APPROXIMATION ALGORITHMS FOR FACILITY LOCATION PROBLEMS

A thesis submitted to University of Delhi

for the award of the degree of

Doctor of Philosophy

by

Manisha Bansal

Department of Computer Science

University of Delhi

Delhi-110007, India

October, 2013

APPROXIMATION ALGORITHMS FOR FACILITY LOCATION PROBLEMS

A thesis submitted to University of Delhi

for the award of the degree of

Doctor of Philosophy

by

Manisha Bansal

Department of Computer Science

University of Delhi

Delhi-110007, India

October, 2013

Declaration

The thesis entitled “*APPROXIMATION ALGORITHMS FOR FACILITY LOCATION PROBLEMS*”, which is being submitted for the award of the degree of Doctor of Philosophy is a record of original and bona fide research work carried out by me in the Department of Computer Science, University of Delhi, Delhi, India.

The work presented in this thesis has not been submitted to any other university or institution for any academic award.

Manisha Bansal
Department of Computer Science,
University of Delhi,
Delhi, India.

Certificate

This is to certify that the thesis entitled “*APPROXIMATION ALGORITHMS FOR FACILITY LOCATION PROBLEMS* ” being submitted by Manisha Bansal in the Department of Computer Science, University of Delhi, Delhi, for the award of degree of Doctor of Philosophy is a record of original research work carried out by her under the supervision of Dr. Neelima Gupta.

The thesis or any part thereof has not been submitted to any other University or institution for any academic award.

Supervisor	**Head**
Dr. Neelima Gupta	Mr. P.K. Hazra
Department of Computer Science	Department of Computer Science
University of Delhi	University of Delhi
Delhi, India.	Delhi, India.

Acknowledgment

I would never be able to adequately thank Dr. Neelima Gupta, my supervisor and mentor, for helping me in developing my research topics. Her continued guidance and support during last four years has helped me accomplish this work. She allowed me to work freely, but was always there whenever I needed advice or guidance. I want to thank her not only for her tremendous guidance and encouragement throughout my study, but also her endless trust and understanding.

I thank Prof. Naveen Garg for his continual support during all these years. He would never say no whenever I sought time for discussion.

I also thank members of my advisory committee, Dr. Naveen Kumar and Dr. Vasudha Bhatnagar for their guidance and support in the last four years.

I am grateful to The Principal of my college, Dr. Babli Moitra Saraf, for her cooperation during these years. She inspires me in many ways.

I am also deeply thankful to my head of department, Mr. P.K. Hazra, for his cooperation during the research.

My deepest appreciation goes to my entire family, including my husband, Deepak, my kids, Jiya and Manu, my mother, my brothers and my father-in-law. I could not have finished this research without their support, love, and encouragement. Most specially, I want to thank my husband, who has always supported me in my endeavors and my kids who have silently lent me all the support they can. I wish my father was here to see my achievement.

I thank my friends, Geeta, Seema, Veenu, Vikas, Rampal, Ritu, Archana, Manoj, and many others who have in some way or the other helped me through this journey.

Last but not the least, my sincere thanks also goes to support staff at Department of Computer Science who were always ready to help with smiling faces and supported me throughout studies.

Manisha Bansal

Abstract

Given a set of facilities F, a set of clients C, demand d_j with each client j, facility opening cost f_i for a facility i and c_{ij}, the service cost of assigning a client j to facility i, the aim of *facility location problem* is to open a set of facilities such that the total cost of opening these facilities together with the service cost of all the clients is minimized. The problem addressed in the thesis is a metric facility location problem in which the service costs satisfy the metric property.

We discuss the capacitated version of the problem in which a capacity constraint is associated with each facility. Local search based approximation results are presented for three variants of the problem. First is *uniform capacitated facility location problem* in which all facilities in F have the same capacity, denoted by U. We analyze a local search based heuristic proposed by Kuehn and Hamburger [KH63] to show that this heuristic provides a (3+ϵ)-factor approximation (for the problem) which improves upon the (5.83+ϵ)-factor of Chudak and Williamson [CW99, CW05]. We give an example to show that the analysis is tight.

In the second variant different facilities have different capacities and it is called *non-uniform capacitated facility location problem* or just capacitated facility location problem. For this problem, we give a (5+ϵ)-factor approximation algorithm which improves the current best of (5.83+ϵ) given by Zhang, Chen and Ye [ZCY05]. For this algorithm also we provide a tight example.

The third problem we consider is *universal facility location problem* which is a gen-

eralization of many variants of facility location problem including the first two problems. In this problem facility cost of a facility $i \in F$ is given by a function $f_i(.)$ and is determined by the capacity allocated at the facility. For this problem, we give a simpler algorithm and show that the cost of the solution is bounded by (5+ϵ) times the cost of the optimum solution. The result is an improvement upon the (6.702+ϵ)-factor of Vygen [Vyg07] and also upon the (5.83+ϵ)-factor given by Angel, Thang and Regnault [ATR13] in a parallel work. This also implies a simpler algorithm for non-uniform capacitated facility location with the same factor.

The key ideas of our analysis are: after assigning the clients of facility being closed to the facility being opened in the operation if the opened facility has some capacity remaining, clients of other facilities in our solution are assigned to it if it results in cost saving; we take a linear combination of some inequalities in a smart way to obtain the claimed approximation guarantees.

We also performed some experiments with our third algorithm for a particular case of nonuniform capacitated facility location problem and found that the algorithm works well in practice. The cost of solutions were found to be within $(1 + 0.12)$ times the optimum solution's cost.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Consider a situation in which a retailer is interested in opening supermarket stores at strategic locations so that the cost of establishing the desired infrastructure is not too high while maintaining close vicinity to a large number of customers/clients. Consider another example where a bank wants to set up its ATMs across the city. Since installation of an ATM machine incurs a cost apart from the price of the land, it is not possible to open a large number of ATMs. Bank needs to choose the locations in such a manner so that the cost of setting up ATMs together with the cost incurred by its customers in traveling to an ATM is minimized. Consider a network service provider interested in providing wireless services to its customers. He would like to select the locations to install the base stations so as to maximize the coverage area. Similarly, a government of a state may want to select locations for schools, hospitals, fire stations and other such utilities so as to provide an easy and fast access to a large number of citizens. These are typical examples of what is called a *facility location problem (FLP)*. What is common among these examples is that the locations for facilities need to be identified in a cost efficient manner (set of potential locations may be fixed). The goal is to minimize the cost incurred (if any) in setting up the facilities at the selected locations with an objective to meet the demand of customers in the best possible way.

Facility location problems have been widely studied since 1960's. These problems are known to be strongly NP-hard. An instance of a minimum weight set cover problem with unit weights, which is an NP-hard problem, can be transformed into metric uncapacitated facility location problem [KV05]. Metric uncapacitated FLP is the simplest amongst the various variants of the problem. Earlier studies discuss different heuristic methods to solve these problems. Kuehn and Hamburger [KH63] suggested one of the earliest heuristic to solve plant location problem. The drawback of heuristic methods is that they do not guarantee the quality of the solution.

First algorithm for FLP with performance guarantee was given by Shmoys, Tardos and Aardal [STA97]. Many different variants of FLP have been studied since then. A variety of approximation algorithmic techniques have been applied to solve these problems. In this work, we present (3+ϵ)-factor approximation for *uniform capacitated facility location problem* which improves the current best of (5.83+ϵ)-factor due to Chudak and Williamson [CW99, CW05] (using the scaling techniques of Charikar and Guha [CG99, CG05]); (5+ϵ)-factor for *non-uniform capacitated facility location problem* which improves the (5.83+ϵ) factor given by Zhang, Chen, and Ye [ZCY05]; and (5+ϵ)-factor for *universal facility location problem* which improves (5.83+ϵ) factor given by Angel, Thang and Regnault [ATR13].

In this chapter we discuss some variants of facility location problems, different techniques which have given good approximation factors for these type of problems and our contributions in brief.

1.1 Facility Location Problem

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 facility. Opening a facility at a location $i \in F$ costs f_i (the facility opening cost). The cost of servicing a client $j \in C$ by a facility i is given by c_{ij} (the service cost). The objective is to determine a set $F' \subseteq F$ and an assignment of clients to facilities in F' so that the total cost of opening the facilities in F' along with the total cost of serving the demands of the clients is minimized. For the rest of the thesis, we assume that service costs form a metric and whenever we say *facility location problem* we mean *metric facility location problem*. Thus, for facilities i, i' and clients j, j', $c_{i'j'} \leq c_{i'j} + c_{ij} + c_{ij'}$. 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. $c_{ii'} = \min_{j \in C}(c_{ji} + c_{ji'})$.

1.1.1 Uncapacitated Facility Location Problem (UFLP)

Consider a manufacturing company which wants to set up regional warehouses in every state so as to minimize the sum of the fixed set up costs of warehouses and the cost of transportation to its customers. It is assumed that a warehouse has sufficient supplies so that any number of customers can be served at a particular instant of time. This is an example of an *uncapacitated facility location problem (UFLP)*. As mentioned earlier, UFLP is the simplest variant of the problem. This problem can be formulated as the following Integer Linear Program, where the indicator variables y_i represent whether a facility is open or not and the indicator variables x_{ij} represent whether a client j is assigned to facility i or not.

$$\begin{array}{rcll} \text{min} & & \displaystyle\sum_{i \in F} f_i y_i + \sum_{i \in F} \sum_{j \in C} c_{ij} d_j x_{ij} & \\ \text{s.t.} & & & \\ x_{ij} & \leq & y_i & \forall\, i \in F, j \in C \\ \displaystyle\sum_{i \in F} x_{ij} & = & 1 & \forall\, j \in C \\ x_{ij} & \in & \{0,1\} & \forall\, i \in F, j \in C \\ y_i & \in & \{0,1\} & \forall\, i \in F \end{array}$$

A feasible solution to this integer program is a set $F' \subseteq F$ and an assignment

$\sigma : C \rightarrow F'$ where $F' = \{i : y_i = 1\}$ and $\sigma(j) = i$ if $x_{ij} = 1$. In fact once F' is known, each client can be assigned to the nearest open facility. Thus F', the set of open facilities, completely defines a solution to the uncapacitated problem.

1.1.2 Capacitated Facility Location Problem (CFLP)

Consider an example in which there is a need to set up wired LANs to satisfy the connectivity needs of an institution. Let us assume that the switches are facilities which facilitate connections among the computers connected through that switch. Each switch has a limited number of slots available which restricts the number of computers that can be connected through it. Since the facilities (switches) have capacities (number of slots) on the number of clients (computers) it can serve, it makes an instance of *capacitated facility location problem (CFLP)*.

More formally, in capacitated facility location, each facility i has a capacity u_i specifying the maximum amount of demand it can serve. There are two variants of this problem: CFLP with unsplittable demands (all the demand of a client must be served by the same facility) and CFLP with splittable demands (demand of a client can be split and assigned to multiple open facilities). The first variant is even hard to approximate unless P = NP [BH12]. When capacities of all the facilities are same, the problem is known as *uniform capacitated facility location problem (UCFLP)*. Rightly so, when capacities are not necessarily the same, it is called *non-uniform capacitated facility location problem* or just *capacitated facility location problem*.

The problem variant with splittable demands can be formulated as the following mixed integer linear program (MILP), wherein x_{ij} variables are allowed to be non-integral to capture the splittable nature of demands.

$$\begin{array}{rcll}
\text{min} & & \displaystyle\sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in C} c_{ij} d_j x_{ij} & \\
\text{s.t.} & & & \\
x_{ij} & \leq & y_i & \forall\, i \in F, j \in C \\
\displaystyle\sum_{i\in F} x_{ij} & = & 1 & \forall\, j \in C \\
\displaystyle\sum_{j\in C} d_j x_{ij} & \leq & u_i y_i & \forall\, i \in F \\
x_{ij} & \geq & 0 & \forall\, i \in F, j \in C \\
y_i & \in & \{0,1\} & \forall\, i \in F
\end{array}$$

A feasible solution to the above MILP is given by a set $F' \subseteq F$ and an assignment of clients to the facilities in F' which obeys the capacity constraints where F' is the set of facilities i for which $y_i = 1$. Note that in CFLP clients cannot always be assigned to the nearest open facilities as it may lead to violation of capacity constraints. Once F' is known, best assignment of clients can still be found in polynomial time by just solving an assignment problem. Thus any solution to CFLP is also completely defined by the set of open facilities.

1.1.3 Universal Facility Location Problem (UniFLP)

Many a times facilities do not have fixed facility costs. For example suppose a corporation office needs to develop playgrounds in a city. The number of children under 15, whose playing needs are to be addressed by these playgrounds, is known in advance. Various locations identified for the purpose have different per-square feet rates. Locations and their sizes need to be identified in such a manner so that every child has easy access to a playground and total cost of establishing playgrounds is minimized. In this particular example, cost of a playground depends both on its location as well as its size. This is an example of *universal facility location problem (UniFLP)*. Formally, FLP is said to be UniFLP if the facility opening cost f_i of a facility i is a function of the capacity allocated to the facility. Thus if u_i is the capacity allocated at a facility location i then the cost incurred for opening the facility is $f_i(u_i)$, which is a monotonically non-decreasing function. The objective is to minimize the sum of the total facility cost $\sum_{i \in F} f_i(u_i)$ plus the total service cost.

This problem can be formulated as the following non-linear program (NLP), where variables u_i denote the amount of capacity allocated to facility i and x_{ij} variables are non-integral and denote the amount of demand of client j served by facility i.

$$\begin{array}{rcll}
\text{min} & & \displaystyle\sum_{i \in F} f_i(u_i) + \sum_{i \in F} \sum_{j \in C} c_{ij} d_j x_{ij} & \\
\text{s.t.} & & & \\
\displaystyle\sum_{i \in F} x_{ij} & = & 1 & \forall\, j \in C \\
\displaystyle\sum_{j \in C} d_j x_{ij} & \leq & u_i & \forall\, i \in F \\
x_{ij} & \geq & 0 & \forall\, i \in F, j \in C \\
u_i & \geq & 0 & \forall\, i \in F
\end{array}$$

A solution to UniFLP is characterized by (U, σ) where $U = \langle u_1, u_2, \cdots, u_n \rangle$ is an allocation vector and $\sigma : C \to F$ is an assignment which obeys the capacity constraints. Universal facility location problem generalizes many other facility location problems including UFLP and CFLP. It reduces to UFLP when $f_i(u_i) = f_i'$ for $u_i > 0$ and to CFLP when we define $f_i(u_i) = f_i'$ for $0 < u_i \leq c_i$, c_i is the capacity of facility i, and $f_i(u_i) = \infty$ otherwise.

1.1.4 k-Median Problem

Consider the warehouse example again. Suppose there is a licensing authority which gives fixed number of licenses say k for warehouses and no cost of establishing warehouses is incurred by the company. The aim of the company is now to choose the locations intelligently within the budget constraint on the number of licenses for the warehouses available to it so as to minimize the total cost of transportation to its customers. A warehouse can serve any number of customers. This is an example of *uncapacitated k-median problem*. The problem is better known as just *k-median problem*.

This problem can be formulated as the following Integer Linear Program:

$$
\begin{aligned}
\min \quad & \sum_{i\in F}\sum_{j\in C} c_{ij} d_j x_{ij} \\
\text{s.t.} \quad & \\
x_{ij} &\leq y_i && \forall\, i \in F, j \in C \\
\sum_{i\in F} x_{ij} &= 1 && \forall\, j \in C \\
\sum_{i\in F} y_i &\leq k \\
x_{ij} &\in \{0,1\} && \forall\, i \in F, j \in C \\
y_i &\in \{0,1\} && \forall\, i \in F
\end{aligned}
\tag{1.1}
$$

A solution to this problem is described by a set $F' \subseteq F$ where $|F'| \leq k$ and an assignment $\sigma : C \rightarrow F'$ where F' and σ are as defined earlier for UFLP. Since there are no capacities associated with a facility, a client gets assigned to the nearest open facility. Therefore solution to the k-median problem is also completely determined by the set of open facilities.

1.1.5 Other variants

Some other existing variants of FLP which are similar to these problems in some way or the other are: k-facility location problem [Zha07], soft capacitated facility location problem [CS03, AGK$^+$01], red-blue median problem [HKK12] and mobile facility location problem [AFS13]. k-facility location problem is UFLP with an added constraint which specifies the maximum number of open facilities in a solution. Red-blue median problem is a generalization of k-median where the facility set F is partitioned into two sets: red and blue, at most k_r red facilities and at most k_b blue facilities can be opened in any solution. Mobile facility location problem is also a generalization of k-median problem. Soft-capacitated FLP is an easier version of CFLP where we can open multiple copies of a facility.

1.2 Algorithmic Techniques for Approximation Algorithms for FLP

The problems discussed in the previous section are known to be the NP-hard [KV05]. One common approach to solve these problems is to provide approximate solutions. An α-approximation algorithm is a polynomial time algorithm which finds a solution whose cost is within α-factor of the cost of an optimum solution. For a minimization problem $\alpha > 1$ and for a maximization problem $\alpha < 1$.

Many techniques have been used for the design of approximation algorithms for FLP. These include LP-rounding, primal-dual approach, greedy approach, local search technique to name a few. Each of these techniques has its own advantages as well as disadvantages. The first approximation algorithm for a facility location problem uses LP-rounding approach [STA97]. This approach involves solving LP relaxation of an Integer programming formulation of the problem and hence is generally quite time consuming. Primal-dual algorithms are relatively faster. Greedy and local search algorithms are generally simple to implement but hard to analyze. There are very few problems to which local search technique has been applied successfully. We next review these techniques briefly.

1.2.1 LP-Rounding

Linear programming has been used extensively in designing approximation algorithms. Many of the optimization problems can be formulated as an Integer Linear Program. Some of them admit a polynomial time solution but many of them are NP-hard. LP-relaxation followed by rounding the solution of the relaxed LP is often used to get good approximation algorithms for these problems. Sometimes the rounding technique is trivial as is the case in Vertex Cover problem [Hoc82a] but more often than not, it requires sophisticated techniques to round the fractional solution of the linear program to obtain an integral solution.

First approximation algorithm for uncapacitated facility location problem by Shmoys, Tardos and Aardal [STA97] uses LP-rounding to give 3.16-factor approximation algorithm. The technique has since been used to develop several approximation algorithms for UFLP with improved approximation factors [CS03, Svi02, Byr07] with the current best being 1.488 by Shi Li [Li13]. This is a very recent result which shows that the attempts are on to close the gap between the lower bound of 1.463 [GK99] and the upper bound for the problem.

LP-rounding has been applied to obtain good approximation factors for other variants of the problem as well. For example, Charikar and Li [CL12] gave a 3.25-factor for the k-median problem in a recent result and, Chudak and Shmoys [SC99] gave a 3-factor approximation for the soft-capacitated version. Levi, Shmoys and Swamy [LSS12] gave a 5-factor algorithm for CFLP for a restricted version of the problem in which all facility costs are same. This algorithm is a combination of rounding approach in the first phase followed by greedy in the second phase. More approximation algorithms for other related problems which have used LP-rounding technique can be found in [BSS10, XX05, BGRS10, SS02, CCGG98, AAK99, HJC08].

1.2.2 Primal-Dual Approach

Primal-dual approach is a technique that uses the dual of the problem to obtain a good integral solution to the relaxed linear program. Starting with a feasible solution of the dual and an infeasible solution to the primal, a feasible solution to the primal is built in steps. Iteratively the feasibility of the primal solution is improved, which is guided by the successive improvements in the optimality of the dual solution. Primal solution is always modified integrally in every iteration, so that the resultant solution is an integral solution. The cost of the primal solution is bounded by an appropriate factor of the dual feasible thereby providing an approximation of the primal optimal.

Jain and Vazirani [JV01] used primal-dual approach to give a 3-factor algorithm for uncapacitated facility location problem and a 6- factor algorithm for the k-median problem. There are many network design problems which have used primal-dual successfully [AKR95, SK04, KGR02, AZ02]. For a survey on approximation algorithms based on primal-dual refer to [Wil02].

1.2.3 Greedy Approach

Greedy approach is one of the simplest techniques used to design algorithms for the optimization problems. At any point in time, we have a partially constructed solution and the solution is extended based on some greedy choice that is the current best without bothering about its impact on the future solutions. For many problems like *minimum spanning tree, shortest path, Huffman codes* to name a few, the technique provides a global optimum.

For its simplicity, the technique has been applied to obtain approximation algorithms also. For example, the best known approximation algorithm for the classical problem of *set cover* is greedy [Joh73, Lov75, Chv79]. Surprisingly, this is the best that is possible for the problem [Fei98]. The technique has also been used to provide several results for

UFLP [Hoc82b, JMS02, GK99, MYZ02, MMSV01] and a 4-factor approximation for k-median problem by Charikar and Guha [CG99]. Some other approximation results using greedy approach can be found in [CEK06, Cha00].

1.2.4 Local Search Algorithms

Local search as an algorithm design technique has been around since 1950's. However for the design of approximation algorithms, the success using this approach came only during the last 15 years. The technique of local search is straightforward and simple to apply but notoriously hard to analyze. In this approach we begin with an initial feasible solution which is improved iteratively in every step doing some local improvements. This process is repeated until it is no more possible to improve the cost of the solution locally. The solution thus obtained is said to be locally optimal solution. In the next chapter we discuss the various aspects of local search paradigm in more detail.

1.3 Our contribution

In this thesis, we use local search technique to provide improved results for two variants of facility location problem: CFLP and UniFLP. We consider CFLP with splittable demands in two flavors: uniform (capacities of all the facilities are same) and non-uniform (capacities of different facilities may be different). In all the problems discussed, we have made an assumption that demand of a client $j \in C$ is one i.e. $d_j = 1$. Since demands are splittable this assumption can be easily gotten rid of.

Korupolu, Plaxton and Rajaraman [KPR00, KPR98] first analyzed the *add-delete-swap* heuristic of Kuehn and Hamburger to show that it is an (8+ϵ)-factor algorithm for the uniform capacitated case. Later Chudak and Williamson [CW05, CW99] improved the analysis to improve the factor for the same heuristic to (5.83+ϵ). We further improve the analysis for the same algorithm to give factor (3+ϵ). The key ideas of our analysis are:

1. instead of assigning the clients of the facilities being closed only to the facilities being opened, we also assign them to other facilities in our solution using a mapping similar to the one discussed in Arya *et al.*[AGK$^+$01].

2. after assigning the clients of facility being closed to the facility being opened in the operation if the opened facility has some capacity remaining, clients of other facilities in our solution are assigned to it if it results in cost saving.

3. we take a linear combination of some inequalities in a smart way to obtain the claimed approximation guarantees.

We also give a tight example to show that this factor cannot be further reduced for this heuristic.

A simple *add-delete-swap* heuristic of Kuehn *et al.*, which is so appropriate for CFLP with uniform capacities, is not so good for the case when the capacities are non-uniform. Pal, Tardos and Wexler [PTW01] gave an algorithm with two new operations: *open* and *close*, which are extensions of swap operation in the changed scenario of non-uniform capacities, and Zhang *et al.* [ZCY05] added another operation called *multi* to these operations. We modify these operations to be able to use our key ideas developed for the uniform capacity case to obtain a (5+ϵ)-factor for the problem. We also show that the analysis is tight by providing a tight example.

For the universal facility location problem we suggest two operations *open* and *close* similar to *mopen* and *mclose*, drop the expensive *mmulti* operation altogether and provide a (5+ϵ)-factor approximation for the problem. As a particular case, it provides a simpler algorithm for (non-uniform) CFLP.

All the three results given by us are currently the best results for the respective problems. The results are consolidated in Table 1.3. We also performed an experimental study of our algorithm for the case of non-uniform capacities. The aim of this study is to show that the (5+ϵ)-factor for this algorithm is just an upper bound and in practice the results are much closer to the optimum solution. In this study we present results for the random instances. The results obtained are within $(1 + 0.12)$-factor of the cost of the optimum solution.

	Problem	Current best	Our results
1	uniform capacitated facility location problem	5.83+ϵ [CW05]	3+ϵ
2	non-uniform capacitated facility location problem	5.83+ϵ [ZCY05]	5+ϵ
3	universal facility location problem	5.83+ϵ [ATR13]	5+ϵ

Table 1.1: Results presented in the thesis

The thesis is organized as follows: in Chapter 2 we discuss the local search paradigm in detail; in Chapter 3 we give an improved analysis of the algorithm for uniform capacitated facility location problem; Chapter 4 discusses the (5+ϵ)-factor algorithm for non-uniform capacitated facility location problem; in Chapter 5 we present the (5+ϵ)-factor algorithm for universal facility location problem and the experimental study of the algorithm. At the end we give our conclusive remarks in Chapter 6.

Chapter 2

Local Search Technique

Local search technique (denoted by LS for brevity) has been applied to various combinatorial optimization problems as a solution method. LS is quite powerful, but difficult to analyze. Johnson, Papadimitriou and Yannakakis [JPY85] studied some local search heuristics and defined a class called PLS for them. A local search heuristic for a problem belongs to the class PLS if the local optimal can be verified in polynomial time. Yannakakis [Yan90] gave a detailed study of local search algorithms. For some problems the technique is known to provide good solutions in practice but without any performance guarantees. Famous Kerninghan-Lin algorithm for Graph Partitioning [KL70] and Lin-Kerninghan algorithm for traveling salesman problem (TSP) [LK73] are known to produce good solutions quickly and are widely used and implemented [Hel00, BS89]; however no known performance guarantees exist for these heuristics.

The technique is also known to provide good approximation factors for some well known problems like Max-Cut, satisfiability [Ali94], and various variants of facility location problem. Most of the problems for which local search has been successfully applied with guaranteed quality of solution are different variants of facility location problems [CG99, KPR00, AGK$^+$01, CW05, PTW01, MP03, Vyg07, ALB$^+$13, BGG12, AFS13, BSSS13]. In fact the only approximation algorithms for capacitated facility location problem with constant approximation factors are based on local search technique.

2.1 The general outline of the technique

Many combinatorial optimization problems can be described as: given a set F, find a feasible set $S \subseteq F$ that (satisfies certain criteria and) minimizes/maximizes the cost/value of the solution. Each instance is associated with a finite set of feasible solutions, each solution has some cost. Finding optimal solutions for many of these problems (for example set-cover, vertex-cover etc.) is NP-hard. One way to get around this problem is to look for locally optimal solutions which are known to be optimal/close to optimal for many of them.

A local search algorithm starts with a feasible solution and improves upon it locally as long as it is possible to do so. When it is not possible to further improve the solution locally we arrive at a local optimal. Starting with a different (initial) feasible solution might lead to a different local optimal solution. For an instance I, let $LOPT(I)$ be the cost of a locally optimal solution and $OPT(I)$ be the cost of a globally optimum solution. The supremum of the ratio $\frac{LOPT(I)}{OPT(I)}$, over all such instances I, is called the *locality gap*.

In a local improvement step of a local search procedure, we move from the current solution to a neighboring solution. A neighboring solution to a solution S is determined by a neighborhood relation. The neighborhood relation determines the structure of the neighborhood of a solution. As an example, for the vertex cover problem neighbor of a solution S is another solution T such that $|S - T| = 1$. Different local search heuristics may lead to different neighborhood structures for the same problem.

A local search procedure restricts the search space of the problem instance, thus making it possible to find a locally optimal solution quickly in comparison to the global optimum solution. To perform a local search algorithm in polynomial time it is important that:

- the number of iterations performed by the procedure is polynomial.

- a local improvement step can be performed in polynomial time.

Many a times, the local search heuristic does not lead to locally optimal solution in polynomial number of iterations. Johnson, Papadimitriou and Yannakakis [JPY85] and Yannakakis [Yan90] discuss these type of algorithms and put them in PLS class of algorithms. Lin-Kerninghan and Kerninghan-Lin heuristics also belong to this class. A well known technique, to reduce the number of iterations to polynomial, is to perform an improvement step only when the improvement is substantial. In the lemma below, we show that if the cost of the solution is decreased by at least $\frac{c(S)}{p(n,\epsilon)}$ for a minimization problem then the algorithm terminates in polynomial number of iterations, where S is the current solution and $p(n,\epsilon)$ is a polynomial in n, size of the problem instance, and $\frac{1}{\epsilon}$ for a carefully chosen value ϵ. With this modification the algorithm can be made to terminate in polynomial time.

