An approximation algorithm for Uniform Capacitated $k$-Median problem with $1 + \epsilon$ capacity violation

Jarosław Byrka, Bartosz Rybicki, and Sumedha Uniyal

Abstract. We study the Capacitated $k$-Median problem, for which all the known constant factor approximation algorithms violate either the number of facilities or the capacities. While the standard LP-relaxation can only be used for algorithms violating one of the two by a factor of at least two, Shi Li [SODA’15, SODA’16] gave algorithms violating the number of facilities by a factor of $1 + \epsilon$ exploring properties of extended relaxations.

In this paper we develop a constant factor approximation algorithm for hard Uniform Capacitated $k$-Median violating only the capacities by a factor of $1 + \epsilon$. The algorithm is based on a configuration LP. Unlike in the algorithms violating the number of facilities, we cannot simply open extra few facilities at selected locations. Instead, our algorithm decides about the facility openings in a carefully designed dependent rounding process.

1 Introduction

In capacitated $k$-median we are given a set of potential facilities $F$, capacity $u_i \in \mathbb{N}^+$ for each facility $i \in F$, a set of clients $C$, a metric distance function $d$ on $C \cup F$ and an integer $k$. The goal is to find a subset $F' \subseteq F$ of $k$ facilities to open and an assignment $\sigma : C \rightarrow F'$ of clients to the open facilities such that $|\sigma^{-1}(i)| \leq u_i$ for every $i \in F'$, so as to minimize the connection cost $\sum_{j \in C} d(j, \sigma(j))$. In the uniform capacity case, $u_i = u, \forall i \in F$.

The standard $k$-median problem, where there is no restriction on the number of clients served by a facility, can be approximated up to a constant factor $\frac{3}{2}$.

The current best is the $(2.675 + \epsilon)$-approximation algorithm of Byrka et al. [5], which is a result of optimizing a part of the algorithm of Li and Svensson [12].

Capacitated $k$-median is among the few remaining fundamental optimization problems for which it is not clear if there exist constant factor approximation algorithms. All the known algorithms violate either the number of facilities or the capacities. In particular, already the algorithm of Charikar et al. [7] gave 16-approximate solution for uniform capacitated $k$-median violating the capacities.

* Research supported by NCN 2012/07/N/ST6/03068 grant

** Partially supported by the ERC StG project NEWNET no. 279352.
by a factor of 3. Then Chuzhoy and Rabani \cite{8} considered general capacities and gave a 50-approximation algorithm violating capacities by a factor of 40.

Perhaps the difficulty is related to the unbounded integrality gap of the standard LP relaxation. To obtain integral solutions that are bounded w.r.t. a fractional solution to the standard LP, one has to either allow the integral solution to open twice more facilities or to violate the capacities by a factor of two. Recently, LP-rounding algorithms essentially matching these limits were obtained \cite{14}.

Next, Li broke this integrality gap barrier by giving a constant factor algorithm for uniform capacitated $k$-median by opening $(1 + \epsilon)k$ facilities \cite{10}. The algorithm is based on rounding a fractional solution to an extended LP. More recently, he gave an algorithm working with general soft capacities and still opening $(1 + \epsilon)k$ facilities \cite{11}. (In the soft capacitated version we can open multiple copies of the same facility, whereas in the hard version we can open at most one copy.) This new algorithm is based on an even stronger configuration LP. Notably, each of the extended linear programs is not solved exactly by the algorithms, but rather a clever "round-or-separate" technique is applied. This technique was previously used in the context of capacitated facility location in \cite{2}. It essentially allows not to solve the strong LP upfront, but only to detect violated constraints during the rounding process. If a violated constraint is detected, it is returned back to the "feasibility-checking" ellipsoid algorithm. While it is not clear if the strong LP with all the constraints can be solved efficiently, it can be shown that the above described process terminates in polynomial time, see \cite{2}.

1.1 Our results and techniques
We give an algorithm for uniform capacitated $k$-median rounding a fractional solution to the configuration LP, from \cite{11}, via the "round-or-separate" technique. We obtain a constant factor approximate integral solution violating capacities by a factor of $1 + \epsilon$. We utilize the power of the configuration LP in effectively rounding small size facility sets, and combine it with a careful dependent rounding to coordinate the opening between these small sets. The main result of this paper is described in the following theorem.

**Theorem 1.** There is a bi-factor randomized rounding algorithm for hard uniform capacitated $k$-median problem, with $O(1/\epsilon^2)$-approximation under $1 + \epsilon$ capacity violation.

Our algorithm utilizes the white-grey-black tree structure from \cite{11}, but the following rounding steps are quite different. In particular, the handling of the small "black components" differs. While aiming for a solution opening $(1 + \epsilon)k$ facilities Li \cite{11} can treat each black component independently, we are forced to precisely determine the number of open facilities. Hence we cannot allow a single black component to individually decide to open more facilities than in the fractional solution. Instead, we first use a preprocessing step which we call massage that reduces the variance in the number of open facilities in the fractional solution within each "black component". Then we use a form of a
facilities. Let \( S \in \epsilon \) approximation algorithm by opening \((1 + \epsilon)\) introduced in [11] a stronger LP called the Configuration LP and got a constant

\[ \leq \]

every client \( j \) cardinality constraint that exactly

indicates whether client \( j \) is connected to facility \( i \) or not. Constraint \((1)\) is the cardinality constraint that exactly \( k \) facilities are open, Constraint \((2)\) requires every client \( j \) to be connected, Constraint \((3)\) says that a client can only be connected to an open facility and Constraint \((4)\) is the capacity constraint.

