A 5-approximation for Universal Facility Location

Manisha Bansal[1], Naveen Garg[2], and Neelima Gupta[3]

1 **University of Delhi**
2 **Indian Institute of Technology Delhi**
3 **University of Delhi**

Abstract

In this paper, we propose and analyze a local search algorithm for the *Universal facility location problem*. Our algorithm improves the approximation ratio of this problem from 5.83 [1] to 5. A second major contribution of the paper is that it gets rid of the expensive `multi` operation that was a mainstay of all previous local search algorithms for capacitated facility location and universal facility location problem. The only operations that we require to prove the 5-approximation are `add, open`, and `close`. `multi` operation is basically a combination of the `open` and `close` operations. 5-approximation algorithm for the capacitated facility location problem [2] also uses `multi`. However, on careful observation, it turned out that `add, open, and close` operations are sufficient to prove a 5- factor for the problem. This resulted into an improved algorithm for the universal facility location problem, with an improved factor.

1998 ACM Subject Classification Dummy classification – please refer to `http://www.acm.org/about/class/ccs98-html`

Keywords and phrases Facility location, Approximation Algorithms, Local Search

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

In a *facility location problem* we are given a set of clients $C = \{1, \ldots, m\}$ and a set of facilities $F = \{1, \ldots, n\}$. A client j has a demand d_j which needs to be serviced by some facilities, i.e., the demand is splittable. The cost of servicing a client $j \in C$ by a facility i is given by c_{ij} (the service cost). The service costs form a metric. Further let, for $i, i' \in F$ $i \neq i'$, $c_{ii'}$ be the cost of the shortest path between i and i', i.e. $c_{ii'} = \min_{j \in C}(c_{ji} + c_{ji'})$. For the sake of simplicity, we consider the case when demand of a client $j \in C$ is one. Arbitrary demands can be easily handled by doing slight modifications, details of which can be found in Pal *et al.* [4] and Mahdian *et al.* [3].

In the classical uncapacitated facility location problem (UFL), we are also given $f : F \to \Re^+$, and f_i is the cost of opening a facility at location i. In the capacitated version of the facility location problem, besides the cost of opening a facility at location i we are also given an upper bound u_i on the number of clients that can be served at location i. The Universal facility location (UniFL) problem is a further generalization of the capacitated facility location problem. Now the cost of opening a facility at $i \in F$ depends on the number of clients that this facility would served and is given by a cost function $f_i(.)$, which is a monotonically non-decreasing function of the capacity allocated to facility i. Thus if u_i is the capacity allocated to facility i, then $f_i(u_i)$ is its facility opening cost. The aim is to determine a capacity allocation vector $U = \langle u_1, u_2, \cdots, u_n \rangle$ such that the total cost of opening facilities and the cost of serving clients by the open facilities, while respecting capacity constraints, is minimized. Once the allocation vector U is known it is easy to determine the assignment of clients by solving a mincost flow problem. Therefore the capacity allocation vector U

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Acces; Article No. 23; pp. 23:1–23:12
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

completely determines the solution. Note that if $f_i(u_i) = \infty$ for $u_i > c_i$ and $f_i(u_i) = f_i$ otherwise, then we have an instance of the capacitated facility location problem where c_i is the fixed capacity of facility i.

The Universal facility location problem was introduced by Mahdian and Pal [3] who gave a 7.88-approximation algorithm which was improved by Vygen [5] to a 6.702-approximation. This has been further improved to a 5.83-approximation by Angel *et al.* [1]. Our main result in this paper, is a 5-approximation algorithm for this problem. Our algorithm extends our earlier 5-approximation algorithm for the capacitated facility location problem [2] and borrows heavily from that work. Bansal *et al.* [2] gave a local search algorithm for capacitated facility location Problem that uses the operations `add`, `mopen`, `mclose`, and `mmulti`. Our algorithm too uses the operations `add`, `mopen`, and `mclose` (which we call `open`, and `close` in this paper) but doesn't require the `mmulti` operaion. The `mmulti` operation which is a combination of an `open` and `close` operation is an expensive operation to perform and has appeared in some form in all previous works on universal facility location problem and capacitated facility location problem. By getting rid of this expensive operation we hope that our algorithm would be simpler to implement and faster in practice. This paper, thus, not only extends but also simplifies the result in [2]. Bansal *et al.* [2] argue that the locality gap of any procedure for the capacitated facility location problem that uses the operations `add, mopen, mclose, and mmulti` is atleast 5. This lower bound also applies in our setting since we consider a subset of these operations for a more general problem. When analyzing the cost of an operation we sometimes assign clients fractionally to the facilities. This can be done as it is well known that a fractional assignment cannot be better than the integral optimum assignment, in an assignment problem.

The remainder of this paper is organized as follows: We begin with some preliminaries and in section 3 we present the local search steps. In Section 4 we analyse the local search algorithm by identifying a suitable set of inequalities and finally in Section 5 we put the various pieces together to prove our main theorem. As mentioned earlier, our paper borrows heavily from ideas developed in [2, 6] and other previous work. At many places we have rephrased key arguments to keep the paper self-contained.