Lemma 2.1 *Let S_0 be an initial feasible solution with cost $c(S_0)$ and let S^* be a global optimum solution with cost $c(S^*)$. A local search procedure terminates in at most $O\left(p(n,\epsilon) \cdot log\frac{c(S_0)}{c(S^*)}\right)$ iterations if a local search move is performed only when the cost of the current solution S is reduced by at least $\frac{c(S)}{p(n,\epsilon)}$.*

Proof Local search procedure begins with an initial feasible solution S_0 of cost $c(S_0)$. In each iteration cost is reduced by at least $\frac{c(S)}{p(n,\epsilon)}$, where S is the current solution. A local search procedure cannot reduce the cost of solution to less than $C(S^*)$. Let t be the number of iterations used to obtain a local optimal solution S, then

$$c(S) \leq \left(1 - \frac{1}{p(n,\epsilon)}\right)^t \cdot c(S_0)$$

and

$$c(S) \geq c(S^*)$$

therefore,

$$\left(1-\frac{1}{p(n,\epsilon)}\right)^{t}\cdot c(S_0)\geq c(S^*)$$

For $t = p(n,\epsilon).l$ the cost would be

$$\left(1-\frac{1}{p(n,\epsilon)}\right)^{p(n,\epsilon)\cdot l}\cdot c(S_0)\approx\frac{c(S_0)}{e^l}$$

This implies

$$\frac{c(S_0)}{e^l}\geq c(S^*)$$

or

$$l\leq log\frac{c(S_0)}{c(S^*)}$$

This means that the procedure terminates in $O\left(p(n,\epsilon)\cdot log\frac{c(S_0)}{c(S^*)}\right)$ iterations. ■

2.2 Local search heuristics with no performance guarantees

Kuehn and Hamburger [KH63] in early 1960's gave a simple local search heuristic for uncapacitated and capacitated facility location problems. For the capacitated version they gave experimental results for the case when the capacities are uniform. It was much later, almost after 35 years, shown by Korupolu *et al.* [KPR00] that this heuristic provides a locality gap of 8 for the case of uniform capacities.

The other well known local search heuristics which evolved during 1960's to 1990's are for graph partitioning problem and traveling salesman problem. These heuristics evolved from 2-opt heuristic for traveling salesman problem given by Croes [Cro58] in 1958. 2-opt is a simple local search heuristic: a) it starts with a traveling salesman tour, b) swaps 2 edges in the tour with two edges not in the tour so that result is a cheaper tour, c) continues this way until no more improvements are possible. If the number of edges exchanged in each step is λ then the algorithm is said to be λ-opt. Kerninghan-Lin (KL) gave a generalization of λ-opt for graph partitioning problem in 1969. This is an adaptive algorithm and decides a new value of λ in each iteration.

Lin-Kerninghan's algorithm (LK) for traveling salesman problem was designed in 1971 which draws ideas from KL's graph partitioning algorithm and is a generalization of λ-opt for TSP. KL and LK are known to be the most successful heuristic procedures for these problems which use local search approach. These algorithms are very sophisticated. Both these algorithms are known to provide good solutions which are within $1\% - 2\%$ of the optimal solution for most instances. Even after 40 years since its inception LK is one of the best heuristics for the TSP. It has also been adapted to solve generalised travelling salesman problem [KG11]. Chandra, Karloff and Tovey [CKT99] show that λ-opt heuristic for TSP can have an arbitrarily large locality gap. The same result applies to Lin-Kerninghan algorithm also.

2.3 Approximation algorithms based on local search technique

Beginning with an approximation algorithm for satisfiability in 1994 by Alimonti [Ali94] several approximation algorithms have been designed using local search. As mentioned in the previous section, Korupolu *et al.* [KPR00, KPR98] in 1998 analyzed the local search heuristic of Kuehn *et al.* to show that it is within an (8+ϵ)-factor of the optimal for CFLP with uniform capacities and within (5+ϵ)-factor of the optimal for UFLP. Arya *et al.* [AGK$^+$04, AGK$^+$01] improved the locality gap for UFLP from (5+ϵ) to (3+ϵ) and gave a tight example for the same. Chudak *et al.* [CW99, CW05] improved the analysis of Korupolu *et al.* for CFLP to give a factor of (5.83+ϵ) for uniform capacities. In our work we have further improved the analysis to obtain (3+ϵ)-factor [ALB$^+$13, AAB$^+$10]. We also provide a tight example to put to rest any further scope of improvement in the analysis of the heuristic given by Kuehn *et al.*. This means that to seek any further improvement in the locality gap for the problem, we need to devise new local search heuristics for the problem.

Kuehn *et al.* deals with CFLP only with uniform capacities. The neighborhood structure of the approach does not deal with the case of non-uniform capacities. In 2001 Pál, Tardos and Wexler [PTW01] gave a different local search algorithm which is well suited for the case of non-uniform capacities and is the first such result for this problem. They showed that their algorithm has a locality gap of (8.53+ϵ). Mahdian and Pál [MP03] gave an LS algorithm for universal facility location problem with (7.88+ϵ)-factor. Since UniFL is a generalization of capacitated facility location problem, the result also implied an improvement for non-uniform capacities. Local search algorithm by Zhang, Chen and Ye [ZCY05] improved the locality gap to (5.83+ϵ) in 2004. We modify the operations of Pal *et al.* [PTW01] and Zhang *et al.* [ZCY05] and give a solution which is at most (5+ϵ) times the cost of an optimum solution [BGG12]. By giving a tight example, we prove that the analysis for our algorithm cannot be strengthened any further.

Result of Mahdian *et al.* [MP03] for universal facility location problem was improved by Vygen in 2007 [Vyg07] to (6.702+ϵ). The line of their analysis is very similar to that of Zhang *et al.*. Recently in a parallel work with ours, Angel, Thang and Regnault in [ATR13] have reduced this factor further to (5.83+ϵ). However, we give a better factor of (5+ϵ) for the problem by extending our results for CFLP to this problem.

There are other variants of facility location problems which have benefited from the local search paradigm. Charikar and Guha [CG99, CG05] gave a local search based algorithm with (2.414 + ϵ)-factor for soft-capacitated version in 1999. In 2001, Arya *et al.* [AGK$^+$04, AGK$^+$01] gave first LS algorithm for the k-median problem with (3+ϵ)-factor. Gupta and Kanat [GT08] simplified the analysis given by Arya *et al.* for the same factor. In 2005 Devanur *et al.* [DGK$^+$05] gave a (5+ϵ)-factor algorithm for k-facility location problem and showed that it can be used to bound the price of anarchy of the defined game. In 2007, we saw a result, a (2+$\sqrt{3}$)-factor, on k-facility location by Zhang

[Zha07] . In 2010, a constant factor on budgeted red-blue median problem was given by Hajiaghayi, Khandekar and Kortsarz [HKK10, HKK12]. Ahmadian, Friggstad and Swamy gave a (3+ϵ)-factor algorithm for the mobile facility location problem [AFS13] and suggest that any improvement in factor for this problem means a similar improvement for the k-median problem. Currently their factor matches the best known for k-median problem using local search.

Other problems for which local search heuristic has provided approximation algorithms include k-means clustering problem [KMN$^+$02]. Microprocessor scheduling by Shuurman *et al.* [SV07], balls into bins by Bogdan *et al.* in [BSSS13], Maximum coverage over a matroid by Filmus *et al.* [FW12], max 3-cover problem by Caragiannis *et al.*in [CM13], k-exchange systems [FNSW11], ranking tournaments [CW09], the Maximum Set Packing Problem [SW13]. Refer to [Ang06] for more approximation results using local search.

Chapter 3

A (3+ϵ)-Approximation Algorithm for the Facility Location Problem with Uniform Capacities

In this chapter we deal with the *uniform capacitated facility location problem* in which the number of clients that a facility i can serve is bounded by u_i and $u_i = U$, for all i. For this problem of uniform capacities the first approximation factor was due to Korupolu, Plaxton and Rajaraman [KPR00, KPR98] who analyzed the local search algorithm by Kuehn *et al.* [KH63] and proved that any locally optimal solution has cost no more than (8+ϵ) times cost of an (global) optimum solution. Chudak and Williamson [CW05, CW99] strengthened the analysis in [KPR00, KPR98] to obtain a (5.83+ϵ)-approximation.

Given a set of open facilities, the best way of serving the clients, can be determined by solving an assignment problem [STA97]. Thus any solution is completely determined by the set of open facilities. The local search procedure proposed by Kuehn *et al.* starts with an arbitrary set of open facilities and then updates this set, using one of the operations *add, delete, swap,* whenever that operation reduces the total cost of the solution. We show that a solution which is locally optimal with respect to this same set of operations is a

(3+ϵ)-approximation. We then show that our analysis of this local search algorithm is best possible by demonstrating an instance where the locally optimum solution is three times the (global) optimum solution.

All earlier work for capacitated facility location (uniform or non-uniform), in their analysis, is able to capture rerouting all the clients in a swap operation from the facility which is being closed to the one being opened. This however can be quite expensive and cannot lead to the tight bounds that we achieve. We use the idea of Arya et.al. [AGK$^+$01, AGK$^+$04] to reassign some clients of the facility being closed in a swap operation to other facilities in our current solution. However, to be able to handle the capacity constraints in this reassignment we need to extend the notion of the mapping between clients used in [AGK$^+$01, AGK$^+$04] to a fractional assignment. We also show rerouting of clients from their current facilities to the facility opened in the swap operation to better utilize its remaining available capacity.

As in earlier work, we use the fact that when we have a local optimal, no operation leads to an improvement in cost. However, we now take carefully defined *linear combination* of the inequalities capturing this local optimality. All previous work that we are aware of seems to only use the *sum* of such inequalities and therefore requires additional properties like the integrality of the assignment polytope to identify suitable swaps [CW05, CW99]. Our approach is therefore more general and amenable to better analysis. The idea of doing things fractionally appears more often in our analysis. Thus, when analyzing the cost of an operation we assign clients fractionally to the facilities and rely on the fact that such a fractional assignment cannot be better than the optimum assignment which follows from the integrality of the assignment polytope.

In Section 3.5 we give a tight example that requires the construction of a suitable set-system. While this construction itself is quite straightforward, this is the first instance we know of where such an idea has been applied to prove a large locality gap.

3.1 Preliminaries

Let C be the set of clients, F denote the facility locations and U be the capacity of each facility in F. Let $|C| = m$ and $|F| = n$. Let S (resp. O) be the set of open facilities in our solution (resp. optimum solution). With abuse of notation we use S (resp. O) to denote our solution (resp. optimum solution). Initially S is an arbitrary set of facilities which can serve all the clients. Let $c(S) = c_f(S) + c_s(S)$ denote the total cost (facility cost plus the service cost) of solution S. The three operations that make up the local search algorithm of Kuehn *et al.*are

add For $s \notin S$, if $c(S + \{s\}) < c(S)$ then $S \leftarrow S + \{s\}$.

delete For $s \in S$, if $c(S - \{s\}) < c(S)$ then $S \leftarrow S - \{s\}$.

Swap For $s \in S$ and $s' \notin S$, if $c(S - \{s\} + \{s'\}) < c(S)$ then $S \leftarrow S - \{s\} + \{s'\}$.

S is locally optimal if none of the three operations are possible and at this point the algorithm stops.

Recall that $f_i, i \in F$ is used to denote the cost of opening a facility at location i and c_{ij} is used to denote the cost of assigning client j to facility i. Let S_j and O_j denote the service-cost of client j in the solutions S and O respectively. The presence of the *add* operation ensures that the total service cost of the clients in any locally optimal solution is at most the total cost of the optimum solution [KPR00]. Formally,

Lemma 3.1 ([KPR00]) *For any locally optimal solution* S, $\sum_{j \in C} S_j \leq \sum_{j \in C} O_j + \sum_{o \in O} f_o$.

For the sake of completeness we prove this lemma in Section 3.2 for the case when $S \cap O = \phi$ and again in Section 3.4 for the general case.

We next show that the facility cost of a locally optimal solution is no more than 2 times the cost of an optimum solution. We prove this by identifying a suitable set of local operations and determine the increase in cost if these operations were to be performed. Since the solution is locally optimal, the increase in cost due to these operations is non-negative[1]. This gives us a set of inequalities and a suitable linear combination of these inequalities yields the bound on the facility cost of the locally optimal solution. Note that the inequalities generated are only for the purpose of analysis; we do not actually perform those local operations since we are already at a locally optimal solution.

Combining the bounds of the service cost and the facility cost of a locally optimal solution then gives us our main theorem:

Theorem 3.2 *For any locally optimal solution S and an optimum solution O to the facility location problem with uniform capacities, $c(S) \leq 3c(O)$.*

To ensure that our procedure has a polynomial running time we use an idea first proposed in [KPR00] — a local step is performed only if the cost of a solution S' reduces by more than $(\epsilon/4n)c(S')$ where $\epsilon > 0$ and $n = |F|$ is the number of facility locations. It is immediate from Lemma 2.1 that as a result of this modification the number of local search steps done is at most $4n\epsilon^{-1}\log(c(S_0)/c(O))$ where S_0 is the initial solution. In Section 3.3 we argue that the approximation guarantee of this modified local search procedure increases to at most $3/(1-\epsilon)$.

The rest of the chapter is organized as follows. In Section 3.2 we give a little loose bound of (3,2) on the facility costs of the locally optimal solution which means that the facility cost of S is at most 3 times the facility cost plus 2 times the service cost of the optimal solution. We improve upon this bound in Section 3.3 by giving a better utilization of opened facilities. In both these sections we assume that the facilities in the locally optimal solution S are disjoint from the facilities in the optimum solution O. Most of the new ideas for this problem appear in these two sections. In Section 3.4 we extend the argument to the case when the facilities in S and O are not disjoint. In Section 3.5 we

[1]In fact, we do not determine the exact increase in cost when a local operation is performed but only an upper bound on this quantity.

give an example of a solution which is locally optimal with respect to the operations of *add, delete, swap* and has cost three times the optimum. This establishes that our analysis is tight.

3.2 Bounding the cost when $S \cap O = \phi$: A loose bound on facility cost

For this section and the next, we assume that the sets S and O are disjoint. This assumption allows us to add any facility of O or to swap any facility in S with a facility in O without worrying about the possibility that the facility of O included in our solution might already be a part of S. Let $N_S(s)$ denote the clients served by facility s in the solution S and $N_O(o)$ denote the clients served by facility o in solution O. Let N_s^o denote the set of clients served by facility s in solution S and by facility o in solution O i.e. $N_s^o = N_S(s) \cap N_O(o)$. For a client $j \in C$, let $\sigma(j)$(respectively $\tau(j)$) be the facility which serves j in solution S(respectively O).

We first give the proof of Lemma 3.1 to bound the service cost of solution S under the assumption that $S \cap O = \phi$.

Proof Consider the operation $\texttt{add}(o)$ for a facility $o \in O$. Since S is a locally optimal solution therefore by assigning clients from $N_O(o)$ to o, instead of best assignment of clients to facilities in $S \cup \{o\}$, would not improve the cost of the solution. We can therefore write the following inequality due to this operation:

$$f_o + \sum_{j \in N_O(o)} (O_j - S_j) \geq 0 \tag{3.1}$$

We can write one such inequality w.r.t. each $o \in O$. Adding all these inequalities, we get

$$\sum_{o \in O} f_o + \sum_{o \in O} \sum_{j \in N_O(o)} (O_j - S_j) \geq 0$$

or

$$\sum_{o \in O} f_o + \sum_{j \in C} (O_j - S_j) \geq 0$$

This implies for a locally optimal solution S,

$$\sum_{j \in C} S_j \leq \sum_{j \in C} O_j + \sum_{o \in O} f_o \tag{3.2}$$

■

To bound the facility cost of S, we will consider *swap/delete* operations in which we close a facility s and assign the clients served by s to other facilities in S and, some facility in O if required. Recall that this is done only for the purpose of analysis and to help generate inequalities which arise from the fact that S is locally optimal. The reassignment of a client j served by s to the facilities in S is done using a *fractional assignment* $\pi_o : N_O(o) \times N_O(o) \rightarrow \Re^+$ where $o = \tau(j)$. $\pi_o(j, j')$ defines the extent up to which j is assigned to the facility $\sigma(j')$ when the facility $\sigma(j) = s$ is closed. Clearly, for this fractional assignment we need to choose j' such that $\sigma(j') \neq s$ (we call this the *separation property* of the fractional assignment). For the purpose of analysis, we also require that $\pi_o(j', j) = \pi_o(j, j')$.

Let $\texttt{wt}(j)$ denote the total extent up to which j is assigned to other facilities in S using this assignment. Therefore for $j \in N_S(s)$

$$\sum_{j' \in C : j' \neq j} \pi_o(j', j) = \sum_{j' \in C : j' \neq j} \pi_o(j, j') = \texttt{wt}(j)$$

We call these equalities as the *balance property* of the fractional assignment.

Note that $1 - \texttt{wt}(j)$ fraction of j remains unassigned. We define the *residual weight* of j as $1 - \texttt{wt}(j)$ and denote it by $\texttt{res-wt}(j)$. This much extent of j will be assigned to some facility in O.

Before we formally explain how to compute $\texttt{wt}(j)$ and π_o, let's consider the following examples ($U = 100$):

1. A facility s_1 is serving 80 clients in S, i.e. $|N_S(s_1)| = 80$. When s_1 is closed, at most 80 clients from s_1 can be reassigned using the fractional assignment, to other facilities in S. But s_1 can receive at most 20 clients, through the fractional assignment, from other facilities in $S - \{s_1\}$ when they are closed. Therefore due to the balance property of fractional assignment, when s_1 is closed only 20 clients can be reassigned using fractional assignment.

2. a facility s_2 is serving 20 clients in S, i.e. $|N_S(s_2)| = 20$. Though s_2 can receive 80 clients from other facilities in S when they are closed, but at most 20 clients from s_2 can be reassigned using fractional assignment, to other facilities in S when s_2 is closed. Again, due to the balance property of fractional assignment s_2 can receive atmost 20 clients from other facilities using fractional assignment.

The above examples show that at most $\min(U - |N_S(s)|, |N_S(s)|)$ clients from a facility s can be reassigned using the fractional assignment. However instead of assigning these many clients from s, when s is closed, we will assign all clients $j \in N_S(s)$ up to at most $\min\left(1, \frac{U-|N_S(s)|}{|N_S(s)|}\right)$ extent, i.e.

$$\texttt{wt}(j) \le \min\left(1, \frac{U - |N_S(s)|}{|N_S(s)|}\right).$$

In order to be able to define a valid fractional assignment, we require that for all $o \in O$ and $s \in S$, $\texttt{wt}(N_s^o) \le \texttt{wt}(N_O(o))/2$. Here for $X \subseteq C$, $\texttt{wt}(X)$ denotes the sum of the weights of the clients in X.

Therefore, $\texttt{wt}(j)$ for each $j \in C$ satisfies the following two properties:

1. $\texttt{wt}(j) \le \min\left(1, \frac{U-|N_S(\sigma(j))|}{|N_S(\sigma(j))|}\right)$.
2. For all $o \in O$ and $s \in S$, $\texttt{wt}(N_s^o) \le \texttt{wt}(N_O(o))/2$.

Now we will explain how to compute $\texttt{wt}(j)$ that satisfies these properties. Let $\texttt{init-wt}(j) = \min\left(1, \frac{U-|N_S(\sigma(j))|}{|N_S(\sigma(j))|}\right)$. Since $|N_S(\sigma(j))| \le U$, we have that $0 \le$

$\texttt{init-wt}(j) \leq 1$. To determine $\texttt{wt}(j)$ so that the above two properties are satisfied we start by assigning $\texttt{wt}(j) = \texttt{init-wt}(j)$. However, this assignment might violate the second property. We say that a facility $s \in S$ *captures* a facility $o \in O$ if $\texttt{init-wt}(N_s^o) > \texttt{init-wt}(N_O(o))/2$. Note that at most one facility in S can capture a facility o. If s does not capture o then for all $j \in N_s^o$ define $\texttt{wt}(j) = \texttt{init-wt}(j)$. However if s captures o then for all $j \in N_s^o$ define $\texttt{wt}(j) = \alpha \cdot \texttt{init-wt}(j)$ where $\alpha < 1$ is such that $\texttt{wt}(N_s^o) = \texttt{wt}(N_O(o))/2$. Note that if $N_s^o = N_O(o)$ then $\alpha = 0$.

Next, we proceed to formally define fractional assignment $\pi_o : N_O(o) \times N_O(o) \rightarrow \Re^+$ for a facility $o \in O$ with the following properties.

separation $\pi_o(j, j') > 0$ only if j and j' are served by different facilities in S.

balance $\sum_{j' \in N_O(o)} \pi_o(j', j) = \sum_{j' \in N_O(o)} \pi_o(j, j') = \texttt{wt}(j)$ for all $j \in N_O(o)$.

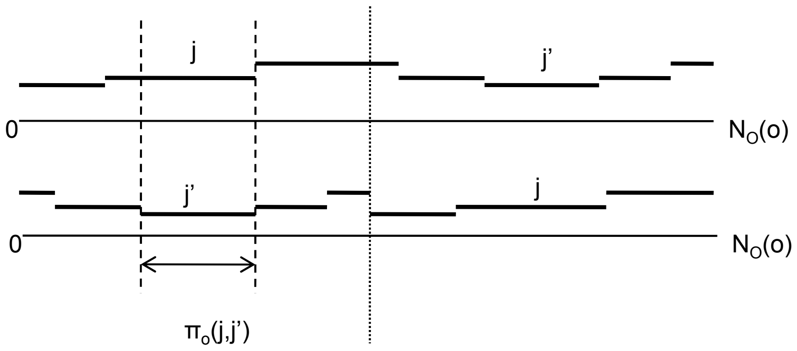

Figure 3.1: Defining π_o. The lower arrangement is obtained by splitting the top arrangement at the central dotted line and swapping the two halves.

The fractional assignment π_o can be obtained along the same lines as the mapping in [AGK+04]. Associate an interval of length $\texttt{wt}(j)$ for each $j \in N_O(o)$ and arrange these intervals on a line segment of length $\texttt{wt}(N_O(o))$ (see Figure 3.1). The intervals are ordered so that intervals corresponding to clients served by the same facility in S appear together. Consider another arrangement of intervals obtained from the first by splitting the line segment at the center and swapping the two halves. As a consequence, one interval

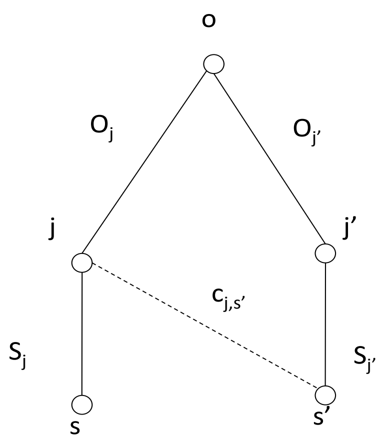

Figure 3.2: $\sigma(j) = s$ and $\sigma(j') = s'$. $\tau(j) = \tau(j') = o$

might be split and be non-contiguous in the second arrangement. Superimpose these two arrangements. $\pi_o(j, j')$ is now defined as the overlap between the interval corresponding to j in the first arrangement and the interval j' in the second. The second property of the weights ensures that there is no overlap between an interval in the first arrangement and the corresponding interval in the second arrangement. Further, it is easy to see that the mapping π_o as defined here satisfies the properties of separation and balance.

The fractional assignments π_o are extended to a fractional assignment (over all clients), $\pi : C \times C \to \Re^+$ in the obvious way — $\pi(j, j') = \pi_o(j, j')$ if $j, j' \in N_O(o)$ and is 0 otherwise.

Thus when a facility s is closed, a client $j \in N_S(s)$ is assigned to $\sigma(j')$ to an extent of $\pi(j, j')$. Let $\Delta(s)$ denote the increase in the service-cost of the clients served by s due to this reassignment. Recall that O_j and S_j denote the service costs of client j in solution O and S respectively. In the following lemma we bound the cost of this reassignment.

Lemma 3.3 $\sum_{s \in S} \Delta(s) \leq \sum_{j \in C} 2O_j \mathtt{wt}(j)$.

Proof Let $\pi(j, j') > 0$. When the facility $\sigma(j)$ is closed and $\pi(j, j')$ fraction of client

j assigned to facility $\sigma(j')$, the increase in service cost is $\pi(j,j')(c_{j\sigma(j')} - c_{j\sigma(j)})$. Since $c_{j\sigma(j')} \leq O_j + O_{j'} + S_{j'}$ (see Figure 3.2) we have

$$\begin{aligned}\sum_{s \in S} \Delta(s) &= \sum_{j,j' \in C} \pi(j,j')(c_{j\sigma(j')} - c_{j\sigma(j)}) \\ &\leq \sum_{j,j' \in C} \pi(j,j')(O_j + O_{j'} + S_{j'} - S_j) \\ &= 2\sum_{j \in C} O_j \texttt{wt}(j)\end{aligned}$$

where the last equality follows from the balance property. ■

Let $S' \subseteq S$ be the set of facilities such that for $s \in S'$ $\texttt{wt}(j) = 1\ \forall j \in N_S(s)$ i.e. $\texttt{res-wt}(j) = 0\ \forall j \in N_S(s)$. We perform a *delete* operation for each of the facilities in S'.

A facility $s \in S'$ can be deleted from S using operation `delete(s)` and its clients reassigned completely to the other facilities in S. Since S is locally optimal, therefore cost of this operation is non-negative, i.e.

$$-f_s - c_s(S) + c_s(S - \{s\}) \geq 0$$

where the reassignment of clients is done in best possible way. Cost of reassigning clients of s using π assignment, when s is deleted, will only be more or equal. Therefore

$$\Delta(s) \geq -c_s(S) + c_s(S - \{s\})$$

and hence

$$-f_s + \Delta(s) \geq 0$$

We write such an inequality for each $s \in S'$ and add all of them to get

$$\sum_{s \in S'} (-f_s + \Delta(s)) \geq 0. \tag{3.3}$$

For $s \in S - S'$, facilities for which $\texttt{res-wt}(j) > 0$ for some $j \in N_S(s)$, *delete* operation is not sufficient as $\texttt{res-wt}(j)$ extent of j needs to be assigned to a facility not in S. Thus for $s \in S - S'$, we need to perform a *swap* operation by swapping s with some facility $o \in O$. $\texttt{res-wt}(j)$ extent of a client j will be assigned to o and $\texttt{wt}(j)$ extent of client j will be assigned to facilities in $S \setminus s$, as is determined by the π assignment. The total extent to which clients are assigned to o in this operation equals $\texttt{res-wt}(N_S(s))$ which is at most U.

Let us consider swapping of facilities s, o where $s \in S - S'$ and $o \in O$. The service cost of a client j, which is assigned to o instead of s would increase by $c_{jo} - c_{js}$. Since $c_{jo} - c_{js} \leq c_{so}$, the total increase in service cost of all clients in $N_S(s)$ which are assigned (partly) to o is at most $c_{so}\texttt{res-wt}(N_S(s))$.

Since S is locally optimal we have

$$f_o - f_s + c_s(S - \{s\} \cup \{o\}) - c_s(S) \geq 0.$$

Let $\langle s, o\rangle$ denote swapping of facilities s, o together with reassignment of clients served by s to facilities in $S - \{s\} \cup \{o\}$ as follows: $\texttt{wt}(j)$ extent of each client $j \in N_S(s)$ is reassigned to facilities in S using π assignment and $\texttt{res-wt}(j)$ extent of j is assigned to o. Since in the actual operation `swap(s,o)` the reassignment of clients is done in best possible way, cost of reassignment of clients in $\langle s, o\rangle$ will only be more or equal. Therefore,

$$c_{so}\texttt{res-wt}(N_S(s)) + \Delta(s) \geq c_s(S - \{s\} \cup \{o\}) - c_s(S).$$

and therefore

$$f_o - f_s + c_{so}\texttt{res-wt}(N_S(s)) + \Delta(s) \geq 0. \tag{3.4}$$

The above inequalities are written for every pair $(s, o), s \in S - S', o \in O$. We take a linear combination of these inequalities with the inequality corresponding to $\langle s, o\rangle$ having a weight $\lambda_{s,o}$ in the combination to get

$$\sum_{s\in S-S',o} \lambda_{s,o} f_o - \sum_{s\in S-S',o} \lambda_{s,o} f_s + \sum_{s\in S-S',o} \lambda_{s,o} c_{so}\texttt{res-wt}(N_S(s)) + \sum_{s\in S-S',o} \lambda_{s,o}\Delta(s) \geq 0. \tag{3.5}$$

We need to define $\lambda_{s,o}$ values carefully to ensure that the reassignment costs of $\texttt{res-wt}(N_S(s))$ clients is not very high. Define

$$\lambda_{s,o} = \frac{\texttt{res-wt}(N_s^o)}{\texttt{res-wt}(N_S(s))}$$

if $\texttt{res-wt}(N_S(s)) \neq 0$ and is 0 if $\texttt{res-wt}(N_S(s)) = 0$. Note that for all $s \in S - S'$, $\sum_o \lambda_{s,o} = 1$. We show that with these values of $\lambda_{s,o}$ third term of Inequality 3.5 is bounded by $\sum_{j\in C} \texttt{res-wt}(j)(O_j + S_j)$. This is proved in the following lemma.