The basic LP has an unbounded integrality gap even if we allow to violate the cardinality or the capacity constraints by \( 2 - \epsilon \). To overcome this gap, Li introduced in [11] a stronger LP called the Configuration LP and got a constant approximation algorithm by opening \((1 + \epsilon)k\) facilities.

To formulate the configuration LP constraints, let us fix a set \( B \subseteq F \) of facilities. Let \( S = \{ S \subseteq B : |S| \leq \ell_1\} \) and \( \tilde{S} = S \cup \{\perp\} \), where \( \ell_1 \) is some constant we will define later and \( \perp \) stands for "any subset of \( B \) with size greater than \( \ell_1 \)". We treat set \( \perp \) as a set that contains all the facilities \( i \in B \). For every \( S \in \tilde{S} \), let \( z^B_S \) be an indicator variable corresponding to the event that the set of open facilities in \( B \) is exactly \( S \) and \( z^B_\perp \) captures the event that the number of facilities open in \( B \) is more than \( \ell_1 \). For every \( S \in \tilde{S} \) and \( i \in S \), \( z^B_{S,i} \) indicates the event that set \( S \) is open and \( i \) is open as well. Notice that when \( i \in S \neq \perp \), we always have \( z^B_{S,i} = z^B_S \). For every \( S \in \tilde{S} \), \( i \in S \) and \( j \in C \), \( z^B_{S,i,j} \) indicates the event that \( z^B_{S,i} = 1 \) and \( j \) is connected to \( i \). The following are valid constraints for any feasible integral solution.

\[
\sum_{S \subseteq B} z^B_S = 1; \quad z^B_S = z^B_{S,i} \forall S \in S, i \in S; \quad (5)\]

\[
\sum_{S \subseteq \tilde{S} : i \in S} z^B_{S,i} = y_i \forall i \in B; \quad (6)\]

\[
\sum_{S \subseteq \tilde{S} : i \in S} z^B_{S,i,j} = x_{i,j} \forall i \in B, j \in C; \quad (7)\]

\[
0 \leq z^B_{S,i,j} \leq z^B_{S,i} \leq z^B_S \forall S \in \tilde{S}, i \in S, j \in C; \quad \sum_{i \in B} z^B_{1,i} \geq \ell_1 z^B_{1,1}. \quad (8)\]

Constraint \((5)\) says that exactly one set \( S \in \tilde{S} \) is open. Constraint \((6)\) says that if facility \( i \) is open then \( z^B_{S,i} = 1 \) for exactly one set \( S \in \tilde{S} \). Constraint \((7)\) says that if \( j \) is connected to \( i \) then \( z^B_{S,i,j} = 1 \) for exactly one set \( S \in \tilde{S} \).
Constraint (10) says that if \( z_B^B = 1 \), then \( j \) can be connected to at most 1 facility in \( S \). Constraint (11) is the capacity constraint. Constraint (12) says that if \( z_B^B = 1 \), then at least \( \ell_1 \) facilities in \( B \) are open.

The configuration LP is obtained by adding the above set of constraints for all subsets \( B \subseteq F \). As there are exponentially many sets \( B \), we do not know how to solve this LP. But given a fractional solution \((x, y)\), for a fixed set \( B \), we can construct the values of the set of variables \( z \) (see (11)) and also check the constraints in polynomial time since the total number of variables and constraints is \( n^{O(\ell_1)} \). We apply method that has been used in, e.g., [10]. Given a fractional solution \((x, y)\) to the basic LP relaxation, our rounding algorithm either constructs an integral solution with the desired properties or outputs a set \( B \subseteq F \) for which one of the Constraints (5) to (12) is infeasible. In the latter case, we can find a violating constraint and feedback it to the ellipsoid method.

3 Rounding Algorithm

Focus on an optimal fractional solution \((x, y)\) to the basic LP. Let \( d_{av}(j) = \sum_{i \in F} d(i, j)x_{i,j} \) be the average connection cost of a client \( j \in C \). Let \( d(S, T) = \min_{i \in S, j \in T} d(i, j) \), for any \( S, T \subseteq C \cup F \). Also let, \( d(i, S) = d(\{i\}, S) \). Note that the value of the LP solution \((x, y)\) is \( \text{LP} := \sum_{(i, j) \in F \times C} d(i, j)x_{i,j} = \sum_{j \in C} d_{av}(j) \). For any set \( F' \subseteq F \) of facilities, let \( y_{F'} := y(F') := \sum_{i \in F'} y_i \) be the volume of the set \( F' \). For any set \( F' \subseteq F \) and \( C' \subseteq C \) of clients, let \( x_{F', C'} := \sum_{(i, j) \in F' \times C} x_{i,j} \). Also let, \( x_{i,C'} := x_{i,\varepsilon} \) and \( x_{F',\ell} := x_{F',\ell_1} \).

