Algorithmic Aspects of the Internet

A Thesis
Presented to
The Academic Faculty

by

Amin Saberi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Algorithms, Combinatorics and Optimization (ACO), College of Computing
Georgia Institute of Technology
June 2004

Algorithmic Aspects of the Internet

Approved by:

Vijay V. Vazirani, Advisor

Milena Mihail, Advisor

Yan Z. Ding

Dana Randall

Richard Duke (School of Mathematics)

Date Approved: 10 June 2004

ACKNOWLEDGEMENTS

First, I would like to thank my advisors, Milena Mihail and Vijay Vazirani, for all I have learned from them and for sharing their insights with me which was invaluable during these four years. I also thank Dana Randall, Dick Lipton and other faculty of the ACO and College of Computing. I have also been very lucky to have wonderful colleagues and friends in the department including Vangelis Markakis, Christos Gkantsidis, Nikhil Devanur, Aranyak Mehta, Parikshit Gopalan and Nayantara Bhatnagar.

I have been fortunate to experience other exciting research communities. My visit to UC Berkeley was possible with the help of Christos Papadimitriou, Dick Karp, and Umesh Vazirani. I was exposed to many beautiful ideas in ICSI Berkeley and UC Berkeley thanks to Christos Papadimitriou, Dick Karp, Satish Rao, Scott Shenker, Amir Ronen, and many others. I am also grateful to many researchers at Microsoft Research including Jennifer Chayes, Christian Borgs, Kamal Jain, and Mohammad Mahdian.

As a final note, I would like to thank my parents for their support in all these years, and to Azin for her love, support and cheerfulness. It would not be fun without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The goal of this thesis is to use and advance the techniques developed in the field of exact and approximation algorithms for many of the problems arising in the context of the Internet.

We will formalize the method of dual fitting and the idea of factor-revealing LP. We use this combination to design and analyze two greedy algorithms for the metric uncapacitated facility location problem. Their approximation factors are 1.861 and 1.61 respectively.

We also provide the first polynomial time algorithm for the linear version of a market equilibrium model defined by Irving Fisher in 1891. Our algorithm is modeled after Kuhn's primal-dual algorithm for bipartite matching.

We also study the connectivity properties of the Internet graph and its impact on its structure. In particular, we consider the model of growth with preferential attachment for modeling the graph of the Internet and prove that under some reasonable assumptions, this graph has a constant conductance.

CHAPTER 1

INTRODUCTION

The Internet is probably the most notable invention of the late twentieth century. In less than twenty years, it has revolutionized computation and communication in the world and has become the medium for collaboration, interaction and information dissemination for millions of people.

This rapid growth and widespread usage has also brought its own challenges. Currently, the Internet is owned, operated, and used by several entities with different interests and any new algorithm or protocol to be implemented globally over the Internet should be secure against the selfish or malicious behavior of these entities.

The goal of this thesis is to enhance the tools and methods in theoretical computer science and especially algorithms for addressing these new challenges. We will study the ideas, models and solutions proposed for understanding various aspects of the Internet with an algorithmic perspective. In particular, we will try to resolve the issue of scalability or efficiency of these models.

In this quest, we have tried to choose fundamental problems which are also of independent interest. For example, algorithms for finding the prices in a market could have applications outside the context of the Internet as well.

In the next section, we will give a brief description of the problems studied in this thesis.

1.1 Problems Studied and Contributions

First, we study the facility location problem. Recently, this problem has found several new applications in network design problems such as placement of routers and caches, agglomeration of traffic or data, and web server replications in a content distribution network. It is also one of the central problems in operation research. In this problem, we are given a set of clients and a set of locations to build facilities, together with an opening cost for

each facility, and a connection cost between each client and each facility satisfying the metric inequality. The objective is to open a set of facilities and connect each client to an open facility so that the total cost of opening facilities and connecting cities to facilities is minimized.

We (jointly with Jain, Mahdian, Markakis, and Vazirani) will suggest and analyze two natural greedy algorithms for this problem (specifically the metric uncapacitated version). Their approximation factors are 1.861 and 1.61, with running times of $O(m \log m)$ and $O(n^3)$, respectively, where n is the total number of vertices and m is the number of edges in the underlying complete bipartite graph between cities and facilities.

In the third chapter, we (with Devanur, Papadimitriou, and Vazirani) use similar techniques to analyze a problem in game theory with applications in charging and rate control in communication networks. We present the first polynomial time algorithm for the linear version of an old problem, first defined in 1891 by Irving Fisher [71]: Consider a market consisting of buyers and divisible goods. The money possessed by buyers and the amount of each good are specified. Also specified are utility functions of buyers, which are assumed to be linear (Fisher's original statement assumed concave utility functions). The problem is to compute prices for the goods such that even if each buyer is made optimally happy, relative to these prices, there is no deficiency or surplus of any of the goods, i.e. the market clears.

In the fourth chapter, we (jointly with Mihail and Papadimitriou) study the connectivity properties of the Internet and its impact on its structure. We consider the model of *growth with preferential attachment* for modeling the graph of the Internet at the autonomous system level and prove that under some reasonable assumptions, this graph has a constant conductance. As a corollary, approximate multicommodity flow algorithms imply routing with congestion $O(n \log n)$ on these networks. An immediate additional implication is constant spectral gap between the first and second eigenvalues of the stochastic normalization of the adjacency matrix of the graph.

1.2 Methodology

The methodology used in the major part of this thesis is primal-dual method or more generally dual-based analysis. We use natural economic interpretations of the dual to help us design and analyze provably efficient algorithms for problems in optimization as well as in game theory.

In the facility location problem, we view the dual variables as the payments made by the cities towards their connection and opening costs. This suggests a natural modification of the Jain-Vazirani [47] algorithm in which cities withdraw their contributions from other facilities once they are connected to an open facility. This modification improves the approximation factor of the JV algorithm substantially and leads to the the best factor known for this problem. We also formalize and use the novel technique of using factor-revealing programs.

In the market equilibrium problem, we are also using an adaptation of the primal-dual schema. We identify two processes: the "primal process" updates the amount of each good sold to each buyer and the "dual process" updates prices of goods. For this problem, the economic interpretation is the most natural and primal and dual processes are a part of the problem definition.

However, unlike the facility location problem, the primal-dual schema that we have developed for the market equilibrium problem is not based on a linear or integer program. Nevertheless, we can apply a similar paradigm here: our algorithm starts with trivial solutions to the primal and dual programs and alternately improves these solutions using complementary slackness conditions. We bound the time that this process takes to converge to the equilibrium by analyzing the combinatorial structure of the problem.

In the last part of this thesis, we prove a lower bound on the conductance of scale-free graphs by a probabilistic analysis and use theorems from the theory of approximate multicommodity flow and MCMC methods for obtaining bounds on the performance certain algorithms performed on this networks.

CHAPTER 2

APPROXIMATION ALGORITHMS FOR METRIC FACILITY LOCATION PROBLEM

2.1 Introduction

A large fraction of the theory of approximation algorithms, as we know it today, is built around the theory of linear programming, which offers the two fundamental algorithm design techniques of rounding and the primal–dual schema (see [79]). Interestingly enough, the LP-duality based analysis [60, 20] for perhaps the most central problem of this theory, the set cover problem, did not use either of these techniques. Moreover, the analysis used for set cover does not seem to have found use outside of this problem and its generalizations [70], leading to a somewhat unsatisfactory state of affairs.

In this chapter, we formalize the technique used for analyzing set cover as the *method of dual fitting*, and we also introduce the idea of using a *factor-revealing LP*. Using this combination we analyze two greedy algorithms for the metric uncapacitated facility location problem. Their approximation factors are 1.861 and 1.61, with running times of $O(m \log m)$ and $O(n^3)$ respectively, where m and n denote the total number of edges and vertices in the underlying complete bipartite graph between cities and facilities. In other words, $m = n_c \times n_f$ and $n = n_c + n_f$, where n_c is the number of cities and n_f is the number of facilities.

2.1.1 Dual fitting with factor-revealing LP

The set cover problem offers a particularly simple setting for illustrating most of the dominant ideas in approximation algorithms (see [79]). Perhaps the reason that the method of dual fitting was not clear so far was that the set cover problem did not require its full power. However, in retrospect, its salient features are best illustrated again in the simple setting of the set cover problem – we do this in Section 2.9.

The method of dual fitting can be described as follows, assuming a minimization problem: The basic algorithm is combinatorial – in the case of set cover it is in fact a simple greedy algorithm. Using the linear programming relaxation of the problem and its dual, one first interprets the combinatorial algorithm as a primal-dual-type algorithm – an algorithm that is iteratively making primal and dual updates. Strictly speaking, this is not a primal-dual algorithm, since the dual solution computed is, in general, infeasible (see Section 2.9 for a discussion on this issue). However, one shows that the primal integral solution found by the algorithm is fully paid for by the dual computed. By *fully paid for* we mean that the objective function value of the primal solution is bounded by that of the dual. The main step in the analysis consists of dividing the dual by a suitable factor, say γ, and showing that the shrunk dual is feasible, i.e., it *fits* into the given instance. The shrunk dual is then a lower bound on OPT, and γ is the approximation guarantee of the algorithm.

Clearly, we need to find the minimum γ that suffices. Equivalently, this amounts to finding the worst possible instance – one in which the dual solution needs to be shrunk the most in order to be rendered feasible. For each value of n_c, the number of cities, we define a factor-revealing LP that encodes the problem of finding the worst possible instance with n_c cities as a linear program. This gives a family of LP's, one for each value of n_c. The supremum of the optimal solutions to these LP's is then the best value for γ. In our case, we do not know how to compute this supremum directly. Instead, we obtain a feasible solution to the dual of each of these LP's. An upper bound on the objective function values of these duals can be computed, and is an upper bound on the optimal γ. In our case, this upper bound is 1.861 for the first algorithm and 1.61 for the second one. In order to get a closely matching tight example, we numerically solve the factor-revealing LP for a large value of n_c.

The technique of factor-revealing LPs is similar to the idea of LP bounds in coding theory. LP bounds give the best known bounds on the minimum distance of a code with a given rate by bounding the solution of a linear program. (cf. McEliece et al. [62]). In the context of approximation algorithms, Goemans and Kleinberg [33] use a similar method in the analysis of their algorithm for the minimum latency problem.

2.1.2 The facility location problem

In the (uncapacitated) facility location problem, we have a set $\mathcal{F}$ of n_f *facilities* and a set $\mathcal{C}$ of n_c *cities*. For every facility $i \in \mathcal{F}$, a nonnegative number f_i is given as the *opening cost* of facility i. Furthermore, for every facility $i \in \mathcal{F}$ and city $j \in \mathcal{C}$, we have a *connection cost* (a.k.a. service cost) c_{ij} between facility i and city j. The objective is to open a subset of the facilities in $\mathcal{F}$, and connect each city to an open facility so that the total cost is minimized. We will consider the *metric* version of this problem, i.e., the connection costs satisfy the triangle inequality.

This problem has occupied a central place in operations research since the early 60's [8, 52, 55, 73, 74], and has been studied from the perspectives of worst case analysis, probabilistic analysis, polyhedral combinatorics and empirical heuristics (see [22, 65]). Although the first approximation algorithm for this problem, a greedy algorithm achieving a guarantee of $O(\log n)$ in the general (non-metric) case due to Hochbaum [42], dates back to almost 20 years ago, renewed interest in recent years has resulted in much progress. Recently, the problem has found several new applications in network design problems such as placement of routers and caches [38, 59], agglomeration of traffic or data [4, 39], and web server replications in a content distribution network (CDN) [50, 68, 69].