Lemma 3.4 $\sum_{s,o} \lambda_{s,o} c_{so}\texttt{res-wt}(N_S(s)) \leq \sum_{j\in C} \texttt{res-wt}(j)(O_j + S_j)$

Proof The left hand side in the inequality is

$$\sum_{s,o} \lambda_{s,o} c_{so}\texttt{res-wt}(N_S(s)) = \sum_{s,o} c_{so}\texttt{res-wt}(N_s^o).$$

Next, recall that $c_{so} = \min_{j\in C}(c_{js} + c_{jo}) \leq \min_{j\in N_s^o}(O_j + S_j)$. Therefore

$$\begin{aligned} c_{so}\texttt{res-wt}(N_s^o) &= \sum_{j\in N_s^o} c_{so}\texttt{res-wt}(j) \\ &\leq \sum_{j\in N_s^o} \texttt{res-wt}(j)(O_j + S_j) \end{aligned}$$

Therefore

$$\begin{aligned} \sum_{s,o} c_{so}\texttt{res-wt}(N_s^o) &\leq \sum_{s,o}\sum_{j\in N_s^o} \texttt{res-wt}(j)(O_j + S_j) \\ &= \sum_{s\in S}\sum_{j\in N_S(s)} \texttt{res-wt}(j)(O_j + S_j) \\ &= \sum_{j\in C} \texttt{res-wt}(j)(O_j + S_j) \end{aligned}$$

■

We add the Inequality 3.3 to Inequality 3.5 and get

$$\begin{aligned}&\sum_{s\in S'}(-f_s+\Delta(s))+\sum_{s\in S-S',o}\lambda_{s,o}f_o-\sum_{s\in S-S',o}\lambda_{s,o}f_s\\+&\sum_{s\in S-S',o}\lambda_{s,o}c_{so}\texttt{res-wt}(N_S(s))+\sum_{s\in S-S',o}\lambda_{s,o}\Delta(s)\geq 0.\end{aligned}\tag{3.6}$$

As $\sum_o \lambda_{s,o}=1$ for all $s\in S-S'$, we have

$$\sum_{s\in S'}f_s+\sum_{s\in S-S',o}\lambda_{s,o}f_s=\sum_s f_s\tag{3.7}$$

and

$$\sum_{s\in S'}\Delta(s)+\sum_{s\in S-S',o}\lambda_{s,o}\Delta(s)=\sum_s\Delta(s)\leq\sum_{j\in C}2O_j\texttt{wt}(j)\tag{3.8}$$

Where the last inequality follows from Lemma 3.3.

Incorporating equations (3.7), (3.8) and Lemma 3.4 into inequality (3.6) we get

$$-\sum_s f_s+\sum_{s,o}\lambda_{s,o}f_o+\sum_{j\in C}\texttt{res-wt}(j)(O_j+S_j)+\sum_{j\in C}2O_j\texttt{wt}(j)\geq 0$$

or

$$\begin{aligned}\sum_s f_s &\leq \sum_{s,o}\lambda_{s,o}f_o+\sum_{j\in C}\texttt{res-wt}(j)(O_j+S_j)+\sum_{j\in C}2O_j\texttt{wt}(j)\\&=\sum_{s,o}\lambda_{s,o}f_o+2\sum_{j\in C}O_j+\sum_{j\in C}\texttt{res-wt}(j)(S_j-O_j)\end{aligned}\tag{3.9}$$

We will now bound $\sum_s \lambda_{s,o}$ the coefficient of f_o in the Inequality 3.9, which provides us with the number of times a facility of the optimum solution may be opened.

Lemma 3.5 *For all $o\in O$, $\sum_s\lambda_{s,o}\leq 2$.*

Proof We begin with the following observations.

1. For all s,o, $\lambda_{s,o}\leq 1$.

2. Let $I \subseteq S$ be the facilities s such that s does not capture any $o \in O$ and $|N_S(s)| \leq U/2$. Let $s \in I$, $o \in O$ then $\forall\ j \in N_s^o$, $\texttt{wt}(j) = \texttt{init-wt}(j)$ (because s does not capture o) and $\texttt{init-wt}(j) = 1$ (because $N_S(s) \leq U/2$). Thus $\forall\ j \in N_s^o$ $\texttt{wt}(j) = \texttt{init-wt}(j) = 1$ and therefore $\texttt{res-wt}(j) = 0$. This implies that $\texttt{res-wt}(N_s^o) = 0$ and hence $\lambda_{s,o} = 0$ for all $s \in I$.

Thus we only need to show that $\sum_{s \notin I} \lambda_{s,o} \leq 2$. We now consider two cases.

1. o is not captured by any $s \in S$.

   Let's partition $S - I$ further into two sets $I^{'}$ and $I^{''}$, where for $s \in I^{'}$ we have $N_S(s) \leq U/2$ and for $s \in I^{''}$ we have $N_S(s) > U/2$. Let $s \in I^{'}$. Since o is not captured by any facility in S therefore in particular it is not captured by s. Thus for all $j \in N_s^o$, $\texttt{wt}(j) = \texttt{init-wt}(j) = 1$ and so $\texttt{res-wt}(j) = 0$ by the same argument as for the case of $s \in I$. This implies that $\lambda_{s,o} = 0$ for all $s \in I^{'}$.

   Let s be a facility in $I^{''}$. Since s does not capture o, for $j \in N_s^o$, $\texttt{wt}(j) = \texttt{init-wt}(j)$. Thus

$$\texttt{res-wt}(j) = 1 - \texttt{wt}(j) = 1 - \texttt{init-wt}(j) = 2 - \frac{U}{|N_S(s)|}.$$

   However, for $j \in N_S(s)$ we have $\texttt{wt}(j) \leq \texttt{init-wt}(j)$. Thus

$$\texttt{res-wt}(j) = 1 - \texttt{wt}(j) \geq 1 - \texttt{init-wt}(j) = 2 - \frac{U}{|N_S(s)|}.$$

   Therefore

$$\lambda_{s,o} = \frac{\texttt{res-wt}(N_s^o)}{\texttt{res-wt}(N_S(s))} \leq \frac{|N_s^o|}{|N_S(s)|}$$

   Hence

$$\sum_s \lambda_{s,o} = \sum_{s \in I^{''}} \lambda_{s,o} \leq \sum_{s \in I^{''}} \frac{|N_s^o|}{|N_S(s)|} \leq \sum_{s \in I^{''}} \frac{|N_s^o|}{U/2} \leq \frac{|N_O(o)|}{U/2} \leq 2.$$

2. o is captured by $s' \in S$.

   This implies

$$\begin{aligned}
\texttt{init-wt}(N^o_{s'}) &\geq \sum_{s \neq s'} \texttt{init-wt}(N^o_s) \\
&\geq \sum_{s \notin I \cup \{s'\}} \texttt{init-wt}(N^o_s) \\
&= \sum_{s \notin I \cup \{s'\}} |N^o_s| \frac{U - |N_S(s)|}{|N_S(s)|} \\
&= \sum_{s \notin I \cup \{s'\}} \left(U \frac{|N^o_s|}{|N_S(s)|} - |N^o_s| \right)
\end{aligned}$$

   Since $\texttt{init-wt}(N^o_{s'}) \leq |N^o_{s'}|$ rearranging we get,

$$\sum_{s \notin I \cup \{s'\}} \frac{|N^o_s|}{|N_S(s)|} \leq \sum_{s \notin I} \frac{|N^o_s|}{U} \leq 1.$$

   Now

$$\sum_{s \notin I \cup \{s'\}} \lambda_{s,o} \leq \sum_{s \notin I \cup \{s'\}} \frac{|N^o_s|}{|N_S(s)|} \leq 1$$

   and since $\lambda_{s',o} \leq 1$ we have

$$\sum_s \lambda_{s,o} = \sum_{s \notin I \cup \{s'\}} \lambda_{s,o} + \lambda_{s',o} \leq 2.$$

This completes the proof. ■

Incorporating Lemma 3.5 into Inequality (3.9) we get

$$\sum_s f_s \leq 2 \left(\sum_o f_o + \sum_{j \in C} O_j \right) + \sum_{j \in C} \texttt{res-wt}(j)(S_j - O_j)$$

Note that $\sum_{j \in C} \texttt{res-wt}(j)(S_j - O_j)$ is at most $\sum_{j \in C}(S_j - O_j)$ which in turn can be bounded by $\sum_o f_o$ by Lemma 3.1 and thus we have

$$\sum_s f_s \leq 3 \sum_o f_o + 2 \sum_{j \in C} O_j \tag{3.10}$$

This gives us the result of this section. In the next section we will prove the following bound on facility cost of S.

$$\sum_s f_s \leq 2\sum_o f_o + 2\sum_{j\in C} O_j$$

3.3 Our main result

The key to obtaining a sharper bound on the facility cost of our solution is the observation that in the swap $\langle s,o\rangle$ facility o gets only $\texttt{res-wt}(N_S(s))$ clients and can accommodate additional $U - \texttt{res-wt}(N_S(s))$ clients.

Claim 3.6 *A facility $o \in O$ is opened at most twice and gets at most U clients in all the swap operations considered.*

Proof Total clients that a facility $o \in O$ gets over all the operations is

$$\begin{aligned}\sum_s \lambda_{s,o}\texttt{res-wt}(N_S(s)) &= \sum_s \texttt{res-wt}(N_s^o)) &(3.11)\\ &= \texttt{res-wt}(N_O(o)) \leq U &(3.12)\end{aligned}$$

■

In swap $\langle s,o\rangle$, we assign a client $j \in N_S(s)$ to other facilities in S up to the extent of $\texttt{wt}(j)$ and to o up to the extent of $\texttt{res-wt}(j)$. We have considered the reassignment of clients served only by facility s in this *swap*. Note that

1. o is assigned a total of $\texttt{res-wt}(N_S(s))$ clients in this reassignment.
2. o can get an additional $U - \texttt{res-wt}(N_S(s))$ clients.
3. we can assign, to o, additional clients which are served by other facilities in S and by o in the optimal. $\texttt{wt}(j)$ extent for all the clients $j \in C$ is kept aside to participate in π assignment only, therefore only $\texttt{res-wt}(j)$ extent of j is available for any other type of reassignment of j. Thus additional clients from $N_O(o)$

are assigned to o up to a total extent of at most $\texttt{res-wt}(N_O(o))$ This means $\min\left(\texttt{res-wt}(N_O(o)), U - \texttt{res-wt}(N_S(s))\right)$ clients can be assigned to o due to this reassignment.

However, instead of assigning $\min(\texttt{res-wt}(N_O(o)), U - \texttt{res-wt}(N_S(s)))$ clients to o, residual weight of each client $j \in N_O(o)$ is assigned up to an extent of $\min\left(1, \frac{U - \texttt{res-wt}(N_S(s))}{\texttt{res-wt}(N_O(o))}\right)$ in this reassignment. Call this quantity $\beta_{s,o}$, i.e.

$$\beta_{s,o} = \min\left(1, \frac{U - \texttt{res-wt}(N_S(s))}{\texttt{res-wt}(N_O(o))}\right).$$

Let $\Delta'(s,o)$ denote the increase in service cost of the clients of $N_O(o)$ due to this reassignment. i.e.

$$\Delta'(s,o) = \beta_{s,o} \sum_{j \in N_O(o)} \texttt{res-wt}(j)(O_j - S_j). \tag{3.13}$$

We will now put the pieces together to prove the following theorem:

Theorem 3.7 *When $S \cap O = \phi$, the total cost of open facilities in any locally optimal solution is at most twice the cost of an optimum solution.*

Proof The Inequality (3.4) corresponding to the *swap* $\langle s,o \rangle$ would now get an additional term $\Delta'(s,o)$ on the left. Hence the term $\sum_{s,o} \lambda_{s,o}\Delta'(s,o)$ would appear on the left in Inequality (3.6) and on the right in Inequality (3.9).

Now

$$\begin{aligned}\sum_s \lambda_{s,o}\Delta'(s,o) &= \sum_s \left(\lambda_{s,o}\beta_{s,o} \sum_{j \in N_O(o)} \texttt{res-wt}(j)(O_j - S_j)\right) \\ &= \left(\sum_s \lambda_{s,o}\beta_{s,o}\right) \sum_{j \in N_O(o)} \texttt{res-wt}(j)(O_j - S_j).\end{aligned}$$

If $\sum_s \lambda_{s,o}\beta_{s,o} > 1$ then we reduce some $\beta_{s,o}$ so that the sum is exactly 1 (we will later show that this does not affect the analysis). On the other hand if $\sum_s \lambda_{s,o}\beta_{s,o} = 1 - \gamma_o$,

$\gamma_o > 0$, then we take the inequalities corresponding to the operation of adding the facility $o \in O$

$$f_o + \sum_{j \in N_O(o)} \texttt{res-wt}(j)(O_j - S_j) \geq 0 \tag{3.14}$$

and add these to Inequality (3.6) with a weight γ_o. Hence the total increase in the left hand side of Inequality (3.6) is

$$\begin{aligned}
&\sum_{s,o} \lambda_{s,o} \Delta'(s,o) + \sum_o \gamma_o \left(f_o + \sum_{j \in N_O(o)} \texttt{res-wt}(j)(O_j - S_j) \right) \\
&= \sum_o \sum_{j \in N_O(o)} (1 - \gamma_o)\texttt{res-wt}(j)(O_j - S_j) \\
&\quad + \sum_o \gamma_o f_o + \sum_o \sum_{j \in N_O(o)} \gamma_o \texttt{res-wt}(j)(O_j - S_j) \\
&= \sum_o \sum_{j \in N_O(o)} \texttt{res-wt}(j)(O_j - S_j) + \sum_o \gamma_o f_o \\
&= \sum_{j \in C} \texttt{res-wt}(j)(O_j - S_j) + \sum_o \gamma_o f_o
\end{aligned}$$

and so Inequality (3.9) now becomes

$$\begin{aligned}
\sum_s f_s &\leq \sum_o \sum_s \lambda_{s,o} f_o + 2 \sum_{j \in C} O_j + \sum_o \gamma_o f_o \\
&\quad + \sum_{j \in C} \texttt{res-wt}(j)(S_j - O_j) + \sum_{j \in C} \texttt{res-wt}(j)(O_j - S_j) \\
&= \sum_o \left(\gamma_o + \sum_s \lambda_{s,o} \right) f_o + 2 \sum_{j \in C} O_j \\
&= \sum_o \left(1 + \sum_s \lambda_{s,o}(1 - \beta_{s,o}) \right) f_o + 2 \sum_{j \in C} O_j \\
&\leq 2 \left(\sum_o f_o + \sum_{j \in C} O_j \right)
\end{aligned}$$

where the last inequality follows from the next lemma. This completes the proof of the theorem. ■

Lemma 3.8 $\sum_s \lambda_{s,o}(1-\beta_{s,o}) \leq 1$.

Proof When $\sum_s \lambda_{s,o}\beta_{s,o} > 1$ we reduced some $\beta_{s,o}$ to ensure that the sum is exactly 1. In this case

$$\sum_s \lambda_{s,o}(1-\beta_{s,o}) = \sum_s \lambda_{s,o} - 1 \leq 1,$$

since by Lemma 3.5, $\sum_s \lambda_{s,o} \leq 2$.

We now assume that no $\beta_{s,o}$ was reduced. Since

$$\texttt{res-wt}(N_O(o)) \leq |N_O(o)| \leq U$$

we have

$$\begin{aligned}
\beta_{s,o} &= \min\left(1, \frac{U - \texttt{res-wt}(N_S(s))}{\texttt{res-wt}(N_O(o))}\right) \\
&\geq \min\left(1, 1 - \frac{\texttt{res-wt}(N_S(s))}{\texttt{res-wt}(N_O(o))}\right) \\
&= 1 - \frac{\texttt{res-wt}(N_S(s))}{\texttt{res-wt}(N_O(o))}
\end{aligned}$$

Hence

$$\sum_s \lambda_{s,o}(1-\beta_{s,o}) \leq \sum_s \frac{\texttt{res-wt}(N_s^o)}{\texttt{res-wt}(N_O(o))} = 1.$$

■

Recall that to ensure that the local search procedure has a polynomial running time we modified the local search procedure so that a step was performed only when the cost of the solution decreases by at least $(\epsilon/4n)c(S)$. This modification implies that the right hand sides of inequalities (3.4), (3.3) and (3.14) which are all zero should instead be $(-\epsilon/4n)c(S)$. Note that for every choice of $s \in S$ and $o \in O$ we add a $\lambda_{s,o}$ multiple of Inequality (3.4) to obtain Inequality (3.6). Since $\sum_o \lambda_{s,o} = 1$, hence $\sum_{o,s} \lambda_{s,o} = |S| \leq n$. We also add Inequality (3.3) for every $s \in S$ to Inequality (3.6). Similarly, for every $o \in O$, a γ_o ($\gamma_o \leq 1$) multiple of Inequality (3.14) is added to Inequality (3.6).

Putting all these modifications together gives rise to an extra term of at most $(3\epsilon/4)c(S)$. This implies that the facility cost of solution S is at most 2c(O)$+(3\epsilon/4)c(S)$.

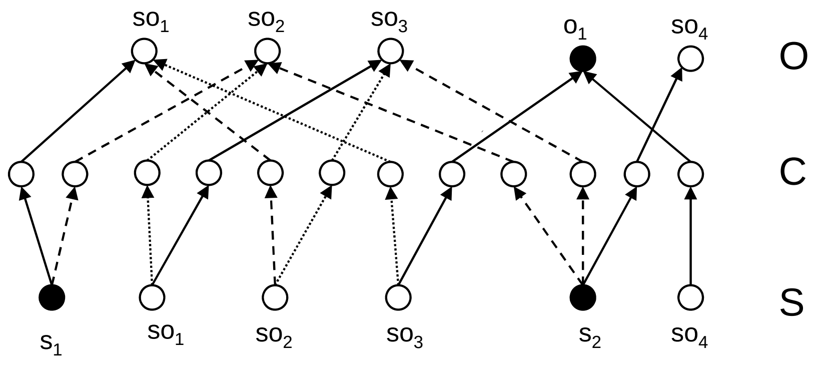

Figure 3.3: An instance showing the decomposition into cycles (dotted arcs), swap paths (solid arcs) and transfer paths (dashed arcs). The facilities labeled so_1, so_2, so_3 and so_4 are in $S \cap O$ and have been duplicated. The cycle is so_1, so_2, so_3, so_1. The transfer paths are (s_1, so_2, so_1), (s_2, so_2) and (s_2, so_3). The swap paths are s_1, so_1, so_3, o_1 and s_2, so_4, o_1.

Similarly, the service cost of solution S can now be bounded by $c(O)+(\epsilon/4)c(S)$. Adding these yields

$$(1-\epsilon)c(S) \leq 3\text{c}(O)$$

which implies that S is a $3/(1-\epsilon)$ approximation to the optimum solution.

3.4 When $S \cap O \neq \phi$

We now consider the case when $S \cap O \neq \phi$. We construct a bipartite graph, G, on the vertex set $C \cup F$ as in [CW05]. Every client $j \in C$ has an edge from the facility $\sigma(j) \in S$ and an edge to the facility $\tau(j) \in O$. Thus each client has one incoming and one outgoing edge. A facility $s \in S$ has $|N_S(s)|$ outgoing edges and a facility $o \in O$ has $|N_O(o)|$ incoming edges. Decompose the graph G into a set of maximal paths, $\mathcal{P}$, and cycles, $\mathcal{C}$, as is explained in Appendix A.1.

Note that all facilities on a cycle are from $S \cap O$. Consider a maximal path, $P \in \mathcal{P}$ which starts at a vertex $s \in S$ and ends at a vertex $o \in O$. Let head(P) denote the client served by s on this path and tail(P) be the client served by o on this path. Let $s_0, j_0, s_1, j_1, \ldots, s_k, j_k, o$ be the sequence of vertices on this path where $s = s_0$. Note that $\{s_1, s_2, \ldots, s_k\} \subseteq S \cap O$. A *shift* along this path is a reassignment of clients so that j_i which was earlier assigned to s_i is now assigned to s_{i+1} where $s_{k+1} = o$. As a consequence of this shift, facility s serves one client less while facility o serves one client more. Let shift(P) denote the increase in service cost due to a shift along the path P. Then

$$\text{shift}(P) = \sum_{j \in C \cap P} (O_j - S_j).$$

We can similarly define a shift along a cycle. The increase in service cost equals the sum of $O_j - S_j$ 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

$$\sum_{Q \in \mathcal{C}} \sum_{j \in Q} (O_j - S_j) = 0.$$

As we did for the case when $S \cap O = \phi$, for this case also we prove the bound on service cost and facility cost separately. Following is the proof of Lemma 3.1 which provides the bound on service cost of a locally optimal solution S when $S \cap O \neq \phi$.

Proof Consider the operation of adding a facility $o \in O$. We shift along all the paths which end at o. The increase in service cost due to these shifts equals the sum of $O_j - S_j$ for all clients j on these paths and this quantity is at least $-f_o$.

$$\sum_{P \in \mathcal{P}} \sum_{j \in P} (O_j - S_j) \geq -\sum_{o \in O} f_o.$$

Thus

$$\sum_{j \in C} (O_j - S_j) = \sum_{P \in \mathcal{P}} \sum_{j \in P} (O_j - S_j)(O_j - S_j) + \sum_{Q \in \mathcal{C}} \sum_{j \in Q} (O_j - S_j) \geq -\sum_{o \in O} f_o$$

which implies that the service cost of S is bounded by $\sum_{o \in O} f_o + \sum_{j \in C} O_j$. ■

To bound the cost of facilities in $S - O$ we only need the paths that start from a facility in $S - O$. Hence we throw away all cycles and all paths that start at a facility in $S \cap O$; this is done by removing all clients on these cycles and paths. Let $\mathcal{P}$ denote the remaining paths and C the remaining clients. Every client in C either belongs to a path which ends in $S \cap O$ (*transfer* path) or to a path which ends in $O - S$ (*swap* path). Let $\mathcal{T}$ denote the set of transfer paths and $\mathcal{S}$ the set of swap paths (see Figure 3.3).

We now define N_s^o to be the set of paths that start at $s \in S$ and end at $o \in O$. Further, define

$$N_S(s) = \cup_{o \in O-S} N_s^o.$$

Note that we do not include the transfer paths in the above definition. Similarly for all $o \in O$ define

$$N_O(o) = \cup_{s \in S-O} N_s^o.$$

Just as we defined the $\texttt{init-wt}()$, $\texttt{wt}()$ and $\texttt{res-wt}()$ of a client, we can define the $\texttt{init-wt}()$, $\texttt{wt}()$ and $\texttt{res-wt}()$ of a swap path. Thus for a path P which starts from $s \in S - O$ we define

$$\texttt{init-wt}(P) = \min\left(1, \frac{U - |N_S(s)|}{|N_S(s)|}\right).$$

The notion of *capture* remains the same and we reduce the initial weights on the paths to obtain their weights. Thus $\texttt{wt}(P) \leq \texttt{init-wt}(P)$ and for every $s \in S$ and $o \in O$, $\texttt{wt}(N_s^o) \leq \texttt{wt}(N_O(o))/2$. For every $o \in O - S$ we define a fractional mapping $\pi_o : N_O(o) \times N_O(o) \to \Re^+$ such that

separation $\pi_o(P, P') > 0$ only if P and P' start at different facilities in $S - O$.

balance $\sum_{P' \in N_O(o)} \pi_o(P', P) = \sum_{P' \in N_O(o)} \pi_o(P, P') = \texttt{wt}(P)$ for all $P \in N_O(o)$.

This fractional mapping can be constructed in the same way as done earlier. The way we use this fractional mapping, π, will differ slightly. When facility s is closed, we will use π to partly reassign the clients served by s in the solution S to other facilities in S. If P is a path starting from s and $\pi(P, P') > 0$, then we shift along P and the client tail(P) is assigned to s', where s' is the facility from which P' starts. This whole operation is done

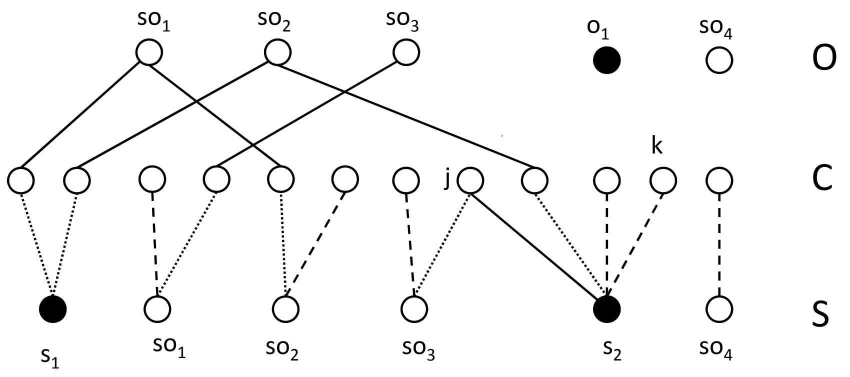

Figure 3.4: The figure shows the assignment of clients to facilities after facility s_1 has been closed ($U = 3$) in the instance given in Figure 3.3. The dotted lines show the earlier assignment while the solid lines show the new assignment. Those assignments which do not change are shown with dashed lines. Note that o serves two clients j, k which are the heads of swap paths s_1, so_1, so_3, o_1 and s_2, so_4, o_1. These two clients are mapped to each other in the mapping π. When facility s_1 is closed we perform a shift along transfer path s_1, so_2, so_1. We also perform a shift along the swap path s_1, so_1, so_3, o_1 with the last client on this path, j now assigned to s_2. Since, s_2 was already serving 3 clients, we move one of its clients along a transfer path (s_2, so_2).

to an extent of $\pi(P, P')$. The cost of assigning client tail(P) to s' can be bounded by the sum of the service cost of tail(P) in solution O and the *length* of the path P' where

$$\text{length}(P') = \sum_{j \in C \cap P'} (O_j + S_j).$$

Let $\Delta(s)$ denote the total increase in service cost due to the reassignment of clients on all swap paths starting from s. Then

$$\begin{aligned}\sum_s \Delta(s) &\leq \sum_s \sum_{P \in N_S(s)} \sum_{P' \in \mathcal{P}} \pi(P, P')(\text{shift}(P) + \text{length}(P')) \\ &= \sum_{P \in \mathcal{S}} \mathtt{wt}(P)(\text{shift}(P) + \text{length}(P)) \qquad (3.15)\end{aligned}$$

As a result of the above reassignment a facility $s' \in S - O, s' \neq s$ might get additional clients whose "number" is at most $\mathtt{wt}(N_S(s'))$. Note that this is less than $\mathtt{init\text{-}wt}(N_S(s'))$ which is at most $U - |N_S(s')|$. The number of clients s' was serving equals $|N_S(s')| + |T(s')|$ where $T(s')$ is the set of transfer paths starting from s'. This implies that the total number of clients s' would have after the reassignment could exceed U. To prevent this violation of our capacity constraint, we also perform a shift along these transfer paths (Figure 3.4).

