IBM Research Report

Improved Local Search Algorithm for Universal Facility Location

Vinayaka Pandit
IBM Research Division
IBM India Research Lab
Block I, I.I.T. Campus, Hauz Khas
New Delhi - 110016. India.

Naveen Garg
Dept. of Computer Science and Engineering
Indian Institute of Technology
New Delhi - 110016. India.

Rohit Khandekar
Dept. of Computer Science and Engineering
Indian Institute of Technology
New Delhi - 110016. India.

Amit Kumar
Dept. of Computer Science and Engineering
Indian Institute of Technology
New Delhi - 110016. India.

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Abstract

In a recent result, M. Pál et al.[2] showed that a local search heuristic can be used to obtain a $9+\epsilon$ approximation for capacitated facility location with *hard capacities*, also known as 1-CFL. Later, Pál, and Mahdian [1] showed that more powerful local operations can be used to obtain $8+\epsilon$ approximation for the Universal Facility Location Problem which includes 1-CFL as a special case. After scaling, their algorithm achieves an approximation ratio of $7.88+\epsilon$. We improve these results incrementally on two fronts. We show that a local search with simpler operations than those used by Pál and Mahdian achieves the same approximation ratio of $8+\epsilon$, and scaling technique can be used to improve it to $7.60+\epsilon$. Secondly, we strengthen the operations used by Pál and Mahdian to improve the approximation guarantee to $7+\epsilon$.

1 Problem Definitions

1.1 1-CFL

In 1-CFL problem, we are given a set of facilities, denoted F and a set of clients, denoted by C. For each facility $i \in F$, the cost of opening it is f_i. Each facility $i \in F$ has a capacity u_i which denotes the maximum amount of demand it can serve. For each client $j \in C$, its demand is denoted by d_j. The goal is to open a set of facilities, and assign clients to the open facilities such that, (i) no open facility serves more clients than its capacity, and (ii) the total cost of opening facilities, and serving clients is minimized.

1.2 Universal Facility Location Problem

In universal facility location problem(UniFL), instad of associating a hard capacity at each facility, we specify the cost of opening a facility as a function of the demand it serves in the solution. Thus, if a facility $i \in F$ serves a demand of u_i, then the cost of the facility i is given by $G_i(u_i)$. A solution to a UniFL instance can be given by a pair (v, x), where v is the *allocation vector* specifying the capacity allocated at each facility $i \in F$, and x is the *assignment matrix*, i.e, x_{ij} denotes the amount of demand of client j served by facility i. We assume that the functions G_is are non-decreasing, and left-continuous mapping from non-negative reals to non-negative reals with infinity. Using these, universal facility location can be specified by the following nonlinear optimization problem. Pál, and Mahdian [1] show that an instance of UniFL has an optimal solution.

$$\begin{array}{lrcll} \text{minimize} & \sum_{i\in F} G_i(u_i) & +\sum_{i\in F, j\in C} c_{ij}x_{ij} & & \\ \text{subject to} & & & & \\ & \sum_{i\in F} x_{ij} & = & d_j & \forall j \in C \\ & \sum_{j\in C} x_{ij} & \leq & u_i & \forall i \in F \\ & u_i, x_{ij} & \geq & 0 & \forall i \in F, \forall j \in C \end{array}$$

Note that, 1-CFL is a special case of the Universal Facility Location Problem. Also note that, given capacities at each facility, the best way to service the clients can be computed by solving a minimum cost network flow problem.

We first present a local search algorithm which yields a $7+\epsilon$ approximation for the Universal Facility Location Problem. This implies a $7+\epsilon$ approximation algorithm for the 1-CFL problem. We then show how to obtain a $8+\epsilon$ algorithm for 1-CFL with simpler operations than considered in [1]. We can use scaling techniques to obtain $7.60+\epsilon$ approximation as opposed to $7.88+\epsilon$ in [1].

2 Algorithm

We now present the two local search algorithms. For Universal Facility Location, we present a local search technique with local operations which strengthen those proposed in [1]. We prove that this heuristic has a locality gap of at most 7. For 1-CFL, we show that a local search technique with simpler operations than considered in [1] has a locality gap of at most 8. It also translates to a $7.60+\epsilon$ approximation for the 1-CFL problem.

Algorithm for the UFL. For Universal Facility Location problem, we present a local search algorithm which considers the following local operation at each step to search for a solution with lower cost. The operations are:

- `add`(s, δ) : Increase the allocated capacity at facility s by δ. For this operation, we compute the actual change in cost, i.e., $G_s(u_s+\delta) - G_s(u_s) + C_s(S') - C_s(S)$ where u_s is current allocated

capacity of s, $C_s(S)$, and $C_s(S')$ are the service cost before, and after the operation. As the number of choices for s, and δ are polynomially bounded in input size, `add` operation can be implemented efficiently.