The first constant factor approximation algorithm for this problem was given by Shmoys, Tardos, and Aardal [72]. Later, the factor was improved by Chudak and Shmoys [17] to $1+2/e$. Both these algorithms were based on LP-rounding, and therefore had high running times.

Jain and Vazirani [47] gave a primal–dual algorithm, achieving a factor of 3, and having the same running time as ours (we will refer to this as the JV algorithm). Their algorithm was adapted for solving several related problems such as the fault-tolerant and outlier versions, and the k-median problem [47, 48, 16]. Mettu and Plaxton [63] used a restatement of the JV algorithm for the on-line median problem.

Strategies based on local search and greedy improvement for facility location problem have also been studied. The work of Korupolu et al. [54] shows that a simple local search heuristic proposed by Kuehn and Hamburger [55] yields a $(5+\epsilon)$-approximation algorithm

with a running time of $O(n^6 \log n/\epsilon)$, for any $\epsilon > 0$. Charikar and Guha [14] improved the factor slightly to 1.728 by combining the JV algorithm, greedy augmentation, and the LP-based algorithm [17]. They also combined greedy improvement and cost scaling to improve the factor of the JV algorithm to 1.853. For a metric defined by a sparse graph, Thorup [76] has obtained a $(3 + o(1))$-approximation algorithm with running time $\tilde{O}(|E|)$. Regarding hardness results, Guha and Khuller [37] showed that the best approximation factor possible for this problem is 1.463, assuming $NP \not\subseteq DTIME[n^{O(\log \log n)}]$.

Since the publication of this chapter as a separate paper, two new algorithms have been proposed for the facility location problem. The first algorithm, due to Sviridenko [75], uses the LP-rounding method to achieve an approximation factor of 1.58. The second algorithm, due to Mahdian, Ye, and Zhang [61], combines our second algorithm with the idea of cost scaling to achieve an approximation factor of 1.52, which is currently the best known factor for this problem.

2.1.3 Our results

Our first algorithm is quite similar to the greedy set cover algorithm: iteratively pick the most cost-effective choice at each step, where cost-effectiveness is measured as the ratio of the cost incurred to the number of new cities served. In order to use LP-duality to analyze this algorithm, we give an alternative description which can be seen as a modification of the JV algorithm – when a city gets connected to an open facility, it withdraws whatever it has contributed towards the opening cost of other facilities. This step of withdrawing contribution is important, since it ensures that the primal solution is fully paid for by the dual.

The second algorithm has a minor difference with the first one: A city 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.

The approximation factor of the algorithms are 1.861 and 1.61, with running times of $O(m \log m)$ and $O(n^3)$ respectively where n is the total number of vertices and m is the number of edges in the underlying complete bipartite graph between cities and facilities.

We have experimented our algorithms on randomly generated instances as well as instances obtained from the Operations Research library [10] and GT-ITM Internet topology generator [82]. The cost of the integral solution found is compared against the solution of the LP-relaxation of the problem, rather than OPT (computing which would be prohibitively time consuming). The results are encouraging: The average error of our algorithms is about 3% and 1% respectively, and is a significant improvement over the JV algorithm which has an error of even 100% in some cases.

The primal-dual algorithm of Jain and Vazirani [47] is versatile in that it can be used to obtain algorithms for many variants of the facility location problem, such as k-median [47], a common generalization of k-median and facility location [47], capacitated facility location with soft capacities [47], prize collecting facility location [16], and facility location with outliers [16]. In Section 2.8, we apply our algorithms to several variants of the problem. First, we consider a common generalization of the facility location and k-median problems. In this problem, which we refer to as the *k-facility location problem*, an instance of the facility location problem and an integer k are given and the objective is to find the cheapest solution that opens at most k facilities. The k-median problem is a special case of this problem in which all opening costs are 0. The k-median problem is studied extensively [6, 14, 15, 47] and the best known approximation algorithm for this problem, due to Arya et al. [6], achieves a factor of $3 + \epsilon$. The k-facility location problem has also been studied in operations research [22], and the best previously known approximation factor for this problem was 6 [47].

Next, we show an application of our algorithm to the facility location game. We also use our algorithm to improve recent results for some other variants of the problem. In the facility location problem with outliers we are not required to connect all cities to open facilities. We consider two versions of this variant: In the robust version, we are allowed to leave l cities unconnected. In facility location with penalties we can either connect a city to a facility, or pay a specified penalty. Both versions were motivated by commercial applications, and were proposed by Charikar et al. [16]. Here, we will modify our algorithm to obtain a factor 2 approximation algorithm for these versions, improving the best known

result of factor 3 [16].

In the fault tolerant variant, each city has a specified number of facilities it should be connected to. This problem was proposed in [48] and the best factor known is 2.47 [40]. We can achieve a factor of 1.61 when all cities have the same connectivity requirement. In addition, we introduce a new variant which can be seen as a special case of the concave cost version of this problem: the cost of opening a facility at a location is specified and it can serve exactly one city. In addition, a *setup cost* is charged the very first time a facility is opened at a given location.

2.2 Algorithm 1

In the following algorithm we use a notion of cost effectiveness. Let us say that a *star* consists of one facility and several cities. The cost of a star is the sum of the opening cost of the facility and the connection costs between the facility and all the cities in the star. More formally, the cost of the star (i, C'), where i is a facility and $C' \subseteq C$ is a subset of cities, is $f_i + \sum_{j \in C'} c_{ij}$. The cost effectiveness of the star (i, C') is the ratio of the cost of the star to the size of C', i.e., $\left(f_i + \sum_{j \in C'} c_{ij}\right) / |C'|$.

Algorithm 1

1. Let U be the set of unconnected cities. In the beginning, all cities are unconnected i.e. $U := C$ and all facilities are unopened.

2. While $U \neq \emptyset$:

   - Among all stars, find the most cost-effective one, (i, C'), open facility i, if it is not already open, and connect all cities in C' to i.
   - Set $f_i := 0$, $U := U \setminus C'$.

Note that a facility can be chosen again after being opened, but its opening cost is counted only once since we set f_i to zero after the first time the facility is picked by the algorithm. As far as cities are concerned, every city j is removed from C, when connected to an open facility, and is not taken into consideration again. Also, notice that although the number

of stars is exponentially large, in each iteration the most cost-effective pair can be found in polynomial time. For each facility i, we can sort the cities in increasing order of their connection cost to i. It can be easily seen that the most cost-effective star will consist of a facility and a set, containing the first k cities in this order, for some k.

The idea of cost effectiveness essentially stems from a similar notion in the greedy algorithm for the set cover problem. In that algorithm, the cost effectiveness of a set S is defined to be the cost of S over the number of uncovered elements in S. In each iteration, the algorithm picks the most cost-effective set until all elements are covered. The most cost-effective set can be found either by using direct computation, or by using the dual program of the linear programming formulation for the problem. The dual program can also be used to prove the approximation factor of the algorithm. Similarly, we will use the LP-formulation of facility location to analyze our algorithm. As we will see, the dual formulation of the problem helps us to understand the nature of the problem and the greedy algorithm.

The facility location problem can be captured by an integer program due to Balinski [8]. For the sake of convenience, we give another equivalent formulation for the problem. Let $\mathcal{S}$ be the set of all stars. The facility location problem can be thought of as picking a minimum cost set of stars such that each city is in at least one star. This problem can be captured by the following integer program. In this program, x_S is an indicator variable denoting whether star S is picked and c_S denotes the cost of star S.

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S x_S \\ \text{subject to} \quad & \forall j \in \mathcal{C} : \sum_{S : j \in S} x_S \geq 1 \\ & \forall S \in \mathcal{S} : \; x_S \in \{0, 1\} \end{aligned} \tag{1}$$

The LP-relaxation of this program is:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S x_S \\ \text{subject to} \quad & \forall j \in \mathcal{C} : \sum_{S : j \in S} x_S \geq 1 \\ & \forall S \in \mathcal{S} : \; x_S \geq 0 \end{aligned} \tag{2}$$

The dual program is:

$$\begin{aligned} \text{maximize} \quad & \sum_{j \in \mathcal{C}} \alpha_j \\ \text{subject to} \quad & \forall S \in \mathcal{S} : \sum_{j \in S \cap \mathcal{C}} \alpha_j \leq c_S \\ & \forall j \in \mathcal{C} : \ \alpha_j \geq 0 \end{aligned} \tag{3}$$

There is an intuitive way of interpreting the dual variables. We can think of α_j as the contribution of city j, or its share toward the total expenses. Note that the first inequality of the dual can also be written as $\sum_{j \in C} \max(0, \alpha_j - c_{ij}) \leq f_i$ for every facility i. We can now see how the dual variables can help us find the most cost-effective star in each iteration of the greedy algorithm: if we start raising the dual variables of all unconnected cities simultaneously, the most cost-effective star will be the first star (i, C') for which $\sum_{j \in C'} \max(0, \alpha_j - c_{ij}) = f_i$. Hence we can restate Algorithm 1 based on the above observation. This is in complete analogy to the greedy algorithm and its restatement using LP-formulation for set-cover.

Restatement of Algorithm 1

1. We introduce a notion of time, so that each event can be associated with the time at which it happened. The algorithm starts at time 0. Initially, each city is defined to be unconnected ($U := C$), all facilities are unopened, and α_j is set to 0 for every j.

2. While $U \neq \emptyset$, increase the time, and simultaneously, for every city $j \in U$, increase the parameter α_j at the same rate, until one of the following events occurs (if two events occur at the same time, we process them in arbitrary order).

   (a) For some unconnected city j, and some open facility i, $\alpha_j = c_{ij}$. In this case, connect city j to facility i and remove j from U.

   (b) For some unopened facility i, we have $\sum_{j \in U} \max(0, \alpha_j - c_{ij}) = f_i$. This means that the total contribution of the cities is sufficient to open facility i. In this case, open this facility, and for every unconnected city j with $\alpha_j \geq c_{ij}$, connect j to i, and remove it from U.

In each iteration of algorithm 1 the process of opening a facility and/or connecting some cities will be defined as an *event*. It is easy to prove the following lemma by induction.

Lemma 1 *The sequence of events executed by Algorithm 1 and its restatement are identical.*

Proof: By induction. □

This restatement can also be seen as a modification of JV algorithm [47]. The only difference is that in JV algorithm cities, when connected to an open facility, are not excluded from U, hence they might contribute towards opening several facilities. Due to this fact they have a second cleanup phase in which some of the already open facilities will be closed down.

Also, it is worth noting that despite the similarity between Algorithm 1 and Hochbaum's greedy algorithm for facility location (which is equivalent to the set cover algorithm applied on the set of stars), they are not equivalent. This is because we set f_i to zero after picking a set containing f_i. As the following example shows, the approximation factor of Hochbaum's algorithm is $\Omega(\frac{\log n}{\log \log n})$ on instances with metric inequality: Consider k facilities with opening cost p^k located in the same place Also $k-1$ groups of cities $S_1, S_2, \ldots, S_{k-1}$. The group S_i consists of p^{k-i+1} cities with distance $\sum_{j=1\ldots i} p^{j-1}$ from the facilities. Other distances are obtained from the triangle inequality. Hochbaum's algorithm opens all facilities and therefore its solution costs more than kp^k. The optimum solution is $p^k + \sum_{i=1\ldots k-1} \sum_{j=1\ldots i} p^{j-1}$. It is easy to show that with a careful choice of k, the ratio of these two expressions is $\Omega(\frac{\log n}{\log \log n})$. We do not know whether the approximation factor of Hochbaum's algorithm on metric instances is strictly less than $\log n$ or not.

