Improved Local Search Based Approximation Algorithm for Hard Uniform Capacitated k-Median Problems

April 25, 2018

Neelima Gupta¹, Aditya Pancholi²

1. Department of Computer Science, University of Delhi, India. ngupta@cs.du.ac.in
2. Department of Computer Science, University of Delhi, India. apancholi@cs.du.ac.in

Abstract

In this paper, we study the hard uniform capacitated k-median problem using local search heuristic. Obtaining a constant factor approximation for the problem is open. All the existing solutions giving constant-factor approximation, violate at least one of the constraints (cardinality/capacity). All except Koruplou et al. [22] are based on LP-relaxation.

We give $(3+\varepsilon)$ factor approximation algorithm for the problem violating the cardinality by a factor of $8/3 \approx 2.67$. There is a trade-off between the approximation factor and the cardinality violation between our work and the existing work. Koruplou et al. [22] gave $(1+\alpha)$ approximation factor with $(5+5/\alpha)$ factor loss in cardinality using local search paradigm. Though the approximation factor can be made arbitrarily small, cardinality loss is at least 5. On the other hand, we improve upon the result of Aardal et al. [11] in terms of factor-loss. They gave $(7+\varepsilon)$ factor approximation, with the cardinality violation by a factor 2. Most importantly, their result is obtained using LP-rounding, whereas local search techniques are straightforward, simple to apply and have been shown to perform well in practice via empirical studies.

We extend the result to hard uniform capacitated k-median with penalties. To the best of our knowledge, ours is the first result for the problem.

1 Introduction

k-median problem is one of the extensively studied problem in literature [1, 3, 4, 5, 11, 12, 20, 21, 27]. The problem is known to be NP-hard. The
input instance consists of a set $F$ of facilities, a set $C$ of clients, a non-negative integer $k$ and a non-negative cost function $c$ defining cost to connect clients to the facilities. Metric version of the problem assumes that $c$ is symmetric and satisfies triangle's inequality. The goal is to select a subset $S \subseteq F$ as centers with $|S| \leq k$ (cardinality constraint) and to assign clients to them such that the total cost of serving the clients from centers is minimum. In capacitated version of the problem, we are also given a bound $u_i$ on the maximum number of clients that facility $i$ can serve. The soft capacity version allows a facility to be opened any number of times whereas the hard capacity version restricts the facilities to be opened at most once. In $k$-median with penalties, each client $j$ has an associated penalty $p_j$ and we are allowed not to serve some clients at the cost of paying penalties for them. In this paper, we address the hard-capacitated $k$ median (C$k$M) problem and its penalty variant (C$k$MP), when the capacities are uniform i.e. $u_i = U$ for all $i \in F$. Our results are stated in Theorems 1 and 2. For the problems, we define an $(a,b)$-approximation algorithm as a polynomial-time algorithm that computes a solution using at most $bk$ number of facilities with cost at most $a$ times the cost of an optimal solution using at most $k$ facilities.

**Theorem 1.** There is a polynomial time local search heuristic that approximates hard uniform capacitated $k$ median problem within $(3 + \epsilon)$ factor of the optimal violating the cardinality by a factor of $\frac{5}{3}$.

In contrast to the LP-based algorithms, local search technique is known to be straightforward, simple to apply and has been shown to perform well in practice via empirical studies [22, 23]. Power of local search technique over the LP-based algorithms is well exhibited by the fact that there are constant factor approximation (3 for uniform and 5 for non-uniform) [2, 6] for capacitated facility location problem whereas the natural LP is known to have an unbounded integrality gap. On the other hand, local search heuristics are notoriously hard to analyze. This is evident from the fact that the only work known, based on local search heuristics, for C$k$M, is due to Korupolu et al. [22] more than 15 years ago.

Our work provides a trade-off between the approximation factor and the cardinality violation with the existing work. Korupolu et al. [22] gave $(1 + \alpha)$ approximation factor with $(5 + 5/\alpha)$ factor loss in cardinality using local search paradigm. Though the approximation factor can be made arbitrarily small, cardinality loss is at least 5. Small approximation factor is obtained at a big loss in cardinality. For example, for $\alpha$ anything less than 1, cardinality violation is more than 10. To achieve 3 factor approximation using their heuristic, cardinality violation is 7.5. Thus, we improve upon their result in terms of cardinality. On the other hand, we improve upon the results in [1, 25, 26] in terms of factor-loss though the cardinality loss is a little more in our case. Aardal et al. [1] gave $(7 + \epsilon)$ factor approximation, with the violation of cardinality by a factor 2 using LP Rounding. $O(1/\epsilon^2)$ factor approximation is given in [25, 26] violating the cardinality by a factor of $(1 + \epsilon)$ using sophisticated strengthened LPs.
Theorem 2. There is a polynomial time local search heuristic that approximates hard uniform capacitated $k$-median problem with penalties within $(3 + \epsilon)$ factor of the optimal violating the cardinality by a factor of $\frac{8}{3}$.

To the best of our knowledge, no result is known for CkMP.

1.1 Related Work

Both, LP-based algorithms as well as local search heuristics, have been used to obtain good approximate algorithms for the (uncapacitated) $k$-median problem. [1, 2, 3, 4, 5, 6, 9, 11, 12, 20, 21, 27]. The best known factor of 2.611 + $\epsilon$ was given by Byrka et al. [9]. Obtaining a constant approximation factor for CkM is an open problem. Natural LP is known to have an unbounded integrality gap when one of the constraints (cardinality/capacity) is allowed to be violated by a factor of less than 2 without violating the other constraint, even for uniform capacities.

Several constant factor approximations are known [8, 11, 12, 24, 16] for the problem that violate the capacities by a factor of 2 or more. A $(7 + \epsilon)$ algorithm was given by Aardal et al. [1] violating the cardinality constraint by a factor of 2. Korupolu et al. [22] gave $(1 + \alpha)$ approximation factor with $(5 + 5/\alpha)$ factor loss in cardinality. Very recently, Byrka et al. [10] broke the barrier of 2 in capacities and gave an $O(1/\epsilon^2)$ approximation violating capacities by a factor of $(1 + \epsilon)$ factor for uniform capacities. For non-uniform capacities, a similar result has been obtained by Demirci et al. in [14]. Li [25, 26] strengthened the LP to break the barrier of 2 in cardinality and gave an $O(1/\epsilon^2 \log(1/\epsilon))$ approximation using at most $(1 + \epsilon)k$ facilities. Though the algorithm violates the cardinality only by $1 + \epsilon$, it introduces a softness bounded by a factor of 2. The running time of the algorithm is $n^{O(1/\epsilon)}$.

The other commonly used technique for the problem is local search [4, 11, 22] with the best factor of $3 + \epsilon$ given by Arya et al. [4]. Local search technique has been particularly useful to deal with capacities for the facility location problem [13, 28, 33, 17, 30, 2, 6].

Some results are known for the penalty variant of (uncapacitated) facilitylocation problems, TSP and steiner network problems [28, 19, 31, 32, 15, 7]. For the capacitated variant of facility location problem with penalties, 5.83 + $\epsilon$ factor approximation for uniform and 8.532 + $\epsilon$ factor for non-uniform capacities were given by Gupta and Gupta in [18]. This is the only result known for the problems with extension on capacities as well as penalties.

1.2 High Level Idea

Let $S$ denote any feasible solution. The algorithm performs one of the following operations if it reduces the cost and it halts otherwise. The local search operations are $\text{swap}(s, o)$: $s \in S, o \in F \setminus S$ and $\text{double-swap}([s_1, s_2], [o_1, o_2])$: $s_1, s_2 \in S, o_1, o_2 \in F \setminus S$. Given a set of open facilities, min-cost flow problem is solved to obtain the optimal assignments of clients to opened facilities.

To define the swaps and the reassignments for the purpose of analysis, we extend the ideas of Arya et al. [4]. Swaps are defined so that every facility
in optimal solution is swapped in at least once and at most thrice whereas facilities in our locally optimal solution is swapped out at most thrice. When a facility in our locally optimal solution is swapped out, some of its clients are reassigned to other facilities in our solution via a mapping similar to the one defined in [4]. However, for the capacitated case, mapping needs to be done a little carefully. An almost fully utilized facility may not be able to accommodate all the clients mapped to it and conversely a partially utilized facility may not be able to accommodate the load of an almost fully utilized facility. To address this concern, we partition the facilities of our locally optimal solution into heavy (denoted by $S_H$) and light (denoted by $S_L$). A facility is said to be heavy if it serves more than $(3U/5)$ clients in our solution and is called light otherwise. Heavy facilities neither participate in swaps nor the mapping. Thus, mapping is defined between the clients of light facilities only. We allow to open $(\frac{\kappa}{3})$ facilities in our solution so that we have at least $k$ light facilities.