- `Single_Pivot`(s, Δ) : This operation allows us to implement a limited version of increasing the allocated capacities at multiple facilities, and decreasing the allocated capacities at multiple facilities. Specifically, the vector Δ indicates the change in allocated capacity at each facility. A facility i is said to *shrink* if $\Delta_i < 0$, and it is said to *grow* if $\Delta_i > 0$. Each shrinking facility i sends $|\Delta_i|$ amount of its demand to the pivot s. For each growing facility i, Δ_i amount of demand is then routed from s. Note that, a valid `Single_Pivot` operation satisfies the property that $\sum_{i \in F} \Delta_i = 0$. We compute the estimated cost of this operation as $\sum_{i \in F}(G_i(u_i + \Delta_i) - G_i(u_i) + c_{si}|\Delta_i|)$. Note that, this is only an upper bound on the change in cost after the operation. The analysis depends on the fact that, at local minima, there is no operation with a negative upper bound.

- `Doublei_Pivot`$(s_1, s_2, \Delta_1, \Delta_2)$: This operation is a generalization of the previous operation to redistribute demand around two pivots. Δ_1 specifies the rerouting of demand through s_1, and Δ_2 specifies the rerouting of demand through s_2. For a valid operation, $\sum_{i \in F} \Delta_{1i} = 0$, and $\sum_{i \in F} \Delta_{2i} = 0$. The cost of this operation can be computed in a manner similar to previous operation, i.e., $\sum_{i \in F}(G_i(u_i + \Delta_{1i} + \Delta_{2i}) - G_i(u_i) + c_{s_1 i}|\Delta_{1i}| + c_{s_2 i}|\Delta_{2i}|)$.

In [1], Pál and Mahdian considered just the first two operations to prove a locality gap of 8. `Double_Pivot`$(s_1, s_2, \Delta_1, \Delta_2)$ is an extention of the `Single_Pivot`(s, Δ) operation, and helps us to improve their analysis.

Simpler operations for 1-CFL. The above result is also valid for 1-CFL as any instance of 1-CFL can be coded as a Universal Facility Location problem. We present a local search algorithm with simpler local operations than those proposed in [1] to prove essentially the same locality gap of 8. The operations are:

- `add`(s) : Add a facility s. If s already exists, then this operation may just reroute some demand to s if its capacity is not fully utilised. For this operation, we compute the actual change in cost, and as the number of choices for s is limited, this can be implemented efficiently.

- `open`(s, T) : In this operation, we open a facility s, and drop a subset of existing facilities, namely, T. If a client j being served by $t \in T$ in the current solution, then the change in service cost of j is bounded by $c_{js} - c_{jt}$. The change in facility cost is the true cost of opening s (i.e., true cost is 0 if s is already open) minus the total cost of facilities in T. The estimated cost of this operation is the sum of change in facility cost plus the change in service cost for clients served by T.

- `close`(s, T) : In this operation, we close a facility $s \in S$, and open a set of facilities $T \subseteq F - s$. The change in facility cost is given by $-f_s + c_f(T - S)$. We estiamte the change in service cost by just reassigning the clients served by s to the facilities in T. Suppose d_{st} amount of demand currently served by s is reassigned to $t \in T$ such that $\sum_{t \in T} d_{st} = d_s$ (d_s is the demand served by s in the current solution), then the estimated change in service cost is $\sum_{t \in T} d_{st} c_{st}$ (using triangular inequality). The estimated change in cost due to the operation is sum of change in facility, and service cost.

- `close_two`(s_1, s_2, T) : This operation is a generalisation of the previous operation to close two facilities in the current solution. The estimated cost of this operation is computed in a manner similar to above.

In [2], Pál et al.show a locality gap of 9 with just the first three operations.

Efficient implementation of operations. We first consider `Single_Pivot`(s, Δ) as it contains all the operations of 1-CFL as special cases. We show how to implement this operation when the demands are small integers, and the facility cost functions are step functions at integer intervals. Approximate versions which incur an additional factor of ϵ in the approximation ratio can be implemented similarly as shown by Pál and Mahdian [1].

