Approximation Algorithms in Network Design

Submitted in partial fulfilment of the requirements for the degree of

Integrated Masters of Technology in Mathematics and Computing

Author:

Piyush Ahuja

Advised by:

Dr. Naveen Garg

Dr. Amitabha Tripathi

Department of Mathematics

Indian Institute of Technology Delhi

August 2013

Declaration of Authorship

I, Piyush Ahuja, declare that this thesis titled, 'Approximation Algorithms in Network Design' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

"The universe ....which stands continually open to our gaze, cannot be understood unless one first learns to comprehend the language and interpret the characters in which it is written. It is written in the language of mathematics, and its characters are triangles, circles, and other geometrical figures, without which it is humanly impossible to understand a single word of it; without these, one is wandering around in a dark labyrinth."

Galileo Galilei, The Assayer

INDIAN INSTITUTE OF TECHNOLOGY DELHI

Abstract

Department of Mathematics

Integrated Masters of Technology in Mathematics and Computing

Approximation Algorithms in Network Design

by Piyush Ahuja

Problems combining facility location with connectivity requirements are fundamental in network design. In this thesis, we investigate the Connected Facility Location (CFL) problem through the non-trivial special case of Single-Sink Rent-or Buy (SRoB) problem. We develop two approaches for this problem, and illustrate the limitations and promises of both. Our first approach is based on a natural linear program for SRoB. Till date, no algorithm based on this linear program is known. We sketch an algorithm which performs well in many instances of the problem, but runs into trouble for certain pathological cases, which we illustrate. Our second approach borrows ideas from the dual fitting algorithm for metric uncapacitated facility location by Jain et al. (called JMS algorithm), and combines it with the Goemans-Williamson moat growing procedure. This approach aims to neutralize the slack in the Goemans-Williamson argument with the one associated with the JMS procedure, and thus holds a lot of promise for achieving an overall 2-approximation for CFL. Apart from these two approaches, we investigate the scope for improvement in the 4.55- approximation algorithm due to Swamy and Kumar, which gives the current best primal-dual based performance guarantee for SRoB. We illustrate a very simple instance where the algorithm produces a 3-approximation, thus giving a lower bound on the performance guarantee for that algorithm.

Acknowledgements

First of all, I'd like to convey a big THANK YOU to Professor Naveen Garg. It has been a real privilege having him as a mentor and getting to work with him. He has this magnificient way of making everything look simple, clean and elegant, and inspiring the same clarity of thought in you, which is hard to describe in words.

I'm grateful to Professor Amitabha Tripathi, for going out of his way to help me. Without his guidance and support this research would not have been possible.

I'd also like to thank Syamantak Das, for listening to all my ideas, and always patiently explaining why some of them don't work.

Contents

List of Figures

Abbreviations

CFL	**C**onnected **F**acility **L**ocation
SRoB	**S**ingle-sink **R**ent **o**r **B**uy
UFL	**U**ncapacitated **F**acility **L**ocation

Dedicated to Mom, Dad, and my sister, Richa

Algorithms, and The Magic of Mathematics

In his 1960 article *The Unreasonable Effectiveness on Mathematics in the Natural Sciences*, Eugene Wigner expresses wonder at the powerful applicability of mathematical concepts, especially in contexts far beyond those that had initially motivated their development. He says,

> *"It is difficult to avoid the impression that a miracle confronts us here, quite comparable in its striking nature to the miracle that the human mind can string a thousand arguments together without getting itself into contradictions, or to the two miracles of laws of nature and of the human mind's capacity to divine them."*

It indeed is a marvel that the same relationship between variables (mathematical labels) which describes the fall of a stone also predicts the motions of stars and planets.[1] That the constant which describes the ratio of a circle's circumference to its diameter also appears in cosmology, statistics, fractals, thermodynamics, mechanics, and electromagnetism and concepts which have nothing to do with a circle.[2] That abstract structures motivated by the problem of traversing a city through bridges find powerful application in telecommunication, linguistics, sociology, chemistry and biology.[3] That a neat, finite, mathematically grounded sequence of operations can dissect the workings of the real world, and solve problems of seemingly colossal complexity.

This miraculous two-way conversation between mathematics and the real world has intensified in our increasingly complex and intricate society. Today, the world is more

[1] Newton's law was not only based on scanty observations, but also contained the physically non-intuitive idea of the second derivative.

[2] Wigner gives an anecdote of two friends, one of whom, a statistician, was working on population distribution. When the statistician explained the symbol π occurring in a particular distribution, the friend, who presumably was not a mathematician, thought it was a joke and said, "surely the population has nothing to do with the circumference of a circle".

[3] Graph theory has been used to form semantic networks of word meanings, in measuring actors' prestige, in exploring social diffusion mechanisms and breeding patterns, in tracking the spread of diseases and parasites, in forming natural models for molecules (representing atoms as nodes and bonds as edges), and in network flows, among many other things.

connected than ever, data and information is exploding, while the natural and technological resources we have at our disposal stay limited. Algorithmic and computational techniques are now central to decision-making, in planning, scheduling, allocation of resources, traffic routing, data organisation[4] and in fields as diverse as Healthcare, Manufacturing, IT, Telecommunications, Agriculture, Finance, Service and Retail, Military, Mining, Shipping and Transportation, and Waste Management. In the other direction, the study of these problems, and the associated complexity and heuristics, has inspired some very deep and beautiful mathematical results over the years, which are, in turn, applicable to a much larger class of problems.

Exposing the underlying mathematical structures of real world problems brings us closer to the very 'language of the universe'. There is something aesthetically pleasing in using 'the language of the universe' to solve real world problems, and in using real world problems to further our understanding of this language. Paul Lockhart, in *A Mathematician's Lament,* refers to a certain mystical feeling, associated with such pursuits:

> *"The thing I want you especially to understand is this feeling of divine revelation. I feel that this structure was "out there" all along I just couldn't see it. And now I can! This is really what keeps me in the math game– the chance that I might glimpse some kind of secret underlying truth, some sort of message from the gods."*

At its very core, this thesis is motivated by this feeling of divine revelation.

[4]Personnel scheduling, facility planning, resource management, process design, forecasting, portfolio management, personal consumption management, pricing, and inventory management....The list is endless.

CHAPTER 1

Introduction

1.1 Network Design and Approximation Algorithms

Many interesting real world scenarios can be modelled through weighted graphs called networks. For example, networks are used to represent transport system (such as highways and mass-transit systems), telecommunication, electrical distribution, fluid flow in pipes, flow of nutrients and energy between different organizations in a food web, among many other processes. Different applications judge the merit of a network in different ways and differing cost criteria. For instance, consider the problem of finding the shortest route from a City A to City B. The scenario can be modelled as a graph with cities as nodes, the connections between cities as edges, and the length of the connections as the weights. Given City A (node s) and City B (node t), this turns into the problem of finding the shortest path between these nodes. The shortest path corresponds to the optimal network in this case.

Unfortunately, most problems in the optimal design of networks are challenging from a computational point of view, which is to say that no polynomial time algorithm for them is known.[1] In the above example, if instead of asking for the shortest path, we wanted to find the longest path between s and t which repeats no vertices, the problem becomes computationally hard. What should be done in such a case?

One way out is to relax the optimality criterion, and design algorithms which are efficient and produce solutions which are near-optimal, over all problem instances.[2] This trade-off

[1] Or even exists, if one is to believe $\mathcal{P} \neq \mathcal{NP}$. For a better understanding of this notion, see Appendix A.

[2] It should be noted that some problems don't even have approximation algorithms. There's a whole class of problems for which designing a good approximation algorithm would imply $\mathcal{P} = \mathcal{NP}$

between optimality and computational tractability is the principle behind approximation algorithms.

The study of these computationally hard problems through approximation algorithms has been very valuable. The push to develop good heuristics often results in a deeper mathematical understanding of the problem's structure, computational properties and new approaches (sometimes for subcases), which leads to better algorithms for solving hundreds of other different but related hard problems.

The problems studied in this thesis falls in the ambit of Connected Facility Location, which combines problems of facility location with problems of connection. The problem arises frequently in telecommunication network infrastructure. It combines two classical problems in combinatorial optimization, and contains an important special case:

- The Steiner tree problem, in which we look for the shortest interconnect for a given set of objects
- Facility Location, in which we look for the optimal placement of facilities
- Single Sink Rent-or Buy, in which we design a network under economies of scale

1.2 Literature Review

The design and analysis of algorithms in combinatorial optimisation has been heavily influenced by Linear programming, particularly for problems which can be naturally formulated as integer programs. Most of the fundamental algorithms in combinatorial optimisation either use this method or can be understood in terms of it, including Dijkstra's shortest path algorithm[3, chap. 7], Ford and Fulkerson's network flow algorithm, Edmonds' non-bipartite matching algorithm and Kuhn's assignment algorithm[4, pg. 145].

The primal dual method has especially flourished in the area of network design. One of the first such algorithms was a moat growing procedure by Agarwal, Klein and Ravi[5] for the generalised Steiner problem on networks, which achieves an approximation factor of 2. Goemans and Williamson generalise the method and apply it to a large class of problems (see [4, 6] for a survey of this). Notably, they develop a 2-approximation algorithm for the Prize Collecting Steiner tree(PCST) problem on the same lines[3, chap. 14]. There is a slack in the Goemans-Williamson argument arising from the observation that we do not need the contribution from one of the active leaf components to get a bound of two. This slack is exploited by Naveen Garg[7] to obtain a 2-approximation

for the k-MST problem. We note that this slack might be useful in improving the ratio of many network design problems that contain elements of PCST.

The primal dual based methods have also been used to obtain good approximation algorithms for metric uncapacitated facility location (UFL). Jain and Vazirani[8] use the technique to achieve an approximation factor of 3. Their algorithm uses the observation that a maximal dual solution corresponds to a feasible set of facilities, and an economically suitable subset of facilities can be chosen from that by forming clusters of demands and facilities. Jain et al.[9] later give two greedy algorithms for UFL, with approximation ratios of 1.861 and 1.61 respectively. Both these algorithms are inspired by the primal dual type algorithm traditionally used for set cover (called dual fitting), and are analyzed using the star formulation for UFL given by Balinsky[10]. In the first of these algorithms, when a client gets connected to an open facility, it withdraws whatever it has contributed towards the opening cost of other facilities. This step of withdrawing contribution ensures that the primal solution is fully paid for by the dual. The second algorithm has a minor difference with the first one: A client might change the facility to which it is connected and connect to a closer facility. If so, it offers this difference toward opening the latter facility. We briefly review these algorithms in Chapter 4.

The problems considered in this thesis, CFL and SROB, have also recently received considerable attention, both in theoretical computer science literature and operations research literature. The current best known results for them are due to Eisenbrand et al.[11], who give a 4-approximation for CFL and a 2.92 approximation for SRoB. They improve the analyses of the Sample-Augment-like algorithm initially given in [12] through a novel method called core-detouring. If we consider primal–dual approaches to CFL, the most significant result is by Swamy and Kumar[13]. Their algorithm combines a modification of the JV algorithm[8] for UFL with the moat growing procedure of [5], achieving an 8.55-approximation ratio for CFL and 4.55 for SRoB (the current best). Their algorithm achieves partial economic viability by opening those facilities where enough local demand can be gathered at a minimum cost . In making this approximation, their algorithm doesn't fully incorporate the overall connectivity requirements in their decision as to which facilities to open. We examine this algorithm in Chapter 5.

1.3 Our Contributions

In this thesis, we investigate the Connected Facility Location problem through the non-trivial special case of Single-Sink Rent-or Buy (SRoB) problem. We develop two approaches for this problem, and illustrate the limitations and promises of both. Apart

from that, we construct an instance of SRoB where the Swamy-Kumar primal dual algorithm gives a 3-approximation, thus giving a lower bound on the performance guarantee of that algorithm.

1. Our first approach (Chapter 3) is based on a natural Linear Program for SRoB. Till date, no algorithm based on this linear program is known. We sketch an algorithm which performs well in many instances of the problem, but runs into trouble for certain pathological cases. A class of these instances can be fixed by replacing edges with bidirectional arcs. The integrality gap for the linear program, and possible modifications to the algorithm which take care of *all* the pathological cases, present interesting open problems for future work.