2.3 Analysis of Algorithm 1

In this section we will give an LP-based analysis of the algorithm. As stated before, the contribution of each city goes towards opening at most one facility and connecting the city to an open facility. Therefore, the total cost of the solution produced by our algorithm will be equal to the sum $\sum_j \alpha_j$ of the contributions. However, $\boldsymbol{\alpha}$ is not a feasible dual solution as it was in JV algorithm. The reason is that in every iteration of the restatement

of Algorithm 1, we exclude a subset of cities and withdraw their contribution from all facilities. So at the end, for some facility i, $\sum_j \max(\alpha_j - c_{ij}, 0)$ can be greater than f_i and hence the corresponding constraints of the dual program is violated.

However, if we find an γ for which $\boldsymbol{\alpha}/\gamma$ is feasible, $\sum_j \alpha_j/\gamma$ would be a lower bound to the optimum and therefore the approximation factor of the algorithm would be at most γ. This observation motivates the following definition.

Definition Given α_j $(j = 1, \ldots, n_c)$, a facility i is called at most γ-overtight if and only if

$$\sum_j \max(\alpha_j/\gamma - c_{ij}, 0) \leq f_i.$$

Using the above definition, it is trivial that $\boldsymbol{\alpha}/\gamma$ is a feasible dual if and only if each facility is at most γ-overtight. Now, we want to find such an γ. Note that in the above sum we only need to consider the cities j for which $\alpha_j \geq \gamma c_{ij}$. Let us assume without loss of generality that it is the case only for the first k cities. Moreover, assume without loss of generality that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_k$. The next two lemmas express the constraints on $\boldsymbol{\alpha}$ imposed by the problem or our algorithm. The first lemma mainly captures metric property and the second one expresses the fact that the total contribution offered to a facility at any time during the algorithm is no more than its cost.

Lemma 2 *For every two cities j, j' and facility i, $\alpha_j \leq \alpha_{j'} + c_{ij'} + c_{ij}$.*

Proof: If $\alpha_{j'} \geq \alpha_j$, the inequality obviously holds. Assume $\alpha_j > \alpha_{j'}$. Let i' be the facility that city j' is connected to by our algorithm. Thus, facility i' is open at time $\alpha_{j'}$. The contribution α_j cannot be greater than $c_{i'j}$ because in that case city j could be connected to facility i' at some time $t < \alpha_j$. Hence $\alpha_j \leq c_{i'j}$. Furthermore, by triangle inequality, $c_{i'j} \leq c_{i'j'} + c_{ij'} + c_{ij} \leq \alpha_{j'} + c_{ij'} + c_{ij}$. □

Lemma 3 *For every city j and facility i, $\sum_{l=j}^{k} \max(\alpha_j - c_{il}, 0) \leq f_i$.*

Proof: Assume, for the sake of contradiction, that for some j and some i the inequality does not hold, i.e., $\sum_{k=j}^{n_c} \max(\alpha_j - c_{ik}, 0) > f_i$. By the ordering on cities, for $k \geq j$, $\alpha_k \geq \alpha_j$. Let time $t = \alpha_j$. By the assumption, facility i is fully paid for before time t. For any city

k, $j \leq k \leq n_c$ for which $\alpha_j - c_{ik} > 0$ the edge (i,k) must be tight before time t. Moreover, there must be at least one such city. For this city, $\alpha_k < \alpha_j$, since the algorithm will stop growing α_k as soon as k has a tight edge to a fully paid for facility. The contradiction establishes the lemma. □

Subject to the constraints introduced by Lemmas 2 and 3, we want to find the minimum γ for which $\sum_{j=1}^{k}(\alpha_j/\gamma - c_{ij}) \leq f_i$. In other words, we want to find the maximum of the ratio $\frac{\sum_{j=1}^{k}\alpha_j}{f+\sum_{j=1}^{k}d_j}$. We can define variables f, d_j, and α_j, corresponding to facility cost, distances, and contributions respectively and write the following maximization program:

$$\begin{array}{lll} z_k = \text{ maximize } & \dfrac{\sum_{j=1}^{k}\alpha_j}{f+\sum_{j=1}^{k}d_j} & \\ \text{subject to } & \alpha_j \leq \alpha_{j+1} & \forall j \in \{1,\ldots,k-1\} \\ & \alpha_j \leq \alpha_l + d_j + d_l & \forall j,l \in \{1,\ldots,k\} \\ & \sum_{l=j}^{k}\max(\alpha_j - d_l, 0) \leq f & \forall j \in \{1,\ldots,k\} \\ & \alpha_j, d_j, f \geq 0 & \forall j \in \{1,\ldots,k\} \end{array} \tag{4}$$

It's not difficult to prove that z_k (the maximum value of the objective function of program 4) is equal to the optimal solution of the following linear program which we call the *factor-revealing LP*.

$$\begin{array}{lll} z_k = \text{ maximize } & \displaystyle\sum_{j=1}^{k}\alpha_j & \\ \text{subject to } & f+\sum_{j=1}^{k}d_j \leq 1 & \\ & \alpha_j \leq \alpha_{j+1} & \forall j \in \{1,\ldots,k-1\} \\ & \alpha_j \leq \alpha_l + d_j + d_l & \forall j,l \in \{1,\ldots,k\} \\ & x_{jl} \geq \alpha_j - d_l & \forall j,l \in \{1,\ldots,k\} \\ & \sum_{l=j}^{k}x_{jl} \leq f & \forall j \in \{1,\ldots,k\} \\ & \alpha_j, d_j, f \geq 0 & \forall j \in \{1,\ldots,k\} \end{array} \tag{5}$$

Lemma 4 *Let $\gamma = \sup_{k\geq 1}\{z_k\}$. Every facility is at most γ-overtight*

Proof: Consider facility i. We want to show that $\sum_j max(\alpha_j/\gamma - c_{ij}, 0) \leq f_i$. Suppose without loss of generality that the subset of cities j such that $\alpha_j \geq \gamma c_{ij}$ is $\{j = 1, 2, \ldots, k\}$ for

some k. Moreover $\alpha_1 \leq \alpha_2 \leq \ldots \alpha_k$. Let $d_j = c_{ij}$, $j = 1, \ldots, k$, and $f = f_i$. By Lemmas 2 and 3 it follows immediately that the constraints of program 4 are satisfied. Therefore, α_i, d_i, f constitute a feasible solution of program 4. Consequently $\frac{\sum_{j=1}^{k} \alpha_j}{f_i + \sum_{j=1}^{k} c_{ij}} \leq z_k$. □

By what we said so far, we know that the approximation factor of our algorithm is at most $\sup_{k \geq 1}\{z_k\}$. In the following theorem, we prove, by demonstrating an infinite family of instances, that the approximation ratio of Algorithm 1 is not better than $\sup_{k \geq 1}\{z_k\}$.

Theorem 5 *The approximation factor of our algorithm is precisely* $\sup_{k \geq 1}\{z_k\}$.

Proof: Consider an optimum feasible solution of program 4. We construct an instance of the facility location problem with k cities and $k+1$ facilities as follows: The cost of opening facility i is

$$f_i = \begin{cases} 0 & \text{if } 1 \leq i \leq k \\ f & \text{if } i = k+1 \end{cases}$$

The connection cost between a city j and a facility i is:

$$c_{ij} = \begin{cases} \alpha_j & \text{if } 1 \leq i = j \leq k \\ d_j & \text{if } 1 \leq j \leq k, i = k+1 \\ d_i + d_j + \alpha_i & \text{otherwise} \end{cases}$$

It is easy to see that the connection costs satisfy the triangle inequality. On this instance, our algorithm connects city 1 to facility 1, then it connects city 2 to facility 2, and finally connects city k to facility k. (The inequality $\sum_{l=j}^{k} \max(\alpha_j - d_l, 0) \leq f$ guarantees that city i can get connected to facility i before facility $k+1$). Therefore, the cost of the restatement of Algorithm 1 is equal to $\sum_{j=1}^{k} c_{jj} + \sum_{i=1}^{k} f_i = \sum_{j=1}^{k} \alpha_j = z_k$.

On the other hand, the optimal solution for this instance is to connect all the cities to facility $k+1$. The cost of this solution is equal to $\sum_{j=1}^{k} c_{k+1,j} + f_{k+1} = f + \sum_{j=1}^{k} d_j \leq 1$.

Thus, our algorithm outputs a solution whose cost is at least z_k times the cost of the optimal solution. □

The only thing that remains is to find an upper bound on $sup_{k \geq 1}\{z_k\}$. By solving the factor-revealing LP for any particular value of k, we get a lower bound on the value of

γ. In order to prove an upper bound on γ, we need to present a general solution to the dual of the factor-revealing LP. Unfortunately, this is not an easy task in general. (For example, performing a tight asymptotic analysis of the LP bound is still an open question in coding theory). However, here empirical results can help us: we can solve the dual of the factor-revealing LP for small values of k to get an idea of how the general optimal solution looks like. Using this, it is usually possible (although sometimes tedious) to prove a close-to-optimal upper bound on the value of z_k. We have used this technique to prove an upper bound of 1.861 on γ.

Lemma 6 *For every $k \geq 1$, $z_k \leq 1.861$.*

Proof: Let $r = 1.8609$. By doubling a feasible solution of 4 it is easy to show that $z_k \leq z_{2k}$ so we can assume, without loss of generality that k is sufficiently large. Consider a feasible solution of the program 4. It is clear from the third inequality that for every j, j' we have

$$\sum_{i=j}^{j'} (\alpha_j - d_i) \leq f. \tag{6}$$

Now, we define l_j and θ_j as follows:

$$l_j = \begin{cases} p_2 k & \text{if } j \leq p_1 k \\ k & j > p_1 k \end{cases}$$

$$\theta_j = \begin{cases} \frac{r+1}{p_2 k} & \text{if } j \leq p_1 k \\ \frac{(r+1)(p_2 - p_1)}{p_2(1-p_1)k} & p_1 k < j \leq p_2 k \\ 0 & j > p_2 k \end{cases}$$

where $p_1 = 0.1991$ and $p_2 = 0.5696$. We consider Inequality 6 for every $j \leq p_2 k$ and $j' = l_j$, and multiply both sides of this inequality by θ_j. By adding up all these inequalities, we obtain

$$\sum_{j=1}^{p_1 k} \sum_{i=j}^{p_2 k} \theta_j (\alpha_j - d_i) + \sum_{j=p_1 k+1}^{p_2 k} \sum_{i=j}^{k} \theta_j (\alpha_j - d_i) \leq (\sum_{j=1}^{p_2 k} \theta_j) f. \tag{7}$$

The coefficient of f in the right-hand side of the above inequality is equal to $\sum_{j=1}^{p_2 k} \theta_j = \frac{r+1}{p_2 k} p_1 k + \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}(p_2 k - p_1 k) \approx 1.8609 < 1.861$. Also, the coefficients of α_j and d_j in the left-hand side of Inequality 7 are equal to

$$\text{coeff}[\alpha_j] = \begin{cases} (p_2 k - j + 1)\theta_j & j \leq p_1 k \\ (k - j + 1)\theta_j & j > p_1 k \end{cases} \tag{8}$$

$$\text{coeff}[d_j] = \begin{cases} \sum_{i=1}^{j} \theta_i & j \leq p_2 k \\ \sum_{i=p_1 k+1}^{j} \theta_i & j > p_2 k \end{cases} \tag{9}$$

Notice that the sum of coefficients of α_j's is equal to

$$\begin{aligned} \sum_{j=1}^{k} \text{coeff}[\alpha_j] &= \sum_{j=1}^{p_1 k} \frac{r+1}{p_2 k}(p_2 k - j + 1) + \sum_{j=p_1 k+1}^{p_2 k} \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}(k - j + 1) \\ &> (r+1)\left(p_1 - \frac{p_1^2}{2p_2} + \frac{(p_2-p_1)^2}{p_2(1-p_1)} - \frac{(p_2-p_1)^2(p_1+p_2)}{2p_2(1-p_1)}\right)k \\ &\approx 1.00004k \\ &> k \end{aligned}$$