There are two situations in which we may not be able to define a feasible mapping between the clients of two light facilities. First situation is explained as follows: Let $O$ denote some optimal solution, let $M_o$ be the number of clients, a facility $o \in O$ shares with the light facilities of our solution. All the clients of a facility $s \in S_L$ cannot be mapped to clients of other facilities of our solution if $s$ shares more than $M_o/2$ clients with $o$. Second situation arises when $s$ shares more than $2U/5$ clients with $o$. In this case, mapping may be possible but it may not be feasible as the other facility $s'$, to which its clients are mapped, may not have sufficient available capacity to accommodate the clients of $s$. We say that $s$ dominates $o$ in the first case and that $s$ covers $o$ in the second case.

Although a facility $s \in S_L$ may dominate several facilities in $O$, it can cover at most one facility in $O$. Whereas, a facility $o \in O$ can be dominated by at most one facility in $S_L$, it can be covered by at most two facilities in $S_L$. The scenario in which a facility $o \in O$ is covered by exactly 2 facilities, say $s_1$ and $s_2$ needs to be handled carefully. In this case, we say that $s_1$ as well as $s_2$ specially covers $o$. We denote the set of such facilities in $O$ as $O_{sp}$. Since mapping of clients of $s_1$ and $s_2$ cannot be done in $o$, we would like to swap $s_1$ and $s_2$ with $o$ and, assign their clients to $o$, i.e. we would like to perform swap$(\{s_1,s_2\},o)$. However since we do not have this operation, we look for one more facility $o'$ in $O$ so that we can perform double-swap of $\{s_1,s_2\}$ with $\{o,o'\}$. First we look for $o'$ such that $\{s_1,s_2\}$ together either dominate or cover $o'$. Clearly neither $s_1$ nor $s_2$, being light, can cover any facility other than $o$. Thus we look for $o'$ that is dominated by them. If $\{s_1,s_2\}$ do not dominate any facility other than $o$, we form a triplet $<s_1,s_2,o>$ and keep it aside. We call such triplets are nice triplets. They will be used to swap in some facilities of $O$ which are not swapped in otherwise. If they dominate exactly one facility $o'$, then we perform double-swap of $\{s_1,s_2\}$ with $\{o,o'\}$. If they dominate at least two facilities other than $o$, then we cannot swap them out at all. We call such a pair of facilities as a bad pair.

Remaining facilities in $S_L$ are classified as good, bad and nice. A facility that does not dominate any facility in $O$ is termed as nice. A nice facility can be swapped in with any facility in $O$. A facility $s$ that dominates exactly one
facility $o$ in $O$ is termed as good. We perform (single) swap($s, o$) in this case. A facility $s$ that dominates more than one facilities in $O$ is termed as bad. Bad facilities cannot participate in swaps. Let $\hat{O}$ denote the set of facilities of $O$ that are either dominated by bad facilities or by bad pairs in $S_L$. Facilities of $\hat{O}$ are swapped in using the triplets (using double-swap($\{\ldots\}, \{\ldots\}$) or the nice facilities (using swap($\ldots$)). We show that the total number of triplets and the nice facilities is at least one thirds of $|\hat{O}|$ so that each facility of $S_L$ is swapped out most 3 times and each facility of $O$ is swapped in at least once and at most 3 times (Note that in the process, the facilities of $O$ which were there in the triplets also get swapped thrice). Swapping in a facility of $\hat{O}$ thrice contributes a factor of 3 and swapping out a facility of $S_L$ thrice contributes a factor of 6 making a total of 9 factor approximation.

Extending swap and double-swap to multi-swap, where upto $p, (p > 2)$ facilities can be swapped simultaneously, we are able to ensure that every $s \in S$ is swapped out at most $1 + 4/(p - 2)$ times, and every $o \in O$ is swapped in at most $1 + 4/(p - 2)$ times thereby reducing the factor to $(3 + \epsilon)$.

For CkMP, we start with an initial feasible solution with $8k/3$ facilities from $\mathcal{F}$. The clients are assigned by solving min cost flow problem over the facilities $S \cup \{\delta\}$, where $u_\delta = |C|$ and $\forall j \in C$, $c(\delta, j) = p_j$. Clients assigned to $\delta$ pay penalty in the solution $S$. We bound the cost of the locally optimal solution, in the same manner as done for CkM.

### 1.3 Local Search Paradigm

Given a problem $P$, local search algorithm starts with a candidate feasible solution $S$. A set of operations are defined such that performing an operation results in a new solution $S'$, called the neighbourhood solution of $S$. A solution $S$ may have more than one neighbourhood solutions. An operation is performed if it results in improvement in the cost. We formally describe the steps of the algorithm for a minimization problem.

The paradigm:

1. Compute an arbitrary feasible solution $S$ to $P$.
2. **while** $S'$ is a neighborhood solution of $S$ such that $cost(S') < cost(S)$
   
   **do** $S \leftarrow S'$.

The algorithm terminates at a locally optimal solution $S$, i.e. $cost(S') > cost(S)$ for every neighborhood solution $S'$.

In the above algorithm presented, we move to a new solution if it gives some improvement in the cost, however small that improvement may be. This may lead to an algorithm taking lot of time. To ensure that the algorithm terminates in polynomial time, a local search step is performed only when the cost of the current solution $S$ is reduced by at least $\frac{cost(S)}{p(n, \epsilon)}$, where $n$ is the size of the problem instance and $p(n, \epsilon)$ is an appropriate polynomial in $n$ and $1/\epsilon$ for a
fixed $\epsilon > 0$. This modification in the algorithm incurs a cost of additive $\epsilon$ in the approximation factor.

1.4 Organization of the paper

For the sake of easy disposition of ideas, we first present a weaker result for CkM in Section 2. The algorithm uses two operations: a (single) swap and a double swap and provides an $(9 + \epsilon, 8/3)$ solution. The factor is subsequently improved to $(3 + \epsilon)$ in Section 3 using multi-swap operation. The results are then extended to CkMP in Section 4.

2 $(9 + \epsilon, 8/3)$ algorithm for Capacitated k-Median Problem

In this section, we present a local search algorithm that computes a solution with cost at most $9 + \epsilon$ times the cost of an optimal. We start with an initial feasible solution selected as an arbitrary set of $8k/3$ facilities. Given a set of open facilities, optimal assignments of the clients is obtained by solving min-cost flow problem.

For any feasible solution $S$, algorithm performs one of the following operations, if it reduces the cost and terminates when it is no longer possible to improve the cost using these operations.

1. swap($s$, $o$): $S \leftarrow S \setminus \{s\} \cup \{o\}$, $o \in F \setminus S, s \in S$.
2. double-swap($\{s_1, s_2\}, \{o_1, o_2\}$): $S \leftarrow S \setminus \{s_1, s_2\} \cup \{o_1, o_2\}$, $o_1, o_2 \in F \setminus S, s_1, s_2 \in S$.

Claim 1. For the locally optimal solution $S$, and optimal solution $O$ we have,

1. $\text{cost}(S \setminus \{s\} \cup \{o\}) \geq \text{cost}(S); \forall s \in S, o \in O$
2. $\text{cost}(S \setminus \{s_1, s_2\} \cup \{o_1, o_2\}) \geq \text{cost}(S); \forall s_1, s_2 \in S, o_1, o_2 \in O$

Proof. The claims follow trivially when $S \cap O = \phi$ by the local optimality of $S$. Next, suppose $S \cap O \neq \phi$. Let $\{s_1, o_1\} \in S \cap O$. Then $S \setminus \{s_1\} \cup \{o_1\} = S \setminus \{s_1\}$ if $s_1 \neq o_1$ and it is $= S$ otherwise. Clearly the cost of assignment to facilities in $S \setminus \{s_1\}$ can not be smaller than the cost of assignment to facilities in $S$. If $s_2 \in S \setminus O, o_2 \in O \setminus S$, then cost($S \setminus \{s_2\} \cup \{o_2\}$) $\geq$ cost($S$) (by the argument of single swap) and cost($S \setminus \{s_1, s_2\} \cup \{o_1, o_2\}$) = cost($S \setminus \{s_1, s_2\} \cup \{o_2\}$) $\geq$ cost($S \setminus \{s_2\} \cup \{o_2\}$). All the other cases can be argued similarly.