2. Our second approach(Chapter 6) borrows ideas from the second dual fitting algorithm for UFL (called JMS algorithm)[9] and combines it with the moat growing procedure for PCST[3, chap. 14][6]. This approach aims to neutralize the slack in the Goemans-Williamson PCST argument with the one associated with that associated with the JMS procedure, and thus holds a lot of promise for obtaining an overall 2-approximation for CFL.

3. We investigate (Chapter 5) the scope for improvement in the 4.55- approximation algorithm due to Swamy and Kumar[13], which gives the current best performance guarantee for SRoB through a primal dual method. The algorithm combines the JV Algorithm for UFL[3, 8] with the moat growing procedure for the Steiner tree[3–5]. We illustrate a very simple instance where the algorithm produces a 3-approximation, thus giving a lower bound on the performance guarantee for that algorithm.

1.4 A Roadmap for the Thesis

Chapter 2 gives an overview of a few preliminary concepts which will make it easier to navigate through the thesis. Chapter 3 introduces the Single Sink Rent-or Buy (SRoB) problem and develops a fresh new approach to solving it, based on a natural linear program. Chapter 4 introduces the more general Connected Facility location problem (CFL). The primal dual approaches to the classical problems of Steiner tree and Facility location which CFL combines are also sketched in this chapter. Chapter 5 examines the Swamy Kumar 4.55-approximation for SRoB. Chapter 6 describes a possible new primal dual approach for solving CFL, which combines the dual fitting Facility location algorithm[9] with the moat growing procedure for Prize Collecting Steiner Tree[3–6].

CHAPTER 2

Preliminaries

2.1 NP Completeness

There are problems in the class $\mathcal{NP}$ that are representative of the entire class, in the sense that if they have polynomial-time algorithms, then $\mathcal{P} = \mathcal{NP}$, and if they do not, then $\mathcal{P} \neq \mathcal{NP}$. These are the $\mathcal{NP} - complete$ problems.

Definition 2.1. (**NP-completeness**) A problem B is $\mathcal{NP} - complete$ if B is in $\mathcal{NP}$, and for every problem A in $\mathcal{NP}$, there is a polynomial-time reduction from A to B.

A polynomial-time solution for any $\mathcal{NP} - complete$ problem would imply one for every problem in $\mathcal{NP}$. The term $\mathcal{NP} - hard$ is usually applied to optimization problems whose corresponding decision problems are $\mathcal{NP} - complete$. For a better understanding of these notions, see Appenfix A, or refer [3, Appendix A].

2.2 Approximation Ratio

A $\rho - approximation$ algorithm for an optimization problem is a polynomial time algorithm that for all instances of the problem produces a solution whose value is within a factor of ρ of the value of an optimal solution.

Definition 2.2. (**ρ-approximation algorithm**) Given an optimization problem $\prod$, an algorithm $\mathcal{A}$ a polynomial time approximation algorithm if it runs in polynomial time (see Appendix A) and returns a solution to the problem "close" to the optimal. If the problem is a minimization problem, then the algorithm $\mathcal{A}$ is called the a $\rho() - factor$ approximation algorithm, where ρ: $\mathbb{R} \to \mathbb{R}$ is a function taking reals to reals, if for any

instance I of the optimization problem $\prod$, the solution returned by the algorithm $\mathcal{A}$ is guaranteed to have cost

$$\mathcal{A}(I) \leq \rho(|I|).OPT(I)$$

where $OPT(I)$ is the value of the optimum solution to I and $|I|$ is the size of the representation of the instance I. Note that $\rho(|I|) \geq 1$ as one cannot possibly do better than the optimum. Furthermore $\mathcal{A}$ runs in time bounded by a polynomial in $|I|$. Similarly, if $\prod$ were a maximization problem, the definition would be the same with the inequality reversed, and $\rho()$ would be less than 1.

For a $\rho - approximation$ algorithm, we will call ρ the approximation factor or performance guarantee of the algorithm.

2.3 Linear Programming

One way in which to find an algorithm $\mathcal{A}$ which is a $\rho - approximation$ algorithm is to find a bound on $OPT(I)$ (a lower bound if $\prod$ is a minimization problem) in polynomial time. An extremely elegant method of obtaining better (efficiently computable) bounds is to model the combinatorial optimization problem as an integer program and then use the solution of its linear programming relaxation as the bound. This procedure will become clearer when we apply it to SRoB in Chapter 3.

2.3.1 Approximation via Primal Dual Method

In the Primal Dual method, we use both the LP relaxation of an integer program and its dual formulation to construct a good solution. The algorithm starts off with a feasible dual solution and a corresponding infeasible primal solution. We then use the feedback from the primal solution to increase our dual solution in such a way that the primal complementary slackness conditions are satisfied. We continue this procedure till we construct a feasible primal solution whose total cost is within a factor ρ of the cost of the feasible dual solution. Since the dual solution provides a lower bound to the optimal cost, the cost of the constructed solution is within ρ times the optimal value.

2.3.2 Approximation via Dual Fitting

The method of dual fitting can be described as follows, assuming a minimization problem: The basic algorithm is combinatorial. Using the LP relaxation of the problem and its dual, we first come up with an algorithm that iteratively makes primal and dual updates.

These updates are made in such a way that the primal integral solution found by the algorithm is fully paid for by the dual computed, i.e., the objective function value of the primal solution is bounded by that of the dual. In this process, the dual solution computed is, in general, infeasible.[1] The main step in the analysis consists of dividing the dual by a suitable factor, say ρ, and showing that the shrunk dual is feasible, that is, it fits into the given instance. The shrunk dual is then a lower bound on OPT, and ρ is the approximation guarantee of the algorithm.

One of the advantages of primal-dual type algorithms is that one does not need to solve any LP . By considering the dual, we reduce a weighted optimization problem to to a purely combinatorial, unweighted problem. However on the flip-side, these algorithms are hard to develop since they typically use only "local information" and designing such algorithms require considerable amount of finesse. Also, these methods provide deep insights we have into the structure of numerous fundamental combinatorial problems.[2]

2.4 Limitations of Linear Programming

The limitations on obtaining good bounds on the optimal value for an optimization problem $\prod$ through LP relaxation of an integer programs is captured by following definition of the integrality gap:

Definition 2.3. (**Integrality Gap**) Given a minimization problem $\prod$ and given an LP-relaxation $LP_{\prod}$ for the problem, the Integrality Gap $I.G$, of the LP-relaxation is defined as

$$I.G.(LP_{\prod}) = \sup_{I} \frac{OPT_{\prod}(I)}{LP_{\prod}(I)}$$

where the supremum is taken over all instances I of the minimization problem. For maximization problems, the integrality gap is defined similarly with the supremum replaced by an infimum.

Given any minimization problem $\prod$ and an LP-relaxation $LP_{\prod}$ for it, the best approximation one can prove using the LP-optimum as a lower bound is at least $I.G.(LP_{\prod})$.

[1] It is here that the algorithm departs from the primal-dual method in the strictest sense

[2] We refer the reader to a guest post by Vijay Vazirani, titled "Seeking Combinatorial Algorithms for Convex Programs" on the blog *Turing's Invisible Hand*. 12 January 2010

CHAPTER 3

A Primal Dual Algorithm for the Single Sink Rent-or Buy problem

3.1 Problem Description

In the single-sink rent-or-buy problem, we are given an undirected graph $G = (V, E)$, non-negative costs (also called lengths) c_e defined on every edge $e \in E$ (c: $E \to \mathbb{R}^+$), a root vertex $r \in V$, a set of clients $X \subseteq V$, and a parameter $M > 1$.[1] In this scenario, we need to design a network connecting all clients to the root; one in which for each client we can specify a path of edges from the client to the root. We say that a client *uses* an edge if the edge is on the client's path to the root. To build the paths, we can both *buy* and *rent* edges. We can buy edges at cost Mc_e, and once bought, any client can use the edge. We can also rent edges at cost c_e, but then we need to pay the rental cost for each client using the edge. The goal is to find a feasible network that minimizes the total cost (of buying and renting edges.) We can formalize the setting by letting $B \subseteq E$ be the set of edges that are bought, and letting R_j be the set of edges that are rented by client $j \in X$. Then for each $j \in X$, the set of edges $B \cup R_j$ must contain a path from j to the root r. Let $c(F) = \sum_{e \in F} c_e$ for any $F \subseteq E$. Then the total cost of the solution is $Mc(B) + \sum_{j \in X} c(R_j)$. We must find edges B to buy and R_j to rent (for each j) that minimizes this overall cost.

[1] In some formulations, we have demands d_j associated with each client $j \in X$ (d: $X \to \mathbb{Q}^+$). Here we assume that every client $j \in X$ has a unit demand, $d_j = 1, \forall j \in X$. This assumption is without loss of generality, as we may replace j by several copies of co-located unit-demand clients.

3.2 Applications

The Rent-or-Buy Network Design problem captures the"economies of scale" property, which says that the per unit cost of installing capacity on edges of the network decreases as more capacity is installed.

Consider an oil company that wishes to connect a network of pipelines to carry oil from several remote wells (clients) to a major refinery (sink). The company can install capacities on the pipelines in two ways: Capacity can be rented, with cost incurred on a per unit of capacity basis, or bought, which allows unlimited use after payment of a large fixed cost. The goal is to design a minimum cost network that would be sufficient to transport the oil to the refinery, assuming fixed oil supply at each well. This is exactly the SRoB problem. It is not difficult to see that SRoB manifests in a number of different situations - in online settings (network leasing), in telecommunication infrastructure, in VLSI design, and so on.

3.3 Previous Literature

SRoB frequently arises in literature as a special case of the Buy-at-bulk network design, as well as Connected Facility Location, which we'll trace in Chapter 4.

The Buy-at-bulk network design is defined in terms of installing cables on edges, with different cable types offering different amounts of capacity and carrying different costs. Andrews and Zhang[15] show that this problem can be rephrased (with a loss of a small constant factor in the approximation ratio) with each cable type carrying a fixed cost (which must be paid irrespective of the capacity needed) and an incremental cost (which is paid for each unit of capacity required). SRoB therefore corresponds to the special case of one cable type with an incremental cost but no fixed cost, one cable type with a fixed cost but no incremental cost, and where all source-sink pairs share a common sink.

Karger and Minkoff[16] gave the first constant approximation algorithm for SRoB in their study of the so-called Maybecast problem, which is the probabilistic version of the Steiner Tree problem. Although the SRoB problem is not explicitly mentioned in [16], the Maybecast problem they studied is actually a variant of the SRoB problem. The approximation ratio is improved to 9 by LP-rounding[17], to 4.55 by primal-dual schema[13], to 3.55 by Sample-Augment with analysis via cost-sharing[12], and finally to 2.92 by Sample-Augment with analysis via core-detouring[11]. The last two are approximation ratios in expectation. For an overview of these, see survey by Zhang[14].

The best primal dual based algorithm is 4.55-approximation is given by Swamy and Kumar[13], which we examine in Chapter 5.