Definition 1. Let \( D_i := \sum_{i \in C} x_{i,j}d(i, j) \) and \( D'_i := \sum_{j \in C} x_{i,j}d_{av}(j) \) for each \( i \in F \). Let \( D_S := D(S) := \sum_{i \in S} D_i \) and \( D'_S := D'(S) := \sum_{i \in S} D'_i \) for every \( S \subseteq F \). Obviously \( D_F = D'_F = \text{LP} \).

First we will partition facilities into clusters (as done in [4, 10]). Each cluster will have a client \( v \) as its representative. We denote the set of cluster representatives by \( R \). Each cluster will contain the set of facilities nearest to a representative \( v \in R \) and the fractional number of open facilities in each cluster will be bounded below by \( 1 - \frac{1}{\ell} \). Let \( U_v \) be the set of facilities in the cluster corresponding to representative \( v \in R \). For any set \( J \subseteq R \) of representatives, we use \( U_J := U(J) = \bigcup_{v \in J} U_v \). Constants \( \ell := O(1/\varepsilon) \) and \( \ell_1 := \ell^2 \) are integers, which we will define later. Since the clustering procedure is the same as in [4, 10], we omit the following Claim (see Claim 4.1 in [10]) captures the key properties of the clustering procedure.

Claim 1. The following statements hold:
1. for all \( v, v' \in R, v \neq v' \), we have \( d(v, v') > 2\ell \max\{d_{av}(v), d_{av}(v')\} \);
2. for all \( j \in C, \exists v \in R \), such that \( d_{av}(v) \leq d_{av}(j) \) and \( d(v, j) \leq 2\ell d_{av}(j) \);
3. \( y_{U_v} \geq 1 - 1/\ell \) for every \( v \in R \);
4. for any \( v \in R, i \in U_v \) and \( j \in C \), we have \( d(i, v) \leq d(i, j) + 2\ell d_{av}(j) \).

We partition the set of representatives \( R \) build a tree and color its edges in the same way as Li [11]. To partition \( R \), we run the Kruskal’s algorithm to find a minimum spanning tree of \( R \). In the Kruskal’s algorithm we maintain a partition
\( J \) of \( R \) and the set of selected edges \( E_{MST} \). Initially \( J = \{ \{ v \} : v \in R \} \) and \( E_{MST} \) is empty. The length of each edge \((u, v) \in (R^2)\) is the distance between \( u \) and \( v \). We sort all edges in \((R^2)\) by length, breaking ties in an arbitrary way. For each edge \((u, v)\) in this order if \( u \) and \( v \) are not in the same group in \( J \), we merge the two groups and add edge \((u, v)\) to \( E_{MST} \).

We now color edges of \( E_{MST} \). For every \( v \in R \), we know that \( y(U_v) \geq 1 - \frac{1}{\ell} \).

For any subset of representatives \( J \subseteq R \) we say that \( S \) is big if \( y(U_J) \geq \ell \) and small otherwise. For each edge \( e \in E_{MST} \) we consider the step in which edge \( e = (u, v) \) was added by the Kruskal’s algorithm to MST. After the iteration we merge groups \( J_u \) (containing \( u \)) and \( J_v \) (containing \( v \)) to one group \( J_u \cup J_v \). If both \( J_u \) and \( J_v \) are small, then we paint edge \( e \) in black. If both are big, we paint the edge \( e \) white. Otherwise if one is small and the other is big then we direct the edge \( e \) towards the big group and paint it grey.

Consider only the black edges from \( E_{MST} \). We define a black component of MST as a connected component in this graph. The following claim (see Claim 4.1 in [11]) is a consequence of the fact that \( J \subseteq R \) appears as a group at some step of the Kruskal’s algorithm.

Claim 2. Let \( J \) be a black component, then for every black edge \((u, v) \in (J^2)\), we have \( d(u, v) \leq d(J, R \setminus J) \).

We contract all the black components and remove all the white edges from MST. The obtained graph \( \Upsilon \) is a forest. Each node \( p \) (vertex in the contracted graph) in \( \Upsilon \) corresponds to a black component and each grey edge is directed. Let \( J_p \subseteq R \) be the set of representatives corresponding to node \( p \). Abusing the notation slightly, we define \( U_p := U(J_p) = \bigcup_{v \in J_p} U_v \). Let's define \( y_p := y(U_p) \). The following lemma follows from the way in which we create our forest. Proof can be found in [11].

**Lemma 1.** For any tree \( \tau \in \Upsilon \), the following statements are true:
1. \( \tau \) has a root node \( r_\tau \) such that all the grey edges in \( \tau \) are directed towards \( r_\tau \);
2. \( J_{r_\tau} \) is big and \( J_p \) is small for all other nodes in \( \tau \);
3. in any leaf-to-root path of \( \tau \), the lengths of grey edges form a non-increasing sequence;
4. for any non-root node \( p \in \tau \), the length of the grey edge in \( \tau \) connecting \( p \) to its parent is exactly \( d(J_p, R \setminus J_p) \);

Consider a tree \( \tau \in \Upsilon \). We group the black components of \( \tau \) top down into sub-trees choosing grey edges in increasing order of their lengths until the volume of the group just exceeds \( \ell \).