Note that, the cost is calculated by first sending the demand at shrinking facilities to s, and then redistributng to the growing facilities. Suppose that a set of shrinking facilities is already identified, then the task of selecting the set of swelling facilities, and the amount of swell is essentially a covering knapsack problem. s can be thought of as one client with demand equal to collected demand of all shrinking facilities, and the goal is to increasing the capacities at the rest of the facilities such that

the demand can be served, and total of cost increasing the capacities, and serving the demand from s is minimized. Let $D = \sum_j d_j$.

Suppose the current solution, S, is given by (v, x), and we are trying to identify if there exists a beneficial pivot operation pivoted at s. The vector v specifies the capacity u_i at a facility i, and in the matrix x, x_{ij} specifies the demand of client j served facility i. For each facility i, we compute a function $g_i(\delta) = c_{si}.|\delta| + G_i(u_i + \delta) - G_i(u_i)$ for $\delta = -D, \ldots, D$. The function computes the cost of shipping δ of its current demand to s (i.e, it shrinks) when δ is negative, and the cost of shipping δ demand from s to i when δ is positive(i.e, it grows). We also capture correctness conditions such as, when δ is negative, $|\delta| \leq u_i$. The idea is to compute a two dimensional table a, such that $a(i, w)$ is the minimum cost of collecting an excess(or deficiency) of w units of demand at s when the set of facilities is limited to $1, \ldots, i$. We compute the table by the following recurrence relation

$$a(i, w) = \begin{cases} g_1(-w) & \text{if } i = 1 \\ min_\delta\{g_i(\delta) + a(i-1, w+\delta)\} & \text{otherwise} \end{cases}$$

The w values in the table vary from $-D$ to D. Clearly, the least cost of an operation with s as the pivot is given by $a(n, 0)$. If this quantity is negative for any choice of s, we have found an admissible pivot operation. The table also intrinsically defines the shrinking and growing facilities(and also the extent).

Similarly, we can implement a stronger version of the pivot operations, which computes the Δ vectors around two possible pivots. The details of the implementation remain same except that we build a three dimensional table using modified g_i functions, and a modified recurrence relation. They are given by:

$$g_i(\delta_1, \delta_2) = \sum_{l=1,2} s_{s_l i}|\delta_l| + G_i(u_i + \delta_1 + \delta_2) - G_i(u_i)$$

$$a(i, w_1, w_2) = \begin{cases} g_1(-w_1, -w_2) & \text{if } i = 1 \\ min_{\delta_1,\delta_2}\{g_i(\delta_1, \delta_2) + a(i-1, w_1+\delta_1, w_2+\delta_2)\} & otherwise \end{cases}$$

The operations $\texttt{open}(s, T)$, and $\texttt{close}(s, T)$ for 1-CFL are special cases of the $\texttt{Single\_Pivot}(s, \Delta)$ operation. In fact, $\texttt{open}(s, T)$ is a simple knapsack problem, and $\texttt{close}(s, T)$ is a covering knapsack problem described in the context of finding a beneficial $\texttt{Single\_Pivot}(s, \Delta)$ operation. Thus, all local operations of the two algorithms can be implemented efficiently.

3 Analysis

In the context of UFL as well 1-CFL we will use S to denote a locally optimum solution and S^* to denote a globally optimum solution. The service cost of the local minima can be bounded in a manner similar to the UFL, and ∞-CFL.

Lemma 3.1 *The service cost of S, $C_s(S)$ is bounded by the cost of the optimial solution, i.e, $C_s(S) \leq C_f(S^*) + C_s(S^*)$.*

Proof. Consider two operations $\texttt{add}(s_1, \delta_1)$, and $\texttt{add}(s_2, \delta_2)$ which decrease the overall cost. It is easy to see that either $\texttt{add}(s_1, \delta_1)$ or $\texttt{add}(s_2, \delta_2)$ improves the cost of the solution. This property can be generalised to show that, there exists a set of add operations, ADD, which improve the cost of the solution if and only if there is a single beneficial $\texttt{add}(s_i, \delta_i)$ operation. Since S is a local minima, there is no beneficial operation. Now, consider a set of add operations in which for each facility $i \in F$ its capacity is set of $\max(u_i, u_i^*)$. So, the change in facility cost is at most $C_f(S^*)$. The capacities at the facilities in the new solution is sufficient to match the service cost of S^*. The change in cost due to these operations, $C_s(S^*) + C_f(S^*) - C_s(S)$ is at least zero. Therefore, $C_s(S) \leq C_s(S^*) + C_f(S^*)$. ∎

It is easy to prove the same lemma for the 1-CFL with non-uniform hard capacities.