Suppose s' gets an additional client, say tail(P), to an extent of $\pi(P, P')$, where $P' \in N_S(s')$. Then for all paths $q \in T(s')$, we would shift along path q to an extent $\pi(P, P')/\texttt{wt}(N_S(s'))$. This ensures that

1. The total extent to which we will shift along a path $q \in T(s')$ is given by

$$\sum_{p} \sum_{P' \in N_S(s')} \frac{\pi(P, P')}{\texttt{wt}(N_S(s'))}$$

which is at most 1. This in turn implies that we do not violate the capacity of any facility in $S \cap O$. This is because, if there are t transfer paths ending at a facility $o \in S \cap O$ then o serves t more clients in solution O than in S. Hence, in solution S, o serves at most $U - t$ clients. Since the total extent to which we could shift along a transfer path ending at o is 1, even if we were to perform shift along all transfer paths ending in o, the capacity of o in our solution S would not be violated.

2. The capacity constraint of no facility in $S - O$ is violated. If a facility $s' \in S - O$ gets an additional x clients as a result of reassigning the clients of some facility $s \neq s'$, then it would also lose some clients, say y, due to the shifts along the transfer paths. Now

$$y = |T(s')| \sum_{p} \sum_{P' \in N_S(s')} \frac{\pi(P, P')}{\texttt{wt}(N_S(s'))} = \frac{x|T(s')|}{\texttt{wt}(N_S(s'))}$$

and hence the additional number of clients served by s' is $x - y$ which equals

$$x\left(1 - \frac{|T(s')|}{\texttt{wt}(N_S(s'))}\right) \leq \texttt{wt}(N_S(s')) - |T(s')| \leq U - |N_S(s')| - |T(s')|,$$

where the first inequality follows from the fact that $x \leq \texttt{wt}(N_S(s'))$ and the second inequality by definition of wt. Since, initially, s' was serving $|N_S(s')| + |T(s')|$

clients, the total number of clients that s' is serving after the reassignment is at most U.

Note that when we close facility s we shift on transfer paths starting from s as well as on some transfer paths starting at $s' \neq s$. Let $\Gamma(s)$ denote the total increase in service cost due to shifts on all transfer paths when facility s is closed. Consider a transfer path, q, starting from s. We would shift once along path q when we close facility s. We would also be shifting along q to an extent of $\sum_P \sum_{P' \in N_S(s)} \pi(P, P')/\texttt{wt}(N_S(s))$ (which is at most 1) when facilities other than s are closed. Hence,

$$\sum_s \Gamma(s) \leq 2 \sum_{q \in \mathcal{T}} \text{shift}(q) \tag{3.16}$$

For a swap path P, define $\texttt{res-wt}(P) = 1 - \texttt{wt}(P)$. Let j be head(P) and define $\texttt{wt}(j) = \texttt{wt}(P)$ and $\texttt{res-wt}(j) = \texttt{res-wt}(P)$. Let P start from facility s. When s is closed, client j is assigned to an extent $\texttt{wt}(j)$ to other facilities in S. We will be assigning the remaining part of j to a facility $o \in O - S$ that will be opened when s is closed. Hence the total number of clients that will be assigned to o is $\texttt{res-wt}(N_S(s))$ which is less than U. The increase in service cost due to this reassignment is at most $c_{so}\texttt{res-wt}(N_S(s))$. As done earlier, the inequality corresponding to the *swap* $\langle s, o\rangle$ is counted to an extent $\lambda_{s,o}$ in the linear combination. Since $c_{so} \leq$ length(P) for all $P \in N_s^o$, we have the following equivalent of Lemma 3.4

$$\sum_{s,o} \lambda_{s,o} c_{so} \texttt{res-wt}(N_S(s)) \leq \sum_{P \in \mathcal{S}} \texttt{res-wt}(P)\text{length}(P). \tag{3.17}$$

The remaining available capacity of o is utilized by assigning each client $j \in N_O(o)$ to an extent $\beta_{s,o}\texttt{res-wt}(j)$, where $\beta_{s,o}$ is defined as before. This assignment is actually done by shifting along each path, $P \in N_O(o)$, by an extent $\beta_{s,o}\texttt{res-wt}(P)$. Let $\Delta'(s, o)$ be the increase in cost due to this reassignment of clients in $N_O(o)$. Then

$$\Delta'(s, o) \leq \beta_{s,o} \sum_{P \in N_O(o)} \texttt{res-wt}(P)\text{shift}(P).$$

This operation is a part of $\langle s, o\rangle$ and hence is counted to an extent $\lambda_{s,o}$ in the linear combination. Therefore the contribution of this term is

$$\sum_{s,o} \lambda_{s,o}\Delta'(s,o) \leq \sum_{o}\left(\sum_{s}\lambda_{s,o}\beta_{s,o}\right)\sum_{P\in N_O(o)} \texttt{res-wt}(P)\text{shift}(P). \tag{3.18}$$

Adding facility $o \in O - S$ and shifting each path $P \in N_O(o)$ by an extent $\texttt{res-wt}(P)$ gives us the following inequality.

$$f_o + \sum_{P\in N_O(o)} \texttt{res-wt}(P)\text{shift}(P) \geq 0 \tag{3.19}$$

As before, if $\sum_s \lambda_{s,o}\beta_{s,o} > 1$ then we reduce some $\beta_{s,o}$ so that the sum is exactly 1. Else, we add a $1 - \sum_s \lambda_{s,o}\beta_{s,o}$ multiple of Inequality (3.19) to Inequality (3.18) to get

$$\sum_{s,o} \lambda_{s,o}\Delta'(s,o) \leq \sum_{o}\gamma_o f_o + \sum_{o}\sum_{P\in N_O(o)} \texttt{res-wt}(P)\text{shift}(P). \tag{3.20}$$

where $\gamma_o = \max\{0, 1 - \sum_s \lambda_{s,o}\beta_{s,o}\}$.

The inequality corresponding to the *swap* $\langle s, o\rangle$ is

$$f_o - f_s + c_{so}\texttt{res-wt}(N_S(s)) + \Delta(s) + \Gamma(s) + \Delta'(s,o) \quad\geq\quad 0,$$

and taking a linear combination of the inequalities corresponding to the swaps $\langle s, o\rangle$, $s \in S - O$, $o \in O - S$ with weights $\lambda_{s,o}$ yields

$$\begin{aligned}\sum_{s,o}\lambda_{s,o}f_o - \sum_{s,o}\lambda_{s,o}f_s + \sum_{s,o}\lambda_{s,o}c_{so}\texttt{res-wt}(N_S(s))& \\ + \sum_{s,o}\lambda_{s,o}(\Delta(s)+\Gamma(s)) + \sum_{s,o}\lambda_{s,o}\Delta'(s,o) \quad&\geq\quad 0.\end{aligned}$$

Since, for all s, $\sum_o \lambda_{s,o} = 1$, we get

$$\begin{aligned}\sum_{s\in S-O} f_s \quad\leq\quad& \sum_{s,o}\lambda_{s,o}f_o + \sum_{s,o}\lambda_{s,o}\Delta'(s,o) \\ &+ \sum_{s,o}\lambda_{s,o}c_{so}\texttt{res-wt}(N_S(s)) + \sum_{s}(\Delta(s)+\Gamma(s))\end{aligned} \tag{3.21}$$

Putting the bounds from inequalities (3.15),(3.16),(3.17) and (3.20) into the right hand side of Inequality (3.21), yields

$$\begin{aligned}
\sum_{s\in S-O} f_s &\le \sum_{o\in O-S}\left(\gamma_o+\sum_s \lambda_{s,o}\right) f_o + \sum_{P\in\mathcal{S}} \texttt{res-wt}(P)\text{shift}(P)\\
&\quad + \sum_{P\in\mathcal{S}} \texttt{res-wt}(P)\text{length}(P) + \sum_{P\in\mathcal{S}} \texttt{wt}(P)(\text{shift}(P)+\text{length}(P))\\
&\quad +2\sum_{P\in\mathcal{T}} \text{shift}(P)\\
&\le 2\sum_{o\in O-S} f_o + \sum_{P\in\mathcal{S}} \texttt{res-wt}(P)(\text{shift}(P)+\text{length}(P))\\
&\quad + \sum_{P\in\mathcal{S}} \texttt{wt}(P)(\text{shift}(P)+\text{length}(P)) + 2\sum_{P\in\mathcal{T}} \text{shift}(P)\\
&= 2\sum_{o\in O-S} f_o + \sum_{P\in\mathcal{S}} (\text{shift}(P)+\text{length}(P)) + 2\sum_{P\in\mathcal{T}} \text{shift}(P)\\
&\le 2\left(\sum_{o\in O-S} f_o + \sum_{j\in C} O_j\right)
\end{aligned}$$

where the first inequality follows from Lemma 3.8 and Lemma 3.5. This implies that

$$\sum_{s\in S} f_s \le 2\left(\sum_{o\in O-S} f_o + \sum_{j\in C} O_j\right) + \sum_{o\in S\cap O} f_o \le 2\left(\sum_{o\in O} f_o + \sum_{j\in C} O_j\right)$$

which is the statement of Theorem 3.7 when $S\cap O \neq \phi$

3.5 A tight example

Our tight example consists of r facilities in the optimum solution O, r facilities in the locally optimum solution S and rU clients. The facilities are $F = O\cup S$. Since no facility can serve more than U clients, each facility in S and O serves exactly U clients. Our instance has the property that a facility in O and a facility in S share at most one client.

We can view our instance as a set-system — the set of facilities O is the ground set and for every facility $s \in S$ we have a subset X_s of this ground set. $o \in X_s$ iff there is a client which is served by s in the solution S and by o in the solution O. This immediately implies that each element of the ground set is in exactly U sets and that each set is of size exactly U. A third property we require is that two sets have at most one element in common.

We now show how to construct a set system with the properties mentioned above. With every $o \in O$ we associate a distinct point $x^o = (x^o_1, x^o_2, \ldots x^o_U)$ in a U-dimensional space where for all i, $x^o_i \in \{1, 2, 3, \ldots, U\}$. For every choice of coordinate i, $1 \leq i \leq U$ we form U^{U-1} sets, each of which contains all points differing only in coordinate i. Thus the total number of sets we form is $r = U^U$ which is the same as the number of points. Each set can be viewed as a line in U-dimensional space. To see that this set system satisfies all the properties note that each line contains U points and each point is on exactly U lines. It also follows from our construction that two distinct lines meet in at most one point.

We now define the facility and the service costs. For a facility $o \in O$, $f_o = 2U$ while for facility $s \in S$, $f_s = 6U - 6$. For a client $j \in N^o_s$, we have $c_{sj} = 3$ and $c_{oj} = 1$. All other service costs are given by the metric property.

Lemma 3.9 *For a client j and facility $s \in S$, the three smallest values that c_{sj} can have are 3,5 and 11. Similarly, the three smallest values that $c_{oj}, o \in O$ can have are 1,7 and 9.*

Proof A client j can be served at a cost 1 by exactly one facility in O and at a cost 3 by exactly one facility in S. The distance between a facility in O and a facility in S is at least 4. ■

Since the service cost of each client in O is 1 and the facility cost of each facility in O is $2U$, we have $c(O) = 3U^{U+1}$. Similarly, $c(S) = (3 - 2/U)3U^{U+1}$ and hence $c(S) = (3 - 2/U)c(O)$. We now need to prove that S is indeed a locally optimum solution with respect to the local search operations of *add*, *delete* and *swap*.

Adding a facility $o \in O$ to the solution S, would incur an opening cost of $2U$. The optimum assignment would reassign only the clients in $N_o(O)$, and all these are assigned to o. The reduction in the service cost due to this is exactly $2U$ which is offset by the increase in the facility cost. Hence the cost of the solution does not improve.

If we delete a facility in the solution S, the solution is no longer feasible since the total capacity of the facilities is now $U^{U+1} - U$ and the number of clients is U^{U+1}.

Now, consider swapping a facility $s \in S$ with a facility $o \in O$. The net decrease in the facility cost is $4U - 6$. To bound the increase in service costs we consider a bipartite graph with the facilities $S \cup \{o\}$ and the clients C forming the two sides of the bipartition. Let E be the edges corresponding to the original assignment of clients to facilities and E' be the edges of the new assignment. The symmetric difference of E and E' is a collection of U edge-disjoint paths between s and o. Let $\mathcal{P}$ be this collection and P be one of these paths. We define the *net-cost* of P as the difference between the costs of the edges of E' and E in P.

Lemma 3.10 *The two paths in $\mathcal{P}$ with the smallest net-cost have a total net-cost of at least 2. All other paths in $\mathcal{P}$ have net-cost of at least 4.*

Note that the increase in service cost as a result of the swap $\langle s, o \rangle$ equals the total net-cost of the paths in $\mathcal{P}$. The lemma implies that the net-cost of the paths is at least $4(U-2)+2$ which is exactly equal to the decrease in facility cost. Hence, swapping any pair of facilities $s \in S$ and $o \in O$ does not improve the solution.

Proof The edges of E on path P have cost 3. From Lemma 3.9 it follows that the edge on path P incident to o has cost 1,7 or higher while the remaining edges of $P \cap E'$ have cost 5,9 or higher. Edges on P alternate between sets E and E'. Hence starting from s we can pair consecutive edges of P with the first edge of each pair from E and the other from E'. Note that every pair, except the last, contributes at least 2 to the net-cost of P while the last pair contributes at least -2.

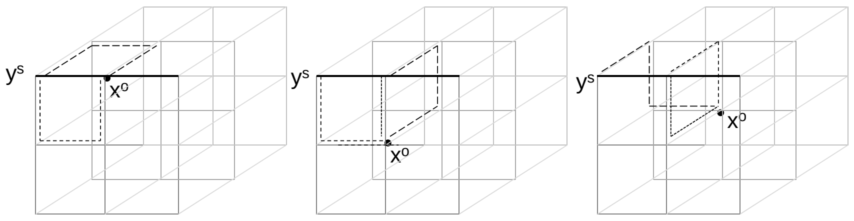

Figure 3.5: $U = 3$.The figure shows the three cases corresponding to P_0 having length 1, 2 and 3. P_0 is the dotted path, P_1 is the path with small dashes while P_2 is the path with longer dashes.

1. If the edge of P incident to o has cost 7 or higher than the last pair contributes at least 4 to the net-cost of P and hence the net-cost of P is at least 4.

2. If any edge of $P \cap E'$ has cost 9 or higher than the corresponding pair contributes at least 6 to the net-cost. Since the last pair contributes at least -2, the net-cost of P is at least 4.

As a consequence of the above we can assume that all edges of $P \cap E'$ have cost 5, except the edge incident to o which has cost 1. This implies that the path P corresponds to a path $S_1, S_2, \ldots S_k$ in our set-system where consecutive sets have a common element and S_1 corresponds to facility s while S_k contains the element corresponding to o. Alternatively, in our construction of the set-system, the path P corresponds to a sequence of lines where consecutive lines in the sequence intersect and the first line is the one corresponding to facility s while the last line contains the point corresponding to o. Note that the paths in $\mathcal{P}$ are edge-disjoint but not vertex-disjoint. Hence the sequence of lines corresponding to two paths in $\mathcal{P}$ may have common lines but no pair of consecutive lines can be common in the two sequences. Further, the sequences should end in different lines.

A path P containing k sets, corresponds to a sequence of lines containing k lines and has a net-cost of $2(k-2)$ and we say that its length is k. Hence paths with 4 or more lines have a net-cost at least 4 and so to prove the lemma we need to argue that there are at most 2 paths in $\mathcal{P}$ having less than 4 lines. Let P_0, P_1 be the two paths with the smallest lengths with P_0 being the smallest.

1. If P_0 has length 1 then the line corresponding to s, say y^s, contains the point corresponding to o, say x^o. From our construction it follows that any other sequence of lines which starts with y^s and ends with a line containing x^o which is different from y^s must contain at least 4 lines (including line y^s). Hence path P_1 has a net-cost at least 4. Thus the total net-cost of paths P_0 and P_1 is at least 2 (see Figure 3.5).

2. If P_0 has length 2 then the line y^s and the point x^o, have identical values for $U - 2$ coordinates. Let y^a be the line in P_0 containing x^o. Once again, from our construction it follows that any other sequence of lines which starts with y^s and ends with a line containing x^o which is different from y^a must contain at least 3 lines. Hence P_1 has length at least 3 and so the total net-cost of paths P_0 and P_1 is at least 2. Further the other paths of $\mathcal{P}$ would end with lines which are in dimensions other than the last lines of P_0, P_1 and so the length of these paths is at least 4 (see Figure 3.5).

3. If P_0 has length 3 then y^s and x^o have identical values for $U - 3$ coordinates. In this case, the net-cost of paths P_0, P_1 is at least 2 and the other paths of $\mathcal{P}$ have at least 4 lines (see Figure 3.5).

■

Chapter 4

A (5+ϵ)-Approximation for Capacitated Facility Location

In this chapter we apply our ideas developed in Chapter 3 to the setting of non-uniform capacities. The first local search algorithm for non-uniform capacities problem was due to Pal, Tardos and Wexler [PTW01] who gave a (8.53+ϵ)-approximation for this problem. Mahdian and Pal [MP03] reduced this to a (7.88+ϵ)-approximation and simultaneously extended it to the more general *universal facility location* problem. The approximation guarantee for the capacitated facility location problem was further reduced by Zhang, Chen and Ye [ZCY05] to (5.83+ϵ).

Zhang *et al.* use three operations *add*, *open* and *close* of Pal *et al.* and introduce a new operation called *multi*. We modify the *open*, *close* and *multi* operations so that apart from the clients served by facilities being closed, some more clients, served by other facilities in the current solution, are assigned to the facilities being opened to utilize them more efficiently. We show that with these modifications we are able to achieve a factor of (5+ϵ).

The remainder of this chapter is organized as follows. In Section 4.1 we present a brief overview of the algorithm and analysis of Zhang *et al.*. In Section 4.2 we first show the underutilization of the opened facilities with *open*, *close* and *multi* operations and then suggest the improved operations *mopen*, *mclose* and *mmulti*. We then bound the facility cost of our solution obtained with these new operations in the same section. We also present a tight example in Section 4.3 where we give a locally optimal solution S whose cost is 5 times the cost of the optimum solution. Throughout this chapter we have made an assumption that $S \cap O = \phi$. This assumption helps us in building new ideas for the problem and putting them in a simpler way. In the next chapter we will get rid of this assumption when we discuss a more general version of the problem.

4.1 Preliminaries and Previous Work

As is true for the uniform capacity case, for a set of facilities $S \subseteq F$, the optimal assignment of clients to the facilities in S can be done by solving a mincost flow problem. Therefore we only need to determine a good subset $S \subseteq F$ of facilities. We reuse the notations of chapter 3 where S denotes both the solution and the set of open facilities in the solution; the cost of the solution S is denoted by $c(S) = c_f(S) + c_s(S)$, where $c_f(S)$ is the facility cost and $c_s(S)$ is the service cost of the solution S.

Pal, Tardos and Wexler [PTW01] suggested a local search algorithm to find a good approximate solution for the problem. Starting with a feasible solution S the following operations are performed to improve the solution if possible.

- **add(s):** $S \leftarrow S \cup \{s\}$, $s \notin S$. In this operation a facility s which is not in the current solution S is added if its addition improves the cost of the solution.
- **open(t, T):** $S \leftarrow (S \cup \{t\}) \setminus T$, $t \notin S, T \subseteq S$. In this operation a facility $t \notin S$ is opened and a subset of facilities $T \subseteq S$ is closed. Since the possibilities for the set T are exponentially large, to make the procedure polynomial, instead of computing the exact cost of the new solution only an estimated cost is computed which overestimates the exact cost. The operation is then performed if the estimated

cost is less than the cost of the solution S. In computing the estimated cost it is assumed that for any $s \in T$, all clients served by s are assigned to t and that each such reassignment costs c_{ts} where $c_{ts} = \min_{j \in C} c_{tj} + c_{js}$. Working with the estimated cost allows to find in polynomial time, for a given t, the set T for which the estimated cost of $(S \setminus T) \cup \{t\}$ is minimum [PTW01] by solving a knapsack problem.

- **close(s, T):** $S \leftarrow (S \cup T) \setminus \{s\}, s \in S, T \subseteq F \setminus S$. In this operation a facility $s \in S$ is closed and a subset of facilities T (disjoint from S) is opened. Once again, an estimated cost of the operation is computed, in which it is assumed that a client which was assigned to s in the solution S will now be assigned to some $t \in T$ and that this reassignment costs c_{st}. As before, working with the estimated cost allows to find in polynomial time, for a given s, the set T for which the estimated cost of $(S \cup T) \setminus \{s\}$ is minimum [PTW01] by computing a covering knapsack problem.

Zhang *et al.* added the following operation to the above set of operations:

- **multi(r, R, t, T):** $S \leftarrow (S \cup R \cup \{t\}) \setminus (\{r\} \cup T), r \in S \setminus T, T \subseteq S \setminus \{r\}, R \subseteq F \setminus (S \cup \{t\}), t \notin S \cup R$. This operation is essentially a combination of a `close`(r, R) and `open`(t, T) with the added provision that clients served by r may be assigned to facility t. For a choice of r and t the operation can be implemented by guessing the number of clients serviced by r that will be assigned to t and then determining the sets R and T which minimize the total expected cost.

S is locally optimal if none of the four operations improve the cost of the solution and at this point the algorithm stops. Polynomial running time can be ensured at the expense of an additive ϵ in the approximation factor by doing a local search operation only if the cost reduces by more than a $1 - \epsilon/5n$ factor, for $\epsilon > 0$.

Let $S \subseteq F$ be a locally optimal solution and $O \subseteq F$ be an optimum solution. As in the case of uniform capacities, the *add* operation allows us to bound the *service cost* of the solution S, i.e.

Lemma 4.1 ([MP03, ZCY05]) $c_s(S) \leq c_s(O) + c_f(O) = c(O)$.

To bound the *facility cost* of S, a suitable set of inequalities is formulated which arise from the fact that S is locally optimal. To identify these inequalities, Pal *et al.* build an exchange graph G whose vertices are the set of facilities in S and the facilities in the optimum solution O. G has an edge $(s, o), s \in S, o \in O$ if there are clients served by s in the solution S and by o in O and the value of this edge, $y(s, o)$, is the number of such clients. Note that $y(s, o) = |N_s^o|$, where N_s^o is the set of clients served by $s \in S$ and by $o \in O$ as defined in Section 3.2. As defined in Section 3.2 $N_S(s)$ ($N_O(o)$) denote the set of clients served by s in S (o in O). Also $N_O(o) = \cup_{s\in S} N_s^o$ and $N_S(s) = \cup_{o\in O} N_s^o$. Also recall that O_j denotes the service cost of client j in the solution O and S_j denotes the service cost of client j in the solution S. The cost of the edge (s, o) is c_{so}. Recall that $c_{so} = \min_{j\in C}(c_{js} + c_{jo}) \leq \min_{j\in N_s^o)}(O_j + S_j)$. Note that

1. $\sum_{s\in S, o\in O} c_{so} y(s, o) \leq c_s(S) + c_s(O)$.

   Proof Since $y(s, o) = |N_s^o|$

$$\begin{aligned}
\sum_{o\in O}\sum_{s\in S} c_{so}\,|N_s^o| &\leq \sum_{o\in O}\sum_{s\in S}\sum_{j\in N_s^o}(O_j + S_j) \\
&= \sum_{s\in S}\sum_{j\in N_S(s)}(O_j + S_j) \\
&= \sum_{j\in C}(O_j + S_j) \\
&= c_s(S) + c_s(O)
\end{aligned}$$

■

2. G is a bipartite graph with S and O defining the sets of the partition.

3. $\forall s \in S, \sum_{o\in O} y(s, o) = |N_S(s)|$ and $\forall o \in O, \sum_{s\in S} y(s, o) = |N_O(o)|$.

The graph G may contain cycles, therefore it is now modified to make it acyclic. Consider a cycle in G and let C be the edges on the cycle. Partition the edges of C into sets C_1, C_2 such that the edges of C_1 (and C_2) are alternate edges on the cycle. Let γ be the minimum value of an edge in C. Consider two operations: one in which we increase the value of edges in C_1 and decrease the value of edges in C_2 by an amount γ and the other in which we do the inverse *i.e.,*decrease the value of the edges in C_1 and increase the value of the edges in C_2. Note that 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 value of 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 modified values of the edges continue to satisfy the three properties listed above (see Appendix A.2 for proof). However, now it is no more the case that value of an edge (s,o) is the number of clients which are served by $s \in S$ and $o \in O$.

Consider a subtree T in G of height 2 rooted at $t \in O$. Figure 4.1 shows one such subtree. Recall that the aim is to formulate a set of inequalities that will let us bound the total facility cost $c_f(S)$ of the solution S. Each inequality is obtained by considering a potential local step and using the fact that S is a locally optimal solution. The inequalities are written such that

1. each facility in S is closed exactly once.

2. each facility in O is opened at most thrice.

3. the total cost of reassigning clients is bounded by $2\sum_{s\in S, o\in O} c_{so}y(s,o)$.

and when added yield

$$-c_f(S) + 3c_f(O) + 2(c_s(S) + c_s(O)) \geq 0$$

or

$$c_f(S) \leq 3c_f(O) + 2(c_s(S) + c_s(O)) \tag{4.1}$$

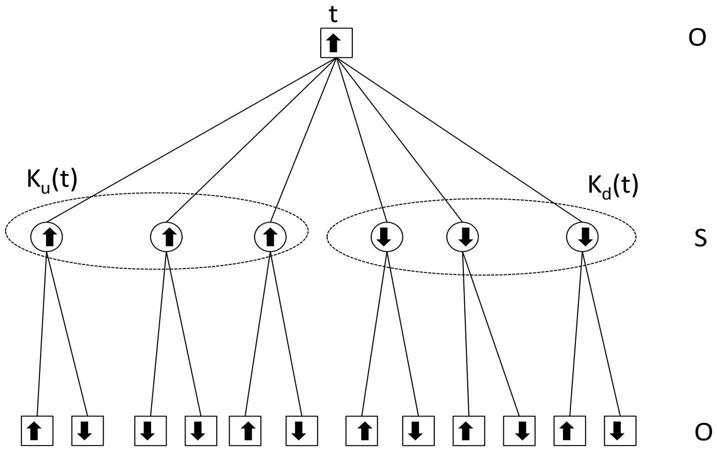

Figure 4.1: The subtree of height 2 showing up-facilities and down-facilities. The square facilities are in the optimum solution while the circular facilities are in the locally optimal solution. The arrow in the facility identifies it as an up/down facility

We now discuss the choice of inequalities as given by Zhang *et al.* in greater detail. For a facility i, let $p(i)$ be the parent and $K(i)$ be 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. Let $K_u(i)$ (respectively $K_d(i)$) denote the children of i which are up-facilities (respectively down-facilities). For a facility $o \in O$ we further insist that

1. if o is an up-facility it is opened at most once in the 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 twice in the operations involving facilities which are descendants of o in the tree and is opened at most once in other operations.

Consider a facility $s \in S$ which is a child of $t \in O$.