**Definition 2.** A black component is called a singleton component if it contains only a single node corresponding to some \( v \in R \). A singleton component which is the very root of some tree \( \tau \in \Upsilon \), is called a singleton root component.

**Observation 1** Consider tree \( \tau \in \Upsilon \). The root-group \( G \) has volume at least \( \ell \). If the root-group is not a singleton root component, then it has volume at most \( 2\ell \).
The leaf-groups might have volume smaller than \( \ell \). All the other internal-groups have volume in the range \([\ell, 2\ell]\).

From now on we will slightly abuse the notation and instead of \( z^p \) and \( x_{U_i} \) we will write \( z^p \) and \( x_p \), respectively. Also, we will assume that any black component \( U_p \) corresponding to a node \( p \in \mathcal{T} \) satisfies the Configuration LP Constraints \([5]\) to \([12]\). If not, then we find the violating constraint and we recompute the LP by applying the ellipsoid method.

In the next lemma we consider edges related to a group \( \mathcal{G} \). The proofs for all the lemmas can be found in [6].

**Lemma 2.** For any tree \( \tau \), group \( \mathcal{G} \) and black component \( p \in \mathcal{G} \), the following properties hold:

1. the total number of grey or black edges within \( \mathcal{G} \) is at most \( O(\ell) \);
2. any grey edge entering \( \mathcal{G} \) is longer (or equal) to any grey or black edge in \( \mathcal{G} \);
3. the total length of the path (including both grey and black edges), from any node \( v \in J_p \) to the root \( r \) of the group \( \mathcal{G} \), is at most \( O(\ell)d(J_p, R \setminus J_p) \); and
4. the length of the path from \( v \in J_p \) to the root \( r' \) of its parent group \( \mathcal{G}' \) (if it exists) is \( O(\ell)d(J_p, R \setminus J_p) \).

**Lemma 3.** Consider any representative \( v \in R \). We can construct a new solution \( \{x', y', z'\} \) such that all the facilities from set \( U_v \) are collocated with \( v \). The cost of the new solution is at most \( O(\ell) \text{LP} \).

For any black component \( p \in \mathcal{T} \), let \( \hat{y}_p = \sum_{S \in \mathcal{S}} S |z^p_S| = \sum_{S \in \mathcal{S}} |z^p_S| \) and \( y'_p = \sum_{i \in U_p} y'_i = \sum_{i \in U_p} \hat{z}'_{p,i} = \sum_{S \in \mathcal{S}} |S| z^p_S \) and \( \sum_{i \in U_p} \hat{z}'_{p,i} \). Moreover, we define \( \pi(J_p) := \sum_{j \in C} x_{p,j} (1 - x_{p,j}) \) for any \( p \in \mathcal{T} \).

Next we show that, we can pre-process each black component \( p \) by opening \( \text{a set randomly from } \mathcal{S} \) and pre-assigning some clients to the open set. We send the demand of the rest of the clients that was served by \( p \) to the root of the parent group. To do that, we first reduce the variance in the size of sets in \( \mathcal{S} \).

For the \( |C| = ku \) case, we perform a massage process in which we move facilities from bigger sets to smaller ones, until the size of each set is either \( |\hat{y}_p| \) or \( |\hat{y}_p| \). Using the saturation property, we can reroute the demand of clients assigned to these facilities, so that the final solution remains feasible. We scale up the opening values of these sets, so that the expected size of sets in \( \mathcal{S} \) is \( \hat{y}_p \).

For the general case, instead, we use a brutal massage process in which we pick a prefix of the smallest sets in set \( \mathcal{S} \), such that their total opening value is at least a constant. Then we add some extra facilities to the selected sets and scale up the opening values of these sets, so that the sets have size either \( |\hat{y}_p| \) or \( |\hat{y}_p| \) and the total opening is exactly \( \hat{y}_p \).

In both cases, we pick a set randomly and pre-assign some clients based on their connection values. The intuition is that the clients which are served by more than \( 1 - \epsilon' \) by the black component \( p \) get assigned to the selected set with high probability and the demand of the other clients can travel to the root of the parent group by paying the total cost of \( O(\ell^2) \text{LP} \).
Lemma 4. Let $p \in \mathcal{T}$ be a black component and $U_p$ satisfies $y_p \leq 2\ell$ and let $Z_p \in \{0, 1\}$ be a random variable, such that $E[Z_p] = \hat{y}_p - \lfloor \hat{y}_p \rfloor$. Moreover, constraints \([5]\) to \([12]\) are satisfied for the solution \(\{x', y', z'\}\) and $U_p$. Then, we can pre-open a set $S \subseteq U_p$ of expected cardinality $\hat{y}_p$, where $|S| = |\hat{y}_p| + Z_p$, and pre-assign a set $C' \subseteq C$ of clients to $S$ such that

