j \leq D_{i'}$ (in case $i' < i''$), we need to upper bound $\beta_{i'} c_j$. By Claim 6.2, $\beta_{i'} c_j \leq \beta_{i''-1} c_j$. Now, if $i''-1$ were overloaded, then $\beta_{i''} D_{i''} = \beta_{i''-1} D_{i''-1} \geq \beta_{i''-1} c_j$ where the last inequality follows from definition of i''. Together, we get $\beta_{i'} c_j \leq \beta_i D_i$. □

Lemma 6.4. *Suppose* $(y_1, \ldots, y_n) \in \mathcal{P}_{\mathsf{ass}}$*. Let* $(\bar{y}_1, \ldots, \bar{y}_n)$ *be a vector such that for all* $1 \leq i \leq n$, $\sum_{j \leq i} \bar{y}_j \geq \sum_{j \leq i} y_i$*. Then* $(\bar{y}_1, \ldots, \bar{y}_n) \in \mathcal{P}_{\mathsf{ass}}$.

Proof. By induction, let us assume the lemma is true for all $\bar{y}$ with $\bar{y}_1 = y_1$ which satisfies the prefix-sum condition. Let $\bar{\bar{y}}$ be the vector with $\bar{\bar{y}}_1 = y_1$, $\bar{\bar{y}}_2 = \bar{y}_2 + \bar{y}_1 - y_1$, and $\bar{\bar{y}}_i = \bar{y}_i$ otherwise. Since $\bar{\bar{y}} \in \mathcal{P}_{\mathsf{ass}}$, there is an assignment z_{ij} satisfying (10)-(12) with $s_j = \bar{\bar{y}}_j$. We now describe a feasible solution $\bar{z}_{ij}$ with $s_j = \bar{y}_j$.

Let $\theta := \bar{y}_2 / \bar{\bar{y}}_2 \leq 1$ since $\bar{y}_1 \geq y_1$. Define $\bar{z}_{i2} = \theta z_{i2}$ for all i, and define $\bar{z}_{i1} = z_{i1} + (1-\theta) z_{i2}$. For $j = 2$, we have $\sum_{i \in M} \bar{z}_{i2} = \theta \sum_{i \in M} z_{i2} \leq \theta \bar{\bar{y}}_2 = \bar{y}_2$. For $j = 1$, we have $\sum_{i \in M} \bar{z}_{i1} = \sum_{i \in M} z_{i1} + (1-\theta) \sum_{i \in M} z_{i2} \leq \bar{\bar{y}}_1 + (1-\theta)\bar{\bar{y}}_2 = \bar{\bar{y}}_1 + \bar{\bar{y}}_2 - \bar{y}_2 = \bar{y}_1$. Since the other z_{ij}'s and $\bar{y}_j$'s are untouched, $\bar{z}_{ij}$ satisfies (10) with $\bar{y}_j$'s.

Now fix a machine i. The 'increase' in the LHS of (11) is $\sum_{j \in J} (\bar{z}_{ij} - z_{ij}) \min(c_j, D_i) = (1-\theta) z_{i2} \min(c_1, D_i) - (1-\theta) z_{i2} \min(c_2, D_i) \geq 0$ since $c_1 \geq c_2$.

□

7 Approximate Supply Polyhedra for $Q|f_i|C_{min}$

Let the instance $\mathcal{I}$ of $Q||C_{min}$ have m machines M with demands $D_1 \geq \cdots \geq D_m$ and cardinality constraints $f_1, \ldots, f_m$, and n types of jobs J with capacities $c_1 \geq \cdots \geq c_n$. We assume $D_1/D_m \leq n^C$ [deepc: i think this can be made wlog with some std trick] A supply vector $(s_1, \ldots, s_n)$ indicates the number of jobs of each type available; a supply vector is feasible if together they can satisfy all the demands. An α-approximate supply polyhedra $\mathcal{P}$ has the following properties: any feasible supply vector lies in $\mathcal{P}$, and given any integral vector $(s_1, \ldots, s_n) \in \mathcal{P}$ there is an allocation of the s_j jobs of capacity c_j which satisfies every demand up to an α-factor. To Do!