2 Preliminaries

A solution to the UniFL problem consists of a capacity allocation vector and an assignment of the clients to the facilities which obey capacity constraints. Let us consider an allocation vector $U = \langle u_1, u_2, \ldots, u_n \rangle$ for a given instance. We abuse notation and use U to denote both the solution and the allocation vector. The cost of a solution U is denoted by $c(U) = c_f(U) + c_s(U)$, where $c_f(U)$ is the facility cost and $c_s(U)$ is the service cost of the solution U.

Let U be a locally optimal solution and U^* be an optimum solution. For each $s \in F$, u_s (respectively u^*_s) denotes the capacity allocated to s in the locally optimal (respectively optimum) solution. Let F_U (respectively F_{U^*}) be the set of facilities for which u_s (respectively u^*_s) is greater than zero. It is no loss of generality to assume that u_s is the number of clients served by s in U, and u^*_s the no. of clients served by s in U^*.

Let $\sigma(j), \tau(j)$ be the facilities serving client j in solutions U, U^* respectively. Construct a bipartite graph, $G = (C \cup F, E)$ where E contains edges $(\sigma(j), j)$ and $(j, \tau(j))$. Thus each client has one incoming and one outgoing edge while a facility s has u_s outgoing edges and u^*_s incoming edges. The graph G is now decomposed into a set of maximal paths, $\mathcal{P}$, and cycles, $\mathcal{C}$. A path $P \in \mathcal{P}$ is a sequence of vertices $s = s_0, j_0, s_1, j_1, \ldots, s_k, j_k, s_{k+1} = o$, which

starts at a vertex $s \in F_U$ and ends at a vertex $o \in F_{U^*}$. Let head(P) denote the client served by s and tail(P) the client served by o on this path. Note that $\{s_1, s_2, \ldots, s_k\} \subseteq F_U \cap F_{U^*}$. Similarly all facilities on a cycle are from $F_U \cap F_{U^*}$.

The *length* of a path P is given by

$$\text{length}(P) = \sum_{j \in C \cap P} (U_j^* + U_j)$$

where U_j^*(respectively U_j) is the service cost of client j in the solution U^*(respectively U). Note that

$$\sum_{P \in \mathcal{P}} \text{length}(P) + \sum_{Q \in \mathcal{C}} \text{length}(Q) = c_s(U) + c_s(U^*).$$

A *shift* along a path P is a reassignment of clients so that j_i which was earlier assigned to s_i is now assigned to s_{i+1}. As a consequence of this shift, facility s serves one client less while facility o serves one client more. shift(P) denotes the increase in service cost due to a shift along P i.e.

$$\text{shift}(P) = \sum_{j \in C \cap P} (U_j^* - U_j).$$

For a cycle in $\mathcal{C}$ the increase in service cost equals the sum of $U_j^* - U_j$ for all clients j in the cycle and since the assignment of clients to facilities is done optimally in our solution and in the global optimum, this sum is zero. Thus

$$\sum_{Q \in \mathcal{C}} \sum_{j \in Q} (U_j^* - U_j) = 0.$$

Let N_s^o be the set of paths that begin at s and end at o. Define $\texttt{out}(s) = \cup_o N_s^o$ as the set of paths starting from s and $\texttt{in}(o) = \cup_s N_s^o$ as the set of paths ending at o. Since the paths chosen are maximal, for any s, at least one of the two sets $\texttt{in}(s)$,$\texttt{out}(s)$ is empty. Let S be the set of facilities for which $\texttt{in}(s)$ is empty and O the set of facilities for which $\texttt{out}(s)$ is empty. Hence, $S \cap O = \phi$. Note that for $s \in S$, $|\texttt{out}(s)| = u_s - u_s^*$ and for $s \in O$, $|\texttt{in}(s)| = u_s^* - u_s$.

Define a capacity function, $\hat{u}$, on the facilities as follows: $\hat{u}(s)$ equals $|\texttt{out}(s)| = u_s - u_s^*$ if $s \in S$, it equals $|\texttt{in}(s)| = u_s^* - u_s$ if $s \in O$ and is 0 for $s \in F \setminus (S \cup O)$. To bound the cost of facilities in S, we reduce the capacity of each facility $s \in S$ from u_s to u_s^* and reassign $u_s - u_s^* = \hat{u}(s)$ clients served by s to facilities in O. We refer to this step as *closing* facility s. Similarly, *opening* a facility $o \in O$ implies increasing its capacity by $u_o^* - u_o = \hat{u}(o)$. `open` and `close` operations are explained in the next section.

To formulate a suitable set of inequalities we formulate an assignment problem where each node $s \in S$ has supply $\hat{u}(s)$ and a node $o \in O$ has demand $\hat{u}(o)$. When a client served by s is transferred to o the increase in service cost is at most c_{so} and hence this is the cost of sending one unit of flow from node s to o. Let $y(s, o)$ be the flow from s to o in an optimum solution to this assignment problem.