The Exchange Graph. For a facility $i \in F$, let $u_i = \sum_{j \in C} x_{ij}$, and $u_i^* = \sum_{j \in C} x_{ij}^*$, i.e, the capacities at each facility is equal to the demand it serves in the solution. Let $\rho_i = u_i - u_i^*$. This denotes the excess, or the deficiency at facility i. Suppose, we route a flow of δ units from $s \in S$ to $t \in S^*$, then this can be thought of as adjusting the capacities at s, and t and the change is service cost is at most $c_{st}.x$, and total change in cost is $G_s(u_i - \delta) - G_s(u_i) + G_t(u_t + \delta) - G_t(u_t) + c_{st}.\delta$. Let us call this hypothetical operation as $\texttt{swap}(s, t, \delta)$. We can bound the facility cost of S by defining

a set of swaps such that, the capacity of excess facilities is same as their capacity in S^*, and the capacity of the deficient facilities is at most their capacity in S^*. Thus, the change in facility cost is bounded by facility cost of S^*, and change in service cost is bounded as described above. But, we do not know how to search for a beneficial $\texttt{swap}(s,t,\delta)$ operation efficiently. Instead, we will find a minimum cost transshipment from excess facilities to deficient facilities, and use this solution to define appropriate `pivot` operations to bound the facility cost. Let $F_e = \{i | i \in F \wedge (u_i > u_i^*)\}$ and $F_d = \{i | i \in F \wedge (u_i^* > u_i)\}$. The transshipment problem set up is:

$$\begin{array}{rlrll} \text{minimize} & \sum_{s\in F_e, t\in F_d} c_{st} y_{s,t} & & & \\ \text{subject to} & & & & \\ & \sum_{t\in F_d} y_{s,t} & = & \rho_s & \forall s \in F_e \\ & \sum_{s\in F_e} y_{s,t} & \leq & |\rho_t| & \forall t \in F_d \\ & y_{s,t} & \geq & 0 & \forall s \in F_e, \forall t \in F_d \end{array}$$

In case of 1-CFL with nonuniform capacities, we construct a similar exchange graph, except that the transshipment is set up between facilities in $S - S^*$ to the facilities in S^*. Based on the solution to transshipment problem, we define a set of open, and close operations to bound the facility cost.

We will first show that the transshipment problems defined above have bounded cost. Let us consider the assignment vectors x, and x^* of our solution and optimal solution respectively. We interpret the assignment vectors as sending x_{ij} amount of flow from i to j. We can extend this notion to compare two assignment vectors by considering the difference in their flow values. So, for a client j and facility s, there is a flow of $x_{ij} - x^*_{ij}$ amount of flow from i to j in a directed sense. This denotes if s is excess or deficient with respect to serving j. In this difference flow, the amount of flow directed into a client is equal to amount flow directed out of it. For a facility s, the flow going out of s is more if it is in F_e, and it is strictly less if it is in F_d. This flow can be decomposed into a set of paths which together can be seen as transferring δ_i amount of flow out of an excess facility i, and δ_l amount of flow into a deficient facility l. Suppose a path P carries $w(P)$ of flow between $s \in F_e$ and $s' \in F_d$. Then, the cost of this can be upper bounded by $w(P).\sum_{(s,j)\in P}$ which is at most the service cost of $w(P)$ amount of demand along these edges in S and S^* respectively. Over the set of all paths in the decomposition, this is upper bounded by $C_s(S) + C_s(S^*)$. Similar path decomposition can be carried out for the 1-CFL solutions as well. Thus, for both universal facility location and 1-CFL we have,

Lemma 3.2 *The transshipment problem has a solution of cost at most $C_s(S) + C_s(S^*)$.*

We observe that there exists an optimal solution for the transshipment problem which does not contain cycles. If there exists a cycle C, then we can remove C by augmenting the flow along its edges such that the resultant cost of transshipment is no more than before. Hence, for both 1-CFL with non-uniform capacities, and Universal Facility Location, we work with an optimal solution to the transshipment problem which does not contain cycles. So the solution, SOL, can be thought of as a collection of trees rooted at a facility in F_d and we use them to define a set of operations which help us bound the facility cost of our solutions. As explained before, we can root each tree $\mathcal{T}_i$ in