A feasible supply vector $(s_1, \ldots, s_n)$ must lie in the following polytope. Let Supp be a set indicating infinitely many copies of all jobs. For every machine i, let $\mathcal{F}_i := \{S \in \mathsf{Supp} : |S| \leq f_i \text{ and } \sum_{j \in S} c_j \geq D_i\}$ denote all the feasible sets that can satisfy machine i. Let $n(S, j)$ denote the number of copies of job of type j.

$$\begin{aligned} \mathcal{P}_{\mathsf{conf}} = \{(s_1, \ldots, s_n) : & \\ \forall i \in M, \quad & \textstyle\sum_{S \in \mathcal{F}_i} z(i, S) = 1 && (7) \\ \forall j \in J, \quad & \textstyle\sum_{i \in M, S \in \mathcal{F}_i} z(i, S) n(S, j) \leq s_j && (8) \\ \forall i \in M, S \in \mathcal{F}_i, \quad & z(i, S) \geq 0\} && (9) \end{aligned}$$

Theorem 7.1. *Given $(s_1, \ldots, s_n) \in \mathcal{P}_{\mathsf{conf}}$, there is an of assignment ϕ of the s_j jobs of capacity c_j to the machines such that for all $i \in M$, $\sum_{j:\phi(j)=i} c_j \geq D_i/\alpha$ for $\alpha = O(\log n)$.*

Proof. Given a feasible fractional solution $\{z(i, S)\}$ to the configuration LP above, we want to efficiently round it to obtain an integer solution which is an α-approximation for the given $Q|k_i|C_{min}$ instance.

We call a job of capacity c_j *large* for machine i if $c_j \geq \frac{D_i}{16C \log n}$, otherwise it is said to be *small* for machine i. For a machine i, we define a relaxed collection of feasible sets $\mathcal{F}_i^{\mathsf{rel}}$ where $S \in \mathcal{F}_i^{\mathsf{rel}}$ if either (a) $S = \{j\}$ and j is large for i, or (b) $c_j < \frac{D_i}{8C \log n}$ for all $j \in S$, $|S| \leq f_i$, and $\sum_{j \in S} c_j \geq D_i/2$.

Partitioning Configurations and Bucketing Demands. Our first step is to modify z such that its support $z(i, S) > 0$ for only $S \in \mathcal{F}_i^{\mathsf{rel}}$ for all i. For every machine i, if $z(i, S) > 0$ and S contains any large job j for i, then we replaces S by $\{j\}$. To be precise, we set $z(i, \{j\}) = z(i, S)$ and $z(i, S) = 0$. We call such singleton configurations *large* for i; all others are small. Note that after this step, $z(i, S) > 0$ only for $S \in \mathcal{F}_i^{\mathsf{rel}}$. Let $\mathcal{F}_i^L$ be the collection of large configurations for i; the rest $\mathcal{F}_i^S$ being small configurations. Define $z^L(i) := \sum_{S \in \mathcal{F}_i^L} z(i, S)$ be the total large contribution to i, and let $z^S(i) := 1 - z^L(i)$ the small contribution.

The next step of our algorithm is to partition the demands into buckets depending on their requirement values D_i. To this end, we say that demand i belongs to *bucket* t if $2^{t-1} \leq D_i < 2^t$ (we assume wlog by scaling that the smallest demand is 1). We let $B^{(t)}$ to denote the bucket t. Note that the number of buckets $K \leq C \log n$; this drives the approximation factor. We make one observation.

Claim 7.2. *For any t, let i and i' be two machines in $B^{(t)}$ and let $f_i \leq f_{i'}$. Let $z(i, T) > 0$ for some small configuration for i. Then $T \in \mathcal{F}_{i'}^{\mathsf{rel}}$ and $\sum_{j \in T} c_j \geq D_{i'}/2$.*

Proof. Note that since $z(i, T) > 0$, we have $\sum_{j \in T} c_j \geq D_i \geq 2^{t-1} \geq D_{i'}/2$. Furthermore, for any $j \in T$, we have $c_j \leq \frac{D_i}{8C \log n} \leq \frac{2^t}{8C \log n}$. Therefore any other machine $i' \in B^{(t)}$, T satisfies two conditions of being in $\mathcal{F}_{i'}^{\mathsf{rel}}$. Now if $f_{i'} \geq f_i$, we get $|T| \leq f_{i'}$ as well. □

Before describing our subroutines, we make a few definitions. All of these are with respect to a solution z. A machine i is called *rounded* if there exists $S \in \mathcal{F}_i^{\mathsf{rel}}$ with $z(i, S) = 1$. We let $\mathcal{R}$ denote the rounded demands. The remaining machines are of three kinds: *large* ones with $z^L(i) = 1$, *hybrid* ones with $z^L(i) \in (0, 1)$ and *small* ones with $z^L(i) = 0$. Let $\mathcal{L}, \mathcal{H}, \mathcal{S}$ denote these respectively.