Case 1: Let s be a down-facility and let $o \in O$ be a child of s. When s is closed we can assign $2y(s,o)$ clients served by s to facility o if o is a down-facility. Else we can

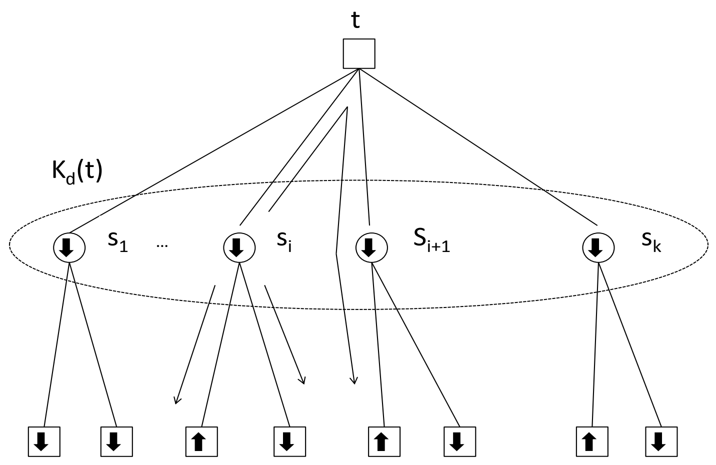

Figure 4.2: The ordering of facilities in $K_d(t)$ and the reassignment of clients when one of these facilities is closed.

assign $y(s,o)$ clients served by s to o. Thus we need to assign

$$\sum_{o \in O} y(s,o) - \sum_{o \in K_u(s)} y(s,o) - 2 \sum_{o \in K_d(s)} y(s,o) = y(s,t) - \sum_{o \in K_d(s)} y(s,o)$$

to facilities other than the children of s; we refer to the above quantity as $\texttt{rem}(s)$. To assign these clients, Zhang *et al.* order the facilities in $K_d(t)$ in increasing order of $\texttt{rem}(s)$; let $s_1, s_2, \ldots, s_k$ be the order (Figure 4.2). For $i \neq k$, they assign the remaining $\texttt{rem}(s_i)$ clients of s_i to facilities in $K_u(s_{i+1})$ with at most $y(s_{i+1}, o)$ clients assigned to facility $o \in K_u(s_{i+1})$. This takes care of all the remaining demand because

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

where the last inequality follows from the fact that s_{i+1} is a down-facility and hence

$$y(s_{i+1}, t) \le \sum_{o \in K_d(s_{i+1})} y(s_{i+1}, o) + \sum_{o \in K_u(s_{i+1})} y(s_{i+1}, o).$$

Thus Zhang *et al.* perform $\texttt{close}(s_i, K(s_i) \cup K_u(s_{i+1}))$ for $i \neq k$ and assign the clients of s_i as described above. The discussion above shows that the operation is feasible. The last facility in the order, i.e. s_k needs to be handled differently. In fact, s_k is closed together with the facilities in $K_u(t)$.

To bound the cost of reassigning clients served by facilities in $K_d(t) \setminus \{s_k\}$, note that

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

2. 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 $\texttt{rem}(s_i)$ and $\texttt{rem}(s_{i-1})$ are at most $y(s_i, t)$ the total contribution 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 $\texttt{rem}(s_1)c_{s_1 t} \leq y(s_1, t)c_{s_1 t}$ and the contribution of the edge (s_k, t) to the reassignment cost is at most $\texttt{rem}(s_{k-1})c_{s_k t} \leq \texttt{rem}(s_k)c_{s_k t} \leq y(s_k, t)c_{s_k t}$

3. for $i \neq k$ 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.

4. the contribution of the edge $(s_i, o), o \in K_u(s_i)$, $i \neq 1, k$ 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 $y(s_1, o)c_{s_1 o}$ and that of $(s_k, o), o \in K_u(s_k)$ is at most $y(s_k, o)c_{s_k o}$.

5. for $i \neq 1, k$, an up-facility $o \in K_u(s_i)$ is opened at most twice when considering facilities of S which are not descendants of o. A down-facility $o \in K_d(s_i)$ is opened once when considering facilities of S which are not descendants of o.

6. for $i = 1$, an up-facility $o \in K_u(s_1)$ is opened once when considering facilities of S which are not descendants of o. A down-facility $o \in K_d(s_1)$ is opened once when considering facilities of S which are not descendants of o.

7. for $i = k$, an up-facility $o \in K_u(s_k)$ is opened once when considering facilities of S which are not descendants of o.

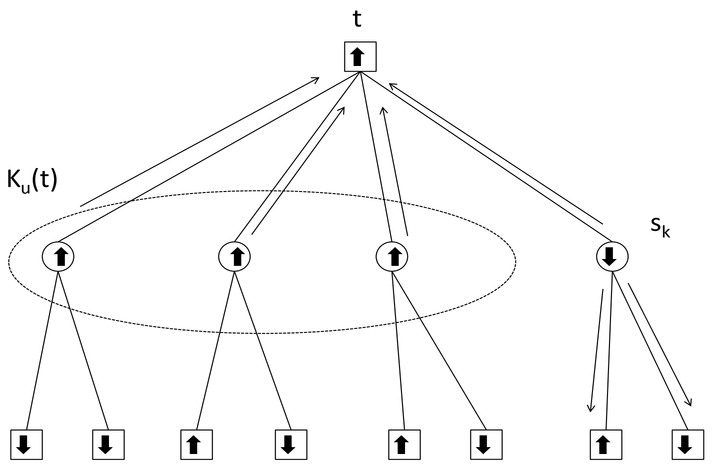

Figure 4.3: The *multi* operation considered when t is an up-facility.

8. t is not opened in these operations.

Case 2: Now consider the facilities in $K_u(t) \cup \{s_k\}$.

Case 2(a): If t is an up-facility then Zhang *et al.* perform a *multi* operation `multi`$(s_k, K(s_k), t, K_u(t))$ which can be viewed as a combination of `open`$(t, K_u(t))$ and `close`$(s_k, K(s_k))$ (Figure 4.3). In this operation clients served by $s \in K_u(t)$ are assigned to t, thus at most $2\sum_{s\in K_u(t)} y(s,t)$ clients served by facilities in $K_u(t)$ are assigned to t, and $y(s_k, o)$ clients served by s_k are assigned to facilities in $o \in K(s_k) \cup \{t\}$. Note that

1. the *multi* operation is feasible because the total number of clients assigned to t is at most $2\sum_{s\in K_u(t)} y(s,t) + y(s_k,t)$ which is at most the capacity of t since t is an up-facility.

2. the contribution of the edges $(s,t), s \in K_u(t)$ to the reassignment cost is at most $2y(s,t)c_{st}$ and that of the edge (s_k, o), $o \in K(s_k) \cup \{t\}$ is at most $y(s_k, o)c_{s_k o}$.

3. the up-facility t is opened once when considering facilities of S which are descendants of t.

4. a facility $o \in K(s_k)$, whether o is an up-facility or a down-facility, is opened once in this operation when considering facilities of S which are not descendants of o.

Case 2(b): Next consider the case when t is a down-facility. Zhang *et al.* partition the facilities in $K_u(t)$ into two sets $A \cup \{h'\}, B$ such that

$$\begin{aligned} \sum_{s \in A} 2y(s,t) + y(h',t) &\leq |N_O(t)| \\ \sum_{s \in B} 2y(s,t) + y(s_k,t) &\leq |N_O(t)| \end{aligned}$$

where $h' \in K_u(t)$ and $h' \notin A \cup B$. This can be done as follows:

1. Let $h \in K_u(t)$ be the facility for which $y(s,t)$ is the maximum for $s \in K_u(t)$. First partition $K_u(t) \setminus \{h\}$ into two sets A' and B' as follows:

   Consider the facilities of $K_u(t) \setminus \{h\}$ in an arbitrary order and continue assigning them to the set A' until $\sum_{s \in A'} 2y(s,t) \geq |N_O(t)|$. The last facility considered and the remaining facilities are assigned to B'. If $\sum_{s \in B'} 2y(s,t) > |N_O(t)|$ then $\sum_{s \in A' \cup B' \cup \{h\}} 2y(s,t) > 2|N_O(t)|$ which is a contradiction. Hence $\sum_{s \in B'} 2y(s,t)$ $\leq |N_O(t)|$

2. Next, construct the sets A and B as follows:

   Let $\sum_{s \in A'} y(s,t) \geq \sum_{s \in B'} y(s,t)$.

   (a) if $\sum_{s \in A'} 2y(s,t) + y(s_k,t) > |N_O(t)|$ then, since $\sum_{s \in A' \cup B'} 2y(s,t) + 2y(h,t) + 2y(s_k,t) \leq 2|N_O(t)|$ therefore $\sum_{s \in B'} 2y(s,t) + 2y(h,t) + y(s_k,t) \leq |N_O(t)|$. In this case, we take $B = B' \cup \{h\}$, note that $\sum_{s \in B} 2y(s,t) + y(s_k,t) \leq |N_O(t)|$. Let h' be any facility in A' then we take $A = A' \setminus \{h'\}$. Since $\sum_{s \in A'} 2y(s,t) \leq |N_O(t)|$, therefore $\sum_{s \in A} 2y(s,t) + y(h',t) \leq |N_O(t)|$.

   (b) If $\sum_{s \in A'} 2y(s,t) + y(s_k,t) \leq |N_O(t)|$, and $\sum_{s \in A'} 2y(s,t) + y(h,t) > |N_O(t)|$, then we must have $\sum_{s \in B'} 2y(s,t) + 2y(s_k,t) + y(h,t) \leq |N_O(t)|$. This implies

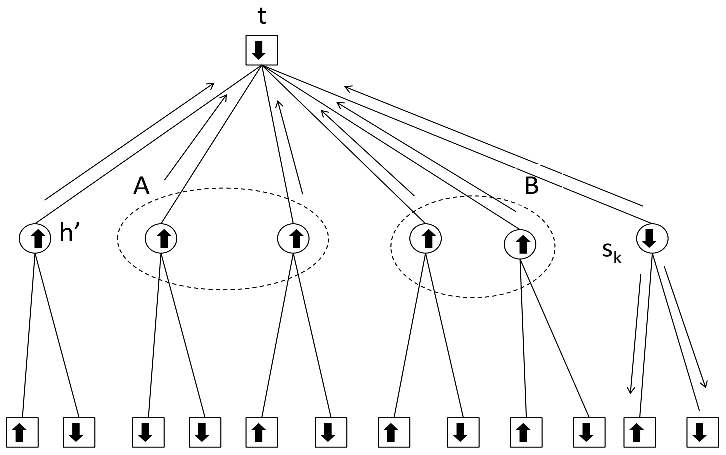

Figure 4.4: The partition of $K_u(t)$ and the *multi* operations considered when t is a down-facility.

$\sum_{s\in B'} 2y(s,t) + y(h,t) \leq |N_O(t)|$. In this case we take $A = B'$, $h' = h$ and $B = A'$. Clearly A, B and h' satisfy the desired property.

(c) if $\sum_{s\in A'} 2y(s,t)+y(s_k,t) \leq |N_O(t)|$ and $\sum_{s\in A'} 2y(s,t)+y(h,t) \leq |N_O(t)|$ then this also implies that $\sum_{s\in B'} 2y(s,t) + y(s_k,t) \leq |N_O(t)|$ and $\sum_{s\in B'} 2y(s,t) + y(h,t) \leq |N_O(t)|$ (by the assumption given above). In this case we can take $A = A'$, $h' = h$ and $B = B'$. Clearly A, B and h' satisfy the desired property

If $\sum_{s\in B'} y(s,t) \geq \sum_{s\in A'} y(s,t)$, then by interchanging the role of A' and B' in the above construction, we can obtain the sets A and B which satisfy the desired property.

Zhang *et al.* now consider two *multi* operations (Figure 4.4). The first is `multi`($s_k, K(s_k), t, B$) which is a combination of `close`($s_k, K(s_k)$) and `open`(t, B) in which $y(s_k, o)$ clients of s_k are assigned to facilities $o \in K(s_k) \cup t$ and clients of facilities in B are assigned to t. Thus at most $\sum_{s\in B} 2y(s,t) + y(s_k,t)$ clients are assigned to t in this operation. The second operation is `multi`($h', K(h'), t, A$) which is a combination of `close`($h', K(h')$) and `open`(t, A) in which $y(h', o)$ clients of h' are assigned to facilities $o \in K(h') \cup t$ and clients of facilities in A are assigned to t. Thus at most $\sum_{s \in A} 2y(s,t) + y(h', t)$ clients are assigned to t in this operation. The properties of the sets A and B ensure that the capacity of t is not violated as the facilities in A and B are up-facilities. This implies that both the *multi* operations are feasible. Note that

1. the contribution of an edge $(s,t), s \in K_u(t)$ to the reassignment cost is at most $2y(s,t)c_{st}$.
2. the contribution of an edge (s_k, o), $o \in K(s_k) \cup \{t\}$ is at most $y(s_k, o)c_{s_k o}$.
3. the contribution of an edge (h', o), $o \in K(h') \cup \{t\}$ is at most $y(h', o)c_{h'o}$.
4. the down-facility t is opened at most twice when considering facilities of S which are descendants of t.
5. a facility $o \in K(s_k)$, whether o is an up-facility or a down-facility, is opened once in `multi`($s_k, K(s_k), t, B$) operation when considering facilities of S which are not descendants of o.
6. a facility $o \in K(h')$, whether o is an up-facility or a down-facility, is opened once in `multi`($h', K(h'), t, A$) operation when considering facilities of S which are not descendants of o.

From the above discussion we can conclude that a facility $o \in O$ is opened at most three times in all the operations considered, as summarized below:

1. When o is an up-facility: While considering the facilities of S which are descendants of o, o would be opened once when it is part of a *multi* operation `multi`($s_k, K(s_k), o, K_u(o)$) as discussed in case 2(a). While considering the facilities of S which are not descendants of o, o would be opened at most twice if $p(o)$

is a down-facility, as discussed in case 1 and would be opened at most once if $p(o)$ is an up-facility.

2. when o is a down-facility While considering the facilities of S which are descendants of o, o would be opened twice, as discussed in case 2(b). 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.

4.2 Improving the Operations

The key contribution of this chapter is to modify the *close* and *open* operations (and consequently the *multi* operation) to exploit the following observation.

Claim 4.2 *In the analysis of Zhang* et al. *a facility $o \in O$ is assigned a total of at most $2\sum_s y(s,o) = 2|N_O(o)| \leq 2u_o$ clients over all operations considered.*

Proof We will first prove the claim for the case when o is an up-facility which is followed by the case when o is a down-facility.

1. When o is an up-facility

   (a) While considering the facilities of S which are descendants of o, o would be part of a *multi* operation `multi`$(s_k, K(s_k), o, K_u(o))$ (as discussed in case 2(a)) and 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 $2\sum_{s\in K(o)} y(s,o) \leq 2\sum_s y(s,o)$.

   (b) We next consider the number of clients assigned to o when considering facilities of S which are not descendants of o. If the parent $p(o)$ of o is an up-facility then o could be assigned at most $y(p(o),o)$ clients in a *multi* operation involving $p(o)$(when $p(o)$ is h') as discussed in case 2(b). If $p(o)$ is a down-facility then o would be assigned at most $2y(p(o),o)$ clients and this can be argued as follows: consider the ordering of the down-facilities which are siblings of $p(o)$.

i. if $p(o)$ is the first facility in the ordering (referred to as s_1) then o is only part of `close`$(s_1, K(s_1) \cup K_u(s_2))$ and is assigned $y(s_1, o)$ clients as is discussed in case 1.

ii. if $p(o)$ is the i^{th} facility in the ordering (referred to as s_i) and is neither the first nor the last facility then o is part of `close`$(s_{i-1}, K(s_{i-1}) \cup K_u(s_i))$ and `close`$(s_i, K(s_i) \cup K_u(s_{i+1}))$ and is assigned $y(s_i, o)$ clients in each of these operations as discussed in case 1.

iii. if $p(o)$ is the last facility in the ordering (referred to as s_k) then o is part of `close`$(s_{k-1}, K(s_{k-1}) \cup K_u(s_k))$ as discussed in case 1 and a *multi* operation involving s_k as discussed in case 2(a)/case 2(b). In both these operations o is assigned $y(s_k, o)$ clients.

Hence the total number of clients assigned to o when considering facilities of S which are not descendants of o is at most $2y(p(o), o)$.

Therefore the total number of clients assigned to o when o is an up-facility is at most $2\sum_s y(s, o)$.

2. when o is a down-facility

(a) While considering the facilities of S which are descendants of o, o would be part of two *multi* operations: first is `multi`$(h', K(h'), o, A)$ and second is `multi`$(s_k, K(s_k), o, B)$ and the number of clients assigned to o in these operations is $2\sum_{s\in A} y(s, o) + y(h', o)$ and $2\sum_{s\in B} y(s, o) + y(s_k, o)$ respectively as discussed in case 2(b). 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 two *multi* operations is at most $2\sum_{s\in K(o)} y(s, o) \leq 2\sum_s y(s, o)$.

(b) We next consider the number of clients assigned to o when considering facilities of S which are not descendants of o. 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 *multi* operation involving $p(o)$ as discussed in case 2(b). If $p(o)$ is a down-facility then o would be assigned at most $2y(p(o), o)$ clients and this can be argued as follows. As before, consider the ordering of the down-facilities which are siblings of $p(o)$.

   i. if $p(o)$ is the i^{th} facility in the ordering (referred to as s_i) and is not the last facility then o is part of `close`$(s_i, K(s_i) \cup K_u(s_{i+1}))$ and is assigned $2y(s_i, o)$ clients as discussed in case 1.

   ii. if $p(o)$ is the last facility in the ordering (referred to as s_k) then o is part of a *multi* operation involving s_k in which o is assigned $y(s_k, o)$ clients as discussed in case 2(a)/case 2(b).

   Hence the total number of clients assigned to o when considering facilities of S which are not descendants of o is at most $2y(p(o), o)$.

Therefore the total number of clients assigned to o when o is a down-facility is at most $2\sum_s y(s, o)$.

. ■

Since each facility $o \in O$ is opened thrice in the analysis of Zhang *et al.* the above claim implies that when a facility is opened we do not use it to its full capacity.

4.2.1 *mopen*: The Modified Open

Recall that, given a feasible solution S', in the operation `open`(t, T) we open a facility $t \in F \setminus S'$ and close a subset of facilities $T \subseteq S'$. Our operation *mopen* is defined in such a manner so that if the capacity of t is more than the total capacity of the facilities in T

we use the remaining capacity of t to service clients j for which S'_j; the service cost of j in S', is larger than c_{tj}. However T is not known beforehand but is computed as a part of the procedure. Thus, we don't know how much capacity of t can be utilized this way. Thus we make a guess for the capacity of t which will be utilized for this purpose and T is computed for the remaining capacity of t as explained below.

1. Let k be a guess of the difference in the capacity of t and the total capacity of the facilities in T.

2. We reduce the capacity of t by k and using the procedure of Pal *et al.* find the subset $T \subseteq S$ which minimizes the estimated cost.

3. To utilize the remaining capacity k of t we do the following: order the clients in decreasing order of $S'_j - c_{tj}$ discarding those for which this quantity is negative. Let k' be the total number of clients for which $S'_j - c_{tj}$ is positive. The first $min\{k, k'\}$ clients in this ordering are assigned to t and the savings arising from this step is reduced from the estimated cost computed in step 2.

4. The process is repeated for all the values of k in$[0..u_t]$ and the solution for which the cost is minimum gives the optimum set T.

For a choice of facility t, steps 1-3 are repeated $u_t = O(m)$ times. Each time a knapsack problem is solved which can be done in polynomial time. Step 3 can be easily done in polynomial time. Therefore the above procedure runs in polynomial time.

Let S be a locally optimal solution with respect to our algorithm and O be an optimum solution. Note that a solution which is locally optimal with respect to our algorithm is also locally optimal with respect to Zhang *et al.*'s algorithm. To bound the facility cost, we consider an operation `mopen`(t, T) where $t \in O$ and $T \subseteq S$, such that $\sum_{s \in T} |N_S(s)| \leq |N_O(t)|$. Since $mopen(t, T)$ does not reduce the cost of the current solution, $open(t, T)$ operation (of Zhang *et al.*) also does not reduce the cost. In this *open*

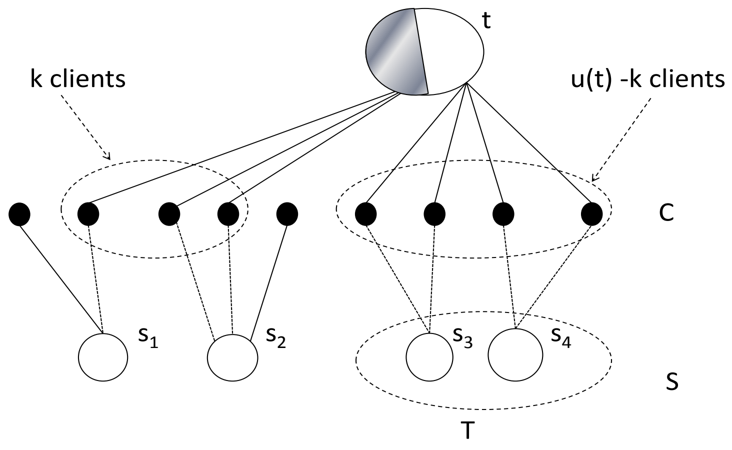

Figure 4.5: Set $T = \{s_3, s_4\}$ is closed and t is opened in the operation $mopen(t, T)$. Unshaded portion of facility t shows the capacity of t utilized for accommodating clients earlier served by facilities in T.

operation $|N_S(s)|$ clients of $s \in T$ are assigned to t. Let $z(s,t)$ denote this quantity. This operation then yields the inequality

$$f_t - \sum_{s \in T} f_s + \sum_{s \in T} z(s,t)c_{st} \geq 0$$

In the modified operation mopen(t, T), $z(s,t)$ number of clients served by each $s \in T$ and $|N_O(t)| - \sum_{s \in T} z(t,s)$ more clients served by facilities in S are assigned to t. We can then formulate the following inequality

$$f_t - \sum_{s \in T} f_s + \sum_{s \in T} z(s,t)c_{st} + \frac{|N_O(t)| - \sum_{s \in T} z(s,t)}{|N_O(t)|} \sum_{j \in N_O(t)} (O_j - S_j) \geq 0 \qquad (4.2)$$

The last term in the above inequality arises from the argument that instead of utilizing the remaining capacity, $|N_O(t)| - \sum_{s \in T} z(s,t)$, in the best possible way we could have assigned each client in $N_O(t)$ to the facility t to an extent $\frac{|N_O(t)| - \sum_{s \in T} z(s,t)}{|N_O(t)|}$ and in doing this we would not have reduced the estimated cost of the new solution. In fact the extent up to which we assign each client of $N_O(t)$ to t can be any value between 0 and $\frac{|N_O(t)|-\sum_{s\in T} z(s,t)}{|N_O(t)|}$. For the moment we assume that each client is assigned up to the extent of $\frac{|N_O(t)|-\sum_{s\in T} z(s,t)}{|N_O(t)|}$. We might reduce this quantity when we add all such inequalities, if the need arises. The purpose of doing this will become clear at the time this is required to be done.

4.2.2 *mclose*: The Modified Close

In `close`(s,T) we close a facility $s \in S$ and open a set of facilities $T \subseteq F \setminus S$. As in the *modified open* operation, if the total capacity of the facilities in T exceeds the number of clients served by s in the solution S, then we would like to use the remaining capacity in T to reduce the estimated cost of the operation.

Note that at most one facility in T could have some remaining capacity. We call this facility the *pivot* facility and denote it by t^*. Since we don't know this pivot facility, we guess t^*. Also since we don't know the excess capacity available with t^*, we guess that also.

Thus, given a facility $s \in S$, to determine the set T for which the estimated cost of the new solution is minimum we proceed as follows:

1. Let $t^* \in F \setminus S$ be a guess of the pivot facility and let k in $[0..u_{t^*}]$ be a guess of the difference in the capacity of s and the total capacity of the facilities in $T\setminus\{t^*\}$, i.e. k is the guess of the number of clients of s that will be served by t^* and $T\setminus\{t^*\}$should have sufficient capacity to serve the remaining $u_s - k$ clients of s i.e. if the capacity of s is u_s then set T of facilities is computed such that $\sum_{t\in T\setminus\{t^*\}} u_t \geq u_s - k$ or equivalently $u_s - \sum_{t\in T\setminus\{t^*\}} u_t \leq k$. Refer to the Figure 4.6.

2. We reduce the capacity of s by k and using the procedure of Pal *et al.* find the subset $T' \subseteq F \setminus (S \cup \{t^*\})$ which minimizes the estimated cost.

3. The remaining k clients of s are assigned to t^* and the estimated cost is increased by kc_{st^*}.

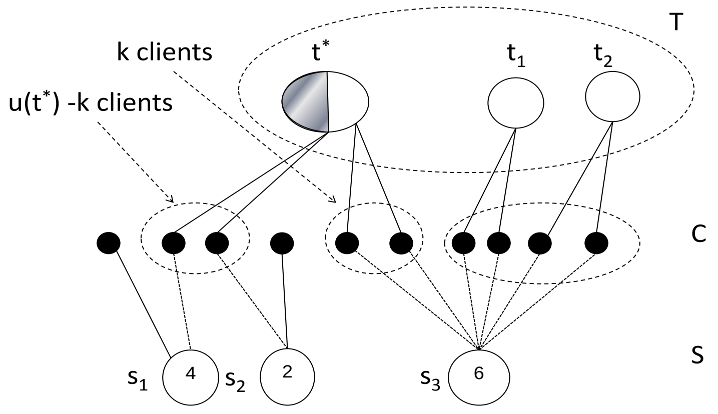

Figure 4.6: Set T is opened and s_3 is closed in the operation $mclose(s_3, T)$. Unshaded portion of facility t^* shows the capacity of t^* utilized for accommodating clients earlier served by s_3.

4. To utilize the remaining capacity of t^* do the following: order clients in decreasing order of $S_j - c_{t^*j}$ discarding those for which this quantity is negative. Let k' be this number. The first $min(k', u_{t^*} - k)$ clients in this ordering are assigned to t^* and the savings arising from this step is reduced from the estimated cost computed in step 3.

5. The process is repeated for all choices of $t^* \in F \setminus S$ and values of k in $[0..u_{t^*}]$ and the solution for which the estimated cost is minimum gives the optimum set $T = T' \cup t^*$.

Given a choice of s, steps 1-4 of the above procedure are repeated for all possible choices of t^* and k. Since the number of choices for t^* is $O(n)$ and that for k is $O(m)$, the procedure is repeated $O(nm)$ times. Step 2 which involves solving a covering knapsack problem runs in polynomial time, step 3 and 4 can be easily done in polynomial time. Therefore the overall running time of the procedure is polynomial.

Recall that O is an optimum solution and S is a locally optimal solution. Let $z(s, t)$

be the number of clients served by the facility $s \in S$ which are assigned to a facility $t \in T \subseteq O$ in the original operation close(s, T) then $\sum_{t \in T} z(s,t) = |N_S(s)|$ and this operation yields the inequality

$$-f_s + \sum_{t \in T} f_t + \sum_{t \in T} z(s,t) c_{st} \geq 0$$

Claim 4.3 *For the modified operation* mclose(s, T) *we can instead formulate the inequality*

$$-f_s + \sum_{t \in T} f_t + \sum_{t \in T} z(s,t) c_{st} + \sum_{t \in T} \sum_{j \in N_O(t)} \frac{|N_O(t)| - z(s,t)}{|N_O(t)|}(O_j - S_j) \geq 0 \tag{4.3}$$

Proof Consider the facilities of T in decreasing order of $z(s,t)/|N_O(t)|$ and keep including them into a set T_1 until the total capacity of the facilities in T_1 exceeds $|N_S(s)|$. Let t_1 be the last facility to be included into T_1. Then an mclose(s, T_1) operation in which t_1 is the pivot and is assigned $k_1 = |N_S(s)| - \sum_{t \in T_1 - t_1} |N_O(t)|$ clients which are served by s and $|N_O(t_1)| - k_1$ more clients served by other facilities in S would yield the inequality

$$\begin{aligned} -f_s + \sum_{t \in T_1} f_t + & \sum_{t \in T_1 - t_1} |N_O(t)| c_{st} + k_1 c_{st_1} + \\ & \frac{|N_O(t_1)| - k_1}{|N_O(t_1)|} \sum_{j \in N_O(t_1)} (O_j - S_j) \geq 0 \end{aligned} \tag{4.4}$$

The last term in the above inequality arises from the argument that instead of utilizing the remaining capacity, $|N_O(t_1)| - k_1$, in the best possible way we could have assigned each client in $N_O(t_1)$ to the facility t_1 to an extent $\frac{|N_O(t_1)| - k_1}{|N_O(t_1)|}$. In fact the extent up to which we assign each client of $N_O(t_1)$ to t_1 can be any value between 0 and $\frac{|N_O(t_1)| - k_1}{|N_O(t_1)|}$. For the moment we assume that each client of$N_O(t_1)$ is assigned to t_1 up to the extent of $\frac{|N_O(t_1)| - k_1}{|N_O(t_1)|}$. We will reduce this quantity when we add all such inequalities, if the need arises. The purpose of doing this will become clear at the time this is required to be done.