▶ **Lemma 1.** *The cost of an optimum flow y is at most $c_s(U) + c_s(U^*)$.*

Proof. Consider a solution $\hat{y}$ defined as $\hat{y}(s, o) = |N_s^o|$. It is easy to check that this is a feasible solution to the assignment problem. Note that for every path $P \in N_s^o$, length$(P) \geq$

c_{so}. Hence

$$\begin{aligned}\sum_s \sum_o c_{so}\hat{y}(s,o) &\leq \sum_s \sum_o \sum_{P \in N_s^o} \text{length}(P) \\ &= \sum_{P \in \mathcal{P}} \text{length}(P) \\ &\leq c_s(U) + c_s(U^*)\end{aligned}$$

◀

It is easy to argue that there is an optimum flow y where the edges carrying non-zero flow form an acyclic subgraph. We call this subgraph as exchange graph. Let $G' = (S \cup O, E')$ where $E' = \{(s,o)|y(s,o) > 0\}$ and suppose G' is not acyclic. Let C be the edges on a cycle. Take alternate edges of C to form sets C_1, C_2. Let γ be the minimum flow on an edge in C. Consider two operations - one in which we increase the flow on edges in C_1 and decrease the flow on edges in C_2 by an amount γ and the other in which we do the inverse. In one of these operations the total cost $\sum_{s \in S, o \in O} c_{so}y(s,o)$ would not increase and the flow on one of the edges would reduce to zero thereby removing it from the graph. This process is continued till the graph becomes acyclic. Note that the total flow from nodes of S to a node $o \in O$ is equal to $\hat{u}(o)$ i.e., $\sum_{s \in S} y(s,o) = \hat{u}(o)$ and total flow from a node $s \in S$ to nodes in O is equal to$\hat{u}(s)$, i.e., $\sum_{o \in O} y(s,o) = \hat{u}(s)$.

3 The local search operations

Starting with a feasible solution U, we perform *add, open* and *close* operations to improve the solution U if possible. Given a solution U, we can assume that for each facility $i \in U$, u_i is exactly equal to the number of clients it is serving for if it is not true then we can reduce u_i and hence the cost of the solution. U is locally optimal if none of these operations improve the cost of the solution and at this point the algorithm stops. The *add* operation is the same as given by Mahdian and Pal [3] while the *open* and *close* are almost the same as in [2].

add(s, δ): In this operation the capacity allocated at a facility s is increased by an amount $\delta > 0$. A mincost flow problem is then solved to find the best assignment of clients to the facilities. As a consequence of this operation the cost increases by: $f_s(u_s + \delta) - f_s(u_s) + c_s(U') - c_s(U)$ where U' is the new solution after increasing the capacity of s. This operation can be performed in polynomial time [3] .

open(t, δ_1, δ_2): This operation is best viewed as a combination of two operations. In the first operation the capacity allocated at $t \in F$ is increased by δ_2 and the total capacity of a set T, to be determined as a part of the operation, is decreased by the same amount. The second operation is $add(t, \delta_1 - \delta_2)$. Our procedure for implementing this operation is as follows.

1. We create an instance of the knapsack problem where the sack has capacity δ_2. For each facility $i \in F, i \neq t$, we have for all $0 < j < u_i$, an object of weight j and profit $f_i(u_i) - f_i(u_i - j) - j \cdot c_{it}$. Picking such an object into the knapsack corresponds to reducing the capacity of facility i by j units and the profit is a lower bound on the savings we get and is obtained by reassigning j clients served by i to t. The knapsack problem is to maximise profit under the constraint that we can pick at most one object corresponding to each facility. Thus, by solving the knapsack problem, we find a set T and for each facility $i \in T$, a quantity j by which the capacity at i is decreased.

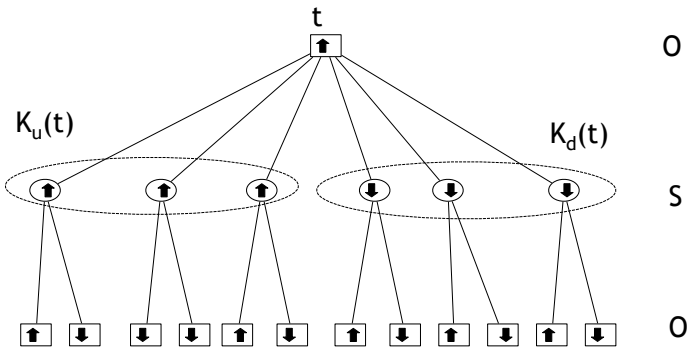

Figure 1 A subtree of height 2 showing up-facilities and down-facilities. The square facilities are in the optimum solution while the circular facilities are in the locally optimum solution. The arrow in the facility identifies it as an up/down facility

2. Independently of the above knapsack procedure we find, by solving a mincost flow problem, the maximum savings in service cost if an additional $\delta_1 - \delta_2$ clients are assigned to t.

The profit in step 1 and the savings in step 2, when reduced by $(f_t(u_t + \delta_1) - f_t(u_t))$ is an estimate of the savings obtained by this operation and if this quantity is positive we have a local improvement. Step 2 can be performed in polynomial time. To perform step 1 in polynomial time, a dynamic programming solution, similar to Mahdian and Pal [3] is used, and values of j are taken to be non-negative integers.

close(t, δ_1, t^*): This operation too is best viewed as a combination of two operations. The first operation is $add(t^*, \delta_2 - \delta_1)$. In the second operation the capacity allocated at $t \in F$ is decreased by δ_1 and the total capacity of a set $T, t^* \in T$, to be determined during the operation, is increased by an amount δ_2, $\delta_2 > \delta_1$. Assuming $\delta_2 \geq 0$ is known, the operation can be implemented as follows.