Subroutine: FixBucket(t). This takes a bucket t with more than one hybrid machine, and modifies the z-solution such that there is at most one hybrid machine in t. Other machines in other buckets are unaffected.

Among the hybrid machines $B^{(t)}$, let i be the one with the smallest f_i. Let i' be any other hybrid machine. We know there is at least one more. We now *modify* z as follows. Since $z^L(i') > 0$, there exists a large configuration $\{j'\}$ for i' with $z(i', \{j'\}) > 0$. Similarly, since $z^L(i) < 1$, there must exist a *small* configuration $T \in \mathcal{F}_i^{\mathsf{rel}}$ such that $z(i, T) > 0$. By Claim 7.2, note that $T \in \mathcal{F}_{i'}^{\mathsf{rel}}$ as well. We then perform the following change: increase $z(i, \{j'\})$ and $z(i', T)$ by δ, and decrease $z(i', \{j'\})$ and $z(i, T)$ by δ, for a $\delta > 0$ such that one of the variables becomes 0 or 1. Note that this keeps (7) and (8) maintained.

We keep on performing this process as long as possible; since we always transfer large configuration assignments to demands which appear earlier in the total order, this process will stop at some point. At this point, we add whichever demands are integrally assigned by large configurations to the set $\mathcal{R}$, i.e., all $i \in B^{(t)}$ for which $z(i, S) = 1$ for some $S \in \mathcal{F}_i^{\mathsf{rel}}$ are added to $\mathcal{R}$.

Claim 7.3. *Fix Bucket on t produces a solution with at most one hybrid machine in $B^{(t)}$.*

Subroutine: FixLargeMachine(i). This takes input a large machine $i \in B^{(t)} \setminus \mathcal{R}$ with $z^L(i) = 1$ and modifies z such that at the end i enters $\mathcal{R}$. Since $i \notin \mathcal{R}$ in the beginning, there must exist then two large configurations with $z(i, \{j_1\}) \in (0, 1)$ and $z(i, \{j_2\}) \in (0, 1)$. Let j_1 be the job with the smallest capacity among all large configurations $(i, \{j\})$ with $z(i, \{j\}) > 0$. Two cases arise. In the simple case, there exists no $i' \notin \mathcal{R}$ and $S' \in \mathcal{F}_{i'}^{\mathsf{rel}}$ with $z(i', S') > 0$ and $j_1 \in S'$. That is, no other machine fractionally claims the job j_1. Since s_{j_1} is an integer, we have slack in (8) and therefore we can round up $z(i, \{j_1\}) = 1$ (zeroing out all other i's $z(i, S)$'s) without violating (8). We then add $(i, \{j_1\})$ to $\mathcal{R}$.

Otherwise, there exists a machine i' (which could be in a different bucket) and a set $S \in \mathcal{F}_{i'}^{\mathsf{rel}}$ such that $z(i', S) \in (0, 1)$ and $j_1 \in S$. Now define the set T as follows. If $c_{j_2} > \frac{D_{i'}}{8C \log n}$, then $T = \{j_2\}$; otherwise $T = S - j_1 + j_2$. Note that in either case $T \in \mathcal{F}_{i'}^{\mathsf{rel}}$. In the first case, j_2 is large for i'. In the second case, $|T| = |S|$ and $\sum_{j \in T} c_j \geq \sum_{j \in S} c_j$ since $c_{j_2} \geq c_{j_1}$ by choice of j_1.

We modify z-as follows. We decrease $z(i, \{j_2\})$ and $z(i', S)$ by δ and increase $z(i, \{j_1\})$ and $z(i', T)$ by δ till one of the values becomes 0 or 1. As before, this preserves the LHS of (7) and can only decrease the LHS of (8) (for jobs $j \in S \setminus j_1$ if $T = \{j_2\}$). This process ends with assigning $z(i, \{j_1\}) = 1$ and we add $(i, \{j_1\})$ to $\mathcal{R}$.