We take a linear combination of a sequence of inequalities of the form 4.4. In the linear combination we take Inequality 4.4 up to an extent of ξ_1 and at the same time for all facilities $t \in T_1 \setminus \{t_1\}$ we reduce $z(s,t)$ by $\xi_1 \cdot |N_O(t)|$ and reduce $z(s,t_1)$ by $\xi_1 \cdot k_1$, where $\xi_1 = min\left(min_{t \in T_1 \setminus \{t_1\}}\left(\frac{z(s,t)}{|N_O(t)|}\right), \frac{k_1}{|N_O(t_1)|}\right)$.

Next inequality in the sequence is obtained by again following the above procedure of selecting a subset of facilities in T with sufficient capacity to accommodate $|N_S(s)|$ clients. However this time the new $z(s,t)$ values are used. If T_2 is the set of facilities now selected and t_2 is the facility with smallest $z(s,t)/|N_O(t)|$ value then we can write another inequality similar to 4.4

$$\begin{aligned} -f_s + \sum_{t \in T_2} f_t + \sum_{t \in T_2 - t_2} |N_O(t)| c_{st} + k_2 c_{st_2} + \\ \frac{|N_O(t_2)| - k_2}{|N_O(t_2)|} \sum_{j \in N_O(t_2)} (O_j - S_j) \geq 0 \end{aligned} \tag{4.5}$$

Inequality 4.5 is included up to an extent of ξ_2 in the linear combination and $z(s,t)$ values are again reduced as earlier. This procedure is repeated until all $z(s,t)$ values become zero.

This process can be viewed as sending $\xi_i \cdot |N_S(s)|$ units of flow from s to facilities in T_i with facility $t \in T_i \setminus \{t_i\}$ receiving $\xi_i \cdot |N_O(t)|$ flow and facility t_i receiving $\xi_i \cdot k_i$ flow. The edges (s,t) have capacity $z(s,t)$ which is reduced by the amount of flow sent. Initially the total capacity of all edges $\sum_{t \in T} z(s,t)$ equals the amount of flow that needs to be sent i.e. $|N_S(s)|$, and this property is maintained with each step. By picking the facilities with the largest values of $z(s,t)/|N_O(t)|$ 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 $z(s,t)$ are zero and $|N_S(s)|$ units of flow have been sent.

Let us suppose that this process is carried out l number of times, then the linear combination looks like

$$\begin{aligned}&(\xi_1 + \ldots + \xi_l)\,(-f_s) + \sum_{i=\{1\ldots l\}} \xi_i \left(\sum_{t\in T_i} f_t\right) + \\ &\sum_{i=\{1\ldots l\}} \xi_i \left(\sum_{t\in T_i\setminus\{t_i\}} |N_O(t)|c_{st} + k_i c_{st_i}\right) + \\ &\sum_{i=\{1\ldots l\}} \xi_i \left(\frac{|N_O(t_i)| - k_i}{|N_O(t_i)|} \sum_{j\in N_O(t_i)} (O_j - S_j)\right) \geq 0.\end{aligned} \tag{4.6}$$

Note that $(\xi_1 + \ldots + \xi_l) = 1$.

Second term in the Inequality 4.6 can be written as

$$\sum_{t\in T} f_t \sum_{i=\{1\ldots l\}:t\in T_i} \xi_i = \sum_{t\in T} \lambda_t \cdot f_t \tag{4.7}$$

where $\lambda_t = \sum_{i=\{1\ldots l\}:t\in T_i} \xi_i$

Third term in the Inequality 4.6 can be written as

$$\begin{aligned}&\sum_{t\in T} c_{st} \sum_{i=\{1\ldots l\}:t\in T_i, t\neq t_i} \xi_i \cdot |N_O(t)| + \\ &\sum_{t\in T} c_{st} \sum_{i=\{1\ldots l\}:t\in T_i, t=t_i} \xi_i k_i = \sum_{t\in T} c_{st} z(s,t)\end{aligned} \tag{4.8}$$

Right hand side of the inequality follows from the construction of the linear combination.

For each $t \in T$, fourth term in the Inequality 4.6 can be written as

$$\sum_{j\in N_O(t)} (O_j - S_j) \sum_{i=\{1\ldots l\}:t=t_i} \xi_i \frac{|N_O(t)| - k_i}{|N_O(t)|} \tag{4.9}$$

When t is not pivot facility in T_i then we can consider fourth term to be zero. Let $k'_t = |N_O(t)|$ if t is not a pivot facility in T_i and $k'_t = k_i$ if t is pivot facility in T_i. 4.9 can now be written as

$$
\begin{aligned}
&\sum_{j \in N_O(t)} (O_j - S_j) \left(\sum_{i=\{1 \ldots l\}: t \in T_i, t \neq t_i} \xi_i \frac{|N_O(t)| - k'_t}{|N_O(t)|} + \sum_{i=\{1 \ldots l\}: t \in T_i, t = t_i} \xi_i \frac{|N_O(t)| - k'_t}{|N_O(t)|} \right) \\
&= \sum_{j \in N_O(t)} (O_j - S_j) \frac{1}{|N_O(t)|} \left(\sum_{i=\{1 \ldots l\}: t \in T_i} \xi_i |N_O(t)| - \sum_{i=\{1 \ldots l\}: t \in T_i} \xi_i k'_t \right) \\
&= \frac{\lambda_t |N_O(t)| - z(s,t)}{|N_O(t)|} \sum_{j \in N_O(t)} (O_j - S_j)
\end{aligned}
\tag{4.10}
$$

Using 4.7 to 4.10, Inequality 4.6 can now be written as:

$$
\begin{aligned}
&- f_s + \sum_{t \in T} \lambda_t \cdot f_t + \\
&\sum_{t \in T} c_{st} z(s,t) + \sum_{t \in T} \frac{\lambda_t \cdot |N_O(t)| - z(s,t)}{|N_O(t)|} \sum_{j \in N_O(t)} (O_j - S_j) \geq 0
\end{aligned}
\tag{4.11}
$$

A facility t would contribute $\lambda_t f_t + z(s,t) c_{st} + \frac{\lambda_t \cdot |N_O(t)| - z(s,t)}{|N_O(t)|} \sum_{j \in N_O(t)} (O_j - S_j)$ to the left hand side of Inequality 4.3. We add a $1 - \lambda_t$ multiple of the inequality

$$
f_t + \sum_{j \in N_O(t)} (O_j - S_j) \geq 0 \tag{4.12}
$$

which corresponds to the operation `add`(t), to the linear combination to match the contribution of t in Inequality 4.3. ■

4.2.3 ***mmulti*: The Modified multi**

Recall that `multi`(r, R, t, T) is a combination of `close`(r, R) and `open`(t, T) with an added provision that clients served by r may be assigned to facility t as well. Therefore the modification to *open* and *close* operations also implies modification to the *multi* operation which we refer to as *mmulti*. For a choice of facilities r, t, *modified multi* operation can be performed as follows:

1. Let k_a be a guess on the number of clients serviced by r that will be assigned to t. We make another guess k_b of the difference in remaining capacity, i.e. $u_t - k_a$, of t and total capacity of facilities in T, as we did for *mopen* operation.

2. We reduce the capacity of t by $k_a + k_b$ and using the procedure of Pal *et al.*find the subset $T \subseteq S$ which minimizes the estimated cost.

3. Next guess a pivot facility $t^* \in F \setminus (S \cup \{t\})$ and let k_c be a guess on the number of clients of r that will be assigned to t^*.

4. We reduce the capacity of r by $k_a + k_c$ and find a subset $R' \subseteq F \setminus (S \cup \{t, t^*\})$ by using the procedure for *close* operation of Pal *et al.*.

5. Of the remaining clients of r, k_a of them are assigned to t and k_c of them are assigned to t^*. The estimated cost is increased by $k_a c_{rt} + k_c c_{rt^*}$.

6. Compute $b_{t^*}(j) = S_j - c_{t^*j}$ and $b_t(j) = S_j - c_{tj}$ values for each client, discarding those for which these quantities are negative. Assign $u_{t^*} - k_c$ clients to t^* and k_b clients to t by finding a matching of maximum cost on the basis of $b_{t^*}(j)$ and $b_t(j)$ values. The savings arising out of this step is reduced from the estimated cost computed in the previous steps.

7. The process is repeated for all choices of k_a in $[0 \cdots u_t]$, k_b in $[0 \cdots u_t - k_a]$, a facility t^* and k_c in $[0 \cdots u_{t^*}]$. The solution for which the cost is minimum gives the sets $R = R' \cup \{t^*\}$ and T.

The total number of possible choices for k_a, k_b, k_c are $O(m)$ and that for t^* is $O(n)$. Thus the procedure is repeated $O(nm^3)$ times. Steps 2 and 4 can be done in polynomial time using knapsack and covering knapsack procedures respectively. Remaining steps can be easily performed in polynomial time. Therefore the overall procedure can be performed in polynomial time.

Let $z(r,o)$ be the number of clients served by r which are assigned to $o \in \{t\} \cup R$ and $z(s,t)$ be the number of clients served by $s \in T$ which are assigned to t in the original operation $multi(r,R,t,T)$ operation, then this operation would yield the inequality

$$-f_r - \sum_{s \in T} f_s + f_t + \sum_{o \in R} f_o + \sum_{o \in \{t\} \cup R} z(r,o) c_{ro} + \sum_{s \in T} z(s,t) c_{st} \geq 0$$

Claim 4.4 *For the modified operation* `mmulti`*(r, R, t, T) we can instead formulate the inequality*

$$\begin{aligned}
& - f_r - \sum_{s \in T} f_s + f(t) + \sum_{o \in R} f(o) + \sum_{o \in \{t\} \cup R} z(r,o) c_{ro} + \sum_{s \in T} z(s,t) c_{st} \\
& + \frac{|N_O(t)| - z(r,t) - \sum_{s \in T} z(s,t)}{|N_O(t)|} \sum_{j \in N_O(t)} (O_j - S_j) \\
& + \sum_{o \in R} \sum_{j \in N_O(o)} \frac{|N_O(o)| - z(r,o)}{|N_O(o)|} (O_j - S_j) \geq 0
\end{aligned} \tag{4.13}$$

Proof Consider the facilities of R in decreasing order of $z(r,o)/|N_O(o)|$ and keep including them into a set R' until the total capacity of the facilities in R' exceeds $|N_S(r)| - z(r,t)$. Let t^* be the last facility to be included into R'. Then an `mmulti`(r, R', t, T) operation in which t^* is the pivot and is assigned $k_c = |N_S(r)| - \sum_{t \in R' \setminus \{t^*\}} |N_O(t)| - z(r,t)$ clients which are served by r and $|N_O(t^*)| - k_c$ more clients served by other facilities in S would yield the inequality

$$\begin{aligned}
& - f_r - \sum_{s \in T} f_s + f(t) + \sum_{o \in R'} f(o) + \sum_{o \in R' \setminus \{t^*\}} |N_O(o)| c_{ro} + k_c c_{rt^*} + \sum_{s \in T} z(s,t) c_{st} \\
& + z(r,t) c_{rt} + \frac{|N_O(t)| - z(r,t) - \sum_{s \in T} z(s,t)}{|N_O(t)|} \sum_{j \in N_O(t)} (O_j - S_j) \\
& + \frac{|N_O(t^*)| - k_c}{|N_O(t^*)|} \sum_{j \in N_O(t^*)} (O_j - S_j) \geq 0
\end{aligned} \tag{4.14}$$

The second last term in the above inequality arises from the argument that instead of utilizing the remaining capacity, $|N_O(t)| - z(r,t) - \sum_{s\in T} z(s,t)$, in the best possible way we could have assigned each client in $N_O(t)$ to the facility t to an extent $\frac{|N_O(t)| - z(r,t) - \sum_{s\in T} z(s,t)}{|N_O(t)|}$ and in doing this we would not have reduced the estimated cost of the new solution.

The last term in the above inequality arises from the argument that instead of utilizing the remaining capacity, $|N_O(t^*)| - k_c$, in the best possible way we could have assigned each client in $N_O(t^*)$ to the facility t^* to an extent $\frac{|N_O(t^*)| - k_c}{|N_O(t^*)|}$.

We take a linear combination of a sequence of inequalities of the form 4.14.

A facility $o \in R$ would contribute $\lambda_o f_o + z(r,o)c_{ro} + \frac{\lambda_o \cdot |N_O(o)| - z(r,o)}{|N_O(o)|} \sum_{j\in N_O(o)} (O_j - S_j)$ to the left hand side of Inequality 4.13. We add a $1 - \lambda_o$ multiple of the inequality

$$f_o + \sum_{j\in N_O(o)} (O_j - S_j) \geq 0 \tag{4.15}$$

which corresponds to the operation add(o), to the linear combination to match the contribution of o in Inequality 4.13. ■

4.2.4 Putting Things Together

Inequalities 4.2, 4.3 and 4.13 have an additional term due to the modifications we have suggested for operations *open*, *close* and *multi*. These additional terms when taken over all the operations considered involving a facility $o \in O$ equals $\left(\alpha - \frac{\beta}{|N_O(o)|}\right) \sum_{j\in N_O(o)} (O_j - S_j)$ where α is the number of times o is opened and β is the total number of clients assigned to o in the operations defined by Zhang *et al.*. Recall that β is at most $2|N_O(o)|$ and α is at most 3. If a facility $o \in O$ is opened less than 3 times in these operations then we add the inequality corresponding to add(o) to our linear combination so that α becomes exactly 3.

Thus $\left(\alpha - \frac{\beta}{|N_O(o)|}\right) \geq 1 \; \forall o \in O$. If $\left(\alpha - \frac{\beta}{|N_O(o)|}\right) > 1$ then we will reduce the extent up to which term corresponding to $\sum_{j\in N_O(o)} (O_j - S_j)$ is included in the respective inequality, so that this quantity $\left(\alpha - \frac{\beta}{|N_O(o)|}\right)$ is exactly 1.

Thus we obtain an additional term of $c_s(O) - c_s(S)$ on the right hand side of Inequality 4.1 i.e.

$$c_f(S) \leq 3c_f(O) + 2(c_s(S) + c_s(O)) + c_s(O) - c_s(S)$$

which together with the bound on the service cost of S, $c_s(S) \leq c_f(O) + c_s(O)$, implies that

$$c_f(S) + c_s(S) \leq 5(c_f(O) + c_s(O)) = 5c(O).$$

Theorem 4.5 *The local search procedure with operations* add, mclose, mopen *and* mmulti *yields a locally optimal solution that is a 5-approximation to an optimum solution.*

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 $(\epsilon/5n)c(S)$. This modification implies that the right hand sides of inequalities 4.2, 4.3, 4.12,4.13,4.15 which are all zero should instead be $(-\epsilon/5n)c(S)$. Note that each $s \in S$ is closed in either an *mopen, mclose* or a *mmulti* operation and therefore appears in exactly one of the inequalities of type 4.2(for mopen), 4.3(for mclose) or 4.13(for mmulti). Further, for every $o \in O$, we add $(1 - \lambda_o <= 1)$ multiple of inequality 4.12/4.15 at most 3 times(because every o appears in atmost 3 inequalities of type 4.3 or 4.13 and $(1 - \lambda_o)$ multiple of inequality 4.12/4.15 is added to 4.3/4.13).

Putting all these modifications together gives rise to an extra term of at most $(4\epsilon/5)c(S)$. This implies that the facility cost of solution S is at most $4c(O)+(4\epsilon/5)c(S)$. Similarly, the service cost of solution S can now be bounded by $c(O)+(\epsilon/5)c(S)$. Adding these yields $(1 - \epsilon)c(S) <= 5c(O)$ which implies that S is a $5/(1 - \epsilon)$ approximation to the optimum solution.

At the beginning of the chapter, we made an assumption that $S \cap O = \phi$. When we remove this assumption, we need to do small modification to the operations as follows:

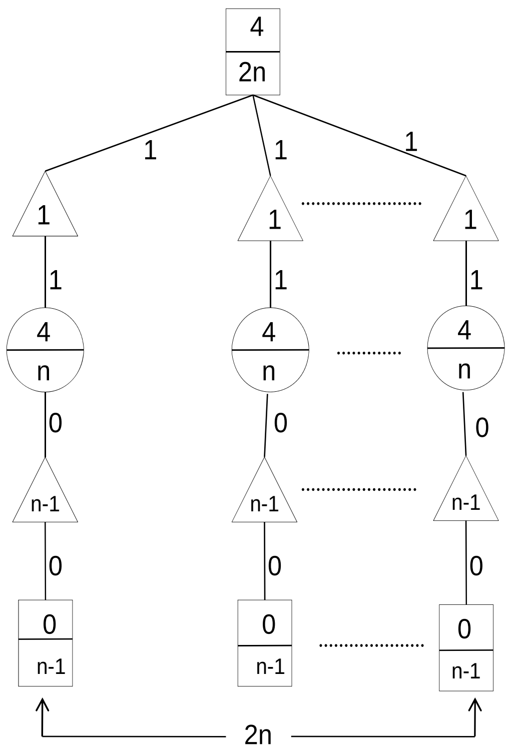

Figure 4.7: The tight example

Step 3 of *mopen*, step 4 of *mclose* and step 6 of *mmulti* are replaced by computatation of min-cost flow. In this general scenario, with the help of path decomposition, it can be shown that

$$c_f(S) + c_s(S) \leq 5(c_f(O) + c_s(O)) = 5c(O).$$

This is explained in detail in Chapter 5.

4.3 The Tight Example

Zhang *et al.*[ZCY05] provide an example (see Figure 4.7) to show that their analysis is tight. The rectangular boxes (respectively circles) in the Figure are the facilities in the optimum solution O (respectively S). The optimum solution has $2n$ facilities each with facility cost 0 and capacity $n-1$ and one facility with facility cost 4 and capacity $2n$. The solution S has $2n$ facilities each having a facility cost 4 and a capacity n. The clients are represented by triangles and there are $2n$ clients each having a demand of $n-1$ and $2n$ clients with 1 demand each. The numbers on the edges represent edge costs.

Zhang *et al.* argue that the solution S is locally optimal with respect to the *add, open, close* and *multi* operations. The cost of the optimal solution is $4+2n$ and that of S is $10n$ which gives a ratio of 5. However, scaling costs, as done in Zhang *et al.* would destroy the local optimality of this solution.

It is easy to confirm that this example is tight with respect to the *add* operation and the modified operations *mopen, mclose, mmulti* operations. We will one by one show that the solution S is locally optimal with respect to *add* and three modified operations.

1. **Local optimality with respect to *add* operation:** Adding a facility with facility cost 4 increases the facility cost of solution by 4 and total reassignment cost of clients does not change. Net cost of operation is positive and therefore does not reduce the cost of the solution. Adding a facility with facility cost 0 does not lead to any change in cost of the solution. Therefore we can conclude that solution S is locally optimal with respect to *add* operation.

2. **Local optimality with respect to *mopen* operation** `mopen`(t, T)**:** If t is a rectangle facility with capacity $n-1$, then $|T| = \phi$ due to capacity constraints. This operation is then equivalent to `add`(t) and we have already argued about this case. If t is facility with capacity 2n and cost 4, then $|T| \leq 2$. The case when $|T| = \phi$, is equivalent to `add`(t). When $|T| = 1$, which means only one circle facility is closed, then cost of the operation is determined by change in facility cost which is 4-4, estimated change in assignment cost of clients which consists of rerouting cost of n clients served by $t' \in T$ which is $2n$. Therefore cost of cost of operation when $|T| = 1$ is $4-4+2n > 0$.

If $|T| = 2$ which means any two circle facilities are closed in the operation, then change in facility cost is -8+4, estimated change in assignment cost is 4n. Therefore cost of this operation when $|T| = 2$ is $-8 + 4 + 4n > 0$.

From the above discussion we can conclude that S is locally optimal with respect to *mopen* operation.

3. **Local optimality with respect to *mclose* opeartion:** If we consider `mclose`(s, T) where s is any facility of S, then $|T| \geq 1$. If $|T| = 1$, then facility with facility cost 4 must be there in T. Let us call this facility t'. For this case we have already argued that the estimated cost of the operation is positive. For the case when $|T| > 1$ and $t' \in T$, then *mclose* operation with minimum estimated cost is one in which square facility which is at a distance 0 from s, say t'', is included in T. Now if $|T| = 2$ and $T = \{t', t''\}$, then change in facility cost is 0, estimated change in assignment cost of clients is 2. Therefore the estimated cost of the operation is $2 > 0$. If $|T| > 2$ then also estimated cost of operation is 2 which is positive.

   If $t' \notin T$, then *mclose* operation has minimum cost when facility at distance 0 from s belongs to T, and at least one more facility should be there in T to satisfy capacity constraints. If $|T| = 2$ then change in facility cost is -4. Estimated change in assignment cost is 4. Therefore estimated cost of the operation is 0. If $|T| > 2$ then also estimated cost of operation is 0.

   If facility at distance 0 from s is not included in T then estimated cost of the operation is at least 4n-4 which is positive.

   All these cases are exhaustive and imply that solution is locally optimal with respect to *mclose* operation.

4. **Local optimality with respect to *mmulti* operation:**

   When we consider modified operation `multi`(r, R, t, T), r can be any circle facility. If t is a square facility at distance 0 or 4 from r, then $|T| = \phi$ irrespective of what R is due to capacity constraints. And this case is essentially `mclose`$(r, R \cup \{t\})$ which we have already discussed,. If $t = t'$, then $|T| = 1$ and all the demand of facility in $|T|$ will be routed to $t = t'$. Change in facility cost due to this operation is -8+4, minimum rerouting cost of clients of r is 2 if facility at distance 0 belongs to R and because client at a distance 1 from r can be assigned to $t = t'$ at cost 2. Cost of rerouting clients of facility in T is 2n and any other reassignment of any other client from facilities in $S \setminus \{r + t"\}$ has 0 cost. Therefore this *mmulti* operation would cost $-8 + 4 + 2 + 2n > 0$. Any other *multi* operation would cost even more.

From all the above arguments it is clear that the given solution is locally optimal.

Chapter 5

A (5+ϵ)-Approximation Algorithm for Universal Facility location (UniFL) Problem

Universal facility location (UniFL) problem is a generalization of many variants of facility location problem. Facility location problem is said to be a *universal facility location* problem when facility cost of every facility $i \in F$ is defined by a cost function $f_i(.)$, which is a monotonically non-decreasing function, that is dependent on the capacity allocated at facility i. Therefore if u_i is the capacity allocated at 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 allocated capacity of the facilities is sufficient to serve all the clients and the total cost of opening facilities and assignment of clients to open facilities 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 completely determines the solution. Uncapacitated FLP, CFLP, and the k-median problem are all restricted variants of UniFLP. If $f_i(u_i) = f_i'$ for all $u_i > 0$, then it is just another way to describe uncapacitated facility location problem. Facility location problem is capacitated when $f_i(u_i) = \infty$ for $u_i > c_i$ and $f_i(u_i) = f'_i$ otherwise, where c_i is the fixed capacity of facility i.

The problem was introduced by Mahdian and Pal [MP03] who gave a (7.88+ϵ)-factor approximation which was improved by Vygen [Vyg07] to (6.702+ϵ)-factor. Recently, in a parallel work, Angel *et al.* [ATR13] proposed a new algorithm with (5.83+ϵ) approximation factor.

In this chapter we propose two operations for the problem: *open* and *close* that are extensions of *mopen* and *mclose* respectively. Using the ideas developed in chapters 3 and 4 we show that the algorithm provides a solution whose cost is within (5+ϵ) times the cost of an optimum solution. The arguments presented in this chapter suggest that we can achieve the same (5+ϵ)-approximation factor for non-uniform CFLP without using the *mmulti* operation.

The remainder of this chapter is organized as follows: in Section 5.1, we first give a broad overview of the arguments to show that we can achieve (5+ϵ) factor for non-uniform CFLP without the *mmulti* operation. And this fact is the main motivation behind the (5+ϵ)-factor algorithm for UniFLP. In Section 5.2 we describe our proposed algorithm. In Section 5.3 we prove the upper bound on the cost of the solution computed by the proposed algorithm. In Section 5.4 we give preliminary results on the basis of experiments done on the algorithm for the particular case of (non-uniform) CFLP.

5.1 A (5+ϵ)-factor algorithm for (non-uniform) CFLP without *mmulti*

In this section we'll give a broad idea as to how we can obtain (5+ϵ) factor for (non-uniform) CFL by dropping *mmulti* operation. Recall that *mmulti* is used at three places in the analysis given in chapter 4: case 2(a) - `mmulti`($s_k, K(s_k), t, K_u(t)$), case 2(b) - `mmulti`($s_k, K(s_k), t, B$), `mmulti`($h', K(h'), t, A$). Now, suppose instead of assigning

$y(s_k, t)$ ($/y(h', t)$) clients of s_k ($/h'$) to t in a *mmulti* operation, we open t for s_k ($/h'$) and utilize the remaining capacity of t in an efficient manner i.e. `mmulti`$(s_k, K(s_k), t, K_u(t))$ operation is replaced with `mopen`$(t, K_u(t))$ and `mclose`$(s_k, K(s_k) \cup \{t\})$ operations; `mmulti`$(s_k, K(s_k), t, B)$ operation is replaced with two operations `mopen`(t, B) and `mclose`$(s_k, K(s_k) \cup \{t\})$; and, `mmulti`$(h', K(h'), t, A)$ is replaced with `mopen`(t, A) and `mclose`$(h', K(h') \cup \{t\})$. For each such replacement we pay an additional cost of opening t but we also get an additive term of $\sum_j (O_j - S_j)$ which leads to saving an additive term in the service cost $c_s(S)$. The net result remains the same.

5.2 The local search operations

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. With abuse of notation we 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.

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. *add* operation is the same as given by Mahdian *et al.*. We propose two new operations: *open* and *close*. The operations are as given below:

- **add(s, δ):** In this operation capacity allocated at a facility s, say u_s is increased by an amount $\delta > 0$. Mincost flow problem is then solved to find the best assignment of clients to the facilities. Cost of the operation is given 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.

For a combination (s, δ), *add* operation, which mainly involves solving a mincost flow problem to find an assignment of clients, can be performed in polynomial time.

- **open(t, T, Δ):** In this operation, capacity allocated at $t \in F \setminus T$ is increased from the current u_t to $u_t + \Delta_t$, $\Delta_t > 0$ and $T \subseteq F \setminus \{t\}$ is such that capacity allocated at a facility $s \in T$ is decreased from u_s to $u_s - |\Delta_s|$, $\Delta_s < 0$, and $\sum_{s \in T} |\Delta_s| \leq \Delta_t$. Δ is an n-dimensional vector which defines the change in capacity allocation at facilities.

  Finding best such operation in polynomial time is not possible because exploring all the possibilities for set T and vector Δ cannot be done in polynomial time. We therefore search for a set T and a vector Δ for a given (t, Δ_t) combination as follows:

  1. Let $k \leq \Delta_t$ be a guess on the total capacity to be decreased at facilities in T.
  2. We solve a knapsack problem with capacity k to find a set T and vector Δ. A dynamic programming solution similar to that of Mahdian and Pal is used for the purpose. Let $c_s(-\delta_s) = c_{ts} \cdot \delta_s + f_s(u_s - \delta_s) - f_s(u_s)$ be the estimated decrease in cost of solution when δ_s number of clients served by facility s are reassigned to facility t. Rename facilities in $F \setminus \{t\}$ as $\{z_1, z_2, \cdots, z_{n-1}\}$. Let $b(i, w)$ denote the best possible benefit of moving $w(= 0, \cdots, k)$ amount of demand to t from the set of facilities $\{z_1, z_2, \cdots, z_i\}$. $b(i, w)$ can be computed as follows:

     $$b(i,w) = \begin{cases} c_{z_1}(-w) & (i = 1) \\ \min_{\delta=0 \ to \ w}(c_{z_i}(-\delta) + b(i-1, w-\delta)) & (i = 2, \cdots, n-1) \end{cases}$$

     Best T is computed by backtracking from $b(n-1, k)$ including facilities for which $\delta_{z_i} = argmin(\min_{\delta=0 \ to \ w}(c_{z_i}(-\delta) + b(i-1, w-\delta))) > 0$ for an appropriate w (determined while backtracking).