3.4 Integer Programming Formulation

SRoB can be formulated naturally as an Integer Program.
We need to decide which edges to buy (for all), and we need to decide which edges to rent for each client. We create *decision variables* tp represent this choice: let integer variable b_e represent the 'bought' status for edge e (whether edge e has been bought or not), and let integer variable $r_{e,j}$ represent the 'rented' status with respect to edge e and client j (whether edge e has been rented for client j i.e. client j it uses the edge on its path as a rented edge).

$$b_e = \begin{cases} 1 & \text{if } b_e \in B \\ 0 & \text{if } b_e \notin B \end{cases} \qquad r_{e,j} = \begin{cases} 1 & \text{if } r_{e,j} \in R_j \\ 0 & \text{if } r_{e,j} \notin R_j \end{cases}$$

To reflect the above nature of the decision variables, we constrain the decision variables to non-negative integers : $b_e, r_{e,j} \in \{0, 1\}$. We need to enforce that the decision variables represent a feasible solution. i.e. a network in which all the clients are connected to the root. From every *cut* that separates a client j from the root, the client must *use* atleast one edge. We formalize this notion. Given a non-empty set of vertices $S \subset V$, let $\delta(S)$ denote the set of edges in the cut defined by S; that is, $\delta(S)$ is the set of all edges with exactly one endpoint in S. Let S_j be the subsets of vertices which contain j but not r; that is, $S_j = S \subseteq V : j \in S, r \notin S$. We introduce the following constraint:

$$\sum_{e \in \delta(S)} (b_e + r_{e,j}) \geq 1$$

for each j and each $S \in S_j$. We want to find a feasible network of minimum cost. Given the decision variables, and the constraints described above, the cost of the network is $M(\sum_{e \in E} c_e b_e) + \sum_{j \in X} \sum_{e \in E} c_e r_{e,j}$. Thus, we model the Single-Sink Rent-or-Buy problem with the following integer program:

$$\begin{aligned} \text{minimize } & M \sum_{e \in E} c_e b_e + \sum_{j \in X} \sum_{e \in E} c_e r_{e,j} \\ \text{subject to } & \sum_{e \in \delta(S)} (b_e + r_{e,j}) \geq 1 && , \forall j \text{ and } \forall S \in S_j \\ & b_e, r_{e,j} \in \{0, 1\} && , \forall e \in E \text{ and } \forall j \in X \end{aligned}$$

Claim. The above integer program models the single-sink rent-or-buy problem.

Proof. Take any feasible solution and pick any client j. Let $P_j = \{e \in E : b_e = 1 \text{ or } r_{e,j} = 1\}$. The constraints ensure that for any $j - r$ cut S, there must be at least one edge of P_j in $\delta(S)$: that is, the size of the minimum $j - r$ cut in $G_j = (V, P_j)$ must be at least one. Thus, by the max-flow/min-cut theorem the maximum $j - r$ flow in G_j is at least one, which implies that there is a path from j to r in the graph. Similarly, if a solution is not feasible, then there is some j for which there exists a $j - r$ cut S such that there are no edges of P_j in $\delta(S)$, which implies that the size of minimum $j - r$ cut is zero, and thus the maximum $j - r$ flow is zero. Hence, there is a path from each $j \in X$ to r if and only if these constraints are satisfied, and if they are all satisfied, there will be a graph connecting all the clients to the root corresponding to the decision variables which have value one. Thus, the solution space of the integer program and of the single-sink rent or buy problem are the same (upto an isomorphism), and so is their cost function.

3.5 Linear Programming Relaxation

Unless $\mathcal{P} = \mathcal{NP}$ the above integer program can't be solved in polynomial time. SRoB is $\mathcal{NP} - hard$, and solving the integer program for any SRoB input would amount to $\mathcal{P} = \mathcal{NP}$. However, if we change the integer program to a linear program by relaxing the integer conditions, we will be able to derive useful information, since linear programs can be solved in polynomial time.

We replace the constraints $b_e, r_{e,j} \in \{0, 1\}$ with the constraints $b_e, r_{e,j} \geq 0$.[2]

Claim. The linear program thus obtained (given in Section 3.5.1 below) is a relaxation of the original integer program.

Proof. To prove that its a relaxation, two things need to be shown: every feasible solution for the original integer program (Section 3.4) is feasible for this linear program; and second, the value of any feasible solution for the integer program has the same value in the linear program.

Any solution for the integer program will certainly satisfy all the constraints of the linear program.

$$b_e, r_{e,j} \in \{0, 1\} \Rightarrow \ b_e, r_{e,j} \geq 0, \forall e \in E \text{ and } \forall j \in X$$

[2]We could also add the constraints $b_e, r_{e,j} \leq 1$, but they would be redundant: in any optimal solution to the problem, we can reduce $b_e, r_{e,j} > 1$ to $b_e, r_{e,j} = 1$ without affecting the feasibility and without increasing the cost.

Furthermore, the objective functions of both the integer and linear programs are the same, so that any feasible solution for the integer program has the same value for the linear program.

Let Z_{LP} denote the optimum value of this linear program, and Z_{IP} denote the optimal value of the integer program. Since optimal solution to the integer program is feasible for the linear program, $Z_{LP} \leq Z_{IP} = OPT$ (since this minimization linear program finds a feasible solution of lowest possible value). We'd try to construct a feasible solution to the integer program in polynomial time, and use the above fact to obtain good bounds on its value.

3.5.1 Primal Problem

$$\begin{aligned} \text{minimize } & M \sum_{e \in E} c_e b_e + \sum_{j \in X} \sum_{e \in E} c_e r_{e,j} \\ \text{subject to } & \sum_{e \in \delta(S)} (b_e + r_{e,j}) \geq 1 && \forall j \text{ and } \forall S \in S_j \\ & b_e,\, r_{e,j} \geq 0 && \forall e \in E \text{ and } \forall j \in X \end{aligned}$$

3.5.2 Dual Problem

We'll motivate the dual formulation from an appeal to intuition, though it can be arrived by purely analytic procedures. In SRoB, we need to construct paths from each client to the root. The cost involved is the cost of connecting each client to the root, i.e. the cost of constructing these paths. We can attribute these costs explicitly to the edges which comprise these paths, as in the primal formulation. Or we can charge these costs in a more implicit way, by attributing it to satisfying the connection requirements of each client. The latter allows us to arrive at a lower bound by eliminating the edge costs, and charging the clients instead. For each client j, we define costs $\theta_{S,j}$ - the cost of satisfying the cut corresponding to sets $S \in S_j$. The client only incurs this cost if it actually uses an edge from the cut, i.e, if $S \in S_j$. Intuitively, we are charging a portion of costs of the edges from each client and cut. The total contribution of all clients towards satisfying their connection requirements shouldn't be greater than the price Mc_e of an edge which does the same (satisfies the same connection requirements). Hence for each edge $e \in E$:

$$\sum_{j \in X} \sum_{\substack{S \in S_j \\ S:\ e \in \delta(S)}} \theta_{s,j} \leq M c_e$$

Also, the contribution of a particular client should't be greater than the renting cost, because renting by itself is sufficient to satisfy the connectivity requirement, which implies:

$$\sum_{S \in S_j,\, S:\, e \in \delta(S)} \theta_{s,j} \leq c_e$$

Any contribution from the clients following the above constraints would give a lower bound to the optimal solution. Since we are interested in the sharpest lower bound, we maximise it:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in X} \sum_{S \in S_j} \theta_{S,j} \\
\text{subject to} \quad & \sum_{\substack{S \in S_j \\ S:\, e \in \delta(S)}} \theta_{S,j} \leq c_e && \forall j \in X \text{ and } \forall e \in E \\
& \sum_{j \in X} \sum_{\substack{S \in S_j \\ S:\, e \in \delta(S)}} \theta_{S,j} \leq M c_e && \forall e \in E \\
& \theta_{S,j} \geq 0 && \forall j \in X \text{ and } \forall S \in S_j
\end{aligned}$$

A common way to visualize these costs is by thinking of them as 'moats' around sets which separate a client from the root[18] (see Figure 3.1). These moats can overlap each other, as long as the total sum of moats on a particular edge doesn't exceed the price Mc_e of the edge.

3.6 A Primal Dual Approximation Algorithm

In the primal dual technique, we construct a feasible solution to the dual of the LP and a corresponding approximate solution to the primal (a solution satisfying the primary complementary slackness conditions). We prove the performance guarantee by comparing values of both the solutions. This can be done, as both the solutions have a mathematical relation to each other (provided by complementary slackness); moreover, the cost of the dual solution provides a lower bound for the optimal solution of the primal.

3.6.1 The Algorithm

We construct a moat packing, drawing inspiration from similar algorithms in [3–5, 7, 18]

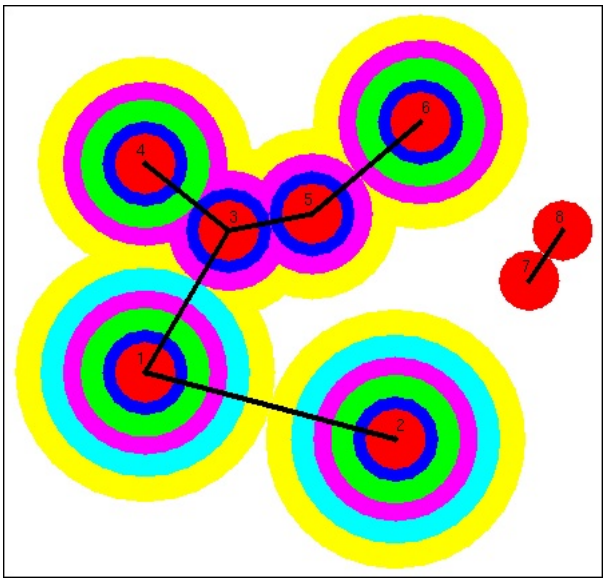

FIGURE 3.1: The variables $\theta_{S,j}$ can be visualized as the width of moats around the sets S. The above figure shows one such visualization, in the context of the Goemans-Williamson procedure[4], where the moats are not allowed to overlap. In our dual formulation, the moats can overlap, but in a constrained manner, i.e., as long as they satisfy certain constraints.

For each client j, we maintain a *reachability set* V_j, and a *tentatively rented set* R_j. We also maintain a *tentatively bought set* B and a notion of time t. At $t = 0$ (initially):

$$\begin{aligned} V_j &= \{j\} && \forall j \in X \\ R_j &= \phi && \forall j \in X \\ B &= \phi \\ \theta_{S,j} &= 0 && \forall j \in X \text{ and } \forall S \in S_j \end{aligned}$$

We start raising the j dual variables corresponding to $\theta_{V_j,j}$ at the same rate, until one of the dual constraints becomes tight. If the dual constraint corresponds a 'renting constraint', for edge e and a particular client j, then we add e to the tentatively rented set R_j and update the reachability set V_j to include the vertices of the edge e. If the dual constraint corresponds to a 'buying constraint' for an edge e, then we update the reachability the of *all* clients j to include the vertices of edge e and add e to the tentatively bought set B. As soon as there is a path from client j to r in $(V_j, R_j \cup B)$, we stop raising the dual variables corresponding to client j. We continue the algorithm until all the clients $j \in X$ are connected to root r in this way.

Discussion 3.1. Our algorithm tries to simultaneously constructs paths from each client to the root by raising the appropriate dual variables. But such an algorithm is profligate, as there are many edges which are tentatively rented and bought but which

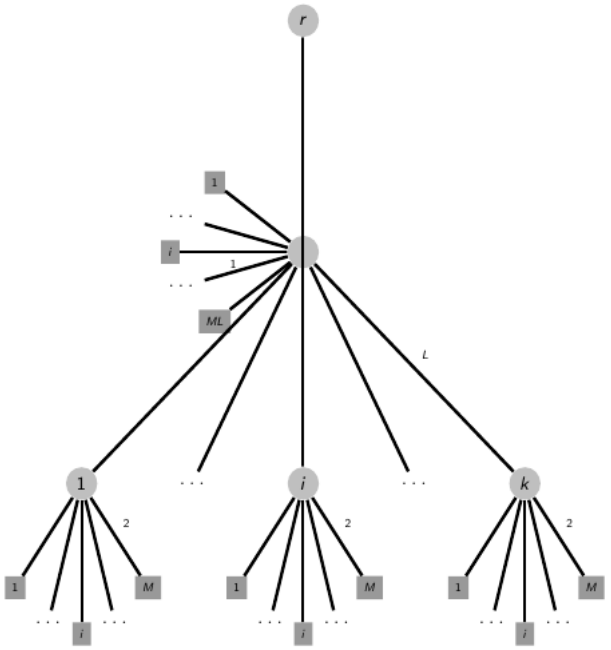

FIGURE 3.2: At $t = 0$, all dual variables are zero, and no paths have been constructed. The ϵ length is just for illustrative purposes and might as well be ignored.

we don't need. We'll need to prune our solution in a second phase, as done in similar moat growing algorithms for network design[5, 6][3, chap. 7].

Unfortunately, we'll see that the algorithm doesn't provide a very good lower bound to the optimal solution over all instances of the SRoB problem, even when we restrict it to the class of trees.

3.6.2 Difficulties

3.6.2.1 Tree Pathological Example

Consider a run of the algorithm on the example given in Figure 3.2.