Claim 7.4. *Subroutine Fix Large Machine i modifies the LP solution and adds i to $\mathcal{R}$.*

Note that Fix Large Machine can make a small machine i' hybrid or large for its bucket since $z^L(i')$ could potentially increase. We run the following while loop in Step 1 of te algorithm.

Step 1: Taking care of large machines

While $\mathcal{L}$ is non-empty:

- If $i \in \mathcal{L}$, then Fix-Large-Machine(i). Note that i enters $\mathcal{R}$ after this. This can increase the number of hybrid machines across buckets.
- For all $1 \leq t \leq K$, if $B^{(t)}$ contains more than one hybrid machine, then Fix-Bucket(t). This can increase the number of machines in $\mathcal{L}$.

The above while loop terminates in at most m iterations, since the first bullet point adds a machine to $\mathcal{R}$.

Claim 7.5. *At the end of* **Step 1**, *we have for every bucket t, at most one $i \in B^{(t)} \setminus \mathcal{R}$ has $z^L(i) \in (0, 1)$ and the rest have $z^S(i) = 1$. Furthermore, for every $i \in \mathcal{S}$ and $z(i, S) > 0$, we have $\sum_{j \in S} c_j \geq D_i/2$.*

Step 2: Taking care of hybrid machines. Let $\mathcal{H}$ be the set of hybrid machines at this point. We know that $|\mathcal{H}| \leq K \leq C \log n$ since each bucket has at most one hybrid machine. For any machine $i \in \mathcal{H}$ with $z^L(i) \leq 1 - 1/K$, we zero-out all its large contribution. More precisely, for all j large for i we set $z(i, \{j\}) = 0$. Note that (7) is no longer true, but it holds with RHS $\geq 1/K$. Note that these machines enter $\mathcal{S}$.

At this point, we have $\mathcal{H}$ where every $i \in \mathcal{H}$ has $z^L(i) > 1 - 1/K$. Let $K' := |\mathcal{H}|$. Let J' be the set of jobs j which are large for some machine $i \in \mathcal{H}$ and $z(i, \{j\}) > 0$. Let G be a bipartite graph with $\mathcal{H}$ on one side and J' on the other and we draw an edge (i, j) iff j is large for i.

Claim 7.6. *There is a matching in G saturating all $i \in \mathcal{H}$.*

Proof. Pick a subset $\mathcal{H}' \subseteq \mathcal{H}$ and let J'' be its neighborhood in G. We need to show $\sum_{j \in J''} s_j \geq \mathcal{H}'$. Since z satisfies (8), we get

$$\sum_{j \in J''} s_j \geq \sum_{j \in J''} \sum_{i \in \mathcal{H}'} z(i, \{j\}) = \sum_{i \in \mathcal{H}'} \sum_{j \in J''} z(i, \{j\}) > (1 - 1/K)|\mathcal{H}'| \geq |\mathcal{H}'| - 1$$

The inequality follows since J'' is the neighborhood of $\mathcal{H}'$ and the fact that $z^L(i) > 1 - 1/K$ for all $i \in \mathcal{H}$. The claim follows since s_j's are integers. □

If machine $i \in \mathcal{H}$ is matched to job j, then we assign i this job and add i to $\mathcal{R}$. Let $J_M \subseteq J'$ be the subset of jobs allocated; note $|J_M| \leq K$. After this point we have only small machines remaining. For every $i \in \mathcal{S}$ and every small configuration S with $z(i, S) > 0$, we move this mass to $z(i, S \setminus J_M)$. Note that $\sum_{j \in S \setminus J_M} c_j \geq \sum_{j \in S} c_j - |J_M| \cdot \frac{D_i}{8C \log n} \geq 3D_i/8$ where we use the fact that $\sum_{j \in S} c_j \geq D_i/2$ (by Claim 7.5) and $K \leq C \log n$.