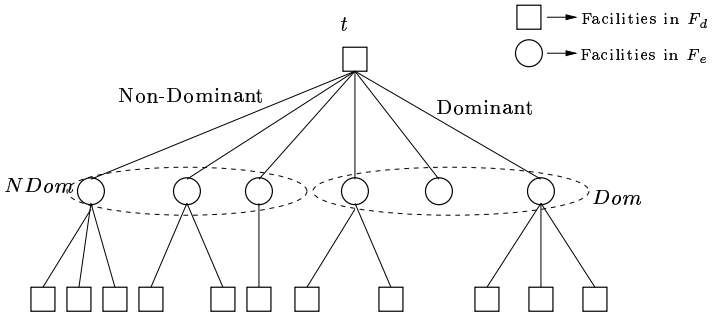

Figure 1: The subtree $\mathcal{T}_t$

SOL at a facility $r_i \in F_d$. For each facility s in a tree, we define $C(s)$ to be the children of s in the flow. So, $C(t) \subseteq F_e$ for a $t \in F_d$. For each facility $t \in F_d$ which is not the leaf of a tree in SOL, we define $\mathcal{T}_t$ to be the subtree rooted at t of depth at most two(See Fig. 1. Now, for each such t we define a set of operations depending on the flow in T_t.

Consider a subtree $\mathcal{T}_t$ rooted at a facility $t \in F_d$. We classify the facilities in $C(t) \subseteq F_e$ into two categories. A facility $s \in C(t)$ is called *dominant* if atleast half of its demand is served by t (i.e. $y_{s,t} \geq \sum_{t' \in F_d} y_{s,t'}$). A facility $s \in C(t)$ is called *non-dominant* if it is not dominant. We denote the set of dominant facilities by Dom, and set of non-dominant facilities by $NDom$(See Fig. 1). Further, we order the facilities in $NDom$ by the non-decreasing order of the demand served by t. If there are k facilities in $NDom$ they are ordered $s_1, s_2, \ldots, s_k$ such that $y_{s_1,t} \leq y_{s_2,t} \leq \ldots \leq y_{s_k,t}$(See Fig. 2).

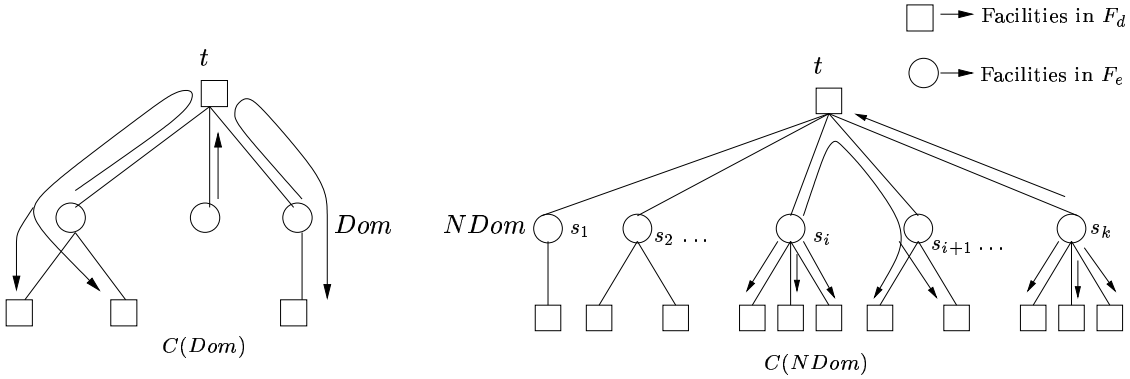

Figure 2: Reassigning the demand using `Single_Pivot`, and `Double_Pivot` operations.

We consider a single `Double_Pivot`$(t, s_k, \Delta_1, \Delta_2)$ operation. In Δ_1, all the facilities in Dom shrink by an amount equal to their excess and the facilities in the set $\{t\} \cup C(Dom)$ grow by an amount equal to their deficiency. The pivot for the vector Δ_1 is t. In Δ_2, the facility s_k in $NDom$ (See Fig. 2.) shrinks by an amount equal to its excess. The facilities in $C(NDom)$ grow by an amount equal to their deficiency. The growth of facility t in Δ_1 is sufficient to send the demand of s_k to t without exceeding its capacity. Equivalently, this can be thought of as closing of $Dom \cup \{s_k\}$, and opening of $\{t\} \cup C(Dom) \cup C(s_k)$. This operation is feasible as the increase in capacities of $\{t\} \cup C(Dom) \cup C(s_k)$ is sufficient to satisfy the total demand of s_k to t.