Now, we use the inequality $\alpha_i \geq \alpha_j - d_j - d_i$ on the expression on the left hand side of inequality 7 to reduce the coefficients of α_j's that are greater than 1, and increase the coefficient of α_j's that are less than 1. Since the sum of these coefficients is greater than k, using this inequality and the inequality $\alpha_j \geq 0$ we can obtain an expression E that is less than or equal to the left hand side of inequality 7, and in which all α_j's have coefficient 1. The coefficient of d_j in this expression will be equal to its coefficient in the left hand side of inequality 7, plus the absolute value of the change in the coefficient of the corresponding α_j. Therefore, by equations 8 and 9 this coefficient is equal to:

$$\text{coeff}_E[d_j] = \begin{cases} \sum_{i=1}^{j} \theta_i + |(p_2 k - j + 1)\theta_j - 1| & j \leq p_1 k \\ \sum_{i=1}^{j} \theta_i + |(k - j + 1)\theta_j - 1| & p_1 k < j \leq p_2 k \\ \sum_{i=p_1 k+1}^{j} \theta_i + |(k - j + 1)\theta_j - 1| & j > p_2 k \end{cases}$$

If $j \leq p_1 k$, we have $(p_2 k - j + 1)\theta_j > (p_2 k - p_1 k)\frac{r+1}{p_2 k} = (r+1)(p_2 - p_1)/p_2 \approx 1.8609 > 1$ Therefore,

$$\begin{aligned}\text{coeff}_E[d_j] &= \sum_{i=1}^{j} \theta_i + (p_2 k - j + 1)\theta_j - 1 \\ &= r + O(\frac{1}{k}) \\ &< 1.861\end{aligned}$$

Similarly, if $p_1 k < j \leq p_2 k$, we have $(k-j+1)\theta_j > (k-p_2 k)\frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k} = \frac{(r+1)(p_2-p_1)(1-p_2)}{p_2(1-p_1)} \approx 1.00003 > 1$. Therefore,

$$\begin{aligned}\text{coeff}_E[d_j] &= \sum_{i=1}^{j} \theta_i + (k - j + 1)\theta_j - 1 \\ &= r + O(\frac{1}{k}) \\ &< 1.861\end{aligned}$$

Finally, if $j > p_2 k$, the coefficient of d_j is equal to

$$\begin{aligned}\text{coeff}_E[d_j] &= \sum_{i=p_1 k}^{j} \theta_i + |0 - 1| \\ &= \frac{(r+1)(p_2 - p_1)}{p_2(1-p_1)k}(p_2 k - p_1 k) + 1 \\ &\approx 1.8609 \\ &< 1.861\end{aligned}$$

Therefore, in each case, the coefficient of d_j is less than or equal to 1.861. Thus, we have proved that

$$\sum_{j=1}^{k} \alpha_j - \sum_{j=1}^{k} 1.861 d_j < 1.861 f.$$

This clearly implies that $z_k < 1.861$. □

Figure 1 shows a tight example for $k = 2$, for which the approximation factor of the algorithm is 1.5. The cost of the missing edges is given by triangle inequality. Numerical computations using the software CPLEX show that $z_{300} \approx 1.81$. Thus, the approximation factor of our algorithm is between 1.81 and 1.861. We do not know the exact approximation ratio.

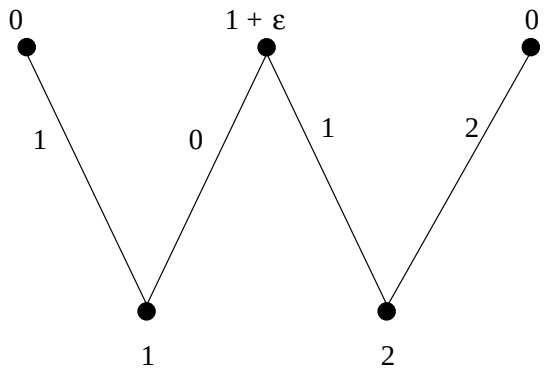

Figure 1: The approximation ratio of Algorithm 1 is at least 1.5

2.4 Algorithm 2

Algorithm 2 is similar to the restatement of Algorithm 1. The only difference is that in Algorithm 1 cities stop offering money to facilities as soon as they get connected to a facility, but here they still offer some money to other facilities. The amount that an already-connected city offers to a facility j is equal to the amount that it would save in connection cost by switching its facility to j. As we will see in the next section, this change reduces the approximation factor of the algorithm from 1.861 to 1.61.

Algorithm 2

1. We introduce a notion of time. The algorithm starts at time 0. At this time, each city is defined to be unconnected ($U := C$), all facilities are unopened, and α_j is set to 0 for every j.

   At every moment, each city j offers some money from its contribution to each *unopened* facility i. The amount of this offer is computed as follows: If j is unconnected, the offer is equal to $\max(\alpha_j - c_{ij}, 0)$ (i.e., if the contribution of j is more than the cost that it has to pay to get connected to i, it offers to pay this extra amount to i); If j is already connected to some other facility i', then its offer to facility i is equal to $\max(c_{i'j} - c_{ij}, 0)$ (i.e., the amount that j offers to pay to i is equal to the amount j would save by switching its facility from i' to i).

2. While $U \neq \emptyset$, increase the time, and simultaneously, for every city $j \in U$, increase the

parameter α_j at the same rate, until one of the following events occurs (if two events occur at the same time, we process them in an arbitrary order).

(a) For some unopened facility i, the total offer that it receives from cities is equal to the cost of opening i. In this case, we open facility i, and for every city j (connected or unconnected) which has a non-zero offer to i, we connect j to i. The amount that j had offered to i is now called the *contribution* of j toward i, and j is no longer allowed to decrease this contribution.

(b) For some unconnected city j, and some open facility i, $\alpha_j = c_{ij}$. In this case, connect city j to facility i and remove j from U.

Clearly the main issue in the facility location problem is to decide which facilities to open. Once this is done, each city should be connected to the closest open facility. Observe that Algorithm 2 makes greedy choices in deciding which facilities to open and once it opens a facility, it does not alter this decision. In this sense, it is also a greedy algorithm.

2.5 Analysis of Algorithm 2

The following fact should be obvious from the description of Algorithm 2.

Lemma 7 *The total cost of the solution found by Algorithm 2 is equal to the sum of α_j's.*

Now, as in the analysis of Algorithm 1, we need to find a number γ, such that for every star S, $\sum_{j \in S \cap \mathcal{C}} \alpha_j \leq \gamma c_S$. Such a γ will be an upper bound on the approximation ratio of the algorithm, since if for every facility i that is opened in the optimal solution and the collection A of cities that are connected to it, we write the inequality $\sum_{j \in A} \alpha_j \leq \gamma(f_i + \sum_{j \in A} c_{ij})$ and add up these inequalities, we will obtain that the cost of our solution is at most γ times the cost of the optimal solution.

2.5.1 Deriving the factor-revealing LP

Our proof follows the methodology of Section 2.3: express various constraints that are imposed by the problem or by the structure of the algorithm as inequalities and get a bound on the value of γ defined above by solving a series of linear programs.

Consider a star S consisting of a facility having opening cost f (with a slight misuse of the notation, we call this facility f), and k cities numbered 1 through k. Let d_j denote the connection cost between facility f and city j, and α_j denote the contribution of the city j at the end of Algorithm 2. We may assume without loss of generality that

$$\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_k. \tag{10}$$

We need more variables to capture the execution of Algorithm 2. For every i ($1 \leq i \leq k$), consider the situation of the algorithm at time $t = \alpha_i - \epsilon$, where ϵ is very small, i.e., just a moment before city i gets connected for the first time. At this time, each of the cities $1, 2, \ldots, i-1$ might be connected to a facility. For every $j < i$, if city j is connected to some facility at time t, let $r_{j,i}$ denote the connection cost between this facility and city j; otherwise, let $r_{j,i} := \alpha_j$. The latter case occurs if and only if $\alpha_i = \alpha_j$. It turns out that these variables (f, d_j's, α_j's, and $r_{j,i}$'s) are enough to write down some inequalities to bound the ratio of the sum of α_j's to the cost of S (i.e., $f + \sum_{j=1}^{k} d_j$).

First, notice that once a city gets connected to a facility, its contribution remains constant and it cannot revoke its contribution to a facility, so it can never get connected to another facility with a higher connection cost. This implies that for every j,

$$r_{j,j+1} \geq r_{j,j+2} \geq \cdots \geq r_{j,k}. \tag{11}$$

Now, consider time $t = \alpha_i - \epsilon$. At this time, the amount city j offers to facility f is equal to

$$\begin{aligned} &\max(r_{j,i} - d_j, 0) && \text{if } j < i, \text{ and} \\ &\max(t - d_j, 0) && \text{if } j \geq i. \end{aligned}$$

Notice that by the definition of $r_{j,i}$ this holds even if $j < i$ and $\alpha_i = \alpha_j$. It is clear from Algorithm 2 that the total offer of cities to a facility can never become larger than the opening cost of the facility. Therefore, for all i,

$$\sum_{j=1}^{i-1} \max(r_{j,i} - d_j, 0) + \sum_{j=i}^{k} \max(\alpha_i - d_j, 0) \leq f. \tag{12}$$

The triangle inequality is another important constraint that we need to use. Consider cities i and j with $j < i$ at time $t = \alpha_i - \epsilon$. Let f' be the facility j is connected to at time

t. By the triangle inequality and the definition of $r_{j,i}$, the connection cost $c_{f'i}$ between city i and facility f' is at most $r_{j,i} + d_i + d_j$. Furthermore, $c_{f'i}$ can not be less than t, since if it is, our algorithm could have connected the city i to the facility f' at a time earlier than t, which is a contradiction. Here we need to be careful with the special case $\alpha_i = \alpha_j$. In this case, $r_{j,i} + d_i + d_j$ is not more than t. If $\alpha_i \neq \alpha_j$, the facility f' is open at time t and therefore city i can get connected to it, if it can pay the connection cost. Therefore for every $1 \leq j < i \leq k$,

$$\alpha_i \leq r_{j,i} + d_i + d_j. \tag{13}$$

The above inequalities form the following factor-revealing LP.

$$\begin{aligned}
\text{maximize} \quad & \frac{\sum_{i=1}^{k} \alpha_i}{f + \sum_{i=1}^{k} d_i} \\
\text{subject to} \quad & \forall\, 1 \leq i < k:\ \alpha_i \leq \alpha_{i+1} \\
& \forall\, 1 \leq j < i < k:\ r_{j,i} \geq r_{j,i+1} \\
& \forall\, 1 \leq j < i \leq k:\ \alpha_i \leq r_{j,i} + d_i + d_j \\
& \forall\, 1 \leq i \leq k:\ \sum_{j=1}^{i-1} \max(r_{j,i} - d_j, 0) \\
& \qquad\qquad + \sum_{j=i}^{k} \max(\alpha_i - d_j, 0) \leq f \\
& \forall\, 1 \leq j \leq i \leq k:\ \alpha_j, d_j, f, r_{j,i} \geq 0
\end{aligned} \tag{14}$$

Notice that although the above optimization program is not written in the form of a linear program, it is easy to change it to a linear program by introducing new variables and inequalities.