Claim 7.7. *At the end of* **Step 2**, *we have a set of residual machines $\mathcal{S}$ and a set of residual jobs J_{res} and a solution $z(i, S)$ where*

1. *For all $i \in \mathcal{S}$ we have $z(i, S) > 0$ iff $|S| \leq f_i$, $\sum_{j \in S} c_j \geq 3D_i/8$, and $c_j < \frac{D_i}{8C \log n}$ for all $j \in S$.*
2. *$\forall i \in \mathcal{S}$, $1 \geq \sum_S z(i, S) \geq 1/K \geq \frac{1}{C \log n}$.*
3. *$\forall j \in J_{res}$, $\sum_i z(i, S) n(S, j) \leq s_j$.*

Step 3: Taking care of Small Machines. We convert the LP solution in Claim 7.7 to an assignment LP solution in the standard way. For every $i \in \mathcal{S}$ and $j \in J_{res}$ define $z_{ij} = \sum_S z(i, S) n(S, j)$. Note that this satisfies the constraint of the assignment LP:

$$\begin{aligned} \forall j \in J_{res}, \quad & \textstyle\sum_{i \in \mathcal{S}} z_{ij} \leq s_j \\ \forall i \in \mathcal{S}, \quad & \textstyle\sum_{j \in J_{res}} z_{ij} c_j \geq \frac{3D_i}{8C \log n} \\ \forall i \in \mathcal{S}, \quad & \textstyle\sum_{j \in J_{res}} z_{ij} \leq f_i \\ \forall i \in \mathcal{S}, j \in J_{res} \text{ with } c_j \geq \frac{D_i}{8C \log n}, \quad & z_{ij} = 0 \end{aligned}$$

The last equality follows from point 1 of Claim 7.7. The first inequality follows from point 3 of Claim 7.7. To see the second and third point, note that for any $i \in \mathcal{S}$,

$$\sum_{j \in J_{res}} z_{ij} c_j = \sum_j \sum_S z(i, S) n(S, j) c_j = \sum_S z(i, S) \sum_j n(S, j) c_j \geq \frac{1}{C \log n} \cdot \frac{3D_i}{8}$$

and,

$$\sum_{j \in J_{res}} z_{ij} = \sum_j \sum_S z(i, S) n(S, j) = \sum_S z(i, S) \sum_j n(S, j) \leq f_i$$

since for any S, $\sum_{j \in S} n(S, j) \leq f_i$.

Now we use Theorem 7.8 to find an allocation of J_{res} to machines $\mathcal{S}$ such that every machine gets capacity $\geq \frac{D_i}{4C \log n}$.

□

7.1 Assignment LP for $Q|f_i|C_{min}$

Suppose we are given m machines M with cardinality constraints $f_1, \ldots, f_m$, and n types of jobs J with capacities $c_1 \geq \cdots \geq c_n$. Let $(s_1, \ldots, s_n)$ be a supply vector, that is, there are s_j copies of job j. Suppose there exists a feasible solution to the following LP.

$$\forall j \in J, \quad \textstyle\sum_{i \in M} z_{ij} \leq s_j \tag{10}$$

$$\forall i \in M, \quad \textstyle\sum_{j \in J} c_j z_{ij} \geq D_i \tag{11}$$

$$\forall i \in M, \quad \textstyle\sum_{j \in J} z_{ij} \leq f_i \tag{12}$$

$$\forall i \in \mathcal{S}, j \in J_{res} \text{ with } c_j \geq C_i, \quad z_{ij} = 0 \tag{13}$$

Theorem 7.8. *If (10)-(13) is feasible, then there is an integral assignment z_{ij}^{int} which satisfies (10), (12) and (13), and* $\forall i \in M, \quad \sum_{j \in J} c_j z_{ij}^{\mathsf{int}} \geq D_i - C_i$.

Proof. NEEDS BETTER WRITING We repeat the argument of Shmoys and Tardos [**?**]. Form $\lfloor \sum_{j \in J} z_{ij} \rfloor \leq f_i$ copies of every machine; let N_i be the copies of machine i. Order the jobs with multiplicities s.t. $c_1 \geq c_2 \geq \cdots \geq c_N$ where $N = \sum_j s_j$. Modify z_{ij} to get an assigment z_{ij} for $i \in \cup N_i$ and $j \in [N]$ as follows. We do this for one machine i.

Given z_{ij}'s we form $|N_i| + 1$ groups $S_1, \ldots, S_{|N_i|}, S_{|N_i|+1}$ with $\sum_{j \in S_t} z_{ij} = 1$ for all $1 \leq t \leq |N_i|$ and $\sum_{j \in S_t} z_{ij} < 1$ for $t = |N_i| + 1$. Note that $\sum_{j \in S_t} z_{ij} c_j < c_{j'}$ for $j' \in S_{t-1}$. When we do this modification for all machines, we get a fractional matching solution where all the N_i copies get fractional value 1 but the jobs are at most 1. So, there is an integral matching. The total integral load on machine i is at least $\sum_{t>1} \sum_{j \in S_t} z_{ij} c_j \geq D_i - C_i$ since $z_{ij} = 0$ for $c_j > C_i$.