1. each facility $i \in S$ is pre-assigned at most $u$ clients
2. expected cost of sending not assigned demand $x_{p,C \setminus C'}$ to the root of the parent-group is at most $O(\ell^2)d(J_p, R \setminus J_p)\tau(J_p)
3. $Pr[|S| = |\hat{y}_p|] = \hat{y}_p - \lfloor \hat{y}_p \rfloor$ and $Pr[|S| = |\hat{y}_p|] = 1 - (\hat{y}_p - \lfloor \hat{y}_p \rfloor)$
4. expected cost of pre-assignment and local moving of $x_{p,C \setminus C'}$ is at most $O(\ell) \sum_{j \in C, i \in U_p} d(i, j)x'_{i,j}$

3.1 Dependent Rounding

We will use a dependent rounding (DR) procedure, described in \([9]\), to decide if a particular variable should be rounded up or down. It transforms fractional vector \(\{\tilde{v}_i\}_{i=1}^n\) to a random integral vector \(\{\tilde{v}_i\}_{i=1}^n\). DR procedure has the following properties:

1. Marginal distribution: $Pr[\tilde{v}_i = 1] = \tilde{v}_i$
2. Sum-preservation: $\sum_{i=1}^n \tilde{v}_i \in \{\lfloor \sum_{i=1}^n \tilde{v}_i \rfloor, \lfloor \sum_{i=1}^n \tilde{v}_i \rfloor \}$

In our procedure we first fix a tree $\tau \in \mathcal{T}$. Then we choose a pair of fractional black components according to a predefined order. After that we increase the opening of one and decrease the opening of the other in a randomized way. After each such iteration, at least one black component has an integral opening. Based on the value of the integral opening $y''_p \in \{\lfloor \hat{y}_p \rfloor, \lfloor \hat{y}_p \rfloor \}$ decided for a black component $p \in \mathcal{T}$, we will select a set of facilities $S \in \mathcal{S} : |S| = y''_p$ in a random way (for details see Lemma \([5]\)). First we do dependent rounding among the black components of the children groups of each parent group. After this step each group $\mathcal{G}$ will have at most one fractional black component among all black components in its children groups, and the total opening (capacity) within these black components will be preserved. Finally, once we complete this rounding phase for all the trees in $\mathcal{T}$, then we will do dependent rounding among all the remaining fractional black components across all the trees in $\mathcal{T}$ in an arbitrary order. The procedure will preserve the sum of facility openings, hence in the end we will open exactly $k$ facilities.

In the ”rounding among children groups” step, we will do the rounding among the black components within the children-groups of a group $\mathcal{G}$ in an order defined by non-decreasing distance of these black components to the root $r$ of the parent group $\mathcal{G}$ (breaking ties arbitrarily). This way, we would have an extra property on the number of open facilities for every prefix in this order of the black components belonging to the children-groups of group $\mathcal{G}$.

Before we start the rounding procedure, we will send exactly $\sum_{i \in U_p} z^p_{\bot,i} - \hat{y}_p z^p_{\bot}$ opening, from each black component $p$, to a virtual black component $v_p$. 

7
co-located with the root of the group. Note that since \( z_{p, i}^p = z_{\bot}^p \) and \( z'_{p, i}^p = z'_{\bot}^p \), we can use \( z \) instead of \( z' \). Let us define \( \hat{y}_{\mathcal{G}} = \sum_{p \in \mathcal{G}} \sum_{i \in U_p} \frac{z_{p, i}^p}{\ell} - \hat{y}_p z_{\bot}^p \). We will call this the blue opening. We will treat this blue opening, co-located with the root, as a virtual black component. Since we are in the uniform capacity case, by loosing a constant factor we can assume that \( \mathcal{F} = \mathcal{C} \) [10]. Moreover we can work with soft capacitated version of the problem due to Theorem 1.2 in [10]. Hence, for the blue opening, we can simply open the decided number of co-located facilities at the virtual black component. By \( BCG(\mathcal{G}) \) we denote a set of all, virtual or not, black components in the children groups of group \( \mathcal{G} \). Note that, from now on the group \( \mathcal{G} \) also contains the virtual black component \( v_G \).

Consider the root group of the tree \( \tau \). If it is a singleton root component then we classify it as a virtual black component, otherwise we treat it as a standard black component. Note that the sum \( \hat{y}_p + \sum_{i \in U_p} \frac{z_{p, i}^p}{\ell} - \hat{y}_p z_{\bot}^p = \sum_{i \in U_p} z_{p, i}^p + (1 - z_{\bot}^p) \hat{y}_p = \hat{y}_{p'} = \hat{y}_p' \). Hence, the total opening across all the black components is exactly equal to \( k \).

**Lemma 5.** For any group \( \mathcal{G} \), the total demand is at most \( (1 + O(1/\ell)) \sum_{p \in \mathcal{G}} u \hat{y}_p \).

For simplicity of exposition, we will say that a black component \( p \) is closed, if the procedure decides to round down the opening of that component to \( \lfloor \hat{y}_p \rfloor \), otherwise we say it is opened.