Note that we could have also considered shrinking other facilities in $NDom$ in the above double pivot operation. But, we are forced to use either t or s_k as the pivot for sending the demand of a $s_i \in NDom$ to $C(s_i)$. Now, consider a facility $s_i \in NDom$ such that $c_{s_i t} \gg c_{s_i t'} \forall t' \in C(s_i)$, and $y_{s,t} \ll \sum_{t' \in C(s_i)} y_{s_i,t'}$. The cost of sending the demand served by $C(s_i)$ to either t or s_k is much higher than the actual cost. Thus, to effectively capture the cost of serving these demands, we need to consider operations in which the amount of flow sent up the edge (s,t) is bounded in terms of the demand served by t.

We achieve this with the help of the ordering of facilities $NDom$ as described before. For each facility $s_i \in NDom$ where $i < k$, we consider a `Single_Pivot`(s_i, Δ) operation. The facility s_i shrinks by its excess and the facilities in $C(s_i) \cup C(s_{i+1})$ grow by an amount equal to their deficiency. The increase in capacity of facilities in $C(s_i)$ is sufficient to serve demands served by them in the tree. The demand served by t is sent to the facilities in $C(s_{i+1})$. As both s_i and s_{i+1} are non-dominant, and $y_{s_i,t} \leq y_{s_{i+1},t}$, this can be done, and the cost of doing so can be bounded in terms of the flow on edges (s_i, t), $(s_{i+1}t)$ and $\{(s_{i+1}t | \forall t' \in C(s_{i+1})\}$. Thus, all the operations considered for a subtree $\mathcal{T}_t$ can be realised without exceeding installed capacities. The facilities in F_d at depth two of $\mathcal{T}_t$ increment their capacities by an amount equal to their deficiency in at most two operations, if their parent is a dominant facility s_i such that $i \neq k$. Other facilities in F_d including t are involved in just one operation in which their capacity increases by an amount equal to their deficiency.

Lemma 3.3 *Each facility $i \in F_d$ is considered in at most three operations in which the capacity of i is set to u_i^*.*

Proof. If a facility $r \in F_d$ is the root of a tree, then it is considered in operations of the subtree $\mathcal{T}_r$ only. Consider a facility $t \in F_d$ in a tree $\mathcal{T}$ rooted at $r \neq t$. Let $t' \in F_d$ be the facility such that $t \in C(C(t'))$. If t is a leaf, then it is considered in the operations of only one subtree $\mathcal{T}_{t'}$. Otherwise, it is considered in operations corresponding to two subtrees $\mathcal{T}_t$, and $\mathcal{T}_{t'}$. So the facility t can increase its capacity equal to its capacity in S^* in at most one operation as root in $\mathcal{T}_t$ and in at most two operations as child of a non-dominant facility in the subtree $\mathcal{T}_{t'}$. ∎

Lemma 3.4 *For each edge in the solution to the transshipment problem, the amount of flow sent in the operations considered is at most three times its flow in the solution to the transshipment problem.*

Proof. Let E_d denote the set of edges which connect dominant facilities to their children. The flow along any edge $e \in E_d$ is equal to the flow along e in the solution to the transshipment problem. Let E_n denote the edges which connect non-dominant facilities to their children. Consider an edge $e \in E_n$ connected to a non-dominant facility s_i. In the `Single_Pivot` operation pivoted at s_i, the flow sent along e is equal to its flow in the solution. In the `Single_Pivot` operation pivoted at s_{i-1}, the flow along e is at most its flow in the solution, thus bounding the flow along e by at most twice its flow in the solution. This also holds for edges connecting non-dominant facilities to their parent. Finally, consider the set of edges connecting dominant facilities to their parent. Let e be a edge which connects the root of a subtree $\mathcal{T}_t$ to its dominant facilities, Dom. The pivot operations sends the flow of $C(Dom)$ back and forth along e before sending them to $C(Dom)$. As the flow to $C(Dom)$ is at most equal to the flow from Dom to t, this flow is at most twice the flow along the edge e. The pivot also sends flow to t which is equal to the flow along the edge e in the solution. Thus the flow along e due to the operations is at most three times its flow in the solution. ∎

Lemma 3.5 *The facility cost of the solution S is at most $C_f(S) \leq 3.C_f(S^*) + 3(C_s(S) + C_s(S^*))$.*

Proof. The change in cost due to each of the operations is non-negative as S is locally optimum with respect to these operations. For each operation, we denote the set of facilities whose capacities change by CHG. For a `Single_Pivot`(s, Δ) operation, we write

$$\sum_{i \in CHG} G_i(u_i^* - u_i) + \sum_{i \in CHG} c_{si}.|\Delta_i| \geq 0$$