Cardinality constraint vacuously satisfied. □

8 QPTAS for $Q|k_i|C_{min}$

Let the instance $\mathcal{I}$ given to us have m machines with speeds $s_1, \ldots, s_m$ and cardinality upper bounds $k_1, \ldots, k_m$. We are also given n jobs with loads $c_1, \ldots, c_n$. We assume each $1 \leq c_i, s_j \leq \text{poly}(n)$. Fix an ε. Let us guess the optimum value $D \geq 1$. We wish to either prove D is too large a guess, or find an assignment which assigns machine i a subset of jobs S_i with *load* $\sum_{p \in S_i} c_p \geq D_i := Ds_i$.

We start with a lemma which states that finding solutions satisfying cardinality constraints approximately suffices.

Lemma 8.1. *Given an assignment of jobs such that the load on any machine i is at least $T_i(1-\epsilon_1)$ such that machine i gets $\leq (1+\epsilon_2)k_i$ jobs, we can find another assignment which satisfies the cardinality constraints and the load of any machine i is $\geq T_i(1-\epsilon_3)$ for $\epsilon_3 < 2\epsilon_1 + \epsilon_2$.*

Proof. For every machine i, let S_i be the jobs currently allocated to it. We may assume $\varepsilon_1 k_i \geq 1$, otherwise $|S_i| \leq k_i$. Remove the $\lfloor 2\varepsilon_1 k_i \rfloor \geq \varepsilon_1 k_i$ least capacity jobs to obtain the set S_i'. Note that the total capacity of S_i' is at least $(1-2\varepsilon_1)$ times capacity of S_i, and therefore at least $(1-2\epsilon_1-\epsilon_2)T_i$. □

Input Modification and Grouping. We now modify the data so that everything is rounded to the nearest power of $(1+\epsilon)$. More precisely we round k_i to the *smallest* power of $(1+\epsilon)$ larger than the original value and T_i to the largest power of $(1+\epsilon)$ smaller than the original value. If T was optimum, then there is a feasible allocation in the new instance. For technical reasons, we round c_p to the smallest value of the form $\epsilon(1+\epsilon)^t$ larger than the original value. Let J_p be the set of jobs with modified capacity $c_p = \varepsilon(1+\epsilon)^p$, and let $n_p = |J_p|$. Furthermore, armed with Lemma 8.1, any $(1-\epsilon)$-approximate solution to the new instance gives an $(1-O(\epsilon))$-approximate solution to the original instance.

We now divide the machines into groups. For $0 \leq r \leq O(\log n)$ and $0 \leq s \leq O(\log n)$, let $M^{(r,s)}$ be the number of machines with $T_i = (1+\epsilon)^r$ and $k_i = (1+\epsilon)^s$. Call a job p big for machine i if $c_p \geq \varepsilon T_i$. If $i \in M^{(r,s)}$, then p lies in the set $J_r \cup J_{r+1} \cup \cdots$. Otherwise, p is small for machine i. We define a bipartite graph H with jobs and machines on either side, with an edge (i,p) iff p is small for i.

For every $0 \leq r, s \leq O(\log n)$, we define a set of *feasible configurations* $\Phi^{(r,s)}$. These consist of vectors $\phi \in \mathbb{Z}^K_{\geq 0}$ for $K = O(1/\epsilon)$ corresponding to big jobs assigned to machines i in $M^{(r,s)}$. To be precise, ϕ_k is supposed to count the number of jobs with $c_p = \varepsilon(1+\epsilon)^{r+k}$ contained in the configuration ϕ. The last coordinate ϕ_K counts the number of jobs p with $c_p > (1+\epsilon)^r$. Let $\mathsf{cap}(\phi) := \sum_{k=0}^K \phi_k \epsilon (1+\epsilon)^{r+k}$ be the total load of the configuration and $|\phi| = \sum_{k=0}^K \phi_k$ be its cardinality. We let $\Phi^{(r,s)}$ be the collection of feasible minimal configurations, that is, ϕ's with (a) $|\phi| \leq (1+\epsilon)^s$ and (b) either $\mathsf{cap}(\phi) \leq (1+\epsilon)^r$ or $\mathsf{cap}(\phi) > (1+\epsilon)^r$ and $\mathsf{cap}(\phi') \leq (1+\epsilon)^r$ for any ϕ' obtained by decreasing any positive coordinate of ϕ by exactly 1. Note that $|\Phi^{(r,s)}| \leq N_0 = (1/\epsilon)^{(1/\epsilon)}$. Also note that in any optimal solution, each machine $i \in M^{(r,s)}$ does get one configuration from $\Phi^{(r,s)}$. Our algorithm constructs these classes and arbitrary numbers them. The tth member of $\Phi^{(r,s)}$ will be denotes as $\phi^{(r,s,t)}$.