1. We solve a mincost flow problem to compute the maximum savings in service cost when $\delta_2 - \delta_1$ clients are assigned to t^*.
2. Next we create a knapsack instance with sack capacity δ_1. For all $0 < j < \delta_1$, for each facility $i \in F, i \notin \{t, t^*\}$, we have an object of weight j and profit $f_i(u_i) - f_i(u_i + j) - j \cdot c_{it}$ while for facility $i = t^*$, we have an object of weight j and profit $f_i(u_i + \delta_2 - \delta_1) - f_i(u_i + j + \delta_2 - \delta_1) - j \cdot c_{it}$. As before, the knapsack problem is to maximise profit under the constraint that we can pick at most one object corresponding to each facility.

The savings in step 1 and the profit in step 2, when increased by $(f_t(u_t) - f_t(u_t - \delta_1)) - (f_{t^*}(u_{t^*} + \delta_2 - \delta_1) - f_{t^*}(u_{t^*}))$ is an estimate of the savings obtained by this operation and if this quantity is positive we have a local improvement. To perform step 2 in polynomial time, a dynamic programming solution, similar to Mahdian and Pal [3] is used with values of j being non-negative integers.

4 Bounding the cost of our solution

Recall that U is a locally optimal solution and U^* is an optimum solution. Also, G' is an eschange graph and y defines an optimum flow on the edges in this graph. We consider potential local improvement steps and using the fact that U is a locally optimal solution, formulate suitable inequalities which help us bound the cost of our solution. The inequalities

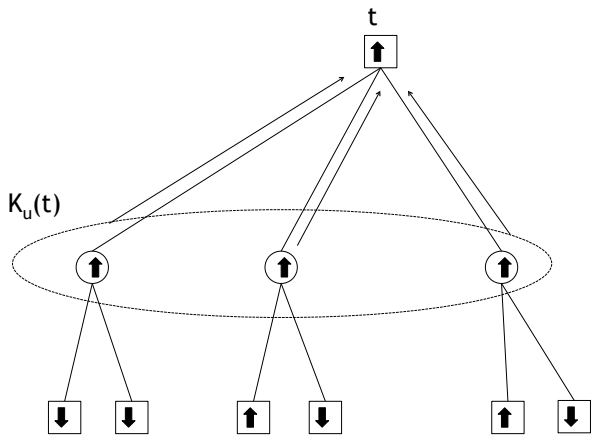

Figure 2 *open* operation considered for handling facilities in $K_u(t)$ when t is an up-facility.

are written such that

1. each facility in S is closed once.
2. each facility in O is opened at most five times.
3. the total cost of reassigning clients is bounded by

$$2\sum_{s\in S, o\in O} c_{so}y(s,o) + 3\sum_{o\in O}\sum_{P\in \mathtt{in}(o)} \text{shift}(P).$$

Every tree in the forest G' is rooted at a facility in O. Consider a subtree T of height 2 having root $t \in O$(Figure 1). For a facility i, let $p(i)$ be the parent and $K(i)$ the children of i. A facility i is an *up-facility* if $y(i,p(i)) \geq \sum_{j\in K(i)} y(i,j)$ and a *down-facility* otherwise. $K_u(i)$ (respectively $K_d(i)$) denote the children of i which are up-facilities (respectively down-facilities). Our choice of operations, considered for the purpose of analysis, is different from the ones considered in [2] and ensure that for a facility $o \in O$:

1. If o is an *up-facility*, it is opened at most twice in operations involving facilities which are descendants of o in the tree and is opened at most twice in other operations.
2. If o is a *down-facility* it is opened at most four times in operations involving facilities which are descendants of o in the tree and is opened at most once in other operations.

4.1 Closing children of t which are up-facilities

Consider the children of t which are *up-facilities*. We first consider the easier case when t is an up-facility(Figure 2). Now,

$$\hat{u}(t) = \sum_{s\in K(t)} y(s,t) + y(p(t),t) \geq \sum_{s\in K_u(t)} y(s,t) + y(p(t),t) \geq \sum_{s\in K_u(t)} 2y(s,t) \geq \sum_{s\in K_u(t)} \hat{u}(s)$$

where the second last inequality is due to the fact that t is an *up-facility* and the last inequality holds since for all $s \in K_u(t)$ we have

$$\hat{u}(s) = y(s,t) + \sum_{o\in K(s)} y(s,o) \leq 2y(s,t).$$

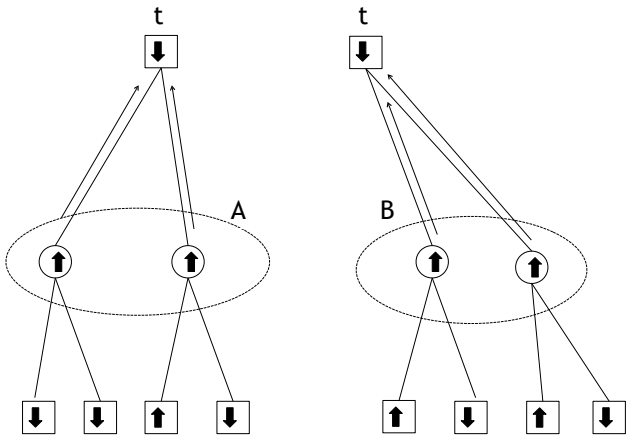

Figure 3 *open* operations considered for facilities in $K_u(t) \setminus \{h\}$ when t is a down facility.