2.1 Notations

Let $S$ denote the locally optimal solution and $O$ denote an optimal solution to the problem. Let $B_S(s)$ be the set of clients served by $s \in S$ and $B_O(o)$ be the set of clients served by $o \in O$. Let $B(s, o)$ denote the set of clients served by
We may not be able to define a feasible mapping for the heavy facilities. Thus other facilities in our solution via a mapping similar to the one defined in \[4\].

Our locally optimal solution is swapped out, some of its clients are reassigned to heavy facilities in our solution via a mapping similar to the one defined in [4]. We may not be able to define a feasible mapping for the heavy facilities. Thus heavy facilities are never swapped out and no client is mapped onto them for reassignment. Consider a facility \( o \in \mathcal{O} \), let \( B^\mathcal{O}_o(o) = B^\mathcal{O}_o(o) \cap \cup_{s \in \mathcal{S}} B^\mathcal{S}_o(s) \). Let \( \mathcal{M}_o = |B^\mathcal{S}_o(o)| \).

We introduce two concepts important to define the swaps and the mapping.

- A facility \( s \in \mathcal{S}_L \) is said to dominate \( o \), if \( B(s,o) > \mathcal{M}_o/2 \). Note that a facility \( o \in \mathcal{O} \) can be dominated by at most one facility in \( \mathcal{S}_L \) where as a facility \( s \in \mathcal{S}_L \) can dominate any number of facilities. Extending the definition to set \( T \), we say that a set \( T \subseteq \mathcal{S}_L \) dominates \( o \in \mathcal{O} \) if \( \sum_{s \in T} B(s,o) > \mathcal{M}_o/2 \). Let \( \mathcal{O}_d(T) \) denote the set of facilities dominated by \( T \). When \( T = \{s\} \), slightly abusing the notation we use \( \mathcal{O}_d(s) \) instead of \( \mathcal{O}_d(\{s\}) \).

- A facility \( s \in \mathcal{S}_L \) is said to cover \( o \in \mathcal{O} \), if \( B(s,o) > \frac{2}{5} U \). Note that if \( s \in \mathcal{S}_L \) then it can cover at most one facility in \( \mathcal{O} \). Also a facility \( o \in \mathcal{O} \) can be covered by at most 2 facilities in \( \mathcal{S}_L \). Extending the definition to set \( T \subseteq \mathcal{S}_L \) covers \( o \in \mathcal{O} \) if \( \sum_{s \in T} B(s,o) > \frac{2}{5} U \). Let \( v(T) \) denote the set of facilities covered by \( T \). Also we will use \( v(s) \) instead of \( v(\{s\}) \) when \( T = \{s\} \).

### 2.2 Analysis: The Swaps

Consider a set of facilities in \( \mathcal{O} \) such that each of them is covered by exactly two light facilities. Let \( \mathcal{O}_{sp} \) denote the set of such facilities. For \( o \in \mathcal{O} \), a \( 1 \leq 1 \) and onto mapping \( \tau : B^\mathcal{O}_o(o) \rightarrow B^\mathcal{S}_o(o) \) can be defined such that the following claim holds.

**Claim 2.** For \( s \in \mathcal{S}_L \) and \( o \in \mathcal{O}|o \notin \mathcal{O}_d(s) \)

1. \( \tau(B(s,o)) \cap B(s,o) = \phi. \)

2. If \( o \notin \mathcal{O}_{sp} \) then \( |\{j \in B(s,o) : \tau(j) \in B(s',o)\}| \leq \frac{2}{9} U \), \( \forall s' \neq s \).

**Proof.** \( \tau \) can be defined as follows: Order the clients in \( B^\mathcal{O}_o(o) \) as \( j_0, j_1, ..., j_{\mathcal{M}_o-1} \) such that for every \( s \in \mathcal{S} \) with a nonempty \( B(s,o) \), the clients in \( B(s,o) \) are consecutive; that is, there exists \( r, s, 0 \leq r \leq s \leq \mathcal{M}_o - 1 \), such that \( B(s,o) = \{ j_r, ..., j_s \} \). Define \( \tau(j_p) = (j_q) \), where \( q = (p + \lfloor \mathcal{M}_o/2 \rfloor) \mod \mathcal{M}_o \).

We show that \( \tau \) satisfies the claim. We prove using contradiction. Suppose if possible that both \( j_p, \tau(j_p) = j_q \in B(s,o) \) for some \( s \), where \( |B(s,o)| = \mathcal{M}_o/2 \). If \( q = p + \lfloor \mathcal{M}_o/2 \rfloor \), then \( |B(s,o)| \geq q - p - 1 = \lfloor \mathcal{M}_o/2 \rfloor + 1 > \mathcal{M}_o/2 \). If
For Claim 3, as \( o \notin \mathcal{O}_{sp} \), then at most one facility can cover \( o \). If \( v(s) = o \) then for all \( s' \neq s \), \( B(s', o) \leq \frac{2}{3} U \). And if \( v(s) \neq o \) then \( B(s, o) \leq \frac{2}{3} U \). In either case the claim \( |\{j \in B(s, o) : \tau(j) \in B(s', o)\}| \leq \frac{2}{3} U \), \( \forall s' \neq s \) holds true.

Mapping \( \tau \) is used to reassign the clients of a facility \( s \) that is swapped out to other facilities \( s' \in \mathcal{S}_L \). Claim 2 ensures that if \( s \) does not dominate \( o \), then the client \( j \in B_L(s, o) \) is mapped to some \( s' \neq s \), whereas claim 2.2 ensures that if \( o \notin \mathcal{O}_{sp} \), then no more than \( \frac{2}{3} k \) clients are mapped to \( s' \). But if \( o \in \mathcal{O}_{sp} \) such that \( P(o) = \{s, s'\} \), then more than \( \frac{2}{3} k \) clients may get mapped to \( s' \). This scenario poses a major challenge: thus facilities in \( \mathcal{O}_{sp} \) are considered separately while defining the swaps.

For \( o \in \mathcal{O}_{sp} \), \( \exists s_1, s_2 \in \mathcal{S}_L \) such that \( s_1 \neq s_2 \) and \( v(s_1) = v(s_2) = o \). Consider a facility \( o \in \mathcal{O}_{sp} \), then let \( P(o) \) denote the set \( \{s_1, s_2\} \) such that \( s_1 \neq s_2, v(s_1) = v(s_2) = o \). Let \( \mathcal{S}_{sp} = \bigcup_{o \in \mathcal{O}_{sp}} P(o) \). Let \( D(o) = \mathcal{O}_d(P(o)) \) and \( D'(o) = D(o) \setminus \{o\} \). Let \( \mathcal{D}_{sp} = \bigcup_{o \in \mathcal{O}_{sp}} D'(o) \). Figure 1(a) shows the relationship between \( \mathcal{S}_{sp} \) and \( \mathcal{D}_{sp} \cup \mathcal{O}_{sp} \). The following claims hold.

Claim 3. \( \forall o, o' \in \mathcal{O}_{sp}, o \neq o' \) we have \( P(o) \cap P(o') = \phi \).

Proof. Suppose if possible \( P(o) \cap P(o') \neq \phi \). Let \( s \in P(o) \cap P(o') \). This implies \( B(s, o) \geq 2U/5 \) and \( B(s, o') \geq 2U/5 \) which is a contradiction as \( s \in \mathcal{S}_L \).

Claim 4. \( \forall o, o' \in \mathcal{O}_{sp}, o \neq o' \) we have \( D(o) \cap D(o') = \phi \).

Proof. Suppose if possible let \( o_1 \in D(o) \cap D(o') \). This implies \( o_1 \in \mathcal{O}_d(P(o)) \) and \( o_1 \in \mathcal{O}_d(P(o')) \). This is a contradiction as \( P(o) \cap P(o') = \phi \) from Claim 3 and \( o_1 \) cannot be dominated dominated by two disjoint set of facilities.

Claim 5. \( \mathcal{D}_{sp} \cap \mathcal{O}_{sp} = \phi \).

Proof. Suppose if possible let \( o \in \mathcal{D}_{sp} \cap \mathcal{O}_{sp} \). As \( o \in \mathcal{O}_{sp} \), we have \( o \in \mathcal{O}_d(P(o)) \). By the definition we have \( o \notin D'(o) \) thus \( o \in D'(o') \) for some \( o' \neq o, o' \in \mathcal{O}_{sp} \). This implies \( o \in \mathcal{O}_d(P(o')) \) which is a contradiction as \( P(o) \cap P(o') = \phi \) using Claim 3.