Enumeration. For every $0 \leq r, s \leq O(\log n)$ and $1 \leq t \leq N_0$, we *guess* the integer $\mathbf{b}_t^{(r,s)} \in \mathbb{Z}_{\geq 0}$ which indicates the number of machines in $M^{(r,s)}$ who are allocated the configuration $\phi^{(t)}$. These guesses must satisfy

$$\forall 0 \leq r, s \leq O(\log n), \qquad \sum_{t=1}^{N_0} \mathbf{b}_t^{(r,s)} = |M^{(r,s)}| \tag{14}$$

The number of such guesses is $\leq \prod_{r,s} |M_{r,s}|^{N_0} \leq C_\varepsilon^{O(\log^3 n)}$. Since machines in $M^{(r,s)}$ are all equivalent (in terms of demand and cardinality constraint), by symmetry we can assign the $\mathbf{b}_t^{(r,s)}$ copies of $\phi(r,s,t)$ as we like. For a guess to be feasible, for every job of type p, at most n_p copies must be used up in the guessed configurations. For every guess we get a residual problem on the bipartite graph H. Let n_p' be the remaining number of jobs of type p. Let T_i' be the residual demand of machine i, that is, $T_i - \mathsf{cap}(\phi)$ where ϕ is allocated to it by the guess. Let k_i' be the residual cardinality constraint, that is, $k_i' = k_i - |\phi|$.

Rounding. The remaining copies of jobs must satisfy the residual demand. For this we simply write the assignment LP.

$$\forall p, \quad \sum_{i \sim p} z_{ip} \leq n'_p \tag{15}$$

$$\forall i \in [m], \quad \sum_{p \sim i} c_p z_{ip} \geq T'_i \tag{16}$$

$$\forall i \in [m], \quad \sum_{p \sim i} z_{ip} \leq k'_i \tag{17}$$

For the correct guess, the above LP is feasible. If so, using the fact that $c_p \leq T_i$ for all $i \sim p$, we can get a feasible solution with a slight hit in the demand.

Lemma 8.2. *If (10)-(12) is feasible, then there is an integral assignment z^{int}_{ip} which satisfies (15) and (17), and* $\forall i \in [m], \quad \sum_{p \sim i} c_p z^{\mathsf{int}}_{ip} \geq T'_i - O(\epsilon) T_i$.

Proof. The proof is via iterative rounding. We look at a vertex solution and if some z_{ip} is 0 or 1, we set z^{int}_{ip} the same. If for any machine i, the number of z_{ip}s is ≤ 5, we set $z^{\mathsf{int}}_{ip} = 0$ for these and delete the constraint in (16) and (17) corresponding to i. For this machine i, the integral assignment is fixed with cardinality $\leq k'_i$ and with total load $\geq T'_i - 5 \max_p c_p \geq T'_i - 5\varepsilon T_i$.

Therefore, we may assume that the vertex solution z has at least 5 non-zeros for every row i and at least 2 non-zeros for every column p. Let the number of rows (machines) at any point be m' and number of columns (jobs) be n'. The above observation implies the fractional support is $\geq \max(5m', 2n')$ Since z is a vertex solution, the fractional support is of cardinality $\leq n' + 2m'$. This is a contradiction. □

Proof of Theorem ??. The final algorithm is as follows. First modify the inputs and then perform the grouping obtaining the feasible configurations for every group. For every guess of $\mathbf{b}_t^{(r,s)}$'s in the enumeration step, obtain the residual problem and check if (15)-(17) is feasible or not. If not, then the guess is infeasible. If yes, then Lemma 8.2 implies a residual solution which satisfies the cardinality constraints, and every machine i obtains a total load of at least $T_i(1 - 5\epsilon)$. The running time is dominated by the number of guesses which is $C_\epsilon^{O(\log^3 n)}$. □