Lemma 8 *If z_k denotes the solution of the factor-revealing LP, then for every star S consisting of a facility and k cities, the sum of α_j's of the cities in S in Algorithm 2 is at most $z_k c_S$.*

Proof: Inequalities 10, 11, 12, and 13 derived above imply that the values $\alpha_j, d_j, f, r_{j,i}$ that we get by running Algorithm 2 constitute a feasible solution of the factor-revealing LP. Thus, the value of the objective function for this solution is at most z_k. □

Lemmas 7 and 8 imply the following.

Table 1: Solution of the factor-revealing LP

k	$\max_{i \leq k} z_i$
10	1.54147
20	1.57084
50	1.58839
100	1.59425
200	1.59721
300	1.59819
400	1.59868
500	1.59898

Lemma 9 *Let z_k be the solution of the factor-revealing LP, and $\gamma := \sup_k\{z_k\}$. Then Algorithm 2 solves the metric facility location problem with an approximation factor of γ.*

2.5.2 Solving the factor-revealing LP

As mentioned earlier, the optimization program (14) can be written as a linear program. This enables us to use an LP-solver to solve the factor-revealing LP for small values of k, in order to compute the numerical value of γ. Table 1 shows a summary of results that are obtained by solving the factor-revealing LP using CPLEX. It seems from the experimental results that z_k is an increasing sequence that converges to some number close to 1.6 and hence $\gamma \approx 1.6$.

We are using the same idea as Lemma 6 in Section 2.3 to prove the upper bound of 1.61 on z_k.

Lemma 10 *Let z_k be the solution to the factor-revealing LP. Then for every k, $z_k \leq 1.61$.*

Proof: Using the same argument as in Lemma 6, we can assume, without loss of generality, that k is sufficiently large. Consider a feasible solution of the factor-revealing LP. Let $x_{j,i} := max(r_{j,i} - d_j, 0)$. The fourth inequality of the factor-revealing LP implies that for every $i \leq i'$,

$$(i' - i + 1)\alpha_i \leq \sum_{j=i}^{i'} d_j + f - \sum_{j=1}^{i-1} x_{j,i}. \tag{15}$$

Now, we define l_i as follows:

$$l_i = \begin{cases} p_2 k & \text{if } i \le p_1 k \\ k & \text{if } i > p_1 k \end{cases}$$

where p_1 and p_2 are two constants (with $p_1 < p_2$) that will be fixed later. Consider Inequality 15 for every $i \le p_2 k$ and $i' = l_i$, and divide both sides of this inequality by $(l_i - i + 1)$. By adding up these inequalities we obtain

$$\sum_{i=1}^{p_2 k} \alpha_i \le \sum_{i=1}^{p_2 k} \sum_{j=i}^{l_i} \frac{d_j}{l_i - i + 1} + (\sum_{i=1}^{p_2 k} \frac{1}{l_i - i + 1}) f - \sum_{i=1}^{p_2 k} \sum_{j=1}^{i-1} \frac{x_{j,i}}{l_i - i + 1}. \tag{16}$$

Now for every $j \le p_2 k$, let $y_j := x_{j,p_2 k}$. The second inequality of the factor-revealing LP implies that $x_{j,i} \ge y_j$ for every $j < i \le p_2 k$ and $x_{j,i} \le y_j$ for every $i > p_2 k$. Also, let $\zeta := \sum_{i=1}^{p_2 k} \frac{1}{l_i - i + 1}$. Therefore, inequality 16 implies

$$\sum_{i=1}^{p_2 k} \alpha_i \le \sum_{i=1}^{p_2 k} \sum_{j=i}^{l_i} \frac{d_j}{l_i - i + 1} + \zeta f - \sum_{i=1}^{p_2 k} \sum_{j=1}^{i-1} \frac{y_j}{l_i - i + 1}. \tag{17}$$

Consider the index $\ell \le p_2 k$ for which $2d_\ell + y_\ell$ has its minimum (i.e., for every $j \le p_2 k$, $2d_\ell + y_\ell \le 2d_j + y_j$). The third inequality of the factor-revealing LP implies that for $i = p_2 k + 1, \ldots, k$,

$$\alpha_i \le r_{\ell,i} + d_i + d_\ell \le x_{\ell,i} + 2d_\ell + d_i \le d_i + 2d_\ell + y_\ell. \tag{18}$$

By adding Inequality 18 for $i = p_2 k + 1, \ldots, k$ with Inequality 17 we obtain

$$\begin{aligned}
\sum_{i=1}^{k} \alpha_i &\le \sum_{i=1}^{p_2 k} \sum_{j=i}^{l_i} \frac{d_j}{l_i - i + 1} + (2d_\ell + y_\ell)(1 - p_2)k + \sum_{j=p_2 k+1}^{k} d_j - \sum_{i=1}^{p_2 k} \sum_{j=1}^{i-1} \frac{y_j}{l_i - i + 1} + \zeta f \\
&= \sum_{j=1}^{p_2 k} \zeta d_j - \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{d_j + y_j}{l_i - i + 1} + \sum_{j=p_2 k+1}^{k} (1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k - i + 1}) d_j \\
&\quad + (2d_\ell + y_\ell)(1 - p_2)k + \zeta f \\
&\le \sum_{j=1}^{p_2 k} \zeta d_j + \sum_{j=p_2 k+1}^{k} (1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k - i + 1}) d_j + \zeta f \\
&\quad + (2d_\ell + y_\ell) \left((1 - p_2)k - \frac{1}{2} \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{1}{l_i - i + 1} \right),
\end{aligned}$$

where the last inequality is a consequence of the inequality $2d_\ell + y_\ell \le 2d_j + y_j \le 2d_j + 2y_j$ for $j \le p_2 k$. Now, let $\zeta' := 1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k-i+1}$ and $\delta := (1 - p_2) - \frac{1}{2k} \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{1}{l_i - i + 1}$. Therefore, the above inequality can be written as follows:

$$\sum_{i=1}^{k} \alpha_i \leq \sum_{j=1}^{p_2 k} \zeta d_j + \sum_{j=p_2 k+1}^{k} \zeta' d_j + \zeta f + \delta(2d_\ell + y_\ell)k, \tag{19}$$

where

$$\zeta = \sum_{i=1}^{p_2 k} \frac{1}{l_i - i + 1} = \ln \frac{p_2(1-p_1)}{(p_2 - p_1)(1-p_2)} + o(1), \tag{20}$$

$$\zeta' = 1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k - i + 1} = 1 + \ln \frac{1-p_1}{1-p_2} + o(1), \tag{21}$$

$$\begin{aligned} \delta &= 1 - p_2 - \frac{1}{2k} \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{1}{l_i - i + 1} \\ &= \frac{1}{2}(2 - p_2 - p_2 \ln \frac{p_2}{p_2 - p_1} - \ln \frac{1-p_1}{1-p_2}) + o(1). \end{aligned} \tag{22}$$

Now if we choose p_1 and p_2 such that $\delta < 0$, and let $\gamma := \max(\zeta, \zeta')$ then inequality 19 implies that

$$\sum_{i=1}^{k} \alpha_i \leq (\gamma + o(1))(f + \sum_{i=1}^{k} d_j).$$

Using equations 20, 21, and 22, it is easy to see that subject to the condition $\delta < 0$, the value of γ is minimized when $p_1 \approx 0.439$ and $p_2 \approx 0.695$, which gives us $\gamma < 1.61$. □

Also, as in the proof of Theorem 5, we can use the optimal solution of the factor-revealing LP that is computed numerically (see Table 1) to construct an example on which our algorithm performs at least z_k times worse than the optimum. These results imply the following.

Theorem 11 *Algorithm 2 solves the facility location problem in time $O(n^3)$, where $n = \max(n_f, n_c)$, with an approximation ratio between 1.598 and 1.61.*

2.6 The tradeoff between facility and connection costs

We defined the cost of a solution in the facility location problem as the sum of the facility cost (i.e., total cost of opening facilities) and the connection cost. We proved in the previous section that Algorithm 2 achieves an overall performance guarantee of 1.61. However, sometimes it is useful to get different approximation guarantees for facility and connection

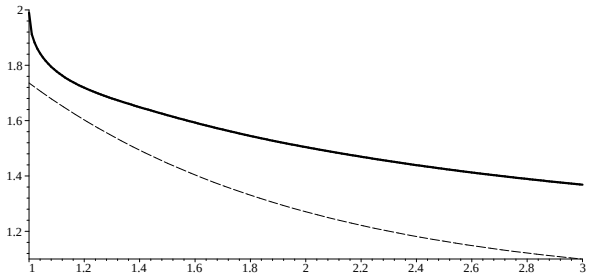

Figure 2: The tradeoff between γ_f and γ_c

costs. The following theorem gives such a guarantee. The proof is similar to the proof of Lemma 9.

Theorem 12 *Let $\gamma_f \geq 1$ and $\gamma_c := \sup_k\{z_k\}$, where z_k is the solution of the following optimization program.*

$$
\begin{aligned}
\text{maximize} \quad & \frac{\sum_{i=1}^{k} \alpha_i - \gamma_f f}{\sum_{i=1}^{k} d_i} \\
\text{subject to} \quad & \forall\, 1 \leq i < k:\ \alpha_i \leq \alpha_{i+1} \\
& \forall\, 1 \leq j < i < k:\ r_{j,i} \geq r_{j,i+1} \\
& \forall\, 1 \leq j < i \leq k:\ \alpha_i \leq r_{j,i} + d_i + d_j \\
& \forall\, 1 \leq i \leq k:\ \sum_{j=1}^{i-1} \max(r_{j,i} - d_j, 0) \\
& \qquad\qquad + \sum_{j=i}^{k} \max(\alpha_i - d_j, 0) \leq f \\
& \forall\, 1 \leq j \leq i \leq k:\ \alpha_j, d_j, f, r_{j,i} \geq 0
\end{aligned}
\tag{23}
$$

Then for every instance $\mathcal{I}$ of the facility location problem, and for every solution SOL for $\mathcal{I}$ with facility cost F_{SOL} and connection cost C_{SOL}, the cost of the solution found by Algorithm 2 is at most $\gamma_f F_{SOL} + \gamma_c C_{SOL}$.

We have computed the solution of the optimization program 23 for $k = 100$, and several values of γ_f between 1 and 3, to get an estimate of the corresponding γ_c's. The result is shown in the diagram in Figure 2. Every point (γ_f, γ_c') on the thick line in this diagram represents a value of γ_f, and the corresponding estimate for the value of γ_c. The dashed line shows the following lower bound, which can be proved easily by adapting the proof of Guha and Khuller [37] for hardness of the facility location problem.

Theorem 13 *Let γ_f and γ_c be constants with $\gamma_c < 1+2e^{-\gamma_f}$. Assume there is an algorithm $\mathcal{A}$ such that for every instance $\mathcal{I}$ of the metric facility location problem, $\mathcal{A}$ finds a solution whose cost is not more than $\gamma_f F_{SOL} + \gamma_c C_{SOL}$ for every solution SOL for $\mathcal{I}$ with facility and connection costs F_{SOL} and C_{SOL}. Then* $\mathbf{NP} \subseteq \mathrm{DTIME}[n^{O(\log \log n)}]$.

Similar tradeoff problems are considered by Charikar and Guha [14]. However, an important advantage that we get here is that all the inequalities $ALG \leq \gamma_f F_{SOL} + \gamma_c C_{SOL}$ are satisfied by a *single* algorithm. In Section 2.8, we will use the point $\gamma_f = 1$ of this tradeoff to design algorithms for other variants of the facility location problem. Other points of this tradeoff can also be useful in designing other algorithms based on our algorithm. For example, Mahdian, Ye, and Zhang [61] use the point $\gamma_f = 1.1$ of this tradeoff to obtain a 1.52-approximation algorithm for the metric facility location problem.