We consider at-most \( k \) swaps, satisfying the following properties.

1. Each \( o \in \mathcal{O} \) is considered in at least one swap and at most three swaps.
2. If \( s \in \mathcal{S}_H \), \( s \) is not considered in any swap operation.
3. Each \( s \in \mathcal{S}_L \) is considered in at most three swaps.
4. If \( \text{swap}(s, o) \) is considered then \( \forall o' \neq o; o' \notin \mathcal{O}_d(s) \) and \( \forall o' \in \mathcal{O}_{sp} : o' \neq o; s \notin P(o') \).
5. If \( \text{double-swap}\{s_1, s_2\}, \{o_1, o_2\} \) is considered then \( \forall o' \neq o_1, o_2; o' \notin \mathcal{O}_d\{s_1, s_2\} \) and \( \forall o' \in \mathcal{O}_{sp} : o' \neq o_1, o_2; s_1, s_2 \notin P(o') \).
Let $S_W \subseteq S_L$ and $O_W \subseteq O$ denote the set of facilities that have participated in the swaps at any point of time. Initially $S_W = O_W = \emptyset$. While considering the facilities, we also maintain the sets $\hat{S} \subseteq S_L$ and $O \subseteq O$; initially $\hat{S} = \emptyset = \emptyset$. The facilities in $\hat{S}$ will never participate in any swap. Facilities in $\hat{O}$ correspond to the facilities in $\hat{S}$ in some way which will become clear when we define the swaps. We also maintain a set of triplets denoted by $B$ (will be defined shortly) and two sets $S_B$ and $O_B$ corresponding to $B$. All the three sets are empty initially. Throughout we maintain that $S_W, S_B, \hat{S}$ are pairwise disjoint and $O_W, O_B, \hat{O}$ are pairwise disjoint.

For $o_1 \in O_{sp}$ with $P(o_1) = \{s_1, s_2\}$.

1. If $|D(o_1)| = 1$ then $D(o_1) = \{o_1\}$. In this case we call $P(o_1)$ a nice pair. Set $B = B \cup \{(s_1, s_2, o_1)\}$, $O_B = O_B \cup D(o_1)$, $S_B = S_B \cup P(o_1)$.

2. If $|D(o_1)| = 2$, let $D(o_1) = \{o_1, o_2\}$ In this case we call $P(o_1)$ a good pair and consider double-swap($P(o_1), D(o_1)$) which is nothing but double-swap($\{s_1, s_2\}, \{o_1, o_2\}$). Set $O_W = O_W \cup D(o_1)$, $S^W = S^W \cup P(o_1)$.

3. If $|D(o_1)| > 2$ then we call $P(o_1)$ a bad pair. Set $\hat{O} = \hat{O} \cup D(o_1)$, $\hat{S} = \hat{S} \cup P(o_1)$. That is, put the bad pairs in $\hat{S}$ and the facilities dominated by them in $\hat{O}$. Note that the cardinality of $\hat{S}$ increased by 2 while cardinality of $\hat{O}$ increased by at least 3.

Figure 1(a) shows the partitions $S_L$ and $O$ at this time. Let $S' = S_W \cup S_B \cup \hat{S}$ and $O' = O_W \cup O_B \cup \hat{O}$. Note that $S' = S_{sp}$ and $O' = O_{sp} \cup O_{sp}$. Also, clearly $|S_W| = |O_W|$, $|S_B| = 2|O_B|$ as for every facility added to $O_B$, two facilities are added to $S_B$. and $3|\hat{S}| \leq 2|O|$. The last claim follows as for every two facilities added to $\hat{S}$, at least three facilities are added to $\hat{O}$. Next, we consider the facilities in $O \setminus O'$ and $S_L \setminus S'$. We say that a facility $s \in S_L \setminus S'$ is good if $|O_d(s)| = 1$, bad if $|O_d(s)| > 1$, else nice (i.e. $|O_d(s)| = 0$). Let $S_g, S_b$, and $S_n$ denote the set of good, bad and nice facilities respectively and, $O \setminus O'$ is partitioned into $O_g, O_b$ and $O_n$. Let $O_g$ denote the set of facilities in $O \setminus O'$ captured by good facilities. Let $O_b$ denote the set of facilities in $O \setminus O'$ captured by bad facilities, and let $O_n$ denote the set of facilities in $O \setminus O'$ not captured by any facility in $S_L \setminus S'$. Figure 1(b) shows the relationship between $S_g, O_g$ and $S_b, O_b$.

Figure 1: (a) Relationship between the partitions of $S_{sp}$ and $D_{sp} \cup O_{sp}$. (b) Relationship between the partitions of $S_g$ and of $O_g, S_b$.
1. For every \( s \in S_n (|O_d(s)| = 0) \): So nothing.

2. For every \( s \in S_g (|O_d(s)| = 1) \): Perform \( \text{swap}(s, O_d(s)) \). Update \( S_W, O_W, O' \) and \( S' \) as \( S_W = S_W \cup S_g, O_W = O_W \cup O_g, O' = O' \cup O_W, S' = S' \cup S_W \) in this order.

3. For every \( s \in S_b (|O_d(s)| = 0) \): Set \( \hat{S} = \hat{S} \cup \{s\}, \hat{O} = \hat{O} \cup O_d(s), O' = O' \cup \hat{O}, S' = S' \cup \hat{S} \) in this order. That is, put the bad facility in \( \hat{S} \) and the facilities dominated by it in \( \hat{O} \). The cardinality of \( \hat{S} \) increased by 1 while the cardinality of \( \hat{O} \) increased by at least 2.

New partitions are shown in Figure 2(b). Let \( \bar{S} \) be the set of facilities in \( S_L \) that have not participated in any swap. Then such a facility is either a nice facility, a bad facility, is in a nice pair or in a bad pair. Similarly, let \( \bar{O} \) be the set of facilities in \( O \) that have not participated in the above swaps. Then, \( O_B \) is the set of facilities in \( O \) that are in a triplet. Let \( T = \bar{O} \setminus O_B \). Then facilities in \( T \) are either dominated by a bad facility, by a bad pair, or are not dominated by any facility or a pair. i.e., \( T = \bar{O} \setminus O_n \). Let \( \ell \) be the number of such facilities i.e., \( \ell = |T| \). Next claim shows that there are at least \( \ell/3 \) nice facilities and nice pairs taken together.

**Claim 6.** \( |B| + |S_n| \geq \frac{1}{3} \left( |\hat{O}| + |O_n| \right) \)

**Proof.** While handling \( O_b \), for every facility added in \( \hat{S} \) atleast 2 facilities are added in \( \hat{O} \) and while handling \( O_{sp} \), atleast 3 facilities are added in \( \hat{O} \) for every 2 facilities added in \( \hat{S} \). Thus we have \( 3 \cdot |\hat{S}| \leq 2 \cdot |\hat{O}| \Rightarrow |\hat{S}| \leq \frac{2}{3} |\hat{O}| \)

Also \( |O| = k \leq |S_L|, O = O' \cup O_n \) and \( O' \cap O_n = \phi \)

Thus \( O = O_W \cup O_B \cup \hat{O} \cup O_n \)

Similarly \( S_L = S' \cup S_n \) and \( S' \cap S_n = \phi \)

Thus \( S_L = S_W \cup S_B \cup \hat{S} \cup S_n \)

Also \( |O_W| = |S_W|, |O_B| = |B|, |S_W| = 2 |B| \) and \( |\hat{S}| \leq \frac{2}{3} |\hat{O}| \)

Thus we get

\[
|S_W| + |\hat{S}| + 2 |B| + |S_n| \geq |O_W| + |B| + |\hat{O}| + |O_n| \\
|B| + |S_n| \geq |\hat{O}| - |\hat{S}| + |O_n| \geq \frac{1}{3} |\hat{O}| + |O_n|
\]
\[ |\mathcal{B}| + |\mathcal{S}_n| \geq \frac{1}{3} \left( |\hat{\mathcal{O}}| + |\mathcal{O}_n| \right) \]

Next, consider the following \( \ell \) swaps in which the facilities in \( \hat{\mathcal{O}} \) are swapped with nice facilities or nice pairs in \( \mathcal{S}_L \) in a way that each nice facility or a facility in a nice pair is considered in at most 3 swaps and each facility in \( \mathcal{O}_B \) is also considered in at most 3 swaps.