The dual variables corresponding to the group of ML clients cover the cost of the paths of many other clients. This happens because the paths are growing in all directions, and in the example given the ML clients buy the k edges of length L which are of use only to the other clients, before those clients get a chance to pay for them. The lower bound found by the algorithm becomes weaker and weaker as values of k and L become larger.

Proposition 3.2. *For any constant ρ, we can find an instance of SRoB such that the optimal value is greater than ρ times the value of the dual returned by the algorithm.*

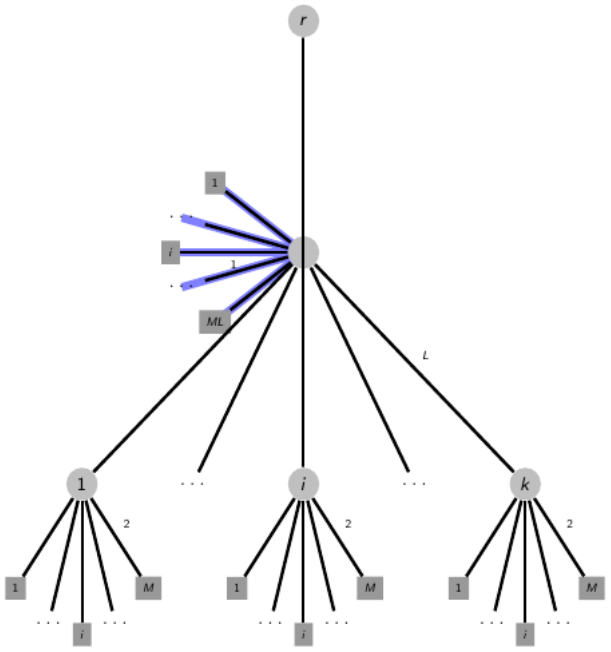

FIGURE 3.3: At $t = 1$. The blue edges correspond to tentatively rented edges.

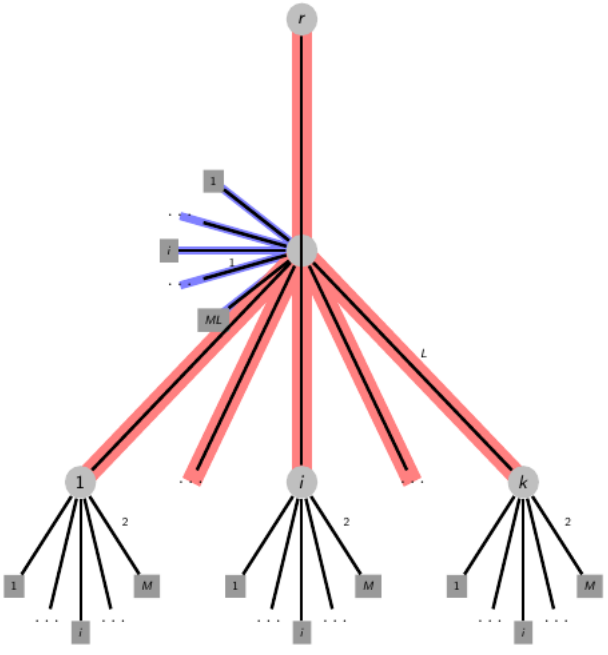

FIGURE 3.4: The red edges correspond to tentatively bought edges. At $t = 2$, the paths of ML clients merge and they buy lots of edges in a short period of time.

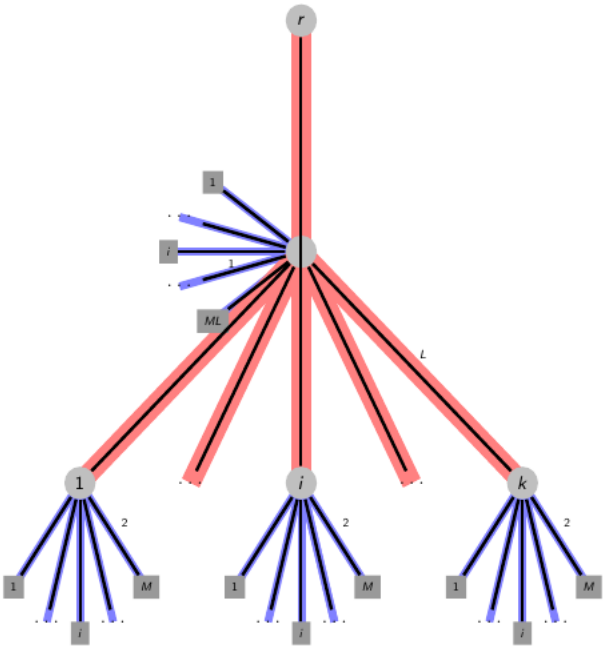

FIGURE 3.5: By the time the path of rest of the clients merge forming k clusters of M clients each (at $t = 2$), their paths have already been constructed by the ML group of clients.

Proof. The optimal solution in the given instance of SRoB is the same as shown in Figure 3.5. The cost of the optimal solution is given by:

$$\begin{aligned} I_{OPT} =& M * (k+1) * L + ML * 1 + k * M * 2 \\ =& \quad (ML * 2 + KM * 2) + M * k * L \end{aligned} \tag{3.1}$$

The value of the dual solution produced by the algorithm is:

$$D_{ALG} = ML * 2 + kM * 2 \tag{3.2}$$

The gap between the value of the optimal and our dual lower bound grows arbitrarily large with k and L. Dividing the quantities in (3.1) and (3.2):

$$\begin{aligned} \frac{I_{OPT}}{D_{ALG}} =& 1 + \frac{k * L}{2 * (k + L)} \\ =& \ 1 + \frac{kL}{2 * (k + L)} \\ =& 1 + \frac{1}{2 * (\frac{1}{L} + \frac{1}{k})} \ \rightarrow \ \infty \text{ as } k, L \rightarrow \infty \end{aligned}$$

□

3.7 Modifications

3.7.1 The bidirected cut relaxation

In the pathological example in Figure 3.6, the dual variables corresponding to the group of ML clients cover the cost of the paths of many other clients. This happens because the paths are growing in all directions, and in the example given the ML clients buy k edges of length L away from the root as well. On the other hand, these edges are on the path to the root for other clients. How do we take care of such cases?

An interesting observation (which we prove in Section 3.9) is that in the optimal solution all the clients that *use* an edge do it only in a particular direction. In the solution, the orientation of the edge in the paths P_j from the client to the root is the same for all clients j. By a similar argument we can say that the final solution to our algorithm would be better off if all the clients use an edge in the same direction. This suggests that we should recoup the cost of the edges direction-wise (from the dual variables). In doing so, situations such as the tree pathological example in Figure 3.6 would be fixed.

We formalize this notion by using the bidirected cut relaxation for our LP. We view every edge as two oppositely directed arcs, each with the same length as the edge. More formally, we replace each undirected edge $e = uv$ with two directed arcs $(u, \vec{v})$ and $(v, \vec{u})$, each of cost c_e. For any set S, let $\delta^+(S)$ be the set of arcs that leave S. Call the set of arcs $\vec{E}$. Let the sets S_j be defined as before (set which contain the client j but not the root r). The observation now is out of all the arcs that leave a set S which contains the client but not the root, there should be either a bought edge or an edge which is rented for client j.

The bidirected cut relaxation is just the directed version of the LP relaxation given above, so there are double the variables $b_e, r_{e,j}$ (one for each arc) and a constraint for every set that contains one client from X but not the root. This relaxation on the lines of a similar bidirected cut relaxation for the Steiner tree problem, first given by Edmonds[19].

3.7.1.1 Primal Problem

$$\begin{aligned}
\text{minimize } & M \sum_{e \in \vec{E}} c_e b_e + \sum_{j \in X} \sum_{e \in \vec{E}} c_e r_{e,j} \\
\text{subject to } & \sum_{e \in \delta^+(S)} (b_e + r_{e,j}) \geq 1 && \forall j \text{ and } \forall S \in S_j \\
& b_e,\, r_{e,j} \geq 0 && \forall e \in \vec{E} \text{ and } \forall j \in X
\end{aligned}$$

3.7.1.2 Dual Problem

The dual formulation is obtained similarly (with double the constraints, each constraint replaced by one for each arc)

$$\begin{aligned}
\text{maximize } & \sum_{j \in X} \sum_{S \in S_j} \theta_{S,j} \\
\text{subject to } & \sum_{\substack{S \in S_j \\ S:\, e \in \delta^+(S)}} \theta_{S,j} \leq c_e && \forall j \in X \text{ and } \forall e \in \vec{E} \\
& \sum_{j \in X} \sum_{\substack{S \in S_j \\ S:\, e \in \delta^+(S)}} \theta_{S,j} \leq M c_e && \forall e \in \vec{E} \\
& \theta_{S,j} \geq 0 && \forall j \in X \text{ and } \forall S \in S_j
\end{aligned}$$

The bidirected cut relaxation solves the problem for trees. Unfortunately, there still exist pathological cases (in the class of general graphs) where our algorithm (with the BCR relaxation) performs arbitrarily bad (Figure 3.6).

3.7.2 More Difficulties

3.7.2.1 General Graph Pathological Example

Consider a run of the algorithm on the example given in Figure 3.6.

Proposition 3.3. *For any constant ρ, we can find an instance of SRoB where the optimal value is greater than ρ times the value of the dual returned by the BCR modified algorithm (Figure 3.12).*

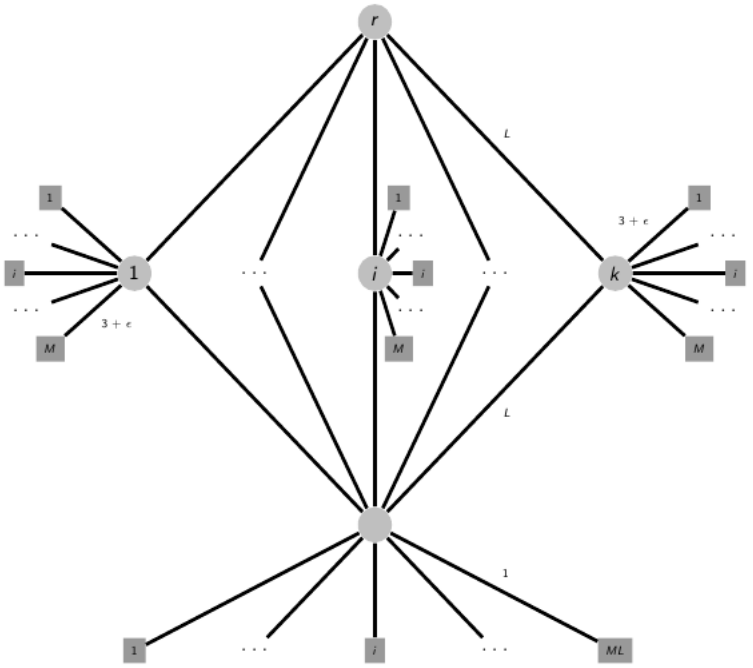

FIGURE 3.6: At $t = 0$, all dual variables are zero, and no paths have been constructed. Note: The ϵ length is just for illustrative purposes and might as well be ignored. The algorithm actually runs in the instance where each edge is replaced by two oppositely directed arcs, but it doesn't make any difference to the visualization in this case.

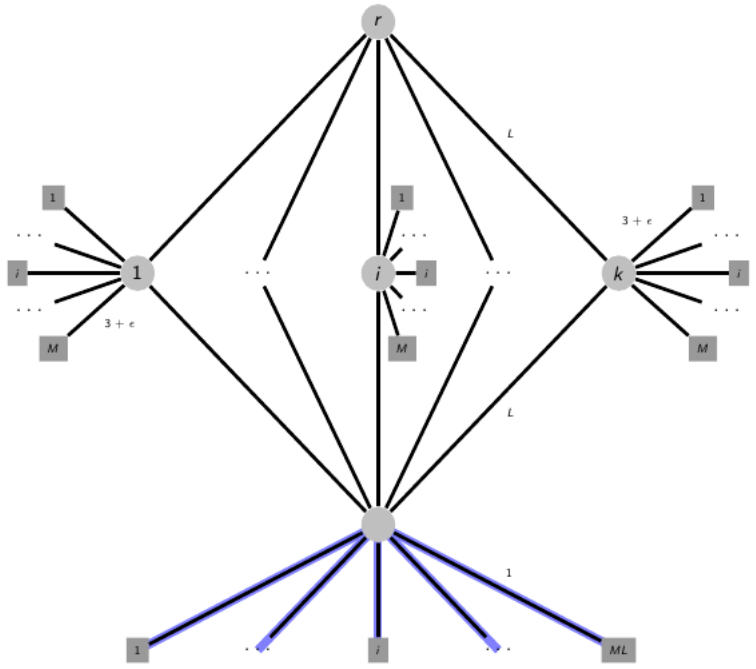

FIGURE 3.7: At $t = 1$. The blue edges correspond to tentatively rented edges.