Let $CHG1$ and $CHG2$ denote the set of facilities whose capacities change because of the two pivot operations in `Double_Pivot`$(s_1, s_2, \Delta_1, \Delta_2)$, then we write

$$\sum_{i \in CHG1 \cup CHG2} G_i(u_i^* - u_i) + \sum_{i \in CHG1} c_{s_1 i}.|\Delta_{1i}| + \sum_{i \in CHG2} c_{s_2 i}.|\Delta_{2i}|$$

From lemmas 3.3 and 3.4 it is clear that, adding equations of the form 3 and 3 for single and double pivot operations, we get

$$3 \sum_{i \in F_d} (G_i(u_i^*) - G_i(u_i)) + 3 \sum_{s,t} c_{st} y_{s,t} \geq \sum_{i \in F_e} G_i(u_i^*) - G_i(u_i)$$

which gives us,

$$3C_f(S^*) + 3(C_s(S) + 3C_s(S^*)) \geq C_f(S).$$

From lemma 3.1, we have

$$6C_f(S^*) + 6C_s(S^*) = 6C(S^*) \geq C_f(S).$$

∎

Lemmas 3.1 and 3.5 imply the following theorem.

Theorem 3.6 *For any instance of UFL, a locally optimum solution with respect to* `add`, `Single_Pivot`, *and* `Double_Pivot` *operations, denoted by S has a total cost $C(S) \leq 7.C_f(S^*) + 7.C_s(S^*)$ where S^* is any optimal solution to the UFL instance.*

The same analysis, with minor modifications, gives a $8+\epsilon$ approximation for the 1-CFL with `add, open, close, and close_two` operations. The operations defined in [1] also give an $8+\epsilon$ approximation. But, the `add, open, close, and close_two` are strictly less powerful than the operations considered in UFL. Upon scaling, our operations give a slightly improved $7.60+\epsilon$ approximation compared with $7.88+\epsilon$ due to strictly more powerful operations. The first modification we need is in classifying the facilities in a subtree $\mathcal{T}_t$. A facility $s \in S - S^*$ is called *Heavy* if $y_{s,t} > 1/2 \sum_{i \in C(t)} y_{i,t}$. A facility $s \in S - S^*$ is said to be *Light Dominant* if $y_{s,t} \geq \sum_{i \in C(s)} y_{s,i}$. Otherwise, a facility in $s \in S - S^*$ is said to be *Light Non-Dominant*. The two light sets are denoted by $LDom$ and $LNDom$ respectively. Refer to Figure 3.

Now, we define a set of operations to be considered for a subtree $\mathcal{T}_t$. If $C(t)$ contain a heavy facility, then we consider a set of operations and we consider a different set of operations if there is no heavy facility. As before, we order the facilities in $LNDom$ from $s_1, s_2, \ldots, s_k$ in the non-decreasing order of their flow to t. We first observe that:

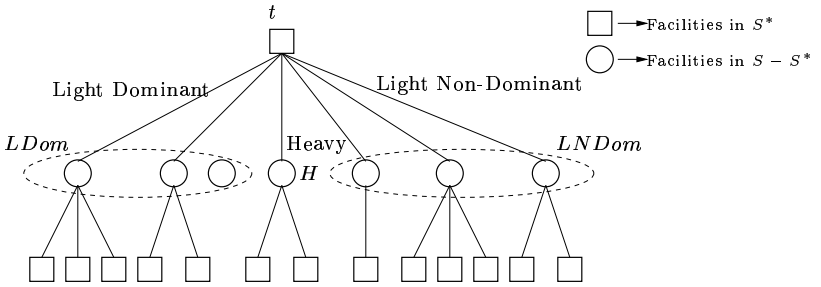

Figure 3: The subtree $\mathcal{T}_t$ in case of CFL

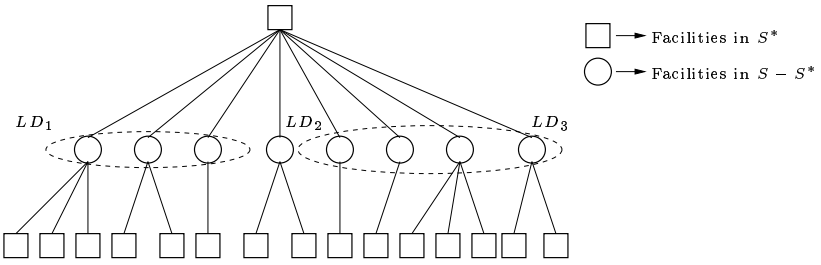

Figure 4: The division of $LDom$ when there is no heavy facility

Lemma 3.7 *When there is no heavy facility in a subtree $\mathcal{T}_t$, the LD set can be divided into three sets, LD_1, LD_2, and LD_3 such that $\sum_{i \in LD_x} y_{i,t} \leq 0.5 \sum_{s \in C(t)} y_{s,t}$ for $x = 1, 2, 3$. Moreover, the set LD_2 consists of a single facility.*