1. Repeat until \(|T| < 3\). Pick \( o_1, o_2, o_3 \in T \).
   
   (a) If \( \mathcal{S}_n \neq \emptyset \). Pick a facility \( s_1 \in \mathcal{S}_n \); perform \( \text{swap}(s_1, o_1), \text{swap}(s_1, o_2), \text{swap}(s_1, o_3) \). Set \( \mathcal{S}_n = \mathcal{S}_n \setminus \{s_1\} \).
   
   (b) Else, pick a triplet \(<s_1, s_2, o>\in \mathcal{B}\), and perform \( \text{double-swap}({s_1, s_2}, \{o, o_1\}), \text{double-swap}({s_1, s_2}, \{o, o_2\}), \text{double-swap}({s_1, s_2}, \{o, o_3\}) \).
   
   (c) Set \( T = T \setminus \{o_1, o_2, o_3\} \)

2. If \(|T| > 0\), either there must be a facility \( s_1 \in \mathcal{S}_n \) or a triplet \(<s_1, s_2, o>\in \mathcal{B}\); accordingly perform swap or double-swap with the facilities in \( T \) in the same manner as described in step 1.

The swaps are summarized in Figure 4 and Figure 3.

![Figure 3](image_url)

Figure 3: (a) Partitions of \( \mathcal{S}_L \) and \( \mathcal{O} \) in terms of \( \mathcal{S}_W, \mathcal{S}_B, \hat{\mathcal{S}}, \mathcal{S}_n \) and \( \mathcal{O}_W, \mathcal{O}_B, \hat{\mathcal{O}}, \mathcal{O}_n \) respectively. (b) Swaps

### 2.3 Analysis: Bounding the Cost

Now we bound the cost of these swaps. Whenever we consider a swap of form \( \text{double-swap}({s_1, s_2}, \{o_1, o_2\}) \), the mapping \( \tau \) as defined in claim [1] cannot be used to reassign the clients of \( s_1 \) and \( s_2 \) as it is possible that some client \( j \) of \( s_1 \) is mapped to a client of \( s_2 \) or vice versa. To address this, we define another mapping \( \tau' \) in a similar way as in claim [2] considering \( s_1 \) and \( s_2 \) as a single facility. Let \( DS = \{\{s_1, s_2\} : \text{double-swap}({s_1, s_2}, \{\ldots\}) \text{ was performed} \} \). \( \tau' \) satisfy the following claim
Claim 7. 1. For all \( o_1 \in O_s \) let \( P(o_1) = \{s_1, s_2\} \).

(a) If \( |D(o_1)| = 1 \) then \( D(o_1) = \{o_1\} \). \( P(o_1) \) is a nice pair. Set \( B = B \cup \{<s_1, s_2, o_1>\}, O_B = O_B \cup D(o_1), S_B = S_B \cup P(o_1) \).

(b) If \( |D(o_1)| = 2 \), let \( D(o_1) = \{o_1, o_2\} \) \( P(o_1) \) is a good pair. Consider double-swap(\( P(o_1), D(o_1) \)). Set \( O_W = O_W \cup D(o_1), S_W = S_W \cup P(o_1) \).

(c) If \( |D(o_1)| > 2 \) then set \( \hat{O} = \hat{O} \cup D(o_1), \hat{S} = \hat{S} \cup P(o_1) \).

2. For every \( o \in O_g \), consider swap(\( s, o \)), where \( s : O_g(s) = o \). Update \( S_W, O_W, O' \) and \( S' \) as \( S_W = S_W \cup S_g, O_W = O_W \cup O_g, O' = O' \cup O_W, S' = S' \cup S_W \) in this order.

3. Next consider \( O_h \) and \( S_h \). Update \( \hat{S}, \hat{O}, O' \) and \( S' \) as \( \hat{S} = \hat{S} \cup S_h, \hat{O} = \hat{O} \cup O_h, O' = O' \cup \hat{O}, S' = S' \cup \hat{S} \) in this order.

4. Let \( T = \hat{O} \cup O_n \). Consider the following swaps:

(a) Repeat until \( |T| < 3 \). Pick \( o_1, o_2, o_3 \in T \).

\begin{itemize}
  \item[i.] If \( S_n \neq \phi \),_pick a facility \( s_1 \in S_n \); perform \( \text{swap}(s_1, o_1), \text{swap}(s_1, o_2), \text{swap}(s_1, o_3) \) Set \( S_n = S_n \setminus \{s_1\} \).
  \item[ii.] Else, pick a triplet \( <s_1, s_2, o > \) \( B \), and perform \( \text{double-swap}(\{s_1, s_2\}, \{o, o_1\}), \text{double-swap}(\{s_1, s_2\}, \{o, o_2\}) \).
  \item[iii.] Set \( T = T \setminus \{o_1, o_2, o_3\} \)
\end{itemize}

(b) If \( |T| > 0 \), either there must be a facility \( s_1 \in S_n \) or a triplet \( <s_1, s_2, o > \) \( B \); accordingly perform swap or double-swap with the facilities in \( T \) in the same manner as described in step 4a.

Figure 4: Summary of the swaps

Claim 7. 1. For \( s \in S_L \) and \( o \in O | o \notin O_d(s) \)

\begin{itemize}
  \item[(a)] \( \tau'(\mathcal{B}(s, o)) \cap \mathcal{B}(s, o) = \phi \).
  \item[(b)] If \( o \notin O_s \) then \(|\{j \in \mathcal{B}(s, o) : \tau'(j) \in \mathcal{B}(s', o)\}| \leq \frac{2}{3} U, \forall s' \neq s \).
\end{itemize}

2. For \( \{s_1, s_2\} \in DS \) and \( o \in O | o \notin O_d(\{s_1, s_2\}) \)

\begin{itemize}
  \item[(a)] \( \tau'(\mathcal{B}(\{s_1, s_2\}, o)) \cap \mathcal{B}(\{s_1, s_2\}, o) = \phi \).
  \item[(b)] If \( o \notin O_s \) then \(|\{j \in \mathcal{B}(\{s_1, s_2\}, o) : \tau'(j) \in \mathcal{B}(s', o)\}| \leq \frac{2}{3} U, \forall s' \neq s_1, s_2 \).
\end{itemize}
Proof. \( \tau' \) can be defined as follows. Consider \( s_1 \) and \( s_2 \) as a single meta-facility, and define the mapping just as earlier. That is no \( j \in B(s_1, o) \) get mapped to \( j' \in B(s_2, o) \) and vice versa. Note that this is possible to create such a \( \tau' \) as \( \{s_1, s_2\} \) together do not dominate \( \hat{o} \).

For \( 2a \), as \( o \notin O_d(\{s_1, s_2\}) \); same argument ensures that \( \tau'(B(\{s_1, s_2\}, o) \cap B(\{s_1, s_2\}, o) = \phi \) still hold true For \( 2b \), as \( o \notin O_{sp} \), then at most one facility can cover \( o \). If \( v(\{s_1, s_2\}) = o \) then for all \( s' \neq s, B(s', o) \leq \frac{2}{5} U \). And if \( v(\{s_1, s_2\}) \neq o \) then \( B(\{s_1, s_2\}, o) \leq \frac{2}{5} U \). In either case the claim \( |\{j \in B(\{s_1, s_2\}, o) : \tau'(j) \in B(s', o)| \leq \frac{2}{5} U, \forall s' \neq s_1, s_2 \) holds true.