Thus all facilities $s \in K_u(t)$ can be closed (*i.e.*,their capacity reduced by $\hat{u}(s)$) in a single operation $open(t, \hat{u}(t), \sum_{s \in K_u(t)} \hat{u}(s))$. We now bound the cost of reassignment of clients as a result of this operation.

1. $\hat{u}(s)$ clients of a facility $s \in K_u(t)$ are assigned to t. Since for $s \in K_u(t)$, $\hat{u}(s) \leq 2y(s,t)$, this reassignment cost is at most $2y(s,t)c_{st}$.
2. We can assign an additional $\hat{u}(t) - \sum_{s \in K_u(t)} \hat{u}(s)$ clients to t. One way of doing this is by shifting to an extent $(1 - \sum_{s \in K_u(t)} \hat{u}(s)/\hat{u}(t))$ along each of the $\hat{u}(t)$ paths in $\texttt{in}(t)$. Since U is locally optimal, this operation will not improve the cost of U. This operation then yields the inequality

$$f_t(u_t^*) - f_t(u_t) - \sum_{s \in K_u(t)} (f_s(u_s) - f_s(u_s^*)) + \sum_{s \in K_u(t)} 2y(s,t)c_{st} + \left(1 - \sum_{s \in K_u(t)} \hat{u}(s)/\hat{u}(t)\right) \sum_{P \in \texttt{in}(t)} \text{shift}(P) \geq 0 \quad (1)$$

We next consider the case when t is a down-facility (Figure 3) and begin by noting that

$$\sum_{s \in K_u(t)} \hat{u}(s) \leq \sum_{s \in K_u(t)} 2y(s,t) \leq 2 \sum_{s \in K(t)} y(s,t) \leq 2\hat{u}(t).$$

Let $h = \arg\max_{s \in K_u(t)} y(s,t)$ and partition the facilities in $K_u(t) \setminus h$ into two sets A and B such that $\sum_{s \in A} \hat{u}(s) \leq \hat{u}(t)$ and $\sum_{s \in B} \hat{u}(s) \leq \hat{u}(t)$. The facilities in sets A and B are closed in two *open* operations $open(t, \hat{u}(t), \sum_{s \in A} \hat{u}(s))$ and $open(t, \hat{u}(t), \sum_{s \in B} \hat{u}(s))$ respectively; see Figure 3.

The extra capacity available at t in each of these open operations is used to assign additional clients to t in the same manner as done earlier.

The facility h is handled together with the facilities in $K_d(t)$ using *close* operations as discussed next.

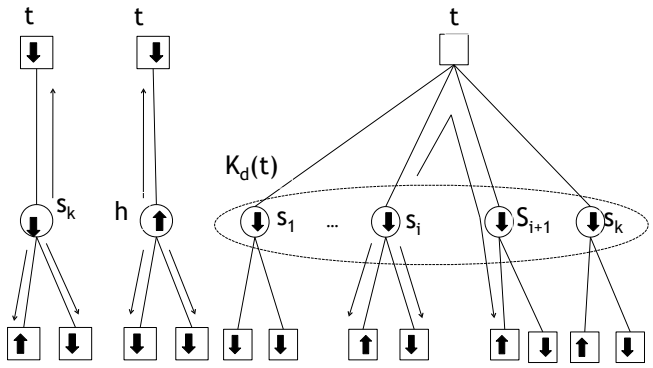

Figure 4 *close* operations for facilities in $K_d(t)$ and facility h showing the reassignment of clients when one of these facilities are closed.

4.2 Closing facility h and down-facilities which are children of t

Now we discuss the operations to close facilities $s \in K_d(t) \cup \{h\}$ and refer to Figure 4. Consider the facilities in $K_d(t)$. As in [2, 5], for every $s \in K_d(t)$ we define $rem(s) = y(s,t) - \sum_{o \in K_d(s)} y(s,o)$ and rename the facilities in $K_d(t)$ so that $rem(s_1) \leq rem(s_2) \leq \cdots rem(s_k)$.

We can close facility $s_i \in K_d(t), i < k$ by reassigning $\hat{u}(s_i)$ of its clients to facilities in $K(s_i) \cup K_u(s_{i+1})$ as follows:

1. $y(s_i, o)$ clients are reassigned to $o \in K_u(s_i)$.
2. $2y(s_i, o)$ clients are reassigned to $o \in K_d(s_i)$. Since o is a down-facility, $y(s_i, o) \leq \sum_{s' \in K(o)} y(s', o)$ and hence $2y(s_i, o) \leq \hat{u}(o)$.
3. This leaves $rem(s_i) = y(s_i, t) - \sum_{o \in K_d(s_i)} y(s_i, o)$ clients, which are reassigned to facilities in $K_u(s_{i+1})$. Doing so is feasible since

$$rem(s_i) \leq rem(s_{i+1}) = y(s_{i+1}, t) - \sum_{o \in K_d(s_{i+1})} y(s_{i+1}, o) \leq \sum_{o \in K_u(s_{i+1})} y(s_{i+1}, o).$$

We denote by $z(s_i, o)$ the number of clients reassigned to $o \in K(s_i) \cup K_u(s_{i+1})$ in the above argument.