3. Once we have determined the set T and change in allocation of capacities for facilities in T, we can solve the mincost flow problem to assign the clients to facilities with their new capacity allocation. This is done to utilize the additional capacity $\Delta_t - k$ available with the facility t. Cost of the operation is given by: $f_t(u_t+\Delta_t)-f_t(u_t)+\sum_{s\in T}(f_s(u_s-|\Delta_s|)-f_s(u_s))+c_s(U')-c_s(U)$ where U' is the new solution after increasing the capacity of t and decreasing the capacity of facilities in T.
4. The process is repeated for all values of k in $[0, \Delta_t]$ and the solution for which the cost is minimum gives the optimum set T and vector Δ.

Steps 1, 2 and 3 of the above procedure can be performed in polynomial time. Further these steps are repeated for all the values of k in $[0, \Delta_t]$. Since Δ_t could be at most m, therefore these steps are repeated at most $O(m)$ times for one choice of Δ_t. Thus the *open* operation can be performed in polynomial time.

- **close(s, T, Δ):** In this operation, capacity allocation at facility $s \in F$ is decreased by amount $|\Delta_s|$, $\Delta_s < 0$ and capacity allocation at $t \in T \subseteq F$ is increased by Δ_t, $\Delta_t > 0$ and $\sum_{t\in T}\Delta_t \geq |\Delta_s|$. Also, a facility $t^* \in T$ is selected as a pivot facility to utilize the excess capacity of the set T, i.e. $\sum_{t\in T}\Delta_t - |\Delta_s|$ in a similar manner as we did in the case of *mclose* operation discussed in chapter 4. We cannot explore all possible sets T and vector Δ in polynomial time. Also pivot facility t^* is not known beforehand. Therefore to perform the operation in polynomial time, we fix Δ_s and determine T, pivot t^* and vector Δ as follows:

1. Let $t^* \in F \setminus \{s\}$ be a guess of a pivot facility and Δ_{t^*} be the guess of increase in capacity at t^*. Let $k \leq \Delta_{t^*}$ be a guess on the number of clients of s that will be assigned to t^*.
2. We solve a covering knapsack problem of capacity $|\Delta_s| - k$ and find a subset T' and vector Δ' as follows: Let $c_t(\delta_t) = c_{st}\cdot\delta_t + f_t(u_t+\delta_t) - f_t(u_t)$ be

the estimated increase in cost of solution when δ_t number of clients served by the facility s are reassigned to facility t. Rename facilities in $F \setminus \{s, t^*\}$ as $\{z_1, z_2, \cdots, z_{n-2}\}$. Let $b(i, w)$ denote the best possible benefit of moving $w(= 0, \cdots, |\Delta_s| - k)$ amount of demand from t to facilities $\{z_1, z_2, \cdots, z_i\}$, which can be computed as follows:

$$b(i, w) = \begin{cases} c_{z_1}(w) & (i = 1) \\ \min_{\delta=0 \ to \ w}(c_{z_i}(\delta) + b(i-1, w-\delta)) & (i = 2, \cdots, n-2) \end{cases}$$

Best set T' is computed by backtracking from $b(n-2, |\Delta_s| - k)$ including the facilities for which $\delta_{z_i} = argmin(\min_{\delta=0 \ to \ w}(c_{z_i}(\delta) + b(i-1, w-\delta))) > 0$ for an appropriate w (determined while backtracking). Set $T = T' \cup \{t^*\}$.

3. Once we have determined the set T and change in allocation of capacities for facilities in T, we can solve the mincost flow problem to assign the clients to facilities with their new capacity allocation. This is done to utilize the additional capacity $\Delta_{t^*} - k$ available with the facility t^*. Cost of the operation is given by: $f_s(u_s - |\Delta_s|) - f_s(u_s) + \sum_{t \in T}(f_t(u_t + \Delta_t) - f_t(u_t)) + c_s(U') - c_s(U)$ where U' is the new solution after decreasing the capacity of s and increasing the capacity of facilities in T.

4. The process is repeated for all choices of $t^* \in F \setminus \{s\}$ and values Δ_{t^*} in $[0, m - u_{t^*}]$. For each choice of (t^*, Δ_{t^*}), the process is repeated for k in $[0, \Delta_{t^*}]$. The solution for which the cost is minimum gives the set T and vector Δ.

Steps 1, 2, 3 and 4 of the above procedure can be performed in polynomial time. Further these steps are repeated for all choices of $t^* \in F \setminus \{s\}$ and values Δ_{t^*} in $[0, m - u_{t^*}]$ and for each choice of (t^*, Δ_{t^*}), the process is repeated for a value of k in $[0, \Delta_{t^*}]$. Since there could be at most $O(mn)$ choices for (t^*, Δ_{t^*}) pair, and Δ_t^*

could be at most m, therefore these steps are repeated at most $O(m^2n)$. Thus the *close* operation can be performed in polynomial time.

Whenever a local search operation is performed u_i values are updated to the number of clients assigned to i by the operation. U is locally optimal if none of these operations improve the cost of the solution and at this point the algorithm stops. The local search algorithm described above runs in polynomial time as:

1. each operation can be performed in polynomial time.
2. number of iterations performed by the algorithm can be bounded by a polynomial at the expense of an additive ϵ in the approximation factor by doing a local search operation only if the cost reduces by more than a $1 - \epsilon/4n$ factor, for $\epsilon > 0$.

5.3 Bounding the cost of our solution

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. Without loss of generality, let $u_s = |N_U(s)|$ be the number of clients served by s in U, similarly $u_s^* = |N_{U^*}(s)|$ be the no. of clients served by s in U^*.

First of all we construct a bipartite graph, $\hat{G}$, on the vertex set $C \cup F$ as explained in Section 3.4. Recall that:

1. Every client $j \in C$ has an edge from the facility $\sigma(j) \in F_U$, where $\sigma(j)$ is the facility which serves j in U, and an edge to the facility $\tau(j) \in F_{U^*}$, where $\tau(j)$ is the facility serving j in U^*. Thus each client has one incoming and one outgoing edge.

2. A facility $s \in F_U$ has $|N_U(s)|$ outgoing edges and a facility $o \in F_{U^*}$ has $|N_{U^*}(o)|$ incoming edges.

3. The graph $\hat{G}$ is decomposed into a set of maximal paths, $\mathcal{P}$, and cycles, $\mathcal{C}$. Note that all facilities on a cycle are from $F_U \cap F_{U^*}$.

4. In a path $P \in \mathcal{P}$ with a sequence of vertices $s_0, j_0, s_1, j_1, \ldots, s_k, j_k, o$, which starts at a vertex $s = s_0 \in F_U$ and ends at a vertex $o \in F_{U^*}$, head(P) denotes the client served by s and tail(P) denotes the client served by o on this path. Note that $\{s_1, s_2, \ldots, s_k\} \subseteq F_U \cap F_{U^*}$.

5. The *length* of a path P is given by

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

   where $U_j^*(U_j)$ is the service cost of client j in the solution $U^*(U)$. Note that $\sum_{P \in \mathcal{P}} \text{length}(P) + \sum_{Q \in \mathcal{C}} \text{length}(Q) = c_s(U) + c_s(U^*)$.

6. A *shift* along P is a reassignment of clients so that j_i which was earlier assigned to s_i is now assigned to s_{i+1} where $s_{k+1} = o$. 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.

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

7. 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

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

We remove all the edges and clients in the cycles from $\hat{G}$. Note that total number of paths beginning from a facility $s \in F_U$ is equal to $max(0, |N_U(s)| - |N_{U^*}(s)|)$ and total number of paths terminating at a facility $o \in F_{U^*}$ is at most $max(0, |N_{U^*}(o)| - |N_U(o)|)$. Let N_s^o be the set of paths that begin at $s \in F_U$ and end at $o \in F_{U^*}$. We redefine $N_U(s) = \cup_{o \in F_{U^*}} N_s^o$ and $N_{U^*}(o) = \cup_{s \in F_U} N_s^o$. Let S be the set of those facilities for which $|N_U(s)| > 0$ and O be the set of facilities for which $|N_{U^*}(o)| > 0$. Note that for a facility $s \in S$, $u_s > u_s^*$ and for a facility $o \in O$, $u_o^* > u_o$. Hence, $S \cap O = \phi$. For the rest of the facilities in F i.e. for $i \in F \setminus (S \cup O)$, $u_i = u_i^*$.

To formulate a suitable set of inequalities an exchange graph G is built whose vertices are the set of facilities in S and the facilities in O.

The exchange graph: An exchange graph is a bipartite graph with S and O defining the two partitions. In order to bound the facility cost of facilities in S, we would close every facility in S and transfer its clients to facilities in O. Here the notion of closing a facility s means that the capacity of facility s is decreased from the current capacity u_s to u_s^*, i.e. by an amount $|N_U(s)|$. Note that when a facility $s \in S$ is closed, total number of clients that needs to be reassigned is $|N_U(s)|$. A facility $o \in O$ can take in at most $|N_{U^*}(o)|$ number of clients along the paths that terminate at o. To obtain a set of feasible operations, we seek a flow in this exchange graph such that the amount of flow leaving a facility $s \in S$ is equal to $|N_U(s)|$ and the amount of flow that enters a facility $o \in O$ is at most $|N_{U^*}(o)|$. A feasible solution and hence in particular an optimal solution to the following linear program provides such a flow.

$$\begin{array}{rcl}
\text{min} & & \displaystyle\sum_{s \in S, o \in O} c_{so}\, y(s,o) \\
\text{s.t.} & & \\
\displaystyle\sum_{o \in O} y(s,o) & = & |N_U(s)| \quad \forall s \in S \\
\displaystyle\sum_{s \in S} y(s,o) & \leq & |N_{U^*}(o)| \quad \forall o \in O \\
y(s,o) & \geq & 0 \quad \forall s \in S, o \in O
\end{array}$$

The path decomposition of graph $\hat{G}$ as discussed above provides us with such a flow y whose cost is at most $c_s(U) + c_s(U^*)$. This is proved in the following lemma.

Lemma 5.1 *The cost of an optimal flow y is bounded by $c_s(U) + c_s(U^*)$.*

Proof To bound the cost of y, a flow $\hat{y}$ is constructed whose cost is at most $c_s(U) + c_s(U^*)$. Consider a facility $s \in S$ and a facility $o \in O$. Let $P = s_1 - o_1 - s_2 - o_2 \dots s_k - o_k$, where $s = s_1$ and $o = o_k$, be a path from s to o as defined by the path decomposition of $\hat{G}$. Let $\hat{y}(s,o)$, a flow on edge (s,o) be equal to the number of paths that begin from s and terminate at o . Note that flow $\hat{y}$ defined in this manner is a feasible flow which satisfies the constraints of the LP given above as $\sum_{o \in O} \hat{y}(s,o) = \sum_{o \in O} |N_s^o| = |N_U(s)|$ and $\sum_{s \in S} \hat{y}(s,o) = \sum_{s \in S} |N_s^o| = |N_{U^*}(o)|$. Also by triangle inequality $c_{so} \leq$ length(P) $\forall P \in \mathcal{P}$ such that P begins at s and ends at o. Thus,

$$c_{so}\hat{y}(s,o) \leq \sum_{P \in N_s^o} \text{length}(P).$$

Therefore

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

which proves the claimed bound on the cost of an optimal flow y. ■

Let G' be the bipartite graph with S and O defining the partitions of the graph and the edges be those which have non-zero flow y on them where y is an optimum flow of the above LP. If this graph is not acyclic, then by modifying the flow y, it can be turned into one without increasing the cost of the flow as follows: Consider a cycle in G' and let C_E be the edges on the cycle. Partition the edges of C_E into sets C_1, C_2 such that the edges of C_1 (and C_2) are alternate edges on the cycle. Let γ be the minimum value of an edge in C_E. Consider two operations: one in which we increase the value of edges in C_1 and decrease the value of edges in C_2 by an amount γ and the other in which we do the inverse *i.e.*,decrease the value of the edges in C_1 and increase the value of the edges in C_2 by γ. Note that 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 value of 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 $\sum_{s \in S} y(s,o) = |N_{U^*}(o)|$ and $\sum_{o \in O} y(s,o) = |N_U(s)|$ still holds. However $y(s,o)$ is no more equal to $|N_s^o|$.

We consider potential local 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 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

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

.

and when added yield

$$\begin{aligned} &- \sum_{s \in S} (f_s(u_s) - f_s(u_s^*)) + 5 \sum_{o \in O} (f_o(u_o^*) - f_o(u_o)) \\ &+ 2 \sum_{s \in S, o \in O} c_{so} y(s,o) + 3 \sum_{o \in O} \sum_{P \in N_{U^*}(o)} \text{shift}(P) \geq 0 \end{aligned} \tag{5.1}$$

Also for $i \in A = F \setminus (S \cup O)$, $f_i(u_i^*) = f_i(u_i)$, therefore we get

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

or

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

since

$$
-\sum_{s\in S\cup O\cup A}f_s(u_s)\geq-\sum_{s\in S}f_s(u_s)-5\sum_{o\in O}f_o(u_o)-5\sum_{i\in A}f_i(u_i) \tag{5.2}
$$

and

$$
5\sum_{s\in S\cup O\cup A}f_s(u_s^*)\geq\sum_{s\in S}f_s(u_s^*)+5\sum_{o\in O}f_o(u_o^*)+5\sum_{i\in A}f_i(u_i^*) \tag{5.3}
$$

Therefore, we get

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

Third term in the above inequality can be bounded by $2(c_s(U)+c_s(U^*))$ by Lemma 5.1. Also, fourth term can be written as

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

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 we get,

$$-c_f(U) + 5c_f(U^*) + 2(c_s(U) + c_s(U^*)) + 3(c_s(U^*) - c_s(U)) \geq 0 \tag{5.4}$$

and finally we get the following bound on the cost of our solution

$$c_f(U) + c_s(U) \leq 5c_f(U^*) + 5c_s(U^*) \tag{5.5}$$

Consider a subtree T in G' of height 2 rooted at $t \in O$. Figure 4.1 shows one such subtree. We use the same terminology as was used in chapter 4 to denote the parent child relationship. Recall that for a facility i, $p(i)$ is the parent and $K(i)$ are 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). The operations that we consider for the analysis are such that for a facility $o \in O$:

1. If o is an *up-facility*, then 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* then 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.

5.3.1 Closing *up-facilities* which are children of t

Consider *up-facilities* of S which are children of t. If t is an up-facility then all the facilities $s \in K_u(t)$ can be closed in a single operation $open(t, K_u(t), \Delta)$ in which clients of a facility $s \in K_u(t)$ are assigned to t and $\Delta_s = -|N_U(s)|\ \forall s \in K_u(t)$ and $\Delta_t = |N_{U^*}(t)|$, refer to Figure 5.1. Note that

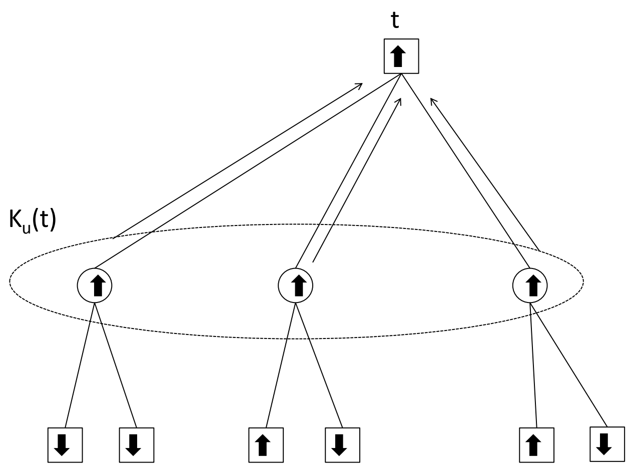

Figure 5.1: *open* operation considered for handling facilities in $K_u(t)$ when t is an up-facility.

1. the operation is feasible since

$$\begin{aligned} |N_{U^*}(t)| &= \sum_{s\in K(t)} y(s,t) + y(p(t),t) \\ &\geq \sum_{s\in K_u(t)} y(s,t) + y(p(t),t) \\ &\geq \sum_{s\in K_u(t)} 2y(s,t) \\ &\geq \sum_{s\in K_u(t)} |N_U(s)| \end{aligned}$$

   where second last inequality is due to the fact that t is an *up-facility* and the last inequality follows from

$$|N_U(s)| = y(s,t) + \sum_{o\in K(s)} y(s,o) \leq 2y(s,t) \quad \forall s \in K_u(t)$$

   as s is an up-facility.

2. the contribution of an edge (s,t), where $s \in K_u(t)$, to the reassignment cost is at most $2y(s,t)c_{st}$.

3. If $|N_{U^*}(t)| - \sum_{s \in K_u(t)} |N_U(s)| > 0$ then we can assign $|N_{U^*}(t)| - \sum_{s \in K_u(t)} |N_U(s)|$ more clients to t by shifting along these many paths in $N_{U^*}(t)$. Since U is locally optimal, this operation will not improve the cost of U. This operation then yields the inequality

$$\begin{aligned} & f_t(u_t^*) - f_t(u_t) - \sum_{s \in K_u(t)} (f_s(u_s) - f_s(u_s^*)) + \\ & \sum_{s \in K_u(t)} |N_U(s)| c_{st} + \frac{|N_{U^*}(t)| - \sum_{s \in K_u(t)} |N_U(s)|}{|N_{U^*}(t)|} \sum_{P \in N_{U^*}(t)} \text{shift}(P) \geq 0 \end{aligned} \tag{5.6}$$

The last term in the above inequality arises from the argument that instead of utilizing the remaining capacity, $|N_{U^*}(t)| - \sum_{s \in K_u(t)} |N_U(s)|$, in the best possible way we could shift along each path in $N_{U^*}(t)$ up to an extent $\frac{|N_{U^*}(t)| - \sum_{s \in K_u(t)} |N_U(s)|}{|N_{U^*}(t)|}$ and assign the client tail(P) to t up to this much extent only and in doing so the cost would not reduce. We can reduce this quantity, if required, to get the desired inequalities.

If t is a down-facility, then let $h \in K_u(t)$ be the facility with largest $y(h,t)$ value. Rest of the facilities $s \in K_u(t) \setminus h$ can be partitioned into two sets A and B such that $\sum_{s \in A} |N_U(s)| \leq |N_{U^*}(t)|$ and $\sum_{s \in B} |N_U(s)| \leq |N_{U^*}(t)|$. The partitioning procedure similar to the one described in Section 4.1(case 2b) is used for the purpose. The facilities in A and B are closed in two *open* operations $open(t, A, \Delta')$ and $open(t, B, \Delta'')$ respectively where $\Delta'_t = \Delta''_t = |N_{U^*}(t)|$; $\Delta'_s = -|N_S(s)|$ for $s \in A$ and $\Delta''_s = -|N_S(s)|$ for $s \in B$, refer to Figure 5.2. Feasibility of the operations follows from the construction. Inequalities similar to the Inequality 5.6 are formulated due to these operations in which $K_u(t)$ is replaced with A and B respectively. Also note that the contribution of an edge (s,t), where $s \in K_u(t) \setminus \{h\}$, to the reassignment cost is at most $2y(s,t)c_{st}$.

The facilities in $K_d(t)$ and h are handled using *close* operations as discussed next.

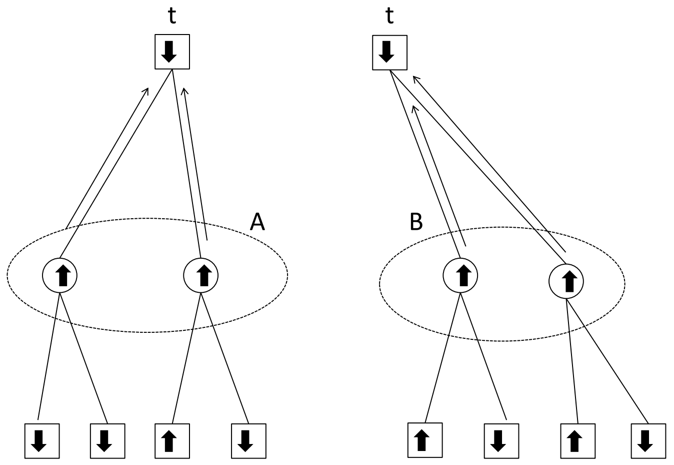

Figure 5.2: *close* operations considered for facilities in $K_u(t) \setminus \{h\}$ when t is a down facility.

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

Now we discuss the operations to close facilities $s \in K_d(t) \cup \{h\}$, refer to Figure 5.3. Consider the facilities in $K_d(t)$. Order the facilities in $K_d(t)$ according to their $rem(s)$ values as discussed in chapter 4. Recall that $rem(s) = y(s,t) - \sum_{o \in K_d(s)} y(s,o)$. Let $s_1, s_2, \cdots, s_k$ be the order so defined. A facility $s_i \in K_d(t)$ for $i < k$ can be closed in an operation $close(s_i, K(s_i) \cup K_u(s_{i+1}), \Delta)$. In this operation $\Delta_{s_i} = -|N_U(s_i)|$ and $\Delta_o = |N_{U^*}(o)|$, $\forall o \in K(s_i) \cup K_u(s_{i+1})$. The operation is feasible as can be argued on the same lines as in case 1 of Section 4.1. In this operation $y(s_i, o)$ flow is sent to $o \in K_u(s_i)$, $2y(s_i, o)$ flow is sent to $o \in K_d(s_i)$ (since o is a down-facility, therefore $y(s_i, o) \leq \sum_{s' \in K(o)} y(s', o)$). The remaining flow from s_i i.e. $rem(s_i)$ is sent to facilities in $K_u(s_{i+1})$. Let's denote amount of flow sent on an edge (s_i, o), $o \in K(s_i) \cup K_u(s_{i+1})$ in this operation by $z(s_i, o)$.

If $\sum_{o \in K(s_i) \cup K_u(s_{i+1})} |N_{U^*}(o)| > |N_U(s_i)|$ then we can find a subset $T' \subseteq K(s_i) \cup K_u(s_{i+1})$ such that $\sum_{o \in T'} |N_{U^*}(o)| \geq |N_U(s_i)|$ and set T' is such that except for one facility $t^* \in T'$, available capacity i.e. $|N_{U^*}(o)|$, of all facilities in $T' \setminus \{t^*\}$ is fully

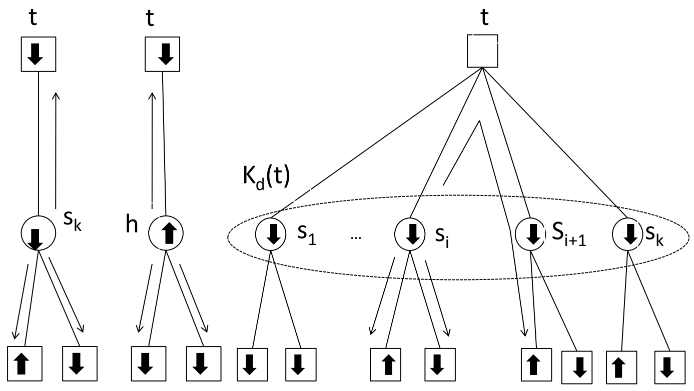

Figure 5.3: *close* operations for facilities in $K_d(t)$ and facility h showing the reassignment of clients when one of these facilities are closed.

exhausted in accommodating the clients coming from s_i. Some capacity of t^* say k is used to accommodate the clients of s_i and the remaining available capacity $|N_{U^*}(t^*)| - k$ is filled with clients which are assigned to t^* when we shift along some paths in $N_{U^*}(t^*)$.

Claim 5.2 *For the operation $close(s_i, K(s_i) \cup K_u(s_{i+1}), \Delta)$ we can formulate the inequality*

$$- (f_{s_i}(u_{s_i}) - f_{s_i}(u^*_{s_i})) + \sum_{o \in K(s_i) \cup K_u(s_{i+1})} (f_o(u^*_o) - f_o(u_o)) + \sum_{o \in K(s_i) \cup K_u(s_{i+1})} z(s_i, o) c_{s_i o}$$
$$+ \sum_{o \in K(s_i) \cup K_u(s_{i+1})} \sum_{P \in N_{U^*}(o)} \frac{|N_{U^*}(o)| - z(s_i, o)}{|N_{U^*}(o)|} \textit{shift(P)} \geq 0 \tag{5.7}$$

Proof Denote the set $K(s_i) \cup K_u(s_{i+1})$ by T. Consider the facilities of T in decreasing order of $z(s_i, o)/|N_{U^*}(o)|$ and keep including them into a set T' until the total capacity of the facilities in T', i.e. $\sum_{o \in T'} |N_{U^*}(o)|$, exceeds $|N_U(s_i)|$. Let t^* be the last facility to be included into T'. Then a $close(s_i, T', \Delta')$ operation, where $\Delta'_o = |N_{U^*}(o)|$ $\forall o \in T'$ and $\Delta'_{s_i} = -|N_U(s_i)|$, t^* is the *pivot* facility and is assigned $k = |N_U(s_i)| -$

$\sum_{o\in T'\setminus\{t^*\}} |N_{U^*}(o)|$ clients served by s_i, would yield the inequality

$$- (f_{s_i}(u_{s_i}) - f_{s_i}(u^*_{s_i})) + \sum_{o\in T'}(f_o(u^*_o) - f_o(u_o)) + \\ \sum_{o\in T'\setminus\{t^*\}} |N_{U^*}(o)|c_{s_i o} + kc_{s_i t^*} + \frac{|N_{U^*}(t^*)| - k}{|N_{U^*}(t^*)|} \sum_{P\in N_{U^*}(t^*)} \text{shift}(P) \geq 0 \tag{5.8}$$

The last term in the above inequality arises from the argument that instead of utilizing the remaining capacity, $|N_{U^*}(t^*)| - k$, in the best possible way we could have assigned for each path in $N_{U^*}(t^*)$ the client tail(P) to the facility t^* to an extent $\frac{|N_{U^*}(t^*)|-k}{|N_{U^*}(t^*)|}$ without reducing the cost by shifting along these paths up to an extent $\frac{|N_{U^*}(t^*)|-k}{|N_{U^*}(t^*)|}$.

We take a linear combination of a sequence of inequalities of the form 5.8 in a similar way as done in Section 4.2.2. In the linear combination we take 5.8 up to an extent of ξ and at the same time for all facilities $o \in T' \setminus \{t^*\}$ we reduce $z(s_i, o)$ by $\xi \cdot |N_{U^*}(o)|$ and reduce $z(s_i, t^*)$ by $\xi \cdot k$, where $\xi = min\left(min_{o\in T'\setminus\{t^*\}}\left(\frac{z(s_i,o)}{|N_{U^*}(o)|}\right), \frac{k}{|N_{U^*}(t^*)|}\right)$.

This process can be viewed as sending $\xi \cdot |N_U(s_i)|$ units of flow from s_i to facilities in T' with facility $o \in T' \setminus \{t^*\}$ receiving $\xi \cdot |N_{U^*}(o)|$ 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. Initially the total capacity of all edges $\sum_{o\in T} z(s_i, o)$ equals the amount of flow $|N_U(s_i)|$ that needs to be sent and this property is maintained at each step. By picking the facilities with the largest values of $z(s_i, o)/|N_{U^*}(o)|$ 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 $|N_U(s_i)|$ units of flow have been sent.

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

$$f_o(u^*_o) - f_o(u_o) + \sum_{P\in N_{U^*}(o)} \text{shift}(P) \geq 0 \tag{5.9}$$

which corresponds to the operation $\texttt{add}(o, |N_{U^*}(o)|)$, to the linear combination to match the contribution of o in Inequality 5.7. ■

Next we will close s_k and open $K(s_k) \cup \{t\}$ in an operation $close(s_k, K(s_k) \cup \{t\}, \Delta)$. In this operation $\Delta_{s_k} = -|N_U(s_k)|$ and $\Delta_o = |N_{U^*}(o)| \; \forall o \in K(s_k) \cup \{t\}$. Operation for closing s_k is feasible since $\sum_{o \in K(s_k) \cup \{t\}} |N_{U^*}(o)| \geq \sum_{o \in K(s_k) \cup \{t\}} y(s_k, o) = |N_U(s_k)|$ and by sending $y(s_k, o)$ flow to each $o \in K(s_k) \cup \{t\}$ we are done. We can now formulate the following inequality for this operation, as we did in case when we considered *close* operation for facility s_i.

$$- (f_{s_k}(u_{s_k}) - f_{s_k}(u^*_{s_k})) + \sum_{o \in K(s_k) \cup \{t\}} (f_o(u^*_o) - f_o(u_o)) + \sum_{o \in K(s_k) \cup \{t\}} y(s_k, o) c_{s_k o}$$
$$+ \sum_{o \in K(s_k) \cup \{t\}} \sum_{P \in N_{U^*}(o)} \frac{|N_{U^*}(o)| - y(s_k, o)}{|N_{U^*}(o)|} \text{shift}(P) \geq 0 \tag{5.10}$$

Note that, due to the operations considered above in which we close facilities of $K_d(t)$,

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

2. 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 $\texttt{rem}(s_i)$ and $\texttt{rem}(s_{i-1})$ are less than $y(s_i, t)$ the total contribution is at most $2y(s_i, t) c_{s_i t}$.