Set \( \tau = \tau' \). For \( swap(s_1, o_1) \) reassignment is done as follows: for \( j \in B_O(o_1) \) assign \( j \) to \( o_1 \) and for \( j \in B_S(s_1) \setminus B_O(o_1) \), assign \( j \) to \( \pi_S(\tau(j)) \). The cost of the operation is

For \( j \in B_S(s_1) \setminus B_O(o_1), \pi_S(\tau(j)) \neq \tau_S(j) \). Thus, since \( j \) was assigned to \( s \) and not to \( \pi_S(\tau(j)) \) (call it \( s' \)), \( c(j, s) \leq c(j, s') \) (since \( s' \), being a light facility, had sufficient room to accommodate \( j \) ). Thus, \( c(j, s) \leq c(j, s') \leq c(j, o') + c(o', \tau(j)) + c(\tau(j), s') \) or \( S_j \leq (O_j + O_{\tau(j)} + S_{\tau(j)}) \). However, if \( j \in B_S(s_1) \cap B_O(o_1), \pi_S(\tau(j)) \) may be same as \( \tau_S(j) \). \( S_j \leq (O_j + O_{\tau(j)} + S_{\tau(j)}) \) follows trivially in this case by triangle inequality. Thus we can write

\[
\sum_{j \in B_O(o_1)} (O_j - S_j) + \sum_{j \in B_S(s_1)} (O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) > 0 \quad (1)
\]

For \( double-swap(\{s_1, s_2\}, \{o_1, o_2\}) \), reassignment is defined as, assign \( j \in B_O(o_1) \) to \( o_1 \), assign \( j \in B_O(o_2) \) to \( o_2 \), and assign \( j \in B_S(s_1) \cup B_S(s_2) \setminus B_O(o_1) \cup B_O(o_2) \) to \( \pi_S(\tau^*(j)) \). The cost of the operation is

\[
\sum_{j \in B_O(o_1) \cup B_O(o_2)} (O_j - S_j) + \sum_{j \in B_S(s_1) \cup B_S(s_2) \setminus B_O(o_1) \cup B_O(o_2)} (O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) > 0 \quad (2)
\]

Also, by similar argument as above, we have

\[
\sum_{j \in B_O(o_1) \cup B_O(o_2)} (O_j - S_j) + \sum_{j \in B_S(s_1) \cup B_S(s_2)} (O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) > 0 \quad (3)
\]

Now, a facility \( o \in O \) may be considered at max three times. Let \( O_1, O_2, O_3 \) denote the set of facilities in \( O \) that are considered in 1, 2 and 3 swaps respectively. A facility \( s \in S_L \) may be swapped out at most 3 times. Thus, we can
write
\[
\sum_{o \in O_1} \sum_{j \in B_0(o)} (O_j - S_j) + \sum_{o \in O_2} \sum_{j \in B_0(o)} 2(O_j - S_j) + \sum_{o \in O_2} \sum_{j \in B_0(o)} 3(O_j - S_j) + \\
\sum_{s \in S_L} \sum_{j \in B_S(s)} 3.(O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) > 0 \quad (4)
\]
as \(O_j + O_{\tau(j)} + S_{\tau(j)} - S_j > 0\) as argued above.

\[
\sum_{o \in O_1} \sum_{j \in B_0(o)} (S_j) + \sum_{o \in O_2} \sum_{j \in B_0(o)} 2(S_j) + \sum_{o \in O_2} \sum_{j \in B_0(o)} 3(S_j) < \\
\sum_{o \in O_1} \sum_{j \in B_0(o)} (O_j) + \sum_{o \in O_2} \sum_{j \in B_0(o)} 2(O_j) + \sum_{o \in O_2} \sum_{j \in B_0(o)} 3(O_j) + \\
\sum_{s \in S_L} \sum_{j \in B_S(s)} 3.(O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) \quad (5)
\]

\[
\sum_{j \in C} (S_j) < \sum_{j \in C} 3(O_j) + \sum_{s \in S_L} \sum_{j \in B_S(s)} 3.(O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) \quad (6)
\]

\[
\text{cost}(S) < 3\text{cost}(O) + \sum_{s \in S_L} \sum_{j \in B_S(s)} 3.(O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) \quad (8)
\]

And as \(\tau\) is a 1−1 and onto mapping,

\[
\sum_{s \in S_L} \sum_{j \in B_S(s)} O_j = \sum_{s \in S_L} \sum_{j \in B_S(s)} O_{\tau(j)}
\]

and

\[
\sum_{s \in S_L} \sum_{j \in B_S(s)} (S_{\tau(j)} - S_j) = 0
\]

Thus,

\[
\sum_{s \in S_L} \sum_{j \in B_S(s)} 3.(O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) = 2 \sum_{s \in S_L} \sum_{j \in B_S(s)} 3.(O_j) = 6 \sum_{s \in S_L} \sum_{j \in B_S(s)} (O_j) \leq \\
6 \sum_{s \in S} \sum_{j \in B_S(s)} (O_j) = 6\text{cost}(O) \quad (9)
\]
Thus we have

\[
\text{cost}(S) \leq 3\text{cost}(O) + \sum_{s \in S} \sum_{j \in B_S(s)} 3(O_j + O_{r(j)} + \hat{S}_{r(j)} - S_j) \\
\leq 3\text{cost}(O) + 6\text{cost}(O) = 9\text{cost}(O) \quad (10)
\]

3 \ (3 + \epsilon, 8/3) algorithm using multi-swaps

In this section, we reduce the factor to \((3 + \epsilon)\) using multi-swap operation. Let \(p = 2p' + 2\) for some integer \(p'\). Let \(p > 2\). The algorithm performs the following operation if it reduces the cost of the solution and it terminates otherwise.

\[
\text{multi-swap}(A, B) : S = S \setminus B \cup A; B \subseteq S, A \subseteq F, |A| = |B| \leq p
\]

The operation can be performed in \(O(n^p)\) time. For a fixed \(\epsilon > 0\) and \(p = O(1/\epsilon)\), it runs in \(O(n^{1/\epsilon})\) time.

3.1 Defining the swaps

Following swaps are considered. Partitions are formed and the facilities in \(O_{sp}\) and \(O_W\) are treated in the same manner as described in Section 2.2. We write inequalities corresponding to different operations and take their weighted sums. Inequalities corresponding to the swaps defined for facilities in \(O_{sp}\) and \(O_W\) are assigned weight 1.

Let \(T_O = O_B \cup \hat{O} \cup O_b \cup O_n\) and \(T_S = S_B \cup \hat{S} \cup S_b \cup S_n\). \(T_S\) and \(T_O\) are partitioned into \(A_1, A_2, \ldots, A_r\) and \(B_1, B_2, \ldots, B_r\) respectively using the Partition Algorithm. such that the partitions satisfy the following properties:

1. For \(1 \leq i \leq (r - 1)\), we have \(|A_i| = |B_i|, B_i = O_d(A_i)\) and \(|A_r| = |B_r|\).
2. For \(1 \leq i \leq (r - 1)\), the set \(A_i\) has exactly one facility \(s\) from \(S_b\) or exactly one bad pair \(<s_1, s_2>\) from \(\hat{S}\).
3. The set \(A_r\) contains facilities only from \(S_B \cup S_n\).

Next, we define the following swaps:

1. For the sets \(A_i, B_i\) for some \(i \leq i \leq (r - 1)\), such that \(|A_i| = |B_i| \leq p\), perform \text{multi-swap}(A_i, B_i): for all \(o \in B_i, j \in B_O(o)\), reassign \(j\) to \(o\), for all \(j \in (\cup_{s \in A_i} B_S(s)) \setminus (\cup_{o \in B_O(B_O(o))})\), assign \(j\) to \(\pi_S(\tau(j))\). Inequalities are assigned weight 1.
2. For the sets \(A_i, B_i\) for some \(i \leq i \leq (r - 1)\), such that \(|A_i| = |B_i| = q > p\), we perform \text{shrinking} as follows: for every \(<s_1, s_2, o_1, \in \hat{B}\) such that \(s_1, s_2 \in A_i\) and \(o_1 \in B_i\) \textbf{do}, \(A_i = A_i \setminus \{s_1, s_2\}, B_i = B_i \setminus o_1, A_i = A_i \cup s_1^*\), where \(s_1^*\) denote a meta node corresponding to nodes \(s_1, s_2\). Note that after shrinking we still have \(|A_i| = |B_i| = q' \geq q/2\). Also as \(q > p = 2p' + 2\), we have \(q' > p' + 1 = p/2\). We consider the swaps as follows

15
Algorithm 1 Partition Algorithm

1: \( i = 0. \)
2: while \( \exists \) a facility in \( \hat{S} \cup S_0 \) do
3: \( i = i + 1 \)
4: \( A_i = \{ s \} \) where \( s \in S_0 \) or \( A_i = \{ s_1, s_2 \} \) where \( \{ s_1, s_2 \} \) is a bad pair from \( \hat{S} \).
5: \( B_i = \mathcal{O}_d(A_i) \)
6: while \( |A_i| \neq |B_i| \) do
7: \( A_i = A_i \cup \{ q \} \) where \( g \in T_S \cap \{ S_B \cup S_a \} \setminus A_i. \)
8: \( B_i = \mathcal{O}_d(A_i) \)
9: end while
10: \( T_O = T_O \setminus B_i, \) \( T_S = T_S \setminus A_i. \)
11: end while
12: \( A_r = T_S, B_r = T_O. \)

(a) If \( s \in A_i : s \in S_0 \) then we consider exactly \( q'(q' - 1) \) swaps as follows: for all \( o \in B_i, \) for all \( s' \in A_i \setminus \{ s \}, \) if \( s' = s_j \) was created by shrinking of \( < s_{j_1}, s_{j_2}, o_{j_1} >, \) then we perform \( \text{double-swap}(\{ s_{j_1}, s_{j_2} \}, \{ o_{j_1}, o \}). \)