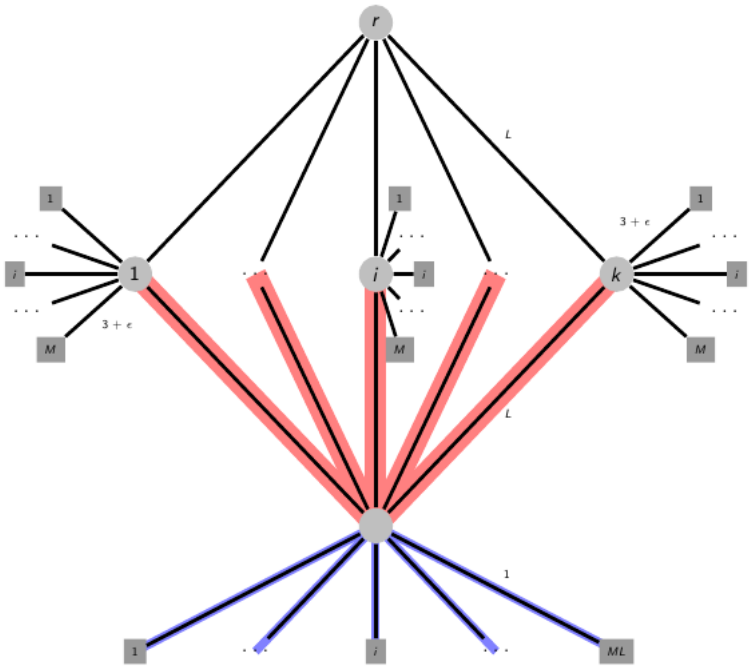

FIGURE 3.8: At $t = 2$, the paths of ML clients merge and they buy lots of edges in a short period of time.

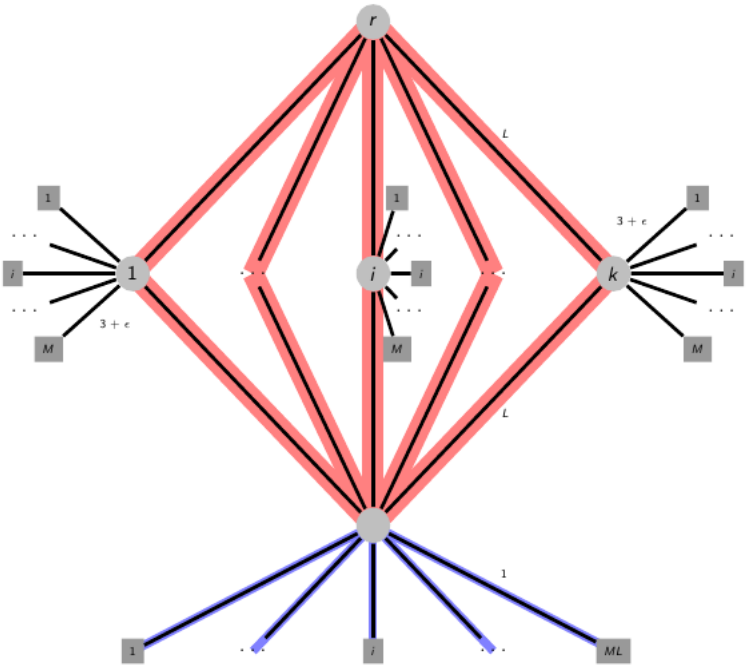

FIGURE 3.9: At $t = 3$, ML clients reach the root.

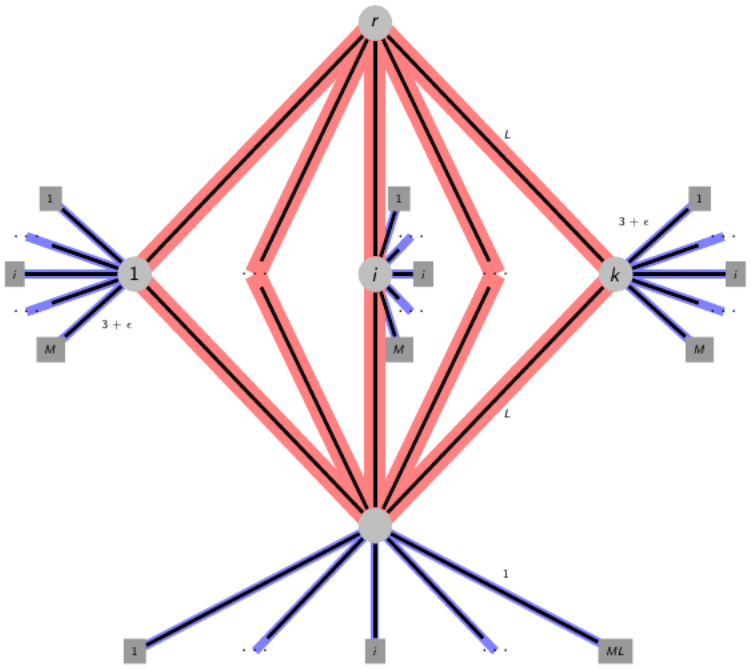

FIGURE 3.10: The path of rest of the clients merge and form k clusters of M clients each.

Proof. The optimal solution in the above instance of SRoB is not hard to visualize (Figure 3.11). The cost of the optimal solution is given by:

$$\begin{aligned} I_{OPT} &= M * (k+1) * L + ML * 1 + k * M * 3 \\ &= (ML * 3 + KM * 3) + M * (k-1) * L \end{aligned}$$

The value of the dual solution produced by the algorithm is:

$$D_{ALG} = ML * 3 + kM * 3$$

The lower bound found by the algorithm becomes weaker and weaker as values of k and L become larger.

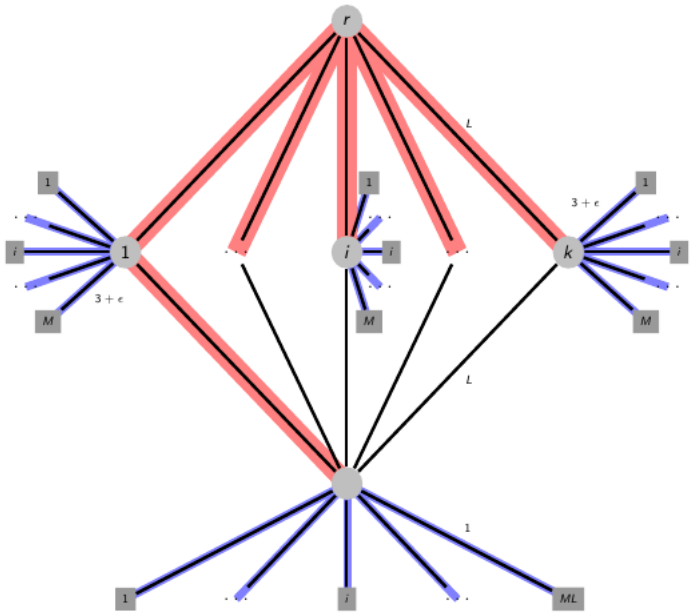

FIGURE 3.11: An optimal solution. Due to the symmetry properties, there are several different optimal solutions, all having the same value.

$$\begin{aligned}\frac{I_{OPT}}{D_{ALG}} &= 1 + \frac{(k-1)*L}{3*(k+L)} \\ &\approx 1 + \frac{kL}{3*(k+L)} \\ &= 1 + \frac{1}{3*(\frac{1}{L}+\frac{1}{k})} \rightarrow \infty \text{ as } k, L \rightarrow \infty\end{aligned}$$

□

3.8 Limitations of LP relaxation

Recall that the integrality gap $I.G.$ of an integer program is the worst-case ratio (over all instances of the problem) of the value of an optimal solution to the integer programming formulation to value of an optimal solution to its linear programming relaxation. Thus, over all problem instances, $I_{OPT} \leq I.G. * LP_{OPT}$, and this is the sharpest upper bound that possible using the linear program. We aim to design an algorithm which provably achieves a constant performance guarantee. For this we'll need to show that the value of its solution I_{ALG} is atmost a constant ρ times the value of the linear programming

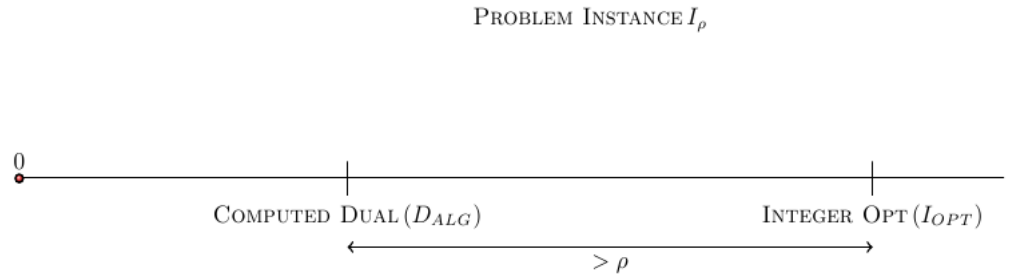

Figure 3.12: For any constant ρ, we can find an instance I_ρ of SRoB where the optimal value is greater than ρ times the value of the dual returned by the algorithm. As an example, in the instance in Figure 3.6 with $L = 1000$, the gap between the optimal value and the value of the dual solution computed by the algorithm can be made ≈ 100 if we take more than 430 branches ($k > 430$)

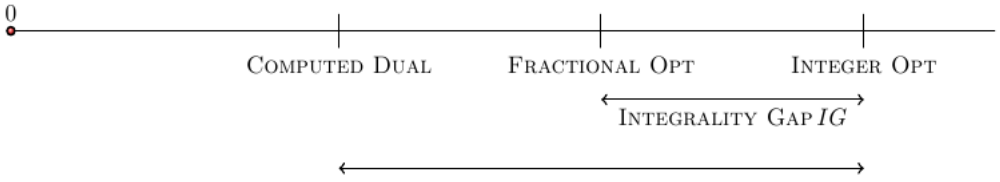

Figure 3.13

relaxation, for all problem instances. Such a ρ would always be lower bounded by the integrality gap IG. Thus, over all problem instances:

$$I_{OPT} \leq I_{ALG} \leq \alpha * LP_{OPT}, \text{ which implies } \alpha \geq I.G.$$

Unfortunately, the integrality gap of the above integer program(with BCR relaxation) is not yet known, and we do not know how close the fractional solution is to the optimal integer solution, in the general case. The LP contains the bidirected cut relaxation for the Steiner tree as a special case ($M = 1$), whose integrality gap is upper bounded by 2[3], which follows for the gap for the undirected relaxation[5]. We believe that the gap for the formulation we use is also close to 2, and finding the actual gap presents an interesting open problem.

[3]It is conjectured to be close to 1

3.9 Proofs

Proposition 3.4. *Let $P = \{P_j : j \in X\}$ represent the set of paths from client j to root r ($P_j \subseteq B \cup R_j$) in an optimal network. If an edge $e = uv \in P_i, P_j$ for two distinct clients $i, j \in X$, such that u precedes v in P_i, then u precedes v in P_j.*

Proof. Assume, for the sake of contradiction, that v precedes u in P_j. Let $R_j(u, r)$ be the set of rented edges that occur in the path for client j from node u to root r; $R_j(u, r) = \{e' \in P_j \cap R_j : e \text{ precedes } e' \text{ in } P_j\}$. Define $R_i(v, r)$ similarly. If the total renting cost for j on the path from node u to root r is greater than (or equal to) that for client i (from v to r), then we can lower the cost by removing e and renting those edges for j instead (the other case is handled similarly). Formally, if $c(R_i(v, r)) \leq c(R_j(u, r))$, then we can replace P_j with $P_j \cup R_i(v, r) - R_j(u, r) - \{e\}$. This would give a feasible solution having strictly lower cost. Hence the contradiction.