In this dependent rounding procedure, in contrast to [4], we will also be able to pull demand to the black components where we decided to open an extra facility. Cost of pulling can be bounded by the LP cost for sending the demand out of a black component. This new strategy is crucial to bring down the capacity violation from \( 2 + \epsilon \) to \( 1 + \epsilon \).

### 3.2 Rounding among children groups

Consider any tree \( \tau \in \mathcal{T} \) and its root \( r \). For simplicity of description, we add a fake single node parent group and attach the root \( r \) to this fake group node with a grey edge of length exactly \( d(J_p, R \setminus J_p) \), where \( p \) corresponds to the only the black component in the root group. Notice that from now on even the original root group is a child-group of some other group.

In the first phase of dependent rounding, we select the deepest (w.r.t. the number of edges) leaf-group and let its parent group be \( \mathcal{G} \). Let \( \hat{y}_p = \hat{y}_p - \lfloor \hat{y}_p \rfloor \) for each \( p \in \mathcal{T} \). For performing this dependent rounding procedure within children groups of \( \mathcal{G} \), we use the root \( r \) of \( \mathcal{G} \) as a accumulator, which will temporarily store all the not assigned demand from children groups of \( \mathcal{G} \). Let \( n_G = |BCG(\mathcal{G})| \).

To perform the dependent rounding procedure, we would order the components in \( BCG(\mathcal{G}) = \{p_1, p_2, \ldots, p_n\} \) by non-decreasing distance from the root \( r \) of \( \mathcal{G} \), so \( d(p_i, r) \leq d(p_{i+1}, r) \) for \( i < n_G \). We define the vectors \( \hat{y}_{\mathcal{G}} = (\hat{y}_{p_1}, \hat{y}_{p_2}, \ldots, \hat{y}_{p_{n_G}}) \) and \( \hat{y}_G = (\hat{y}_{p_1}, \hat{y}_{p_2}, \ldots, \hat{y}_{p_{n_G}}) \). Now we apply dependent rounding between the two fractional components in the \( i \)th prefix of vector \( \hat{y}_{\mathcal{G}} \), for each \( i \) starting from \( i = 2 \) until \( i = n_G \). Note that, after applying dependent rounding on the \( i \)th prefix of \( \hat{y}_{\mathcal{G}} \), at most one component will remain fractional in the prefix and one will become integral. If the black component \( p \) which become integral is not virtual, we apply Lemma [4] with \( r \in \mathcal{G} \) as a root and, with
$Z_p = 1$ if component is open and $Z_p = 0$ if it is closed. Let the output vector be $Z_G = (Z_{p_1}, Z_{p_2}, \ldots, Z_{p_G})$. If $\sum Z_p(i)$ is not integral then the output vector will have one fractional variable, otherwise it will be a vector of all integral values. Notice that by the property (1) of dependent rounding $E[Z_p] = \bar{y}_p$ and by the property (2) the sum of the facility opening $\hat{y}_p$ is preserved.

From now on, we will ignore the presence of all the children of the group $G$ in our procedure. We repeat this process until our tree $\tau$ has only the added fake group left. Note that the root group will contain at most one black component which is fractional. After we finish the first phase, for each group $G$, at most one component of $Z_G$ will be fractional.

For any vector $v$, let $v[i_1, i_2] = \sum_{k=i_1}^{i_2} v(k)$. Due to the ordering which we follow in the above dependent rounding procedure and the fact that we didn’t move any opening out of (or into) set $BCG(G)$ for each group $G$, the following observation holds.

**Observation 2** After the first phase of the rounding procedure, $Z_G[1, i] \in [\lfloor \bar{y}_G[1, i] \rfloor, \lceil \bar{y}_G[1, i] \rceil]$ holds for each $i$ and, for each non-leaf group $G$. Moreover, $Z_G[1, n_G] = \bar{y}_G[1, n_G]$.

Once we complete phase one of rounding for each tree $\tau \in \Upsilon$, the second phase of the rounding procedure starts. In the second phase of the rounding procedure we just apply dependent rounding among all the remaining fractional variables, in an arbitrary order, until everything is integral. We apply Lemma 4 to all the non-virtual components with $Z_p = 1$, if it was open, and $Z_p = 0$ otherwise. Notice that for the black component from the root group we will use the root of a fake group as an accumulator. We open $\lceil \hat{y}_{vG} \rceil$ facilities in each virtual component $v_G$ if it was rounded up, and $\lfloor \hat{y}_{vG} \rfloor$ otherwise.

Since the last fractional component in $BCC(G)$ could be either opened or closed, the total $Z_G[1, n_G]$ is either $\lceil \hat{y}_G[1, n_G] \rceil$ or $\lfloor \hat{y}_G[1, n_G] \rfloor$ respectively. And since each of the components of vector $Z$ is integral, the following observation is true.

**Observation 3** After the second phase of the rounding procedure, $Z_G[1, i] \in \{\lfloor \bar{y}_G[1, i] \rfloor, \lceil \bar{y}_G[1, i] \rceil\}$ holds for each $i$ and, for each non-leaf group $G$.

**Lemma 6.** The cost of moving the demand from all black components to their respective accumulators can be bounded by $\sum_{p \in \Upsilon} O(\ell^2)d(J_p, R \setminus J_p)\pi(J_p)$.