If \( s' \) was not created by shrinking, then we perform \( \text{swap}(s', o). \)

Each inequality is assigned weight \( 1/(q' - 1). \) Then, each \( s' \in A_i \) participates in exactly \( (q')/(q' - 1) = 1 + 1/(q' - 1) \leq 1 + 1/p' = 1 + 2/(p - 2) \) swaps where as each \( o \in B_i \) participate in exactly 1 swap except for facilities of the type \( o_j, \) which participates in as many swaps as \( s_{j_1}, s_{j_2} \) do, which is no more than \( 1 + 2/(p - 2). \)

(b) If \( s_1, s_2 \in A_i : \{ s_1, s_2 \} \) is a bad pair from \( \hat{S} \) then we consider exactly \( q'(q' - 2) \) swaps as follows: for all \( o \in B_i, \) for all \( s' \in A_i \setminus \{ s_1, s_2 \}, \)

if \( s' = s_j \) was created by shrinking of \( < s_{j_1}, s_{j_2}, o_{j_1} >, \) then we perform \( \text{double-swap}(\{ s_{j_1}, s_{j_2} \}, \{ o_{j_1}, o \}). \)

If \( s' \) was not created by shrinking, then we perform \( \text{swap}(s', o). \)

Each inequality is assigned weight \( 1/(q' - 2). \) Then, each \( s' \in A_i \) participates in exactly \( (q')/(q' - 2) = 1 + 2/(q' - 2) \leq 1 + 2/p' = 1 + 4/(p - 2) \) swaps where as each \( o \in B_i \) participate in exactly 1 swap except for facilities of the type \( o_j, \) which participates in as many swaps as \( s_{j_1}, s_{j_2} \) do, which is no more than \( 1 + 4/(p - 2). \)

The cost of the swaps can be analysed as follows. Every \( s \in S \) is swapped out at most \( 1 + 4/(p - 2) \) times and every \( o \in \mathcal{O} \) is swapped at least once and at most \( 1 + 4/(p - 2) \) times. Thus we can now write

\[
\sum_{j \in \mathcal{C}} (S_j) < (1 + 4/(p - 2)) \sum_{j \in \mathcal{C}} (O_j) + (1 + 4/(p - 2)) \sum_{s \in S_L} \sum_{j \in B_S(s)} \{ O_j + O_{T(j)} + S_{T(j)} - S_j \}
\]

which gives

\[
(11)
\]
\[
\sum_{j \in C} (S_j) < (1+4/(p-2)) \text{cost}(\mathcal{O}) + (1+4/(p-2))(2 \text{cost}(\mathcal{O})) = (3+12/(p-2)) \text{cost}(\mathcal{O})
\]  

(12)

4 Capacitated \( k \)-Median with Penalties

CkMP is a variation of CkM where we also have a fixed penalty cost \( p_j \) associated with each client \( j \in C \). For a solution \( S \), let \( \mathcal{P}(S) \) denote the set of clients that pay penalties in \( S \). The clients \( \mathcal{C} \setminus \mathcal{P}(S) \) are serviced by the facilities opened in \( S \).

We borrow the notations from Section 2. Then, \( \text{cost}(S) \) is \( \sum_{s \in S} \sum_{j \in B(s)} S_j + \sum_{j \in \mathcal{P}(S)} p_j \). For easy disposition of ideas, we give \( (9 + \epsilon, 8/3) \) algorithm and its analysis. The factor is reduced to \( (3 + \epsilon) \) in exactly the same manner as is done in Section 3.

4.1 \( (9 + \epsilon, 8/3) \) Algorithm

We start with an initial feasible solution \( S_0 \) such that \( |S| = 8k/3 \). The clients are assigned by solving min cost flow problem over the facilities \( \mathcal{S}_0 \cup \{\delta\} \), where \( u_\delta = |C| \) and \( \forall j \in C, c(\delta, j) = p_j \). Clients assigned to \( \delta \) pay penalty in the solution \( S_0 \). The operations available to the algorithm are swap \( (s, o) \) and double-swap \( \{s_1, s_2\}, \{o_1, o_2\} \). Let \( S \) be the locally optimal solution with \( \text{cost}(S) = \sum_{s \in S} \sum_{j \in B(s)} S_j + \sum_{j \in \mathcal{P}(S)} p_j \).

4.2 Analysis

Let \( \mathcal{O} \) be an optimal solution for the problem with \( \text{cost}(\mathcal{O}) = \sum_{o \in \mathcal{O}} \sum_{j \in B_{\mathcal{O}}(o)} O_j + \sum_{j \in \mathcal{P}(\mathcal{O})} p_j \). Swaps are defined exactly in the same manner as done in Section 2. However, there is a slight change in the re-assignment. Consider the \( \text{swap}(s, o) : s \in \mathcal{S}_t \) and \( o \in \mathcal{O} \); reassignments are done as follows: \( \forall j \in B_{\mathcal{O}}(o), \) assign \( j \) to \( o; \forall j \in B_S(s) \cap \mathcal{P}(\mathcal{O}), \) \( j \) pays penalty \( p_j \) and \( \forall j \in B_S(s) \setminus \{B_{\mathcal{O}}(o) \cup \mathcal{P}(\mathcal{O})\}, \) \( j \) is assigned to \( \pi_S(\tau(j)) \). As \( S \) is locally optimal, we have

\[
\sum_{j \in B_{\mathcal{O}}(o) \setminus \mathcal{P}(S)} (O_j - S_j) + \sum_{j \in B_{\mathcal{O}}(o) \cap \mathcal{P}(S)} (O_j - p_j) + \sum_{j \in B_S(s) \setminus \mathcal{P}(\mathcal{O})} (p_j - S_j) + \sum_{j \in B_S(s) \setminus \mathcal{B}_{\mathcal{O}}(o) \setminus \mathcal{P}(\mathcal{O})} (O_j + S_{\tau(j)} + S_{\tau(j)} - S_j) > 0
\]  

(13)
Similarly for double-swap\(\{s_1, s_2\}, \{o_1, o_2\}\), we have

\[
\sum_{j \in B_{O}(o_1) \cup B_{O}(o_2)} (O_j - S_j) + \sum_{j \in \{B_{O}(o_1) \cup B_{O}(o_2)\} \cap \mathcal{P}(S)} (O_j - p_j) + \\
\sum_{j \in \{B_{S}(s_1) \cup B_{S}(s_2)\} \cap \mathcal{P}(O)} (p_j - S_j) + \\
\sum_{j \in \{B_{S}(s_1) \cup B_{S}(s_2)\} \setminus \{B_{O}(o_1) \cup B_{O}(o_2)\} \cap \mathcal{P}(O)} (O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) > 0 \quad (14)
\]

and a facility \(s \in S_L\) may be swapped out at most 3 times. Thus summing over all swaps we have

\[
\sum_{o \in O_1} \left( \sum_{j \in B_{O}(o) \setminus \mathcal{P}(S)} (O_j - S_j) + \sum_{j \in \mathcal{B}_{O}(o) \cap \mathcal{P}(S)} (O_j - p_j) \right) + \\
\sum_{o \in O_2} \left( \sum_{j \in B_{O}(o) \setminus \mathcal{P}(S)} 2(O_j - S_j) + \sum_{j \in \mathcal{B}_{O}(o) \cap \mathcal{P}(S)} 2(O_j - p_j) \right) + \\
\sum_{o \in O_3} \left( \sum_{j \in B_{O}(o) \setminus \mathcal{P}(S)} 3(O_j - S_j) + \sum_{j \in B_{S}(o) \cap \mathcal{P}(S)} \sum_{s \in S_L} 3(O_j - p_j) \right) + \\
\sum_{s \in S_L} \sum_{j \in B_{S}(s) \cap \mathcal{P}(O)} 3(p_j - S_j) + \sum_{s \in S_L} \sum_{j \in B_{S}(s) \setminus \mathcal{P}(O)} 3(O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) > 0 \quad (15)
\]

Note that for all \(s \in S\), \(p_j - S_j \geq 0\) for all \(j \in B_{S}(s) \cap \mathcal{P}(O)\) otherwise \(j\) would have paid penalty in \(S\) too. Thus we can write \(\sum_{s \in S_L} \sum_{j \in B_{S}(s) \cap \mathcal{P}(O)} 3(p_j - S_j) \leq \sum_{s \in S} \sum_{j \in B_{S}(s) \cap \mathcal{P}(O)} 3(p_j - S_j)\)

Rearranging, we get

\[
\sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in B_{O}(o) \setminus \mathcal{P}(S)} S_j + \sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in \mathcal{B}_{O}(o) \cap \mathcal{P}(S)} p_j + \sum_{j \in \mathcal{P}(O) \setminus \mathcal{P}(S)} S_j \leq \\
\sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in B_{O}(o) \setminus \mathcal{P}(S)} 3O_j + \sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in \mathcal{B}_{O}(o) \cap \mathcal{P}(S)} 3O_j + \sum_{j \in \mathcal{P}(O) \setminus \mathcal{P}(S)} 3p_j + \\
+ \sum_{s \in S_L} \sum_{j \in B_{S}(s) \setminus \mathcal{P}(O)} 6O_j \quad (16)
\]

as for \(\sum_{s \in S_L} \sum_{j \in B_{S}(s) \setminus \mathcal{P}(O)} (O_j + O_{\tau(j)} + S_{\tau(j)} - S_j) = \sum_{s \in S_L} \sum_{j \in B_{S}(s) \setminus \mathcal{P}(O)} 2O_j\)

by the property of \(\tau\).