Proof. For a facility $ld_i \in LD$, let $pl_i = y_{ld_i,t} / \sum_{i \in C(t)} y_{i,t}$ denote its proportion of the flow coming into t from its children. Clearly, $pl_i \leq 0.5$ as it is a light facility. Let us order the facilities in LD as $ld_1, ld_2, \ldots, ld_k$ in the non-increasing order of pls. Clearly, $\sum_{ld_i \in LD} pl_i \leq 1$. It is easy to observe that there is an index x such that, $(\sum_{i \leq x} pl_i \leq 0.5) \wedge (\sum_{i \leq x+1} pl_i > 0.5)$. We define $LD_1 = \{ld_1, \ldots, ld_x\}, LD_2 = \{ld_{x+1}\}$, and $LD_3 = \{ld_{x+2}, \ldots, ld_k\}$. The three sets satisfy all the required properties. ∎

Let $s_1, s_2, \ldots, s_k$ be the facilities in LND ordered as before. Irrespective of the presence of a heavy facility in $C(t)$ we consider a $\texttt{close}(s_i, C(s_i) \cup C(s_{i+1})$ for $1 \leq i < k$. If there is a heavy facility $s_h \in C(t)$, then we consider the following operations.

- $\texttt{close}(s_h, C(s_h) \cup t)$: We close the heavy facility s_h. The demand of s_h is now served by opening the children of s_h and t. From the formulation of the transshipment problem, it is clear that this operation is feasible.
- $\texttt{open}(t, LD)$: Since there is a heavy facility, and the fact that t serves at least half of demand served by LD, it is clear that there is enough capacity at t to serve the entire demand of LD.
- $\texttt{close}(s_k, C(s_k) \cup t)$: Clearly, there is enough capacity at $C(s_k) \cup t$ to serve all the demand of s_k.

If there is no heavy facility in $C(t)$, then $\texttt{open}(t, LD)$ may not always be feasible. So, we split LD into LD_1, LD_2, and LD_3 satisfying the properties mentioned in lemma 3.7. Now, we consider then following operations.

- $\texttt{close\_two}(LD_2, s_k, C(s_k) \cup C(LD_2) \cup t)$: There is enough capacity at $C(s_k) \cup C(LD_2$ to serve the demand of s_k and LD_2. As LD_2 has just one facility, this is a valid $\texttt{close\_two}$ operation.
- $\texttt{open}(t, LD_1)$ and $\texttt{open}(t, LD_3)$: The flow from LD_1 to $C(LD_1)$ is at most flow from LD_1 to t. Also, the total flow of LD_1 to t is at most half of the total flow(or capacity) at t. So, we can close LD_1 and serve all the demand served by them by opening t. The same arguement holds for LD_3 also.

From the foregoing discussion, it is clear that we have considered a set of operations such that:

- Each facility in $S - S^*$ is closed exactly once.
- Each facility in S^* is opened at most three times as root of a subtree and at most twice as depth two facility. Thus each facility in S^* is opened at most five times.
- For each edge in the solution to the transshipment problem, the total flow due to the operations considered is at most twice the flow in the solution.

A lemma similar to the lemma 3.5 can be proved.

Lemma 3.8 *The facility cost of the solution S is at most $C_f(S) \leq 5.C_f(S^*) + 2.(C_s(S) + C_s(S^*)) \leq 7.C_f(S^*) + 4.C_s(S^*)$.*

The following them holds.

Theorem 3.9 *For any instance of 1-CFL, a locally optimum solution with respect to* `add`, `open`, `close`, *and* `close_two` *operations, denoted by S has a total cost $C(S) \leq 8.C_f(S^*) + 5.C_s(S^*)$ where S^* is any optimal solution to the 1-CFL instance.*

The above theorem implies a $8 + \epsilon$ approximation to the 1-CFL problem. The asymmetry in the co-efficients of $C_f(S^*)$ and $C_s(S^*)$ in lemma 3.8 helps us to scale the facility costs appropriately to obtain a $7.60 + \epsilon$ approximation to the 1-CFL problem.

References

[1] M. Mahdian and M. Pál. Universal facility location. In *Proceedings of 11th Annual European Symposium on Algorithm s*, pages 409–422, 2003.

[2] M. Pal, E. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 329–338, 2001.