□

3.10 Conclusion and Future Work

In the pathological cases, the paths of many clients agglomerate (merge) and they end up buying many edges, which they don't use, from the paths of other clients. One way out is to 'withdraw' the contribution of these agglomerated clients from the edges that they don't eventually use. We would have to argue that resulting infeasible dual stays within a factor of a feasible dual. This procedure is similar to the first dual fitting algorithm mentioned in Chapter 4 Section 4.5.3.2, and would allow us to distribute the costs among all the clients more evenly. It is not entirely clear that such a procedure could be applied to our algorithm, but we believe that there's a scope for a fix in such a direction.

It might even be the case that our algorithm performs arbitrarily badly due to pathological cases that do not arise in practice. There's good value in examining the algorithm further, as it might give rise to heuristics that return solutions much closer to optimal than indicated by its performance guarantees.

CHAPTER 4

Connected Facility Location

4.1 The Problem

Given (i) an undirected graph $G = (V, E)$, (ii) non-negative costs (also called lengths) c_e defined on every edge $e \in E$ ($c\colon E \to \mathbb{Q}^+$), (iii) a set of facilities $\mathcal{F} \subseteq V$ with non-negative costs $f_i \in \mathbb{Q}^+$ defined on every facility $i \in \mathcal{F}$, (iv) a set of clients $X \subseteq V$, and (v) a parameter $M > 1$, determine[1]: 1. A subset $F \subseteq F$ of the facilities to be opened
2. An assignment $\sigma\colon X \to F$, from each client $j \in X$ to some open facility $\sigma(j) \in F$
3. A Steiner tree T connecting the open facilities F such as to minimize the total cost:

$$\sum_{i \in F} f_i + M \sum_{e \in T} c_e + \sum_{j \in X} l(j, \sigma(j)),$$

where $l(v, w)$ is the shortest distance between vertices $v, w \in V$ in G (with respect to c).

4.2 Applications

As a practical example, consider the problem of installing a telecommunication network infrastructure (see [11, 16]). The network consists of a central high-bandwidth core with unlimited capacity on the links and individual connections from endnodes to nodes in the core. Among the potential core nodes, we need to select a subset that we connect with each other and then route the traffic from each endnode to a core node. Each core node comes with an installation cost and we assume that the cost of installing the

[1] In some formulations, we have demands d_j associated with each client $j \in X$ ($d\colon X \to \mathbb{Q}^+$). Here we assume that every client $j \in X$ has a unit demand, $d_j = 1, \forall j \in X$. This assumption is without loss of generality, as we may replace j by several copies of co-located unit-demand clients.

high-bandwidth links in the core is larger than the (per unit) routing cost from the end nodes to the core. This problem can be handled as a CFL problem.[2] In other scenarios, facilities could be caches or file servers which need to communicate with each other to maintain consistent data, and the clients could be users or processes requesting data items.

4.3 Previous Literature

CFL has been received considerable attention in both theoretical computer science and operations research literature. Gupta et al. obtain a 10.66-approximation[17], based on rounding an exponential size LP. The ratio is subsequently improved to first 9.01[20] and then to 4[11] via random sampling based algorithms, which is the best known result for CFL till date. The best current primal dual based algorithm is an 8.55-approximation by Swamy and Kumar (SK algorithm)[13](Chapter 5). Their algorithm combines the primal dual based 2-approximation for Steiner trees[5] with the 3-approximation for facility location by Jain and Vazirani[8] (Section 4.5.3.1)

4.4 SRoB special case

Consider a special case of the Connected Facility Location problem: Assume $\mathcal{F} = V$, $f : \mathcal{F} \to \mathbb{Q}^+ = 0$ ($f_i = 0, \forall i \in \mathcal{F}$), and a given vertex r must be contained in the Steiner tree. Consider an optimal solution to this problem. Every client $j \in X$ sends a unit flow to an open facility $\sigma(j)$ through a shortest path from j to $\sigma(j)$, paying cost $l(j, \sigma(j))$. This is equivalent to renting the shortest path. The flow gathered at all open facilities further are sent to vertex r, called the sink, paying cost $M \sum_{e \in T} c_e$. This is equivalent to buying the Steiner tree T (and hence we can send unlimited flow along its edges). Since $f_i = 0 \, \forall i \in F$, the total cost of the solution is the renting cost plus the buying cost. This is just the SRoB problem.

[2]The partial replacement of existing out-of-date copper based networks by modern fiber optic cables can also be handled as a CFL problem. We are given clients, that need to be connected to a central distributor by a tree-shaped network. In commonly used Fiber-To-The-Curb (FTTC) architectures, potential switching locations are given to which the customers may by connected by an existing copper infrastructure. Any choice of switch installations results in a set of terminals that have to be connected to the distributor using new fiber optic technology. The practical objective is to minimize the overall installation costs for cables and switching devices.

4.5 Subproblems

Connected facility location (CFL) combines problems of Facility Location and Steiner Tree. There is an implicit cost imposed by the connectivity requirement on the cost of opening a facility. There are existing 2-approximation primal dual algorithms for both Steiner tree and facility location, and ideas from those would go a long way in helping us design a 2-approximation algorithm for CFL. In the next sections, we review these algorithms.

4.5.1 Steiner Tree

In the Steiner tree problem, we are given an undirected graph $G = (V, E)$, non-negative costs c_e defined on every edge $e \in E$ ($c\colon E \to \mathbb{R}^+$), and a set of clients $X \subseteq V$. The goal is to connect the clients with each other with a tree of minimum cost. The problem can be modelled as the following integer program:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e z_e \\ \text{subject to} \quad & \sum_{e \in \delta(S)} z_e \geq 1 \qquad \forall S \subseteq V : \emptyset \neq S \cap X \neq X \\ & z_e \in \{0, 1\} \quad e \in E \end{aligned}$$

Relaxing the integrality conditions to $z_e \geq 0$, we get the Linear Program. The dual of the program is given by :

$$\begin{aligned} \text{maximize} \quad & \sum_{S \subseteq V\,:\, \emptyset \neq S \cap X \neq X} \theta_S \\ \text{subject to} \quad & \sum_{S\,:\, e \in \delta(S), \emptyset \neq S \cap X \neq X} \theta_S \leq c_e, \forall e \in E \\ & \theta_S \geq 0 \qquad\qquad\qquad\qquad , \forall S \subseteq V : \emptyset \neq S \cap X \neq X \end{aligned}$$

4.5.1.1 AKR Algorithm

In the AKR algorithm for Steiner trees, we raise the dual variables θ_S corresponding to connected components S such that $|S \cap \{r, J\}| = 1$ for some $J \in X$, until some dual inequality associated with some edge $e \in \delta(C)$ becomes tight, and we add this edge to

our primal solution . Once we have a feasible solution such that all clients are connected to the root, we do a clean-up phase where we remove all the unnecessary edges. This algorithm produces a 2-approximation.

4.5.2 Facility Location

In the metric Uncapacitated Facility Location problem, we are given (1) an undirected graph $G = (V, E)$, (2) a set of clients $X \subseteq V$, (3) a set of facilities $\mathcal{F} \subseteq V$ with (4) non-negative *opening* costs $f_i \in \mathbb{Q}^+$ defined on every facility $i \in \mathcal{F}$, and (5) non-negative *connection* costs c_{ij} for assigning client j to demand i ($c\colon X \times \mathcal{F} \to \mathbb{Q}^+$). The costs c_{ij} obey the triangle inequality. The goal is to find a subset $F \subseteq \mathcal{F}$ of the facilities to be opened and an assignment σ of clients to facilities ($\sigma\colon X \to \mathcal{F}$), which minimizes the total cost (the sum of opening and connection costs):

$$\sum_{i \in F} f_i + \sum_{j \in X} c_{\sigma(j),j}$$

The problem can be modelled as an integer program through the Star formulation[10]: A star $K = (i, C)$ consists of one facility and many clients, where facility $i \in \mathcal{F}$ and clients $C \subseteq X$. The cost c_K associated with a star K is $f_i + \sum_{j \in C} c_{ij}$. Let $\mathcal{K}$ be the set of all stars. In the Star formulation, the facility location problem is viewed as a problem of finding a minimum cost set of *stars* such that each client is in atleast one star.

$$\begin{aligned} &\text{minimize } c_K x_K \\ &\text{subject to } \sum_{K=(i,C)\colon j \in C} x_K \geq 1 \ \forall j \in X \\ &\qquad\qquad x_K \in \{0,1\} \qquad\qquad \forall K \in \mathcal{K} \end{aligned}$$

Relaxing the integrality conditions to $x_K \geq 0$, we get the Linear Program. The dual of the program is given by :

$$\begin{aligned} \text{maximize} \sum_{j \in X} \alpha_j & \\ \text{subject to} \sum_{j \in C} \alpha_j \leq c_K \, \forall K = (i, C) \in \mathcal{K} & \\ \alpha_j \geq 0 \qquad\qquad & \forall j \in X \end{aligned}$$

4.5.3 Three Different Algorithms

We review three different primal-dual type algorithms for UFL. The first of these algorithms[8] (called the JV algorithm) is the standard primal dual method (Section 2.3.1), and achieves a 3-approximation. The other two are greedy algorithms[9] and follow the dual fitting methodology (Section 2.3.2).

4.5.3.1 JV Algorithm

The first algorithm (call it the JV algorithm)[8] uses the fact that a maximal dual solution corresponds to a feasible set of facilities, and an economically suitable subset of facilities can be chosen from that by forming *clusters* of demands and facilities.

In the algorithm, we raise the dual variables α_j of all clients simultaneously. At any point of the algorithm, a client contributes $max(\alpha_j - c_{ij}, 0)$ towards the opening of facility i. When the total contribution given to a facility equals its opening cost, we tentatively open the facility and freeze the dual variables of all those facilities which had a non-zero contribution. We continue the process until all the clients have a non-zero contribution to a tentatively open facility. Because a client might contribute to more than one facility, we do a final clean-up phase where we choose only a subset of the tentatively open facilities to open, and then assign clients to the nearest open facility. After the clean-up phase, the cost of the final primal solution is shown to be within a factor of 3 of the dual (here the euclidean distance property is crucial) , and thus the algorithm produces a 3-approximation. This algorithm emulates the standard primal-dual method. We start off with a feasible dual feasible solution and a corresponding infeasible primal solution, and increase the value of the dual in such a way that we obtain a feasible primal solution. We do this by maintaining the primal complementary slackness condition at all times.

4.5.3.2 Dual Fitting

The first dual fitting algorithm[9] is a modification of the JV algorithm. After we tentatively open a facility,all the clients with a non-zero contribution to it are connected to it and their contribution isn't considered for covering the opening costs of other facilities (we say they 'withdraw' their contribution). In doing this, the dual feasibility is compromised. All clients contribute to and are assigned only one facility, and we don't need a clean up phase. Here the value of the primal solution is equal to the dual value (which is infeasible). The final step shows that the dual is within a factor of 1.81 of a feasible dual, thus achieving a 1.861-approximation.

4.5.3.3 JMS Algorithm

The second dual fitting algorithm[9] (called JMS here) has a minor modification from the first one. In the second algorithm a client stops contributing to other facilities once it is connected to a facility, but here it continues to offer some amount to other facilities. The amount is the connection cost it will save by switching to a new facility. This algorithm does slightly better than the second algorithm, producing a 1.61-approximation.

CHAPTER 5

The Swamy Kumar Algorithm

The difficulty in SRoB is due to the tension of "route vs gather". On one hand, we would like to route flow between the clients and the root on a shortest path; on the other, we would like to gather flow from different clients together in order to take advantage of economies of scale. Many approaches have been applied to resolve this tension. The Swamy Kumar algorithm (SK algorithm) makes the tradeoff by first gathering enough demand locally , and then routing the flow from these gathered clusters to the root through a Steiner tree. While they use the primal dual JV algorithm[8] (Section 4.5.3.1) for the gathering phase, for the second phase they use a combination of the primal dual Steiner tree procedure[5] and a dual fitting-type analysis. The algorithm is generalized to CFL with minor modifications, but the principle remains the same. We refer the reader to their paper [13] for a full reading of the algorithm.