Adding \(\sum_{j \in \mathcal{P}(S) \setminus \mathcal{P}(O)} p_j\) on both the sides and re-arranging, we get
∑ \sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in B o \setminus P(S)} S_j + \sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in B o \setminus P(S)} p_j + \sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in B o \setminus P(S)} S_j + \\
\sum_{j \in P(S) \cap P(O)} \sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in B o \setminus P(S)} 3O_j + \sum_{o \in O_1 \cup O_2 \cup O_3} \sum_{j \in B o \setminus P(S)} 3O_j \\
+ \sum_{j \in P(S) \setminus P(O)} 3p_j + \sum_{o \in S_L} \sum_{j \in B o \setminus P(O)} 6O_j + \sum_{j \in P(S) \cap P(O)} p_j 
(17)

Rearranging the terms, we get

\sum_{s \in S} \sum_{j \in B_o \setminus P(O)} S_j + \sum_{j \in P(S) \setminus P(O)} \sum_{s \in S_L} \sum_{j \in B_o \setminus P(O)} 3O_j + \sum_{j \in P(S) \setminus P(O)} 3p_j \\
+ \sum_{j \in P(S) \setminus P(O)} 3O_j + \sum_{j \in P(S) \setminus P(O)} \sum_{s \in S_L} \sum_{j \in B_o \setminus P(O)} 6O_j 
(18)

which gives

\sum_{s \in S} \sum_{j \in B_o \setminus P(O)} S_j + \sum_{j \in P(S)} p_j \leq \sum_{o \in O} \sum_{j \in B_o \setminus P(O)} 3O_j + \sum_{j \in P(O)} 3p_j + \sum_{s \in S_L} \sum_{j \in B_o \setminus P(O)} 6O_j \\
\leq \sum_{o \in O} \sum_{j \in B_o \setminus P(O)} 9O_j + \sum_{j \in P(O)} 3p_j = 9\text{cost}(O) 
(19)

5 Conclusion

In this paper, we presented ((3 + \epsilon), 8/3) factor algorithm for Capacitated k-Median Problem and its penalty version with uniform capacities, using local search heuristic. There is a trade-off between the approximation factor and the cardinality violation between our work and the existing work. Work in [22] is closest to our work as it is the only result for the problem based on local search. We improve upon the results in [22] in terms of cardinality violation.

It would be interesting to see how these results extend to the problems with non-uniform capacities.

References

[1] Karen Aardal, Pieter L. van den Berg, Dion Gijswijt, and Shanfei Li. Approximation algorithms for hard capacitated k-facility location problems. EJOR, 242(2):358–368, 2015.
[2] Ankit Aggarwal, Louis Anand, Manisha Bansal, Naveen Garg, Neelima Gupta, Shubham Gupta, and Surabhi Jain. A 3-approximation for facility location with uniform capacities. In Proceedings of 14th International Conference on Integer Programming and Combinatorial Optimization (IPCO), Lausanne, Switzerland, 2010.

[3] Aaron Archer, Ranjithkumar Rajagopalan, and David B. Shmoys. Lagrangian relaxation for the k-median problem: new insights and continuity properties. In Proceedings of the 11th Annual European Symposium on Algorithms, pages 31–42, 2003.

[4] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In Proceedings on 33rd Annual ACM Symposium on Theory of Computing, Heraklion, Crete, Greece, pages 21–29, 2001.

[5] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. SIAM Journal of Computing, 33(3):544–562, 2004.

[6] Manisha Bansal, Naveen Garg, and Neelima Gupta. A 5-approximation for capacitated facility location. In Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings, pages 133–144, 2012.

[7] Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, and David Williamson. A note on the prize collecting traveling salesman problem. Mathematical Programming, 59(1):413–420, Mar 1993.

[8] Jaroslaw Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In SODA 2015, pages 722–736.

[9] Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Joachim Spoerhase, Aravind Srinivasan, and Khoa Trinh. An improved approximation algorithm for knapsack median using sparsification. In ESA 2015, pages 275–287.

[10] Jaroslaw Byrka, Bartosz Rybicki, and Sumedha Uniyal. An approximation algorithm for uniform capacitated k-median problem with (1 + ϵ) capacity violation. In Integer Programming and Combinatorial Optimization - 18th International Conference, IPCO 2016, Li`ege, Belgium, June 1-3, 2016, Proceedings, pages 262–274, 2016.

[11] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS), New York, NY, USA, pages 378–388, 1999.
[12] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the $k$-median problem (extended abstract). In STOC, 1999, pages 1–10.

[13] Fabián A. Chudak and David P. Williamson. Improved approximation algorithms for capacitated facility location problems. Journal of Mathematical Programming, 102(2):207–222, March 2005.

[14] H. Gökalp Demirci and Shi Li. Constant approximation for capacitated $k$-median with $(1+\epsilon)$-capacity violation. In 43rd ICALP 2016, pages 73:1–73:14.

[15] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. SIAM J. Comput., 24(2):296–317, April 1995.

[16] Sapna Grover, Neelima Gupta, and Aditya Pancholi. Constant factor approximation algorithms for uniform hard capacitated facility location problems: Natural LP is not too bad. CoRR, abs/1606.08022, 2016.

[17] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. CoRR, abs/0809.2554, 2008.

[18] Neelima Gupta and Shubham Gupta. Approximation algorithms for capacitated facility location problem with penalties. CoRR, abs/1408.4944, 2014.

[19] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. Journal of ACM, 50(6):795–824, November 2003.

[20] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In STOC, 2002, pages 731–740.

[21] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and $k$-median problems using the primal-dual schema and lagrangian relaxation. J. ACM, 48(2):274–296, March 2001.

[22] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. Journal of Algorithms, 37(1):146–188, 2000.

[23] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. Journal of Management Science, 9(4):pp. 643–666.

[24] Shanfei Li. An Improved Approximation Algorithm for the Hard Uniform Capacitated $k$-median Problem. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014), Germany, pages 325–338.
[25] Shi 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 2015, San Diego, CA, USA, January 4-6, 2015, pages 696–707, 2015.

[26] Shi 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 2016, Arlington, VA, USA, January 10-12, 2016, pages 786–796, 2016.

[27] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In Proceedings of Symposium on Theory of Computing Conference (STOC), Palo Alto, CA, USA, pages 901–910, 2013.

[28] Charikar M., Khuller S., and Mount D. M. and Narasimhan G. Algorithms for facility location problems with outliers. pages 642–651, 2001.

[29] M. Pál, É. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS), Las Vegas, Nevada, USA, pages 329–338, 2001.

[30] Jens Vygen. From stars to comets: Improved local search for universal facility location. Journal of Operations Research Letters, 35(4):427–433, 2007.

[31] Guang Xu and Jinhui Xu. An lp rounding algorithm for approximating uncapacitated facility location problem with penalties. Information Processing Letters, 94(3):119–123, 2005.

[32] Yicheng Xu, Dachuan Xu, Donglei Du, and Chenchen Wu. Local search algorithm for universal facility location problem with linear penalties. Journal of Global Optimization, 67(1):367–378, Jan 2017.

[33] Jiawei Zhang, Bo Chen, and Yinyu Ye. A multiexchange local search algorithm for the capacitated facility location problem. Journal of Mathematics of Operations Research, 30(2):389–403, 2005.