2.7 Experimental Results

We have implemented our algorithms, as well as the JV algorithm, using the programming language C. We have made four kinds of experiments. In all cases the solution of the algorithms is compared to the optimal solution of the LP-relaxation, computed using the package CPLEX to obtain an upper bound on the approximation factor of the algorithms.

The test bed of our first set of experiments consists of randomly generated instances on a $10,000 \times 10,000$ grid: In each instance, cities and facilities are points, drawn randomly from the grid. The connection cost between a city and a facility is set to be equal to the euclidean distance of the corresponding points. Furthermore, the opening cost of each facility is drawn uniformly at random from the integers between 0 and 9999.

For the second set of experiments, we have generated random graphs (according to the distribution $G(n,p)$) and assigned uniform random weights on the edges. Cities and facilities correspond to the nodes of this graph, and the connection cost between a city and a facility is defined to be the shortest path between the corresponding nodes. The opening costs of facilities are generated at random.

The instance sizes in both of the above types vary from 50 cities and 20 facilities to 400 cities and 150 facilities. For each size, 15 instances are generated and the average error

Table 2: Random Graphs and Random Points on a Grid

		Random Points on a Grid			Random Graphs		
n_c	n_f	JV	ALG 1	ALG 2	JV	ALG 1	ALG 2
50	20	1.0927	1.0083	1.0004	1.0021	1.0007	1.0001
100	20	1.0769	1.0082	1.0004	1.0014	1.0022	1.0
100	50	1.2112	1.0105	1.0013	1.0225	1.0056	1.0005
200	50	1.159	1.0095	1.001	1.0106	1.0094	1.0002
200	100	1.301	1.0105	1.0016	1.0753	1.0178	1.0018
300	50	1.1151	1.0091	1.0011	1.0068	1.0102	1.0002
300	80	1.1787	1.0116	1.001	1.0259	1.0171	1.0004
300	100	1.2387	1.0118	1.0014	1.0455	1.0185	1.0009
300	150	1.327	1.0143	1.0015	1.1365	1.0249	1.0018
400	50	1.0905	1.0092	1.0005	1.0044	1.012	1.0
400	100	1.8513	1.0301	1.0026	1.0313	1.0203	1.0003
400	150	1.8112	1.0299	1.0023	1.1008	1.0234	1.0009

of the algorithm (compared to the LP lower bound) is computed. The results of these experiments are shown in Table 2.

An Internet topology generator software, namely GT-ITM, is used to generate the third set of instances. GT-ITM is a software package for generating graphs that have a structure modeling the topology of the Internet [82]. This model is used because of the applications of facility location problems in network applications such as placing web server replicas [69]. In this model we consider transit nodes as potential facilities and stub nodes as cities. The connection cost is the distance produced by the generator. The opening costs are again random numbers. We have generated 10 instances for each of the 10 different instance sizes. The results are shown in Table 3.

We also tested all algorithms on 15 instances from [10], which is a library of test data sets for several operations research problems. Our results are shown in Table 4.

As we can see from the tables, Algorithm 2 behaves extremely well, giving almost no error in many cases. Algorithm 1 has an error of 7% on the worst instance and an average error of 2-3%. On the other hand, the JV algorithm has much larger error, sometimes as high as 100 %. We should also note that the running times of the three algorithms did not vary significantly. In the biggest instances of 1000 cities and 100 facilities all the algorithms

Table 3: GT-ITM Model

n_c	n_f	JV	ALG 1	ALG 2
100	20	1.004	1.0047	1.0001
160	20	1.5116	1.0612	1.0009
160	40	1.065	1.0063	1.0
208	52	2.2537	1.074	1.019
240	60	1.0083	1.0045	1.0001
300	75	1.8088	1.0478	1.0006
312	52	1.7593	1.0475	1.0008
320	32	1.0972	1.0015	1.0
400	100	1.0058	1.0048	1.0
416	52	1.0031	1.0048	1.0

Table 4: Instances from Operations Research library

n_c	n_f	JV	ALG 1	ALG 2
50	16	1.0642	1.0156	1.0
50	16	1.127	1.0363	1.0
50	16	1.1968	1.0258	1.0
50	16	1.2649	1.0258	1.0022
50	25	1.1167	1.006	1.0028
50	25	1.2206	1.0393	1.0
50	25	1.3246	1.0277	1.0
50	25	1.4535	1.0318	1.0049
50	50	1.3566	1.0101	1.0017
50	50	1.5762	1.0348	1.0061
50	50	1.7648	1.0378	1.0022
50	50	2.0543	1.0494	1.0075
1000	100	1.0453	1.0542	1.0023
1000	100	1.0155	1.0226	1.0
1000	100	1.0055	1.0101	1.0

ran in approximately 1-2 seconds. The implementation of the algorithms as well as all the data sets are available upon request. For other experimental results see [13].

2.8 Variants of the problem

In this section, we show that our algorithms can also be applied to several variants of the metric facility location problem.

2.8.1 The k-median problem

The *k-median problem* differs from the facility location problem in two respects: there is no cost for opening facilities, and there is an upper bound k, that is supplied as part of the input, on the number of facilities that can be opened. The *k-facility location problem* is a common generalization of k-median and the facility location problem. In this problem, we have an upper bound k on the number of facilities that can be opened, as well as costs for opening facilities. The k-median problem is studied extensively [6, 14, 15, 47] and the best known approximation algorithm for this problem, due to Arya et al. [6], achieves a factor of $3 + \epsilon$. It is also straightforward to adapt the proof of hardness of the facility location problem [37] to show that there is no $(1 + \frac{2}{e} - \epsilon)$-approximation algorithm for k-median, unless $\mathbf{NP} \subseteq \text{DTIME}[n^{O(\log \log n)}]$. Notice that this proves that k-median is a strictly harder problem to approximate than the facility location problem because the latter can be approximated within a factor of 1.61.

Jain and Vazirani [47] reduced the k-median problem to the facility location problem in the following sense: Suppose $\mathcal{A}$ is an approximation algorithm for the facility location problem. Consider an instance $\mathcal{I}$ of the problem with optimum cost OPT, and let F and C be the facility and connection costs of the solution found by $\mathcal{A}$. We call algorithm $\mathcal{A}$ a Lagrangian Multiplier Preserving α-approximation (or LMP α-approximation for short) if for every instance $\mathcal{I}$, $C \leq \alpha(OPT - F)$. Jain and Vazirani [47] show that an LMP α-approximation algorithm for the metric facility location problem gives rise to a 2α-approximation algorithm for the metric k-median problem. They have noted that this result also holds for the

k-facility location problem.

Lemma 14 [47] *An LMP α-approximation algorithm for the facility location problem gives a 2α-approximation algorithm for the k-facility location problem.*

Here we use Theorem 12 together with the scaling technique of Charikar and Guha [14] to give an LMP 2-approximation algorithm for the metric facility location problem based on Algorithm 2. This will result in a 4-approximation algorithm for the metric k-facility location problem, whereas the best previously known was 6 [47].

Lemma 15 *Assume there is an algorithm $\mathcal{A}$ for the metric facility location problem such that for every instance $\mathcal{I}$ and every solution SOL for $\mathcal{I}$, $\mathcal{A}$ finds a solution of cost at most $F_{SOL} + \alpha C_{SOL}$, where F_{SOL} and C_{SOL} are facility and connection costs of SOL, and α is a fixed number. Then there is an LMP α-approximation algorithm for the metric facility location problem.*

Proof: Consider the following algorithm: The algorithm constructs another instance $\mathcal{I}'$ of the problem by multiplying the facility opening costs by α, runs $\mathcal{A}$ on this modified instance $\mathcal{I}'$, and outputs its answer. It is easy to see that this algorithm is an LMP α-approximation. □

Now we only need to prove the following. The proof of this theorem follows the general scheme that is explained in Section 2.9.

Theorem 16 *For every instance $\mathcal{I}$ and every solution SOL for $\mathcal{I}$, Algorithm 2 finds a solution of cost at most $F_{SOL} + 2C_{SOL}$, where F_{SOL} and C_{SOL} are facility and connection costs of SOL.*

Proof: By Theorem 12 we only need to prove that the solution of the factor-revealing LP 23 with $\gamma_f = 1$ is at most 2. We first write the maximization program 23 as the following equivalent linear program.

$$\text{maximize} \quad \sum_{i=1}^{k} \alpha_i - f \tag{24}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i=1}^{k} d_i = 1 \\
& \forall\, 1 \le i < k :\ \alpha_i - \alpha_{i+1} \le 0 \\
& \forall\, 1 \le j < i < k :\ r_{j,i+1} - r_{j,i} \le 0 \\
& \forall\, 1 \le j < i \le k :\ \alpha_i - r_{j,i} - d_i - d_j \le 0 \\
& \forall\, 1 \le j < i \le k :\ r_{j,i} - d_i - g_{i,j} \le 0 \\
& \forall\, 1 \le i \le j \le k :\ \alpha_i - d_j - h_{i,j} \le 0 \\
& \forall\, 1 \le i \le k :\ \sum_{j=1}^{i-1} g_{i,j} + \sum_{j=i}^{k} h_{i,j} - f \le 0 \\
& \forall\, i, j :\ \alpha_j, d_j, f, r_{j,i}, g_{i,j}, h_{i,j} \ge 0
\end{aligned}$$

We need to prove an upper bound of 2 on the solution of the above LP. Since this program is a maximization program, it is enough to prove the upper bound for any relaxation of the above program. Numerical results (for a fixed value of k, say $k = 100$) suggest that removing the second, third, and seventh inequalities of the above program does not change its solution. Therefore, we can relax the above program by removing these inequalities. Now, it is a simple exercise to write down the dual of the relaxed linear program and compute its optimal solution. This solution corresponds to multiplying the third, fourth, fifth, and sixth inequalities of the linear program 24 by $1/k$, and the first one by $(2 - 1/k)$, and adding up these inequalities. This gives an upper bound of $2 - 1/k$ on the value of the objective function. Thus, for $\gamma_f = 1$, we have $\gamma_c \le 2$. In fact, γ_c is precisely equal to 2, as shown by the following solution for the program 23.

$$\alpha_i = \begin{cases} 2 - \frac{1}{k} & i = 1 \\ 2 & 2 \le i \le k \end{cases}$$

$$d_i = \begin{cases} 1 & i = 1 \\ 0 & 2 \le i \le k \end{cases}$$

$$r_{j,i} = \begin{cases} 1 & j = 1 \\ 2 & 2 \le j \le k \end{cases}$$

$$f = 2(k-1)$$

This example shows that the above analysis of the factor-revealing LP is tight. □

Lemma 15 and Theorem 16 provide an LMP 2-approximation algorithm for the metric facility location problem. This result improves all the results in Jain and Vazirani [47], and gives straightforward algorithms for some other problems considered by Charikar et al [16].

Notice that Theorem 13 shows that finding an LMP $(1 + \frac{2}{e} - \epsilon)$-approximation for the metric facility location problem is hard. Also, the integrality gap examples found by Guha [36] show that Lemma 14 is tight. This shows that one cannot use Lemma 14 as a black box to obtain a smaller factor than $2 + \frac{4}{e}$ for k-median problem. Note that $3 + \epsilon$ approximation is already known [6] for the problem. Hence if one wants to beat this factor using the Lagrangian relaxation technique then it will be necessary to look into the underlying LMP algorithm as already been done by Charikar and Guha [14].