5.1 Linear Program

Swamy and Kumar use following integer program[13] for their analysis.[1]

[1]Here it is assumed that we know a facility r that is opened and hence belongs to the Steiner tree in the optimal solution. This assumption doesn't affect anything because all $|\mathcal{F}|$ different possibilities for r can be tried.

$$
\begin{aligned}
\text{minimize } & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{j\in X}\sum_{i\in\mathcal{F}} c_{ij}x_{ij} + M\sum_{e\in E} c_e z_e \\
\text{subject to } & \sum_{i\in\mathcal{F}} x_{ij} \geq 1, && \forall j \in X \\
& x_{ij} \leq y_i, && \forall i \in \mathcal{F} \text{ and } \forall j \in X \\
& \sum_{i\in S} x_{ij} \leq \sum_{e\in\delta(S)} z_e, && \forall S \subseteq V, r \notin S \text{ and } \forall j \in X. \\
& y_r = 1 \\
& x_{ij}, y_i, z_e \in \{0,1\} && \forall i \in \mathcal{F}, \forall j \in X, \forall e \in E
\end{aligned}
$$

5.2 SRoB formulation

If we put $f_i = 0$ and $\mathcal{F} = V$, we get another formulation of SRoB. Relaxing the integrality constraints to $x_{ij}, y_i, z_e \geq 0$ gives us a Linear Program.

5.2.1 Primal Problem

$$
\begin{aligned}
\text{minimize } & \sum_{j\in X}\sum_{i\in V} c_{ij}x_{ij} + M\sum_{e\in E} c_e z_e \\
\text{subject to } & \sum_{i\in V} x_{ij} \geq 1, && \forall j \in X \\
& x_{ij} \leq y_i, && \forall i \in V \text{ and } \forall j \in X \\
& \sum_{i\in S} x_{ij} \leq \sum_{e\in\delta(S)} z_e, && \forall S \subseteq V, r \notin S \text{ and } \forall j \in X. \\
& y_r = 1 \\
& x_{ij}, y_i, z_e \in \{0,1\} && \forall i \in \mathcal{F}, \forall j \in X, \forall e \in E
\end{aligned}
$$

5.2.2 Dual Problem

$$\begin{aligned}
&\text{maximize} \sum_{j \in X} \alpha_j \\
&\text{subject to } \alpha_j \leq c_{ij} + \sum_{S \subseteq V\,:\, i \in S, r \notin S} \theta_{S,j} \quad \forall j \in X \text{ and } \forall i \in V, i \neq r \\
&\qquad \alpha_j \leq c_{rj} \qquad\qquad \forall j \in X \\
&\qquad \sum_{j \in X} \sum_{S \subseteq V\,:\, e \in \delta(S)} \theta_{S,j} \leq M c_e \qquad\qquad \forall e \in E \\
&\qquad \alpha_j, \theta_{S,j} \geq 0 \qquad\qquad \forall j \in X \text{ and } \forall S \subseteq V, r \notin S
\end{aligned}$$

5.2.3 Integrality Gap

Swamy and Kumar give a 4.55-approximation algorithm using the above linear program. Thus, the integrality gap of the LP is atmost 4.55.

5.3 Scope for Improvement

We investigate if the analysis in Kumar-Swamy holds a scope for improvement. The problem instance in Example 5.1 throws light on the inefficiencies which arise in the SK algorithm.

Example 5.1.

Analysis. The algorithm produces a worst case $\approx$ 3-approximation for the problem instance in Figure 5.1[2] :

$$\begin{aligned}
\frac{I_{ALG}}{I_{OPT}} &= \frac{M * (2L + \epsilon) + M) * L}{M * (L + \epsilon) + M\delta} \\
&\rightarrow 3 \qquad\qquad \text{as } \epsilon, \delta \rightarrow 0
\end{aligned}$$

Observation 1. Comparing Figure 5.3 and Figure 5.4, we get the ratio between the Steiner Cost of the solution constructed by the algorithm and the Optimal Steiner Cost as $\frac{(2L+\epsilon+\delta)}{\delta}$, which becomes arbitrarily large as δ becomes small or length L becomes large.

[2]This also shows that the Swamy Kumar analysis can't be improved beyond an approximation ratio of 3.

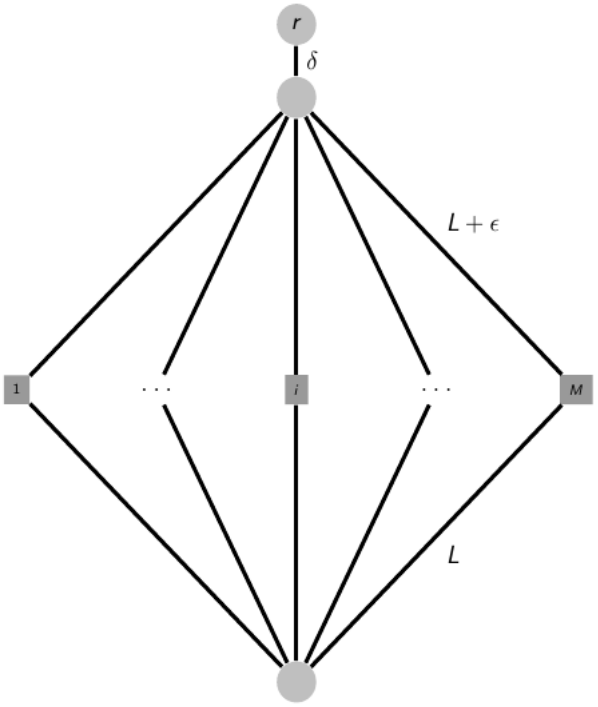

FIGURE 5.1: In the above problem instance, the Swamy-Kumar phase 1 clusters the demands at a location far away from the root and opens it, when paying just a little extra cost would have led to an overall better solution.

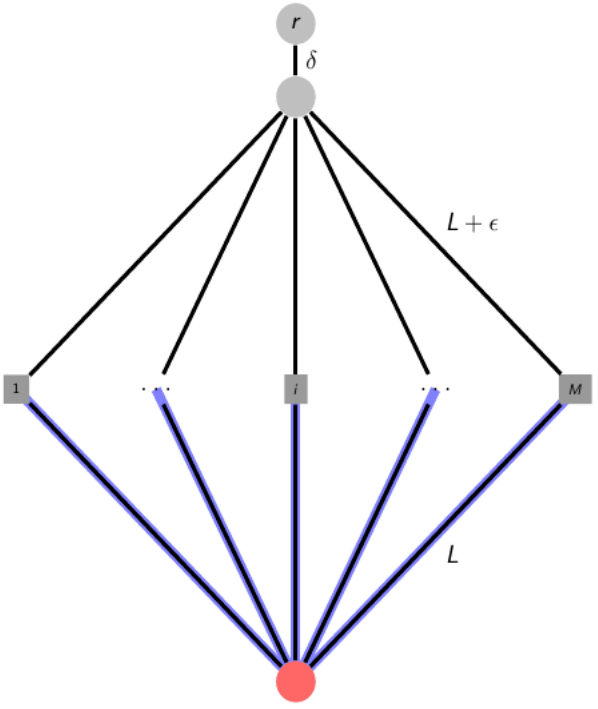

FIGURE 5.2: Phase 1. The red vertex is the facility chosen.

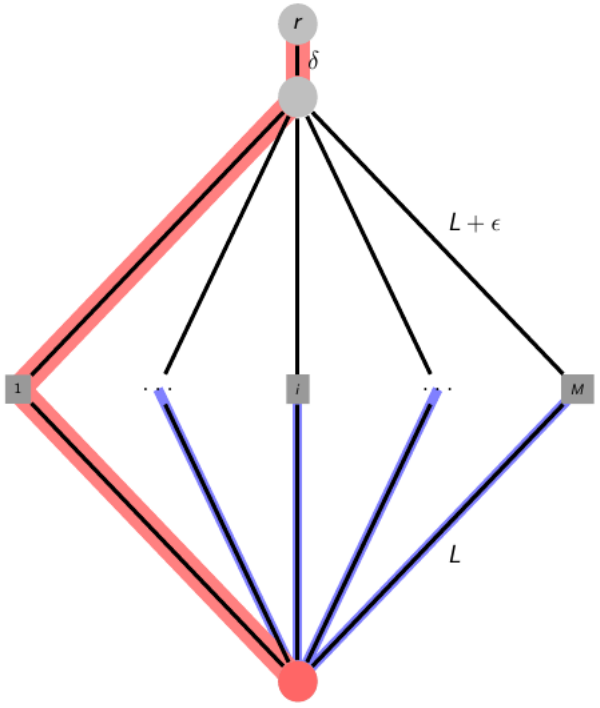

FIGURE 5.3: Phase 2. This phase simulates the primal dual algorithm for the rooted Steiner tree problem with r as the root and the red vertex as the terminal

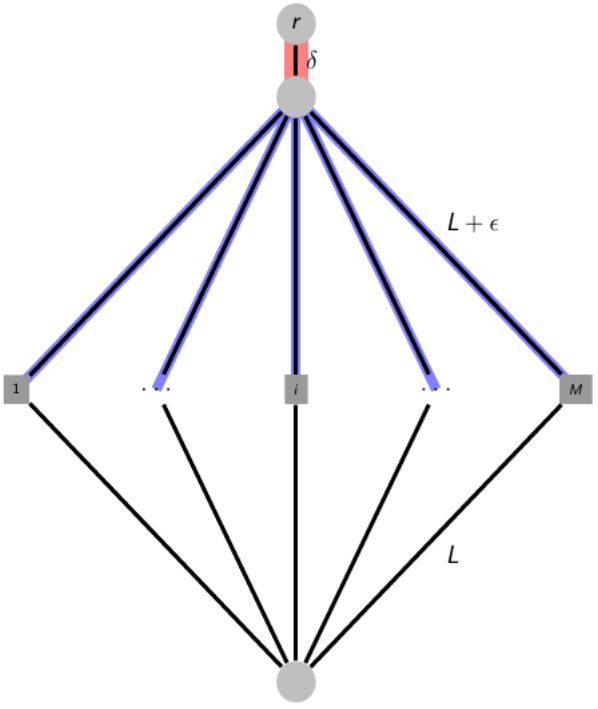

FIGURE 5.4: Optimal Solution

Discussion 5.1. In CFL, the optimal solution is constrained by the demand to satisfy the connectivity requirements economically. The Swamy-Kumar algorithm operates on the principle that partially economic viability is achieved by opening those facilities where enough demand can be clustered at a minimum local connection cost (Figure 5.2). The algorithm ignores the full implicit costs imposed on each facility (e.g, the distance between open facilities) in its decision regarding which facilities to open.

Furthermore, in the Swamy Kumar analysis, the cost of constructing the Steiner tree (in the second phase) is bounded by $(2+\alpha)$ times the overall optimal cost, where α is the Steiner tree approximation factor. A sharper analysis might relate the Steiner cost to the optimal Steiner tree cost, but this is not possible in the SK algorithm, as the gap between the cost of the Steiner tree constructed by the algorithm and the optimal tree might become arbitrarily large (Observation 1)

CHAPTER 6

Combining JMS and Steiner tree algorithms: Towards a 2-approximation for CFL

6.1 Introduction

In Chapter 4, we introduced the JMS algorithm for facility location due to Jain et al.[9]. A closer reading of the analysis of the JMS algorithm shows that there's a slack in the analysis which can possibly be exploited. The dual solution α produced by algorithm on an instance I of the facility location problem, is feasible for an instance I' obtained from I by multiplying only the connection costs (and not the facility costs) by 2. On the other hand, there's also a slack in the primal dual analysis of Prize Collecting Steiner tree algorithm by Goemans-Williamson, where the cost of the constructed Steiner tree has cost at most twice the sum of certain dual variables. In this chapter, we sketch out an approach on combining these two approaches, by relating the relating the facility costs in the JMS algorithm to the vertex potentials in the Goemans-Williamson PCST procedure. This approach holds a lot of promise for obtaining an overall 2-approximation for CFL.