### 3.3 Pulling back demand to the open facilities

Now we will define a single-commodity flow corresponding to distributing the demand from the accumulator co-located with the root of some non-group to the open facilities in its children groups, for each tree $\tau \in \Upsilon$. To do this, we will pull back demand to the black components in a greedy way by pulling the demand first to the component belonging to $BCG(G)$ which is closest to the root $r$. We can bound the cost of pulling demand to the open facilities by charging it to the cost of pushing the demand to the root bounded in Lemma 6. The intuition is, since we are pulling back the demand in a greedy fashion, we can argue that for every demand, the distance which it will travel in the pulling phase is at most the distance it traveled to reach the accumulator $r$ in the pushing phase.
Since the cost for pushing is bounded by \( O(\ell^2) \sum_{p \in BCG(\mathcal{G})} d(J_p, R \setminus J_p) \pi(J_p) \) (see Lemma 6), hence by the above claim the cost of pulling back the demand is bounded as well.

In this procedure, we first fix a tree \( \tau \in \mathcal{T} \). Consider a non-leaf group \( \mathcal{G} \) of \( \tau \) and the set \( BCG(\mathcal{G}) \). In the pre-assignment step (Lemma 4), let \( q_p \) be the amount of demand we assigned to the open facilities in each black component \( p \in BCG(\mathcal{G}) \). Notice that for any virtual black component \( p \in BCG(\mathcal{G}) \) we didn’t assign any demand in a pre-assignment, so \( q_p = 0 \). Now we define vector \( \mathcal{Q}(\mathcal{G}) = (q_{p_1}, q_{p_2}, \ldots, q_{p_{n_{\mathcal{G}}}}) \), to be the vector of the pre-assigned demand, which respects the same order of the components as in vector \( \mathcal{Y}(\mathcal{G}) \).

Now we describe the pulling back procedure which we call the greedy pulling process. First, we freeze \((1 + O(1/\ell))u\) units of demand at the accumulator \( r \) of group \( \mathcal{G} \). Next we start pulling the rest of the demand to the black components \( BCG(\mathcal{G}) \). We do the pulling process in the same greedy order in which we did the dependent rounding among the black components \( BCG(\mathcal{G}) \), i.e. starting from the component closest to \( r \). By definition, the vectors \( \dot{y}_G, Z_G \) and \( q_G \) respect this ordering. We start pulling the demand equal to \((1 + O(1/\ell))(Z_G(i) + \lfloor \dot{y}_G(i) \rfloor)u - q_G(i)\) from the accumulator \( r \) to the \( i \)th component starting from \( i = 1 \), until we have no more demand to pull. We do this process for each non-leaf group \( \mathcal{G} \) in all the trees in our forest \( \mathcal{T} \).

**Observation 4** After the greedy pulling process, each black component \( p \in BCG(\mathcal{G}) \) has a capacity violation by a factor of at most \((1 + O(1/\ell))\).

**Lemma 7.** After the greedy pulling procedure, the left over demand at any accumulator \( r \) of some non-leaf group \( \mathcal{G} \) is exactly equal to \( u(1 + O(1/\ell)) \); which is the demand frozen at the beginning.

**Lemma 8.** For any non-leaf group \( \mathcal{G} \), the distance travelled by any demand in the greedy pulling phase is at most the distance travelled by it in the dependent rounding phase.

Now we would distribute the demand received by any black component \( p \) to the actual open facilities (which are located at the representatives \( J_p \)), such that each facility has a capacity violation of at most \( 1 + O(1/\ell) \). The following lemma bounds the cost of this step.

**Lemma 9.** Any demand that a black component \( p \) received in the greedy pulling back process can be distributed to the open facilities within \( p \). The distance travelled by the demand received by \( p \) in this procedure is at most \( O(\ell)d(J_p, R \setminus J_p) \).

### 3.4 Distributing frozen demand to the open facilities

Now, we distribute the frozen \((1 + O(1/\ell))u\) units of demand located at the accumulators over some open facilities, such that each open facility gets at most \( uO(1/\ell) \) more demand. Let us fix a tree \( \tau \in \mathcal{T} \). To do this distribution, we first send \((1 + O(1/\ell))u\) units of demand from each of the non-fake accumulator to the accumulator of his parent group. Note that, using Lemma 9 we can bound the cost for this movement by paying an additive factor of \( O(\ell)d(J_p, R \setminus J_p) \) in
the distance moved by this demand in Section 3.2. Let \( r \) be the accumulator belonging to the group \( \mathcal{G} \), which received \(|C^c_\mathcal{G}|(1 + O(1/\ell))u = O(u|C^c_\mathcal{G}|)\) units of demand from the accumulators of the non-leaf children groups \( C^c_\mathcal{G} \) of the group \( \mathcal{G} \) in the tree \( \tau \). Note that, \(|C^c_\mathcal{G}| \leq n_\mathcal{G} \), since \( \mathcal{G} \) may have children which are leaf-groups. We start sending \( O(1/\ell)u(|\hat{y}_\mathcal{G}(i)| + Z_\mathcal{G}(i)) \) units of demand to the 0th black component (in the same greedy order defined by the vector \( \hat{y}_\mathcal{G} \)) in the \( BCG(\mathcal{G}) \), starting from \( i = 1 \), until we have no more demand left with \( r \).