We can formulate the following inequality, w.r.t.closing of facility s_i:

▶ **Lemma 2.**

$$f_{s_i}(u_{s_i}) - f_{s_i}(u^*_{s_i}) - \sum_{o \in K(s_i) \cup K_u(s_{i+1})} (f_o(u^*_o) - f_o(u_o))$$

$$\leq \sum_{o \in K(s_i) \cup K_u(s_{i+1})} \left(z(s_i, o) c_{s_i o} + \sum_{P \in \mathtt{in}(o)} \left(1 - \frac{z(s_i, o)}{\hat{u}(o)}\right) shift(P) \right) \tag{2}$$

Proof. Denote the set $K(s_i) \cup K_u(s_{i+1})$ by T. Consider the facilities of T in decreasing order of $z(s_i, o)/\hat{u}(o)$ and keep including them into a set T' until the total capacity of the facilities in T', i.e. $\sum_{o \in T'} \hat{u}(o)$, exceeds $\hat{u}(s_i)$. Let t^* be the last facility to be included into

T' and $k = \hat{u}(s_i) - \sum_{o \in T' \setminus \{t^*\}} \hat{u}(o)$ be the number of clients reassigned from s_i to t^*. Then a $close(s_i, \hat{u}(s_i), t^*)$ operation which reassigns $\hat{u}(o)$ clients from s_i to $o \in T' \setminus \{t^*\}$, k clients from s_i to t^* and an additional $\hat{u}(t^*) - k$ clients to t^* by performing a shift along each path in $\mathtt{in}(t^*)$ to an extent $1 - k/\hat{u}(t^*)$ yields the inequality

$$f_{s_i}(u_{s_i}) - f_{s_i}(u^*_{s_i}) - \sum_{o \in T'} (f_o(u^*_o) - f_o(u_o))$$
$$\leq \sum_{o \in T' \setminus \{t^*\}} \hat{u}(o) c_{s_i o} + k c_{s_i t^*} + (1 - k/\hat{u}(t^*)) \sum_{P \in \mathtt{in}(t^*)} \text{shift}(P) \tag{3}$$

We now build a linear combination by taking inequality 3 to an extent of ξ, reduce $z(s_i, o)$ by $\xi \cdot \hat{u}(o)$ for all facilities $o \in T' \setminus \{t^*\}$ and reduce $z(s_i, t^*)$ by $\xi \cdot k$, where ξ is the largest value such that $z(s_i, o) \geq 0, o \in T'$. We keep building the linear combination in this manner till $\xi = 0$. This process can be viewed as sending $\xi \cdot \hat{u}(s_i)$ units of flow from s_i to facilities in T' with facility $o \in T' \setminus \{t^*\}$ receiving $\xi \cdot \hat{u}(o)$ flow and facility t^* receiving $\xi \cdot k$ flow. The edges (s_i, o) have capacity $z(s_i, o)$ which is reduced by the amount of flow sent in each step. Initially the total capacity of all edges $\sum_{o \in T} z(s_i, o)$ equals the amount of flow $\hat{u}(s_i)$ that needs to be sent and this property is maintained at each step. By picking the facilities with the largest values of $z(s_i, o)/\hat{u}(o)$ we are ensuring that the maximum of these quantities never exceeds the fraction of the flow that remains to be sent. This implies that when the procedure terminates all $z(s_i, o)$ are zero and $\hat{u}(s_i)$ units of flow have been sent.

If a facility $o \in T$ was opened to an extent λ_o in the above process, then its contribution in the linear combination would be $\lambda_o(f_o(u^*_o) - f_o(u_o)) + z(s_i, o)c_{s_i o} + (\lambda_o - z(s_i, o)/\hat{u}(o)) \sum_{P \in \mathtt{in}(o)} \text{shift}(P)$. We add a $1 - \lambda_o$ multiple of the inequality

$$f_o(u^*_o) - f_o(u_o) + \sum_{P \in \mathtt{in}(o)} \text{shift}(P) \geq 0 \tag{4}$$

which corresponds to the operation $\mathtt{add}(o, \hat{u}(o))$, to the linear combination to match the contribution of o in Inequality 2. ◀

Note that due to the above process, s_i is closed to an extent of one and $o \in T$ is opened to an extent of one.

We now consider operations of the kind $close(s_k, \hat{u}(s_k), t^*)$ where $t^* \in K(s_k) \cup \{t\}$ to handle s_k. As before we build a suitable linear combination of the inequalities arising from these operations while ensuring that the total number of clients reassigned from s_k to $o \in K(s_k) \cup \{t\}$ is $z(s_k, o) = y(s_k, o)$. The inequality corresponding to this linear combination is given by

$$(f_{s_k}(u_{s_k}) - f_{s_k}(u^*_{s_k})) - \sum_{o \in K(s_k) \cup \{t\}} (f_o(u^*_o) - f_o(u_o))$$
$$\leq \sum_{o \in K(s_k) \cup \{t\}} z(s_k, o) c_{s_k o} + \sum_{o \in K(s_k) \cup \{t\}} \sum_{P \in \mathtt{in}(o)} (1 - z(s_k, o)/\hat{u}(o)) \text{shift}(P) \tag{5}$$

Note that in the above process, s_k is closed to an extent of one and $o \in K(s_k) \cup \{t\}$ is opened to an extent of one.