6.2 An Integer Program

We use the Star formulation of Facility Location (Section 4.5.2) to obtain a new integer program which captures CFL. As before, we assume that we know a facility r that is opened and hence belongs to the Steiner tree in the optimal solution. This assumption doesn't affect anything because all $|\mathcal{F}|$ different possibilities for r can be tried.

A star $K = (i, C)$ consists of one facility and many clients, where facility $i \in \mathcal{F}$ and clients $C \subseteq X$. The cost c_K associated with a star K is $f_i + \sum_{j \in C} c_{ij}$. Let $\mathcal{K}$ be the set of all stars.

CFL can be viewed as a problem of choosing a set of stars such that each client belongs to atleast one star, and then choosing a set of edges which connect the facilities from each star. We create *decision variables* to represent these choices: let integer variable x_K represent whether star K has been chosen or not, and let integer variable b_e represent whether edge e has been included in the Steiner tree T or not (the *Steiner* or *bought* status of edge e, in our SRoB problem parlance).

$$x_K = \begin{cases} 1 & K \text{ is chosen} \\ 0 & K \text{ is not chosen} \end{cases} \qquad b_e = \begin{cases} 1 & b_e \in T \\ 0 & b_e \notin T \end{cases}$$

To reflect the above nature of the decision variables, we constrain the decision variables to non-negative integers : $x_K, b_E \in \{0, 1\}$. If we choose a particular star (i, C) for our solution, we say we have *opened* the facility i and *assigned* or *connected* clients $j \in S$ to it.

We need to enforce that the decision variables represent a feasible solution. i.e. a network in which (1) each clients is assigned to a facility, and (2) each open facility is connected to the root through a Steiner edge. From every *cut* that separates an *open* facility i (belonging to a star $K = (i, C)$) from the root, there must be an edge *use* atleast one edge. We formalize these requirements: Every client must belong to atleast one of the chosen stars, so for each client j,

$$\sum_{K=(i,C):\, j \in C} x_K \geq 1$$

Given a non-empty set of vertices $S \subset V$, let $\delta(S)$ denote the set of edges in the cut defined by S; that is, $\delta(S)$ is the set of all edges with exactly one endpoint in S. For every facility i and every *cut* separating the root r from the facility i, we introduce the following constraint:

$$\sum_{e \in \delta(S)} b_e \geq \sum_{K=(i,C)} x_K$$

We want to find a feasible solution of minimum cost. Given the decision variables, and the constraints described above, the cost of the network is $M \sum_{e \in E} c_e b_e + \sum_{K \in \mathcal{K}} c_K x_K$.

Thus, the following integer program models CFL:

$$\begin{aligned}
\text{minimize } & M\sum_{e\in E} c_e b_e + \sum_{K\in\mathcal{K}} c_K x_K \\
\text{subject to } & \sum_{K=(i,C):\ j\in C} x_K \geq 1 && \forall j \in X \\
& \sum_{e\in\delta(S)} b_e \geq \sum_{K=(i,C)} x_K && \forall i \in \mathcal{F}, \forall S\colon i \in S, r \notin S \\
& x_K, b_e \in \{0,1\} && \forall K \in \mathcal{K} \text{ and } \forall e \in E
\end{aligned}$$

6.3 Linear Programming Relaxation

We replace the constraints $x_K, b_e \in \{0,1\}$ with the constraints $x_K, b_e \geq 0$ to obtain a Linear Programming relaxation on the integer program.[1]

6.3.1 Primal Problem

$$\begin{aligned}
\text{minimize } & M\sum_{e\in E} c_e b_e + \sum_{K\in\mathcal{K}} c_K x_K \\
\text{subject to } & \sum_{K=(i,C):\ j\in K} x_K \geq 1 && \forall j \in X \\
& \sum_{e\in\delta(S)} b_e \geq \sum_{K=(i,C)} x_K && \forall i \in \mathcal{F}, \forall S\colon i \in S, r \notin S \\
& x_K, b_e \geq 0 && \forall K \in \mathcal{K} \text{ and } \forall e \in E
\end{aligned}$$

6.3.2 Dual Problem

We'll motivate the dual formulation with an appeal to intuition. In CFL, the costs involved are connection costs, facility opening costs, and the cost of building the Steiner tree. Instead of explicitly ascribing these costs to edges and stars, we recoup these costs from the clients instead. Let α_j denote the payment client j is willing to make towards buying a star and building the Steiner tree. For any star $K = (i, C)$, the clients $j \in C$ would not be willing to pay more than the price c_K of the star c_K (connection and facility opening costs) and the cost of constructing the part of the Steiner tree that joins facility i to root r. Under these conditions, the sum of the payments provides a lower

[1] We could also add the constraints $x_K, b_e \leq 1$, but they would be redundant: in any optimal solution to the problem, we can reduce $x_K, b_e > 1$ to $x_K, b_e = 1$ without affecting the feasibility and without increasing the cost.

bound to the cost of any feasible primal solution. Since we are interested in the sharpest lower bound, we maximise these payments:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in X} \alpha_j \\
\text{subject to} \quad & \sum_{j \in C} \alpha_j \leq c_K + \sum_{\substack{S\colon i \in S \\ r \notin S}} \theta_{S,i} && \forall K = (i, C) \in \mathcal{K} \\
& \sum_{i \in \mathcal{F}} \sum_{S\colon e \in \delta(S)} \theta_{S,i} \leq M c_e && \forall e \in E \\
& \alpha_j, \theta_{S,i} \geq 0 && \forall j \in X \text{ and } \forall S \subseteq V, r \notin S
\end{aligned}$$

6.4 SRoB

In the special case of SRoB, all facility opening costs f_i are zero, and hence $c_K = \sum_{j \in C} c_{ij}$. The dual reduces to:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in X} \alpha_j \\
\text{subject to} \quad & \sum_{j \in C} \alpha_j \leq \sum_{j \in C} c_{ij} + \sum_{\substack{S\colon i \in S \\ r \notin S}} \theta_{S,i} && \forall K = (i, C) \in \mathcal{K} \\
& \sum_{i \in \mathcal{F}} \sum_{S\colon e \in \delta(S)} \theta_{S,i} \leq M c_e && \forall e \in E \\
& \alpha_j, \theta_{S,i} \geq 0 && \forall j \in X \text{ and } \forall S \subseteq V, r \notin S
\end{aligned}$$

6.4.1 Combining JMS Algorithm with Goemans-Williamson moat-growing

Compare the above formulation for SRoB with the dual program for metric uncapacitated facility location given in Section 4.5.2 (substituing the cost of the star c_K by $f_i + \sum_{j \in C} c_{ij}$):

$$\begin{aligned} \text{maximize} \sum_{j \in X} \alpha_j & \\ \text{subject to} \sum_{j \in C} \alpha_j \leq \sum_{j \in C} c_{ij} + f_i \; \forall K = (i, C) \in \mathcal{K} & \\ \alpha_j \geq 0 \qquad\qquad\qquad\qquad\qquad\qquad & \forall j \in X \end{aligned}$$

Note the striking similarity between the role of expression $\sum_{\substack{S:\, i \in S \\ r \notin S}} \theta_{S,i}$ in SRoB with the facility opening costs f_i - the connectivity requirements in SRoB show a character like that of facility opening costs in UFL. This observation is useful because it suggests that we might be able to borrow techniques and results from UFL, for which 2-approximation algorithm primal dual algorithm already exists (Section 4.5.3.3), and apply them to SRoB. In the JMS algorithm in particular, the dual variables are constructed in such a way that they are atmost twice the connection cost plus the facility opening cost, for any star:

$$\sum_{j \in C} \alpha_j \leq 2. \sum_{j \in C} c_{ij} + f_i$$

Substituting the facility cost with the Steiner costs associated with the facility , the JMS construction would give:

$$\sum_{j \in C} \alpha_j \leq 2. \sum_{j \in C} c_{ij} + \sum_{\substack{S:\, i \in S \\ r \notin S}} \theta_{S,i} \; \forall K = (i, C) \tag{6.1}$$

The dual variables can also be interpreted as moats around sets, see Figure 3.1 in Chapter 3. In the PCST procedure[4] (and related Steiner tree algorithms) by Goemans and Williamson, the eventual Steiner tree constructed has cost at most twice the sum of certain dual variables, which contribute to the tree. Combining these two approaches would involve defining facility opening costs in such a way that JMS algorithm opens only those facilities which can pay for the tree through Goemans-Williamson moat growing procedure. At this stage, it is not entirely clear how such facility costs (or potentials, in PCST parlance) should be defined and what possible modifications to the JMS algorithm would be needed, but it is a very interesting direction for future work.

APPENDIX A

NP Completeness

One of the most powerful performance measure of an algorithm is the number of steps it takes to produce the solution as a function of input size, over all problem instances. [1] This measure is a machine independent measure of performance, and can be easily translated to the time it takes to execute the algorithm on any given machine, hence its also a measure of the time complexity of the algorithm.

The lower the time complexity, the larger the problem we can hope to solve in a reasonable amount of time. An algorithm whose time complexity can be expressed as a fixed degree polynomial in terms of input size is called a *polynomial-time algorithm*. These problems are efficiently solvable and referred to as tractable.

Definition A.1. (**Polynomial Time Algorithm**) An algorithm for a problem is said to run in polynomial time, or said to be a polynomial-time algorithm, with respect to a particular model of computer (such as a RAM) if the number of instructions executed by the algorithm can be bounded by a polynomial in the size of the input.

The complexity is used to divide all problems into a class hierarchy. The class $\mathcal{P}$ consists of all problems which have polynomial-time solution. For instance, the examples of addition and multiplication mentioned above have a polynomial time algorithm and is thus are $\mathcal{P}$. So is the shortest-path problem mentioned in the introduction which has a polynomial time algorithm (given by Edger Dijkstra).

Unfortunately, most combinatorial optimization problems are $\mathcal{NP} - hard$, that is, if one believe the $\mathcal{P} \neq \mathcal{NP}$ conjecture, then such problems do not have polynomial time algorithms.

[1] For example, assuming each arithmetic operation to be a single step operation, then the number of steps required to add n numbers is $n - 1$, multiplying two $n \times n$ matrices by the elementary method taught in high school requires $2n^3 - n^2$ steps, and so on.

Bibliography

[1] Eugene P Wigner. The unreasonable effectiveness of mathematics in the natural sciences. richard courant lecture in mathematical sciences delivered at new york university, may 11, 1959. *Communications on pure and applied mathematics*, 13(1): 1–14, 1960.

[2] Paul Lockhart. *A mathematician's lament*. Bellevue Literary Press New York, 2009.

[3] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.

[4] Michel X Goemans and David P Williamson. The primal-dual method for approximation algorithms and its application to network design problems. *Approximation algorithms for NP-hard problems*, pages 144–191, 1997.

[5] 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.

[6] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[7] Naveen Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 396–402. ACM, 2005.

[8] 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 (JACM)*, 48(2):274–296, 2001.

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

[10] ML Balinski. On finding integer solutions to linear programs. Technical report, DTIC Document, 1964.

[11] Friedrich Eisenbrand, Fabrizio Grandoni, Thomas Rothvoß, and Guido Schäfer. Approximating connected facility location problems via random facility sampling and core detouring. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1174–1183. Society for Industrial and Applied Mathematics, 2008.

[12] Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 365–372. ACM, 2003.

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

[14] Peng Zhang. Rent-or-buy network design problem and the sample-augment algorithm: A survey. *International Journal of Software and Informatics*, 5(4):607–636, 2011.

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

[16] David R Karger Maria Minkoff. Building steiner trees with incomplete global knowledge.

[17] Anupam Gupta, Jon Kleinberg, Amit Kumar, Rajeev Rastogi, and Bulent Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 389–398. ACM, 2001.

[18] Michael Jünger, Michael Schulz, and Wojciech Zychowicz. Fun with geometric duality.

[19] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.

[20] Anupam Gupta, Aravind Srinivasan, and Éva Tardos. Cost-sharing mechanisms for network design. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 139–150. Springer, 2004.