2.8.2 Facility location game

An important consideration, in cooperative game theory, while distributing the cost of a shared utility, is that the cost shares should satisfy the *coalition participation constraint*, i.e., the total cost share of any subset of the users shall not be larger than their stand-alone cost of receiving the service, so as to prevent this subset from seceding. In general, this turns out to be a stringent condition to satisfy. For the facility location problem, Goemans and Skutella [34] showed that such a cost allocation is only possible for a very special case. Furthermore, intractability sets in as well, for instance, in the case of the facility location problem, computing the optimal cost of serving a set of users is **NP**-hard.

In [49] Jain and Vazirani relax this notion: for a constant k, ensure that the cost share of any subset is no more than k times its stand-alone cost. They also observe that LP-based approximation algorithms directly yield a cost sharing method compatible with this relaxed notion. However, this involves solving an LP, as in the case of LP-rounding. We observe that our facility location algorithms automatically yield such a cost sharing method, with $k = 1.861$ and $k = 1.61$ respectively, by defining the cost share of city j to be α_j.

2.8.3 Arbitrary demands

In this version, for each city j, a non-negative integer demand d_j, is specified. An open facility i can serve this demand at the cost of $c_{ij}d_j$. The best way to look at this modification is to reduce it to unit demand case by making d_j copies of city j. This reduction suggests that we need to change our algorithms , so that each city j raises its contribution α_j at rate d_j. Note that the modified algorithms still have the same running time in more general cases, where d_j is fractional or exponentially large, and achieve the same approximation ratio.

2.8.4 Fault tolerant facility location with uniform connectivity requirements

We are given a connectivity requirement r_j for each city j, which specifies the number of open facilities that city j should be connected to. We can see that this problem is closely related to the set multi-cover problem, in the case that every set can be picked at most once [70]. The greedy algorithm for set-cover can be adapted for this variant of the multi-cover problem achieving the same approximation factor. We can use the same approach to deal with the fault tolerant facility location: The mechanism of raising dual variables and opening facilities is the same as in our initial algorithms. The only difference is that city j stops raising its dual variable and withdraws its contribution from other facilities, when it is connected to r_j open facilities. We can show that when all r_j's are equal, our algorithms can still achieve the approximation factor of 1.861 and 1.61.

2.8.5 Facility location with penalties

In this version we are not required to connect every city to an open facility; however, for each city j, there is a specified penalty, p_j, which we have to pay, if it is not connected to any open facility. We can modify our algorithms for this problem as follows: If α_j reaches p_j before j is connected to any open facility, the city j stops raising its dual variable and keeps its contribution equal to its penalty until it is either connected to an open facility or all remaining cities stop raising their dual variables. At this point, the algorithm terminates and unconnected cities remain unconnected. Using the linear programming

formulation introduced in Charikar et al. ([16] inequalities (4.6)-(4.10)), we can show that the approximation ratio and running time of our modified algorithms have not changed.

2.8.6 Robust facility location

In this variant, we are given a number l and we are only required to connect $n_c - l$ cities to open facilities. This problem can be reduced to the previous one via Lagrangian relaxation. Very recently, Charikar et al. [16] proposed a primal-dual algorithm, based on JV algorithm, which achieves an approximation ratio of 3. As they showed, the linear programming formulation of this variant has an unbounded integrality gap. In order to fix this problem, they use the technique of parametric pruning, in which they guess the most expensive facility in the optimal solution. After that, they run JV algorithm on the pruned instance, where the only allowable facilities are those that are not more expensive than the guessed facility. Here we can use the same idea, using Algorithm 1 rather than the JV algorithm. Using a proof similar to the proof of the Theorem 3.2 in [16], we can prove that this algorithm solves the robust facility location problem with an approximation factor of 2.

2.8.7 Dealing with capacities

In real applications, it is not usually the case that the cost of opening a facility is independent of the number of cities it will serve. But we can assume that we have *economy of scales*, i.e., the cost of serving each city decreases when the number of cities increases (since the publication of this part of the thesis, this problem has also been studied in [41]). In order to capture this property, we define the following variant of the capacitated metric facility location problem. For each facility i, there is an initial opening cost f_i. After facility i is opened, it will cost s_i to serve each city. This variant can be solved using metric uncapacitated facility location problem: We just have to change the metric such that for each city j and facility i, $c'_{ij} = c_{ij} + s_i$. Clearly, c' is also a metric and the solution of the metric uncapacitated version to this problem can be interpreted as a solution to the original problem with the same cost.

We can reduce the variant of the capacitated facility location problem in which each facility can be opened many times [47] to this problem by defining $s_i = f_i/u_i$. If in the

solution to this problem k cities are connected to facility i, we open this facility $\lceil k/u_i \rceil$ times. The cost of the solution will be at most two times the original cost so any α-approximation for the uncapacitated facility location problem can be turned into a 2α-approximation for this variant of the capacitated version. We can also use the same technique as in [47] to give a factor 3-approximation algorithm for this problem based on the LMP 2-approximation algorithm for uncapacitated facility location problem.

2.9 Discussion

The method of dual fitting can be seen as an implementation of the primal-dual schema in which, instead of relaxing complementary slackness conditions (which is the most common way of implementing the schema), we relax feasibility of the dual. However, we prefer to reserve the term primal-dual for algorithms that produce feasible primal and dual solutions.

Let us show how the combination of dual fitting with factor-revealing LP applies to the set cover problem. The duality-based restatement of the greedy algorithm (see [79]) is: All elements in the universal set U increase their dual variables uniformly. Each element contributes its dual towards paying for the cost of each of the sets it is contained in. When the total contribution offered to a set equals its cost, the set is picked. At this point, the newly covered elements freeze their dual variables and withdraw their contributions from all other sets. As stated in the introduction, the latter (important) step ensures that the primal is fully paid for by the dual. However, we might not get a feasible dual solution. To make the dual solution feasible we look for the smallest positive number Z, so that when the dual solution is shrunk by a factor of Z, it becomes feasible. An upper bound on the approximation factor of the algorithm is obtained by maximizing Z over all possible instances.

Clearly Z is also the maximum factor by which any set is over-tight. Consider any set S. We want to see what is the worst factor, over all sets and over all possible instances of the problem, by which a set S is over-tight. Let the elements in S be $1, 2, \cdots, k$. Let x_i be the dual variable corresponding to the element i at the end of the algorithm. Without loss of generality we may assume that $x_1 \leq x_2 \leq \cdots \leq x_k$. It is easy to see that at time

$t = x_i^-$, total duals offered to S is at least $(k - i + 1)x_i$. Therefore, this value cannot be greater than the cost of the set S (denoted by c_S). So, the optimum solution of the following mathematical program gives an upper bound on the value of Z. (Note that c_S is a variable not a constant).

$$\begin{aligned} \text{maximize} \quad & \frac{\sum_{i=1}^{k} x_i}{c_S} \\ \text{subject to} \quad & \forall 1 \leq i < k: \; x_i \leq x_{i+1} \\ & \forall 1 \leq i \leq k: \; (k - i + 1)x_i \leq c_S \\ & \forall 1 \leq i \leq k: \; x_i \geq 0 \\ & c_S \geq 1 \end{aligned} \tag{25}$$

The above optimization program can be turned into a linear program by adding the constraint $c_S = 1$ and changing the objective function to $\sum_{i=1}^{k} x_i$. We call this linear program the *factor-revealing LP*. Notice that the factor-revealing LP has nothing to do with the LP formulation of the set cover problem; it is only used in order to analyze this particular algorithm. This is the important distinction between the factor-revealing LP technique, and other LP-based techniques in approximation algorithms.

One advantage of reducing the analysis of the approximation guarantee of an algorithm to obtaining an upper bound on the optimal solution to a factor-revealing LP is that one can introduce emperical experimentation into the latter task. This can also help decide which aspects of the execution of the algorithm to introduce into the factor-revealing LP to obtain the best possible bound on the performance of the algorithm, e.g., we needed to introduce the variables $r_{j,i}$ in Section 2.5.1 in order to get a good bound on the approximation ratio of Algorithm 2.

In general, this technique is not guaranteed to yield a tight analysis of the algorithm, since the algorithm may be performing well not because of local reasons but for some global reasons that are difficult to capture in a factor-revealing LP. In the case of set cover, this method not only produces a tight analysis, but the factor-revealing LP also helps produce a tight example for the algorithm. From any feasible solution x of factor-revealing LP 25, one can construct the following instance: There are k elements $1, \ldots, k$, a set $S = \{1, \ldots, k\}$ of cost $1 + \epsilon$ which is the optimal solution, and sets $S_i = \{i\}$ of cost x_i for $i = 1, \ldots, k$. It is

easy to verify that the greedy algorithm gives a solution that is $\sum x_i$ times worse than the optimal on this instance. Picking x to be the optimal solution, we get a tight example, and also show that the approximation ratio of the greedy algorithm is precisely equal H_n, the optimal solution of the factor-revealing LP.

Finally, in terms of practical impact, what is the significance of improving the approximation guarantee for facility location from 3 to 1.81 or 1.61 when practitioners are seeking algorithms that come within 2% to 5% of the optimal? The superior experimental results of our algorithms, as compared with the JV algorithm, seem to provide the answer and to support the argument made in [79] (Preface, page IX) that the approximation factor should be viewed as a "measure that forces us to explore deeper into the combinatorial structure of the problem and discover more powerful tools for exploiting this structure" and the observation that "sophisticated algorithms do have the error bounds of the desired magnitude, 2% to 5%, on typical instances, even though their worst case error bounds are much higher".

CHAPTER 3

MARKET EQUILIBRIUM VIA A PRIMAL-DUAL-TYPE ALGORITHM

3.1 Introduction

We present the first polynomial time algorithm for the linear version of an old problem, first defined in 1891 by Irving Fisher [71]: Consider a market consisting of buyers and divisible goods. The money possessed by buyers and the amount of each good are specified. Also specified are utility functions of buyers, which are assumed to be linear (Fisher's original statement assumed concave utility functions). The problem is to compute prices for the goods such that even if each buyer is made optimally happy, relative to these prices, there is no deficiency or surplus of any of the goods, i.e. the market clears.

Our work partially answers the open question raised in [24], of computing equilibrium prices for the case of linear utilities for the Arrow-Debreu model, in which there is no demarcation between buyers and sellers; Fisher's model is a special case of the Arrow-Debreu model. Besides raising this question, [24] also gave polynomial time algorithms for the linear case of the Arrow-Debreu model in case the number of goods or agents is bounded, and initiated an algorithmic theory of market equilibria. Well before this, [66] had considered the question of polynomial time solvability of equilibria and gave a complexity-theoretic framework for establishing evidence of intractability for such issues.

Before our work, the following folklore result was known: there is a PTAS for computing equilibrium prices for the linear version of Fisher's model. This follows from Eisenberg and Gale's [27] result, giving a convex program for computing equilibrium prices, and the use of the ellipsoid algorithm. A corollary of our work is that equilibrium prices have small denominators. As a consequence, the ellipsoid algorithm will compute equilibrium prices exactly in polynomial time. Alternatively, Jain [45] uses diophantine approximation to

show that this approach leads to an exact polynomial time algorithm. Jain has also given a convex program for the linear version of the Arrow-Debreu model and used the ellipsoid algorithm and diophantine approximation to obtain a polynomial time algorithm for this case as well, thereby settling the open problem of [24]. Prior to his result, [46, 25, 51] had given FPTAS's for the same problem.

Fisher's work was done contemporarily and independently of Walras' pioneering work [80] on modeling market equilibria. Through the ensuing years, the study of market equilibria has occupied center stage within Mathematical Economics. Its crowning achievement came with the work of Arrow and Debreu [5] which established the existence of equilibrium prices in a very general setting, through the use of Kakutani's fixed point theorem. The First Welfare Theorem, showing Pareto optimality of allocations obtained at equilibrium prices, provides important social justification for this theory.