3. The contribution of the edge (s_1, t) to the reassignment cost is at most $\texttt{rem}(s_1) c_{s_1 t} \leq y(s_1, t) c_{s_1 t}$.

4. The contribution of the edge (s_k, t) to the reassignment cost is at most $(\texttt{rem}(s_{k-1}) + y(s_k, t)) c_{s_k t} \leq 2y(s_k, t) c_{s_k t}$

5. 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.

6. the contribution of the edge $(s_i, o), o \in K_u(s_i)$, $i \neq 1$ is at most $2y(s_i, o)c_{s_io}$ 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 $y(s_1, o)c_{s_1o}$.

Facility h can be closed in the operation $close(h, K(h) \cup \{t\}, \Delta)$ where $\Delta_o = |N_{U^*}(o)|\ \forall o \in K(h) \cup \{t\}$ and $\Delta_h = -|N_U(h)|$. This operation is feasible because $\sum_{o\in K(h)\cup\{t\}} |N_{U^*}(o)| \geq \sum_{o\in K(h)\cup\{t\}} y(h, o) = |N_U(h)|$. For this operation we can formulate the following inequality:

$$\begin{aligned} &- (f_h(u_h) - f_h(u_h^*)) + \sum_{o\in K(h)\cup\{t\}} (f_o(u_o^*) - f_o(u_o)) + \sum_{o\in K(h)\cup\{t\}} y(h, o)c_{ho} \\ &+ \sum_{o\in K(h)\cup\{t\}} \sum_{P\in N_{U^*}(o)} \frac{|N_{U^*}(o)| - y(h, o)}{|N_{U^*}(o)|} \text{shift}(P) \geq 0 \end{aligned} \tag{5.11}$$

Note that in this operation, the contribution of an edge (h, o), $o \in K(h) \cup \{t\}$ is at most $y(h, o)c_{ho}$.

5.3.3 Putting Things Together

In all the operations considered in the analysis 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 things in the following lemmas.

Lemma 5.3 *A facility $o \in O$ is opened at most 5 times in all the operations considered.*

Proof

1. 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 a *close* operation $close(s_k, K(s_k) \cup \{o\}, \Delta)$ where $s_k \in K_d(o)$ and once when it is part of an *open* operation $open(o, K_u(o), \Delta)$. While considering the facilities of S which are not descendants of o, o would be opened at most twice if $p(o)$ is a down facility and would be opened at most once if $p(o)$ is an up facility.

2. 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 a *close* operation $close(s_k, K(s_k) \cup \{o\}, \Delta)$ where $s_k \in K_d(o)$, once when facility $h \in K_u(o)$ is closed in an operation $close(h, K(h) \cup \{o\}, \Delta)$, and twice as a part of two *open* operations in which sets $A, B \subseteq K_u(o)$ are closed. 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.

■

Claim 5.4 *A facility $o \in O$ is assigned a total of at most $2\sum_s y(s,o) \leq 2|N_{U^*}(o)|$ clients from the facilities closed in the respective operations involving o over all the operations considered.*

Proof

1. When o is an up-facility

   (a) While considering the facilities of S which are descendants of o, o would be part of a $close(s_k, K(s_k) \cup \{o\}, \Delta)$ where $s_k \in K_d(o)$ and an *open* operation $open(o, K_u(o), \Delta)$ and 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 $2\sum_{s \in K(o)} y(s,o)$.

   (b) We next consider the number of clients assigned to o when considering facilities of S which are not descendants of o. If the parent of o, $p(o)$, is an up-facility then o could 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 this can be argued as follows. Consider the ordering of the down-facilities which are siblings of $p(o)$.

      i. if $p(o)$ is the first facility in the ordering (referred to as s_1) then o is only part of $close(s_1, K(s_1) \cup K_u(s_2), \Delta)$ and is assigned $y(s_1, o)$ clients.

ii. if $p(o)$ is the i^{th} facility in the ordering (referred to as s_i) and is neither the first nor the last facility then o is part of $close(s_{i-1}, K(s_{i-1}) \cup K_u(s_i), \Delta)$ and $close(s_i, K(s_i) \cup K_u(s_{i+1}), \Delta)$ and is assigned $y(s_i, o)$ clients in each of these operations.

iii. if $p(o)$ is the last facility in the ordering (referred to as s_k) then o is part of $close(s_{k-1}, K(s_{k-1}) \cup K_u(s_k), \Delta)$ and a *close* operation $close(s_k, K(s_k) \cup \{p(s_k)\}, \Delta)$ involving s_k. In both these operations o is assigned $y(s_k, o)$ clients.

Hence the total number of clients assigned to o when considering facilities of S which are not descendants of o is at most $2y(p(o), o)$.

Therefore the total number of clients assigned to o when o is an up-facility is at most $2\sum_s y(s, o)$.

2. when o is a down-facility

   (a) While considering the facilities of S which are descendants of o, o would be part of two *open* operations $open(o, A, \Delta)$ and $open(o, B, \Delta)$ and two *close* operations: $close(h, K(h) \cup \{o\}, \Delta)$ and $close(s_k, K(s_k) \cup \{o\}, \Delta)$. 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 $2\sum_{s\in K(o)} y(s, o)$.

   (b) We next consider the number of clients assigned to o when considering facilities of S which are not descendants of o. 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 this can be argued as follows. As before, consider the ordering of the down-facilities which are siblings of $p(o)$.

i. if $p(o)$ is the i^{th} facility in the ordering (referred to as s_i) and is not the last facility then o is part of $close(s_i, K(s_i) \cup K_u(s_{i+1}), \Delta)$ and is assigned $2y(s_i, o)$ clients.

ii. if $p(o)$ is the last facility in the ordering (referred to as s_k) then o is part of a *close* operation involving s_k in which o is assigned $y(s_k, o)$ clients.

Hence the total number of clients assigned to o when considering facilities of S which are not descendants of o is at most $2y(p(o), o)$.

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 5.5 *The total reassignment cost of all the operations is bounded by*

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

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 $\sum_{P \in N_{U^*}(o)}$ shift(P) for a facility $o \in O$ appears up to the extent of $\alpha - \beta/|N_{U^*}(o)|$ 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 Claim 5.4, β is at most $2|N_{U^*}(o)|$ and from Lemma 5.3 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 $add(o, N_{U^*}(o))$ to our linear combination so that each facility is now opened exactly five times. Therefore, the coefficient of the term $\sum_{P \in N_{U^*}(o)}$ shift(P) is at least 3. If its greater than 3 for some $o \in O$ then we will reduce the coefficient of $\sum_{P \in N_{U^*}(o)}$ shift(P) in some of the inequalities involving o to make this contribution exactly 3. ■

From Lemma 5.3 and Lemma 5.5, we can conclude that

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

Recall that 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 implies that the right hand sides of inequalities 5.6, 5.7, 5.9, 5.10, 5.11 which are all zero should instead be $(-\epsilon/4n)c(U)$. Note that each $s\in S$ is closed in either an *open* or a *close* operation and therefore appears in exactly one of the inequalities of type 5.6, 5.7, 5.10, 5.11. Further, every $o\in O$ appears in at most 3 close operations and therefore we add $(1-\lambda_o <= 1)$ multiple of inequality 5.9 atmost 3 times.

Putting all these modifications together gives rise to an extra term of at most $(4\epsilon/4)c(U)$. This implies that the cost of solution U is at most $5c(U^*)+\epsilon\cdot c(U)$ which implies that U is a $5/(1-\epsilon)$ approximation to the optimum solution.

Thus we arrive at our main result:

Theorem 5.6 *The local search procedure with operations* add, open and close *yields a locally optimum solution that is a (5+ϵ)-approximation to the optimum solution.*

5.4 Experimental study for non-uniform CFLP

In this section, we show that the algorithm performs well in practice. The experiments were performed for the non-uniform CFLP on the data sets used in earlier studies [CST91, Aar98, BC05, ABSV09].

5.4.1 Data sets used

We performed experiments on three types of random instances which have been used in earlier studies: *type A*, *type B* and *type C*. To construct problem instances of type A, we used the procedure as in [CST91, Aar98, BC05, ABSV09] which is as follows:

1. For a problem instance of size $n \times m$, where n is the number of facilities and m is the number of clients, points are generated uniformly at random in a unit square to represent these many facilities and clients.

2. Euclidean distances are computed between every point representing a facility say i and every point representing a client say j and multiplied by 10.

3. Demands for each client are generated from interval [5,35] uniformly at random i.e. from $U[5, 35]$.

4. Capacity for a facility i is generated from interval [10,160] uniformly at random.

5. Facility costs are computed to reflect economies of scale using the formula $f_i = U[0, 90] + U[100, 110]\sqrt{s_j}$

Problem instances of type B are constructed by multiplying the Euclidean distances, computed in step 2 of the above procedure, by 100 and for type C instances these distances are multiplied by 1000. Rest of the steps remain same for these two instance types. For the problem instances of type A, facility cost component of a solution dominates the cost of the solution. For problem instances of type C, its the service cost component that dominates the cost. Type B instances are somewhere in between the two types of instances.

We give our computational experience for instances of sizes 50×50, 100×100 and 200×200. For these instances we computed optimal solution using LINGO 13 optimization software from LINDO Systems, Inc. and we give the % error i.e. percentage by which the locally optimal solution differs from the optimum solution, thereby giving the quality of the solution produced by the algorithm. Tables 5.4.1, 5.4.1 and 5.4.1 provide these results.

5-approx for non-uniform CFLP			
Filename	Optimum	Local Opt	%error
ndat10_50	10272.66	11456.7	11.53
ndat11_50	10840.81	11830.3	9.13
ndat12_50	11404.18	11526.8	1.08
ndat13_50	10182.61	10840	6.46
ndat14_50	11018.23	12073.2	9.57
ndat10_100	19762.46	20665.2	4.57
ndat11_100	20731.4	21837.9	5.34
ndat12_100	21465.68	23270.5	8.41
ndat13_100	20152.34	21484.6	6.61
ndat14_100	20209.24	21990	8.81
ndat10_200	36288	38395.1	5.81
ndat11_200	38902.9	40748.9	4.75
ndat12_200	38215	41317.6	8.12
ndat13_200	41318.7	43718.6	5.81
ndat14_200	37783.4	40162.6	6.30

Table 5.1: Results for type A instances

These experiments show that the algorithm provides solutions within $(1 + 0.12)$ factor of the optimal solution for the instances tested. These experiments are performed mainly to study the quality of solution. We observe that for a problem of a given size, instances of type A take largest amount of time as compared to the time required by an instance of the same size but of type B or type C to reach a locally optimal solution.

5-approx for non-uniform CFLP			
Filename	Optimum	Local Opt	%error
ndat20_50	22068.7	23036.2	4.38
ndat21_50	20444.49	20892.1	2.19
ndat22_50	25399.5	26620.2	4.81
ndat23_50	22472.35	23623.5	5.12
ndat24_50	23746.76	25109.6	5.74
ndat20_100	38044.36	39431.5	3.65
ndat21_100	35951.7	37258.9	3.64
ndat22_100	38028.82	39462.6	3.77
ndat23_100	36253.68	38050	4.95
ndat24_100	38746.52	39661.8	2.36
ndat20_200	65204	69723.2	6.93
ndat21_200	65706	68270.7	3.90
ndat22_200	65176	68475.1	5.06
ndat23_200	58552.4	61072.5	4.30
ndat24_200	60108.4	62017.7	3.18

Table 5.2: Results for type B instances

5-approx for non-uniform CFLP			
Filename	Optimum	Local Opt	%error
ndat30_50	92713.86	93907.9	1.29
ndat31_50	100713	104694	3.95
ndat32_50	111647	113486	1.65
ndat33_50	92403.3	94981.5	2.79
ndat34_50	99659.35	100759	1.10
ndat30_100	167878	170408	1.51
ndat31_100	178072.3	186263	4.60
ndat32_100	149689.6	154382	3.13
ndat33_100	143649.4	145917	1.58
ndat34_100	141551.7	145525	2.81
ndat30_200	261738	270211	3.24
ndat31_200	226732	230567	1.69
ndat32_200	230017	236346	2.75
ndat33_200	222940	228150	2.34
ndat34_200	222948	232892	4.46

Table 5.3: Results for type C instances

Chapter 6

Conclusion

In this work we presented local search based approximation results for three variants of facility location problem. The first problem we discussed is uniform capacitated facility location problem for which we analyzed the local search heuristic of Kuehn and Hamburger. We showed that the algorithm is (3+ϵ)-factor approximation and also provided a tight example to show that the analysis cannot be strengthened any further. Thus, new operations would be required to improve the approximation factor using local search paradigm.

For non-uniform capacitated facility location problem, we improved the *open, close* operations of Pal *et al.*and *multi* operation of Zhang *et al.*to obtain *mopen, mclose* and *mmulti* operations. It was shown that the cost of the local optimal is no more than (5+ϵ) times the cost of the optimum solution. An example was presented to show that the analysis is tight for the algorithm.

For universal facility location problem, we extended our ideas developed for the second problem and presented a (5+ϵ)-factor approximation algorithm which improved the current best result. The arguments given for the problem suggested that we can obtain (5+ϵ) factor for (non-uniform) CFL without *mmulti* operation as well. We performed an experimental study of the algorithm for the particular case of non-uniform capacities which showed that in practice the results are much closer to the optimum solution.

For the results presented in the thesis we consider the case when demand of a client $j \in C$ is one, i.e. $d_j = 1$. Arbitrary demands can be easily handled by the algorithms preseted in chapters 4 and 5 by doing slight modifications, details of which can be found in Pal *et al.* [PTW01] (for non-uniform capacitated facility location problem) and Mahdian *et al.* [MP03] (for universal facility location problem).

For the problems discussed in the thesis, no constant approximation algorithm based on LP is known. The only known LP-based approximation algorithm for capacitated facility location problem is for the special case when all *facility opening costs are equal* by Levi, Shmoys and Swamy [LSS12]. They gave a 5-factor algorithm for this restricted version of the problem. While the local search algorithms for these problems are not difficult to specify, the analysis, even for the case of uniform capacities, can be quite involved. It would be interesting to explore other, non-local-search approaches to these facility location problems.

Bibliography

[AAB$^+$10] Ankit Aggarwal, L. 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*, pages 149–162. Springer-Verlag, 2010.

[AAK99] Susanne Albers, Sanjeev Arora, and Sanjeev Khanna. Page replacement for general caching problems. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) Baltimore, Maryland*, pages 31–40, 1999.

[Aar98] Karen Aardal. Capacitated facility location: separation algorithms and computational experience. *Journal of Mathematical Programming*, 81(2):149–175, 1998.

[ABSV09] Pasquale Avella, Maurizio Boccia, Antonio Sforza, and Igor Vasilev. An effective heuristic for large-scale capacitated facility location problems. *Journal of Heuristics*, 15(6):597–615, 2009.

[AFS13] Sara Ahmadian, Zachary Friggstad, and Chaitanya Swamy. Local-search based approximation algorithms for mobile facility location problems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Dis-*

crete Algorithms (SODA), New Orleans, Louisiana, USA, pages 1607–1621, 2013.

[AGK$^+$01] 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.

[AGK$^+$04] 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.

[AKR95] Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[ALB$^+$13] Ankit Aggarwal, Anand Louis, Manisha Bansal, Naveen Garg, Neelima Gupta, Shubham Gupta, and Surabhi Jain. A 3-approximation algorithm for the facility location problem with uniform capacities. *Journal of Mathematical Programming*, 141(1-2):527–547, 2013.

[Ali94] Paola Alimonti. New local search approximation techniques for maximum generalized satisfiability problems. In *Proceedings of Second Italian Conference on Algorithms and Complexity, Rome, Italy*, pages 40–53. Springer, 1994.

[Ang06] Eric Angel. A survey of approximation results for local search algorithms. In *Efficient Approximation and Online Algorithms*, pages 30–73. Springer, 2006.

[ATR13] 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.

[AZ02] Matthew Andrews and Lisa Zhang. Approximation algorithms for access network design. *Journal of Algorithmica*, 34(2):197–215, 2002.

[BC05] Francisco Barahona and FabiáN A Chudak. Near-optimal solutions to large-scale facility location problems. *Journal of Discrete Optimization*, 2(1):35–50, 2005.

[BGG12] Manisha Bansal, Naveen Garg, and Neelima Gupta. A 5-approximation for capacitated facility location. In *Proceedings of 20th Annual European Symposium on Algorithms (ESA), Ljubljana, Slovenia*, pages 133–144. Springer-Verlag, 2012.

[BGRS10] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. An improved lp-based approximation for steiner tree. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 583–592, 2010.

[BH12] MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Assignment problem in content distribution networks: Unsplittable hard-capacitated facility location. *ACM Transactions on Algorithms*, 8(3):20, 2012.

[BS89] Robert G Bland and David F Shallcross. Large travelling salesman problems arising from experiments in x-ray crystallography: A preliminary report on computation. *Journal of Operations Research Letters*, 8(3):125 – 128, 1989.

[BSS10] Jaroslaw Byrka, Aravind Srinivasan, and Chaitanya Swamy. Fault-tolerant facility location: a randomized dependent lp-rounding algorithm. In *Proceedings of Integer Programming and Combinatorial Optimization (IPCO), Lausanne, Switzerland*, pages 244–257. Springer, 2010.

[BSSS13] Paul Bogdan, Thomas Sauerwald, Alexandre Stauffer, and He Sun. Balls into bins via local search. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans, Louisiana, USA*, pages 16–34, 2013.

[Byr07] Jaroslaw Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of 10th International Workshop on Approximation and 11th International Workshop on Randomization, and Combinatorial Optimization, (RANDOM-APPROX), Princeton, NJ, USA*, pages 29–43. Springer-Verlag, 2007.

[CCGG98] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: deterministic approximation algorithms for group steiner trees and k-median. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 114–123, 1998.

[CEK06] Chandra Chekuri, Guy Even, and Guy Kortsarz. A greedy approximation algorithm for the group steiner problem. *Journal of Discrete Applied Mathematics*, 154(1):15–34, 2006.

[CG99] 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.

[CG05] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

[Cha00] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. *Journal of Approximation Algorithms for Combinatorial Optimization*, pages 139–152, 2000.

[Chv79] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Journal of Mathematics of Operations Research*, 4(3):233–235, 1979.

[CKT99] Barun Chandra, Howard J. Karloff, and Craig A. Tovey. New results on the old k-opt algorithm for the traveling salesman problem. *SIAM Journal of Computing*, 28(6):1998–2029, 1999.

[CL12] Moses Charikar and Shi Li. A dependent lp-rounding approach for the k-median problem. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I, ICALP (1)*, pages 194–205, 2012.

[CM13] Ioannis Caragiannis and Gianpiero Monaco. A 6/5-approximation algorithm for the maximum 3-cover problem. *Journal of Combinatorial Optimization*, 25(1):60–77, 2013.

[Cro58] GA Croes. A method for solving traveling-salesman problems. *Journal of Operations Research*, 6(6):791–812, 1958.

[CS03] Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal of Computing*, 33(1):1–25, 2003.

[CST91] Gerard Cornuejols, R Sridharan, and Jean-Michel Thizy. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research*, 50(3):280–297, 1991.

[CW99] Fabián A. Chudak and David P. Williamson. Improved approximation algorithms for capacitated facility location problems. In *Proceedings of 7th International Conference on Integer Programming and Combinatorial Optimization (IPCO), Graz, Austria,*, pages 99–113, 1999.

[CW05] 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.

[CW09] Tom Coleman and Anthony Wirth. Ranking tournaments: Local search and a new algorithm. *ACM Journal of Experimental Algorithmics*, 14:6:2.6–6:2.22, 2009.

[DGK$^+$05] Nikhil Devanur, Naveen Garg, Rohit Khandekar, Vinayaka Pandit, Amin Saberi, and Vijay Vazirani. Price of anarchy, locality gap, and a network service provider game. In *Proceedings of First International Workshop on Internet and Network Economics, Hong Kong, China*, pages 1046–1055. Springer, 2005.

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[FNSW11] Moran Feldman, Joseph Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems - (extended abstract). In *Proceedings of 19th Annual European Symposium on Algorithms (ESA), Saarbrücken, Germany*, pages 784–798, 2011.

[FW12] Yuval Filmus and Justin Ward. The power of local search: Maximum coverage over a matroid. In *Proceedings of 29th International Symposium on Theoretical Aspects of Computer Science (STACS), Paris, France*, pages 601–612, 2012.

[GK99] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[GT08] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[Hel00] Keld Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106 – 130, 2000.

[HJC08] Mohammad Khairul Hasan, Hyunwoo Jung, and Kyung-Yong Chwa. Approximation algorithms for connected facility location problems. *Journal of Combinatorial Optimization*, 16(2):155–172, 2008.

[HKK10] MohammadTaghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Budgeted red-blue median and its generalizations. In *Proceedings of 18th Annual European Symposium on Algorithms (ESA), Liverpool, UK*, pages 314–325, 2010.

[HKK12] M Hajiaghayi, R Khandekar, and G Kortsarz. Local search algorithms for the red-blue median problem. *Journal of Algorithmica*, 63(4):795–814, 2012.

[Hoc82a] Dorit S Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

[Hoc82b] Dorit S Hochbaum. Heuristics for the fixed cost median problem. *Journal of Mathematical Programming*, 22(1):148–162, 1982.

[JMS02] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC), Montréal, Québec, Canada*, pages 731–740, 2002.

[Joh73] David S Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC), Austin, Texas, USA*, pages 38–49, 1973.

[JPY85] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? (extended abstract). In *Proceedings of 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA*, pages 39–42, 1985.

[JV01] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, March 2001.

[KG11] Daniel Karapetyan and Gregory Gutin. Lin–kernighan heuristic adaptations for the generalized traveling salesman problem. *European Journal of Operational Research*, 208(3):221–232, 2011.

[KGR02] Amit Kumar, Anupam Gupta, and Tim Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of The 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 333–342, 2002.

[KH63] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. *Journal of Management Science*, 9(4):pp. 643–666, 1963.

[KL70] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–308, 1970.

[KMN$^+$02] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, pages 10–18, 2002.

[KPR98] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, California*, pages 1–10, 1998.

[KPR00] 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.

[KV05] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Algorithms and Combinatorics. Springer, 2005.

[Li13] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Journal of Information and Computation*, 222:45–58, 2013.

[LK73] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Journal of Operations Research*, 21(2):498–516, March-April 1973.

[Lov75] László Lovász. On the ratio of optimal integral and fractional covers. *Journal of Discrete Mathematics*, 13(4):383–390, 1975.

[LSS12] Retsef Levi, David B. Shmoys, and Chaitanya Swamy. Lp-based approximation algorithms for capacitated facility location. *Journal of Mathematical Programming*, 131(1-2):365–379, 2012.

[MMSV01] Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay Vazirani. A greedy facility location algorithm analyzed using dual fitting. In *Proceedings of 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and 5th International Workshop on Randomization and Computation (RANDOM-APPROX) Berkeley, CA, USA*, pages 127–137. Springer, 2001.

[MP03] 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.

[MYZ02] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Improved approximation algorithms for metric facility location problems. In *Proceedings of 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX), Rome, Italy*, pages 229–242. Springer, 2002.

[PTW01] 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.

[SC99] D Shmoys and FA Chudak. Improved approximation algorithms for capacitated facility location problems. In *Proceedings of 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Baltimore, Maryland*, pages S875–S876, 1999.

[SK04] Chaitanya Swamy and Amit Kumar. Primal–dual algorithms for connected facility location problems. *Journal of Algorithmica*, 40(4):245–269, 2004.

[SS02] Andreas S Schulz and Martin Skutella. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics,* 15(4):450–469, 2002.

[STA97] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA,*, pages 265–274, 1997.

[SV07] Petra Schuurman and Tjark Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *INFORMS Journal on Computing*, 19(1):52–63, 2007.

[Svi02] Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of 9th International Conference on Integer Programming and Combinatorial Optimization, Cambridge, MA, USA*, pages 240–257, 2002.

[SW13] Maxim Sviridenko and Justin Ward. Large neighborhood local search for the maximum set packing problem. In *Proceedings of 40th International Colloquium on Automata, Languages, and Programming (ICALP) , Riga, Latvia*, pages 792–803, 2013.

[Vyg07] Jens Vygen. From stars to comets: Improved local search for universal facility location. *Journal of Operations Research Letters*, 35(4):427–433, 2007.

[Wil02] David P. Williamson. The primal-dual method for approximation algorithms. *Journal of Mathematical Programming*, 91(3):447–478, 2002.

[XX05] 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.

[Yan90] Mihalis Yannakakis. The analysis of local search problems and their heuristics. In *Proceedings of 7th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Rouen, France*, pages 298–311, 1990.

[ZCY05] 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.

[Zha07] Peng Zhang. A new approximation algorithm for the k-facility location problem. *Journal of Theoretical Computer Science*, 384(1):126 – 135, 2007.

Appendix A

A.1 Path decomposition of graph G

Path decomposition of graph G can be performed as follows:

Take a graph G' which is initialised to G. Now perform the following steps:

1. If G' has cycles, pick an arbitrary cycle C in G' and add it to $\mathcal{C}$.
2. Drop the edges that occured in C from G'.
3. Repeat steps 1 and 2 until there is any cycle left in G' else go to the next step.
4. Let i be an arbitrary node in G' with atleast one outgoing edge and no incoming edges. Let P be a maximal path which begins at i.
5. Add P to $\mathcal{P}$and remove the edges in P from G'.
6. Continue with steps 4 and 5 until graph G' has no edges.

A.2 Removing cycles from the graph

Lemma A.1 *The exchange graph G, whose vertices are the set of facilities in the locally optimal solution S and the facilities in the optimum solution O, when modified to make it acyclic continue to satisfy the following three properties:*

1. $\sum_{s\in S, o\in O} c_{so} y(s,o) \leq c_s(S) + c_s(O)$.

2. G *is a bipartite graph with* S *and* O *defining the sets of the partition.*

3. $\forall s \in S, \sum_{o\in O} y(s,o) = |N_S(s)|$ *and* $\forall o \in O, \sum_{s\in S} y(s,o) = |N_O(o)|$.

Proof

1. Note that whenever a cycle is removed, total cost $\sum_{s\in S,o\in O} c_{so} y(s,o)$ does not increase. Therefore first property holds when procedure terminates.

2. Property 2 is trivially satisfied.

3. Consider a cycle C and its two partitions C_1 and C_2 as described. Consider a vertex $s \in S$ which lies on C and let $e_1 \in C_1$ and $e_2 \in C_2$ be two edges incident with s. Let o_1 (respectively o_2) be the other vertex incident with e_1 (respectively e_2). Note that $o_1, o_2 \in O$. Initially

$$\sum_{o\in O-\{o_1,o_2\}} y(s,o) + y(s,o_1) + y(s,o_2) = |N_S(s)|$$

Suppose, by increasing the value of edges in C_1 and decreasing the value of edges in C_2 by an amount ϵ, the cycle C is removed. Then the edge e_1 has an increased value $y(s,o_1) + \epsilon$ and the edge e_2 has a reduced value $y(s,o_2) - \epsilon$. Therefore

$$\begin{aligned}
&\sum_{o\in O-\{o_1,o_2\}} y(s,o) + (y(s,o_1) + \epsilon) + (y(s,o_2) + \epsilon) \\
&= \sum_{o\in O-\{o_1,o_2\}} y(s,o) + y(s,o_1) + y(s,o_2) \\
&= |N_S(s)|
\end{aligned}$$

In the same manner we can argue that $\sum_{s\in S} y(s,o) = |N_O(o)|$ even with the modified values of edges. With this we have proved that property 2 still holds.

■