The following lemma bounds the cost of reassigning clients (excluding the cost of shifting along paths in $\mathcal{P}$) in the close operations on facilities in $K_d(t)$.

▶ **Lemma 3.**

$$\sum_{i=1}^{k} \sum_{o} z(s_i, o) c_{s_i o} \leq 2 \sum_{i=1}^{k} \sum_{o \in K(s_i) \cup \{t\}} y(s_i, o) c_{s_i o}$$

Proof. We begin by observing that since edge costs form a metric, $c_{s_i o}, o \in K_u(s_{i+1})$ is at most $c_{s_i t} + c_{t s_{i+1}} + c_{s_{i+1} o}$.

The contribution of the edge $(s_i, t), i \neq 1, k$ to the reassignment cost is at most $(\texttt{rem}(s_i) + \texttt{rem}(s_{i-1}))c_{s_i t}$. Since both $\texttt{rem}(s_{i-1}) \leq \texttt{rem}(s_i) \leq y(s_i, t)$ the total contribution of this edge is at most $2y(s_i, t)c_{s_i t}$. The contribution of the edge (s_1, t) to the reassignment cost is at most $\texttt{rem}(s_1)c_{s_1 t} \leq y(s_1, t)c_{s_1 t}$ while the contribution of the edge (s_k, t) to the reassignment cost is at most $(\texttt{rem}(s_{k-1}) + y(s_k, t))c_{s_k t} \leq 2y(s_k, t)c_{s_k t}$.

The contribution of the edge $(s_i, o), o \in K_d(s_i)$ is at most $2y(s_i, o)c_{s_i o}$ since $2y(s_i, o)$ clients are assigned to o when s_i is closed.

The contribution of the edge $(s_i, o), o \in K_u(s_i)$, $i \neq 1$ is at most $2y(s_i, o)c_{s_i o}$ since at most $y(s_i, o)$ clients are assigned to j once when s_i is closed and once when s_{i-1} is closed. The contribution of the edge $(s_1, o), o \in K_u(s_1)$ is at most $y(s_1, o)c_{s_1 o}$. ◀

Finally, by considering operations $close(h, \hat{u}(h), t^*)$ where $t^* \in K(h) \cup \{t\}$ and taking a suitable linear combination we obtain an inequality similar to inequality 5 with h replacing s_k. Note that in this operation, the contribution of an edge (h, o), $o \in K(h) \cup \{t\}$ is at most $y(h, o)c_{ho}$. Also note that h is closed to an extent of one and $o \in K(h) \cup \{t\}$ is opened to an extent of one in this process.

5 Putting Things Together

In all the operations discussed in the previous section, a facility $o \in O$ is opened at most 5 times and cost of reassignment of clients in all these operations is small. We prove these facts in the following lemmas.

▶ **Lemma 4.** *A facility $o \in O$ is opened at most 5 times and is assigned a total of at most $2\sum_s y(s, o) \leq 2\hat{u}(o)$ clients, from the facilities closed in the respective operations, over all the operations considered.*

Proof. **When o is an up-facility:** While considering the facilities of S which are descendants of o, o would be opened twice, once when it is part of *close* operations $close(s_k, \hat{u}(s_k), t)$, $s_k \in K_d(o), t \in K(s_k) \cup \{o\}$ and once when it is part of an *open* operation $open(o, \hat{u}(o), \sum_{s \in K_u(o)} \hat{u}(s))$. o is assigned at most $2\sum_{s \in K_u(o)} y(s, o) + y(s_k, o)$ clients where $s_k \in K_d(o)$. Note that this is at most $2\sum_{s \in K(o)} y(s, o)$.

While considering the facilities of S which are not descendants of o, if the parent of o, $p(o)$, is an up-facility, o would be opened once if $p(o) = h$. In this case a `close` operation involving $p(o)$ assigns atmost $y(p(o), o)$ clients to o. If $p(o) \neq h$ then o is not opened and no client is assigned to it.

If $p(o)$ is a down-facility then o would be opened at most twice and would be assigned at most $2y(p(o), o)$ clients. This can be argued in a straightforward manner by considering the 3 cases: $p(o) = s_1; p(o) = s_i, i \neq 1, k; p(o) = s_k$.