Lemma 10. After the distribution procedure for some accumulator \( r \) belonging to the group \( \mathcal{G} \), all the demand which \( r \) received from the accumulators of his non-leaf children groups will be distributed fully.

By an argument similar to Lemma 9, we can send this demand to any open facility within \( p \), by loosing an additive factor of \( O(1/\ell) \) in the distance traveled by the demand. Hence, this shows that we can distribute all the demand received by \( r \) from his children-accumulators, corresponding to groups \( C^c_\mathcal{G} \), among the open facilities within black components in \( BCG(\mathcal{G}) \), such that each facility receives at most an extra \( O(1/\ell)u \) units of demand. We keep on doing this process bottoms-up, until we reach the very root fake accumulator. Now, for the demand located in the fake accumulator, we just distribute that demand over the open facilities in the very root group of the tree, which was using this accumulator. Note that since the very root group comprises of only one black component with \( O(1) \) opening, there will be at least \( O(1) \) open facilities in this component and again we will send \( O(1/\ell)u \) units of extra demand to all the open facilities in the very root black component. By Lemma 1, each edge in the black component \( p \) has length at most \( d(J_p, R \setminus J_p) \) and by Lemma 2 the number of edges in the group is \( O(\ell) \). Hence, by loosing an additive factor of \( O(1/\ell)d(J_p, R \setminus J_p) \) in the distance traveled by the demand, we can distribute this demand over open facilities in this black component.

In the following lemma, we bound the cost of distributing the frozen demand by the upper bound which we use to bound the cost of moving this demand from black components in \( BCG(\mathcal{G}) \) to the accumulator.

Lemma 11. The distance travelled by any demand from each non-fake accumulator group \( \mathcal{G} \) in the above re-distribution process is bounded above by \( O(1/\ell)d(J_p, R \setminus J_p) \), which is also a bound on the distance it travelled to reach the accumulator in the dependent rounding phase.

Proof (Theorem 1). We modify the initial solution by ”moving” all facilities to their respective representatives (see Lemma 3). The obtained solution has cost \( O(\ell)\text{LP} \). In the Lemma 4, we pre-assign some demand and all the other demand we send to the respective accumulators. The cost of this operation is \( \sum_{p \in \mathcal{P}} \sum_{(i,j) \in U_p} O(\ell)d(i,j)x'_{i,j} + O(\ell^2) \sum_{p \in \mathcal{P}} d(J_p, R \setminus J_p)\pi(J_p) \leq O(\ell^2)\text{LP} \). The last inequality follows from \( [11] \). By the Lemmas 8, 9 and 11 we can bound the distance travelled by any demand in Sections 3.3, 3.4 by the distance it travelled in Section 3.2. This implies that the cost of moving the demand in Sections 3.3 and 3.4 is bounded by \( O(\ell^2)\text{LP} \). Hence, overall the connection cost of our algorithm is \( O(\ell^2)\text{LP} \).
From the Observation 4 we know that the capacity violation of each facility is at most $1 + O(1/\ell)$. Moreover in Section 3.4 we increase the capacity violation of each facility by at most $O(1/\ell)$. So the final capacity violation is $1 + O(1/\ell)$, which ends the proof of the theorem.

4 Concluding remarks

We showed that Configuration LP helps obtaining an algorithm with $1 + \epsilon$ capacity violation for uniform capacities. It remains open if a similar result is possible for general capacities. It seems that the difficulty of generalizing our algorithm to general case lies in the dependent rounding. It is hard to control the number of open facilities and the capacities at the same time.

References

1. K. Aardal, P. L. van den Berg, D. Gijswijt, and S. Li. Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, 242(2):358–368, 2015.
2. H.-C. An, M. Singh, and O. Svensson. LP-based algorithms for capacitated facility location. In *Foundations of Computer Science (FOCS)*, 2014 IEEE 55th Annual Symposium on, pages 256–265. IEEE, 2014.
3. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
4. J. Byrka, K. Fleszar, B. Rybicki, and J. Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 722–736. SIAM, 2015.
5. J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 737–756. SIAM, 2015.
6. J. Byrka, B. Rybicki, and S. Uniyal. An approximation algorithm for uniform capacitated k-median problem with $1 + \epsilon$ capacity violation. *CoRR*, abs/1511.07494, 2015.
7. M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1999.
8. J. Chuzhoy and Y. Rabani. Approximating k-median with non-uniform capacities. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 952–958. Society for Industrial and Applied Mathematics, 2005.
9. R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360, 2006.
10. S. Li. On uniform capacitated k-median beyond the natural lp relaxation. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 696–707. SIAM, 2015.
11. S. Li. Approximating capacitated k-median with $(1 + \epsilon)k$ open facilities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 786–796. SIAM, 2016.
12. S. Li and O. Svensson. Approximating k-median via pseudo-approximation. In *proceedings of the forty-fifth annual ACM symposium on theory of computing*, pages 901–910. ACM, 2013.