When o is a down-facility: While considering the facilities of S which are descendants of o, o would be opened four times: once when it is part of *close* operations $close(s_k, \hat{u}(s_k), t)$ where $s_k \in K_d(o), t \in K(s_k) \cup \{o\}$, once when it is part of *close* operations $close(h, \hat{u}(h), t)$ where $h \in K_u(o), t \in K(h) \cup \{o\}$, and twice as a part of two *open* operations in which sets $A, B \subseteq K_u(o)$ are closed. The number of clients assigned to o in these operations is $2\sum_{s \in A} y(s, o)$, $2\sum_{s \in B} y(s, o)$, $y(h, o)$ and $y(s_k, o)$ respectively. Since $A \cup B \cup \{h\} = K_u(o)$ and $s_k \in K_d(o)$, the total number of clients assigned to o in these four operations is at most $2\sum_{s \in K(o)} y(s, o)$.

o would be opened at most once while considering the facilities of S, which are not descendants of o irrespective of whether $p(o)$ is an up-facility or a down-facility. If the parent of o, $p(o)$, is an up-facility then o would be assigned at most $y(p(o), o)$ clients in a *close* operation involving $p(o)$. If $p(o)$ is a down-facility then o would be assigned at most $2y(p(o), o)$ clients and once again this can be argued by considering 2 cases: $p(o) = s_i, i \neq k; p(o) = s_k$. Therefore the total number of clients assigned to o when o is a down-facility is at most $2\sum_s y(s, o)$. ◀

▶ **Lemma 5.** *The total reassignment cost of all the operations is bounded by*

$$2 \sum_{s \in S, o \in O} c_{so} y(s, o) + 3 \sum_{o \in O} \sum_{P \in N_{U^*}(o)} \mathit{shift}(P)$$

Proof. The first term in the required expression follows from the fact that in all the operations considered, the contribution of an edge (s, o) of the exchange graph is at most $2c_{so}y(s, o)$.

When all the inequalities are added, the term $\sum_{P \in \mathtt{in}(o)} \text{shift}(P)$ for a facility $o \in O$ appears to the extent of $\alpha - \beta/\hat{u}(o)$ where α is the number of times o is opened and β is the total number of clients assigned to o from the facilities whose capacity allocation decreases in the operation in which o is opened. From Lemma 4, β is at most $2\hat{u}(o)$ and α is at most 5. If a facility $o \in O$ is opened less than five times in these operations then we add the inequality corresponding to $add(o, \hat{u}(o))$ to our linear combination so that each facility is now opened exactly five times. Therefore, the coefficient of the term $\sum_{P \in \mathtt{in}(o)} \text{shift}(P)$ is at least 3. If its greater than 3 for some $o \in O$ then we will reduce the coefficient of $\sum_{P \in \mathtt{in}(o))} \text{shift}(P)$ in some of the inequalities involving o to make this contribution exactly 3. ◀

From Lemma 4 and Lemma 5, we can conclude that

$$-\sum_{s \in S} (f_s(u_s) - f_s(u_s^*)) + 5 \sum_{o \in O} (f_o(u_o^*) - f_o(u_o))$$
$$+ 2 \sum_{s \in S, o \in O} c_{so} y(s, o) + 3 \sum_{o \in O} \sum_{P \in N_{U^*}(o)} \text{shift}(P) \geq 0$$

Rearranging terms, we get

$$-\sum_{i \in F} f_i(u_i) + 5 \sum_{i \in F} f_i(u_i^*) + 2 \sum_{s \in S, o \in O} c_{so} y(s, o) + 3 \sum_{o \in O} \sum_{P \in \mathtt{in}(o)} \text{shift}(P) \geq 0$$

The third term in the above inequality can be bounded by $2(c_s(U) + c_s(U^*))$ (Lemma 1). The fourth term can be written as

$$3 \sum_{o \in O} \sum_{P \in \mathtt{in}(o)} \text{shift}(P) = 3 \sum_{P \in \mathcal{P}} \sum_{j \in P} (U_j^* - U_j) + 3 \sum_{Q \in \mathcal{C}} \sum_{j \in Q} (U_j^* - U_j) = 3 \sum_{j \in C} (U_j^* - U_j)$$

where second term in the middle equality is due to the fact that $\sum_{j \in Q:\, Q \in \mathcal{C}} (U_j^* - U_j) = 0$. Thus,

$$-c_f(U) + 5c_f(U^*) + 2(c_s(U) + c_s(U^*)) + 3(c_s(U^*) - c_s(U)) \geq 0$$

which implies the following bound on the cost of our solution

$$c_f(U) + c_s(U) \leq 5c_f(U^*) + 5c_s(U^*) \tag{6}$$

To ensure that the local search procedure has a polynomial running time we need to modify the local search procedure so that a step is performed only when the cost of the solution decreases by at least $(\epsilon/4n)c(U)$. This modification gives rise to an extra term of at most $(4\epsilon/4)c(U)$ in the above inequality. This implies that the cost of the solution U is at most $5c(U^*) + \epsilon \cdot c(U)$.

Thus we arrive at our main result:

▶ **Theorem 6.** *The local search procedure with operations* add, open and close *yields a locally optimum solution that is a* $(5 + \epsilon)$*-approximation to the optimum solution.*

References

1 Eric Angel, Nguyen Kim Thang, and Damien Regnault. Improved local search for universal facility location. In *Proceedings of 19th International Conference on Computing and Combinatorics (COCOON), Hangzhou, China*, pages 316–324, 2013.

2 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.

3 Mohammad Mahdian and Martin Pál. Universal facility location. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA), Budapest, Hungary*, volume 2832 of *Lecture Notes in Computer Science*, pages 409–421, 2003.

4 M. Pál, É. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 329, Washington, DC, USA, 2001. IEEE Computer Society.

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

6 Jiawei Zhang, Bo Chen, and Yinyu Ye. A multiexchange local search algorithm for the capacitated facility location problem. *Math. Oper. Res.*, 30(2):389–403, 2005.