The highly non-constructive nature of Arrow and Debreu's proof naturally raised questions of efficient computability of equilibrium prices. Despite impressive progress on this issue, e.g., Scarf's work [71], which has been useful in many applications [23], polynomial time algorithms have evaded researchers, even for the case of linear utility functions.

For the case of linear utilities, it is natural to seek an algorithmic answer in the theory of linear programming. However, there does not seem to be any natural linear programming formulation for this problem. The main contribution of this work is to point out that despite this, a suitable adaptation of the primal-dual schema yields a combinatorial solution to Fisher's problem. Our algorithm is modeled after Kuhn's primal-dual algorithm for the bipartite matching problem [56]. At the heart of the primal-dual schema lies the following powerful paradigm: the algorithm starts with trivial solutions to the primal and dual LP's corresponding to the given problem and alternately improves these solutions until a termination criterion is met (see [79] for a detailed discussion); the current primal suggests how to improve the dual, and vice versa. We identify two processes: the "primal process" updates the amount of each good sold to each buyer and the "dual process" updates prices of goods. Throughout the algorithm the prices are such that buyers have surplus money left over. Each update decreases this surplus, and when it vanishes, the prices are right

for the market to clear exactly. Our proof of correctness involves new combinatorial facts: understanding how the min-cut changes in a network derived from a bipartite graph as the capacities of its source edges are increased.

Prior to this work, [47] had used the above-stated paradigm, of two processes making improvements relative to each other, outside of the setting of linear programming. This naturally raises the question of whether there is a formal mathematical framework in which such "primal-dual-type" algorithms can be set and analyzed. Subsequent to this work [46, 26, 78] used the techniques introduced here for different generalizations of this problem.

3.2 Problem

Consider a market consisting of a set B of *buyers* and a set A of divisible *goods*. Assume $|A| = n$ and $|B| = n'$. We are given for each buyer i the amount e_i of money she possesses and for each good j the amount b_j of this good. In addition, we are given the utility functions of the buyers. Our critical assumption is that these functions are linear. Let u_{ij} denote the utility derived by i on obtaining a unit amount of good j. Given prices $p_1, \ldots, p_n$ of the goods, it is easy to compute baskets of goods (there could be many) that make buyer i happiest. We will say that $p_1, \ldots, p_n$ are *market clearing* prices if after each buyer is assigned such a basket, there is no surplus or deficiency of any of the goods. Our problem is to compute such prices in polynomial time.

First observe that w.l.o.g. we may assume that each b_j is unit – by scaling the u_{ij}'s appropriately. The u_{ij}'s and e_i's are in general rational; by scaling appropriately, they may be assumed to be integral. Now, it turns out that there is a market clearing price iff each good has a potential buyer (one who derives nonzero utility from this good). Moreover, if there is a solution, it is unique [30, 27]. We assume that we are in the latter case.

In [30], the problem is formulated as

$$\begin{array}{lll} \text{maximize} & \sum_{i=1}^{n'} m_i \log u_i & \\ \text{subject to} & u_i = \sum_{j=1}^{n} u_{ij} x_{ij} & \forall i \in B \\ & \sum_{i=1}^{n'} x_{ij} \leq 1 & \forall j \in A \\ & x_{ij} \geq 0 & \forall i \in B, \forall j \in A \end{array} \tag{26}$$

where x_{ij} is the amount of good j allocated to buyer i. The price of good j in the equilibrium is equal to the optimum value of the dual variable corresponding to the second constraint in the above program. We will show in Section 3.5 that equilibrium prices are rational numbers with small denominators and therefore they can be found in polynomial time using ellipsoid method. In the rest of the chapter, we will develop a combinatorial algorithm for finding the market equilibrium prices in polynomial time.

3.3 High level idea of the algorithm

Let $\boldsymbol{p} = (p_1, \ldots, p_n)$ denote a vector of prices. If at these prices buyer i is given good j, she derives u_{ij}/p_i amount of utility per unit amount of money spent. Clearly, she will be happiest with goods that maximize this ratio. Define her *bang per buck* to be $\alpha_i = \max_j\{u_{ij}/p_j\}$; clearly, for each $i \in B, j \in A$, $\alpha_i \geq u_{ij}/p_j$. If there are several goods maximizing this ratio, she is equally happy with any combination of these goods. This motivates defining the following bipartite graph, G. Its bipartition is (A, B) and for $i \in B, j \in A$ (i, j) is an edge in G iff $\alpha_i = u_{ij}/p_j$. We will call this graph the *equality subgraph* and its edges the *equality edges*.

Any goods sold along the edges of the equality subgraph will make buyers happiest, relative to the current prices. Computing the largest amount of goods that can be sold in this manner, without exceeding the budgets of buyers or the amount of goods available (assumed unit for each good), can be accomplished by computing max-flow in the following network: Direct edges of G from A to B and assign a capacity of infinity to all these edges. Introduce source vertex s and a directed edge from s to each vertex $j \in A$ with a capacity of p_j. Introduce sink vertex t and a directed edge from each vertex $i \in B$ to t with a capacity

of e_i. The network is clearly a function of the current prices $\boldsymbol{p}$ and will be denoted $N(\boldsymbol{p})$. The algorithm maintains the following throughout:

Invariant: The prices $\boldsymbol{p}$ are such that $(s, A \cup B \cup t)$ is a min-cut in $N(\boldsymbol{p})$.

The Invariant ensures that, at current prices, all goods can be sold. The only eventuality is that buyers may be left with surplus money. The algorithm raises prices systematically, always maintaining the Invariant, so that surplus money with buyers keeps decreasing. When the surplus vanishes, market clearing prices have been attained. This is equivalent to the condition that $(s \cup A \cup B, t)$ is also a min-cut in $N(\boldsymbol{p})$, i.e., max-flow in $N(\boldsymbol{p})$ equals the total amount of money possessed by the buyers.

Remark: With this setup, we can define our market equilibrium problem as an optimization problem: find prices $\boldsymbol{p}$ under which network $N(\boldsymbol{p})$ supports maximum flow.

How do we pick prices so the Invariant holds at the start of the algorithm? The following two conditions guarantee this:

- The initial prices are low enough prices that each buyer can afford all the goods. Fixing prices at $1/n$ suffices, since the goods together cost one unit and all e_i's are integral.

- Each good j has an interested buyer, i.e., has an edge incident at it in the equality subgraph. Compute α_i for each buyer i at the prices fixed in the previous step and compute the equality subgraph. If good j has no edge incident, reduce its price to

$$p_j = \max_i \left\{ \frac{u_{ij}}{\alpha_i} \right\}.$$

The iterative improvement steps follow the spirit of the primal-dual schema: The "primal" variables are the flows in the edges of $N(\boldsymbol{p})$ and the "dual" variables are the current prices. The current flow suggests how to improve the prices and vice versa.

For $S \subseteq B$, define its money $m(S) = \sum_{i \in B} e_i$. W.r.t. prices $\boldsymbol{p}$, for set $S \subseteq A$, define its money $m(S) = \sum_{j \in A} p_j$; the context will clarify the price vector $\boldsymbol{p}$. For $S \subseteq A$, define its *neighborhood in* $N(\boldsymbol{p})$

$$\Gamma(S) = \{j \in B \mid \exists i \in S \text{ with} (i, j) \in G\}.$$

By the assumption that each good has a potential buyer, $\Gamma(A) = B$. The Invariant can now be more clearly stated.

Lemma 17 *For given prices* $\boldsymbol{p}$ *network* $N(\boldsymbol{p})$ *satisfies the Invariant iff*

$$\forall S \subseteq A : m(S) \leq m(\Gamma(S)).$$

Proof: The forward direction is trivial, since under max-flow (of value $m(A)$) every set $S \subseteq A$ must be sending $m(S)$ amount of flow to its neighborhood.

Let's prove the reverse direction. Assume $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ is a min-cut in $N(\boldsymbol{p})$, with $A_1, A_2 \subseteq A$ and $B_1, B_2 \subseteq B$. The capacity of this cut is $m(A_2) + m(B_1)$. Now, $\Gamma(A_1) \subseteq B_1$, since otherwise the cut will have infinite capacity. Moving A_1 and $\Gamma(A_1)$ to the t side also results in a cut. By the condition stated in the Lemma, the capacity of this cut is no larger than the previous one. Therefore this is also a min-cut in $N(\boldsymbol{p})$. Hence the Invariant holds. □

If the Invariant holds, it is easy to see that there is a unique maximal set $S \subseteq A$ such that $m(S) = m(\Gamma(S))$. Say that this is the *tight set* w.r.t. prices $\boldsymbol{p}$. Clearly the prices of goods in the tight set cannot be increased without violating the Invariant. Hence our algorithm only raises prices of goods in the *active subgraph* consisting of the bipartition $(A - S, B - \Gamma(S))$. We will say that the algorithm *freezes* the subgraph $(S, \Gamma(S))$. Observe that in general, the bipartite graph $(S, \Gamma(S))$ may consist of several connected components (w.r.t. equality edges). Let these be $(S_1, T_1), \ldots, (S_k, T_k)$.

Clearly, as soon as prices of goods in $A - S$ are raised, edges (i, j) with $i \in \Gamma(S)$ and $j \in (A - S)$ will not remain in the equality subgraph anymore. We will assume that these edges are dropped. Before proceeding further, we must be sure that these changes do not violate the Invariant. This follows from:

Lemma 18 *If the Invariant holds and* $S \subseteq A$ *is the tight set, then each good* $j \in (A - S)$ *has an edge, in the equality subgraph, to some buyer* $i \in (B - \Gamma(S))$.

Proof: Since the Invariant holds, $j \in (A - S)$ must have an equality graph edge incident

at it. If all such edges are incidents at buyers in $\Gamma(S)$, then $\Gamma(S \cup j) = \Gamma(S)$ and therefore

$$m(S \cup j) > m(S) = m(\Gamma(S)) = m(\Gamma(S \cup j)).$$

This contradicts the fact that the Invariant holds. □

We would like to raise prices of goods in the active subgraph in such a way that the equality edges in it are retained. This is ensured by multiplying prices of all these goods by x and gradually increasing x, starting with $x = 1$. To see that this has the desired effect, observe that (i, j) and (i, l) are both equality edges iff

$$\frac{p_j}{p_l} = \frac{u_{ij}}{u_{il}}.$$

The algorithm raises x, starting with $x = 1$, until one of the following happens:

- **Event 1:** A set $R \neq \emptyset$ goes tight in the active subgraph.
- **Event 2:** An edge (i, j) with $i \in (B - \Gamma(S))$ and $j \in S$ becomes an equality edge. (Observe that as prices of goods in $A - S$ are increasing, goods in S are becoming more and more desirable to buyers in $B - \Gamma(S)$, which is the reason for this event.)

If Event 1 happens, we redefine the active subgraph to be $(A - (S \cup R), B - \Gamma(S \cup R))$, and proceed with the next iteration. Suppose Event 2 happens and that $j \in S_l$. Because of the new equality edge (i, j), $\Gamma(S_l) = T_l \cup i$. Therefore S_l is not tight anymore. Hence we move (S_l, T_l) into the active subgraph.

To complete the algorithm, we simply need to compute the smallest values of x at which Event 1 and Event 2 happen, and consider only the smaller of these. For Event 2, this is straightforward. Below we build an algorithm for Event 1.

3.4 Finding tight sets

Let $\boldsymbol{p}$ denote the current price vector (i.e. at $x = 1$). We first present a lemma that describes how the min-cut changes in $N(x \cdot \boldsymbol{p})$ as x increases. Throughout this section, we will use the function m to denote money w.r.t. prices $\boldsymbol{p}$. W.l.o.g. assume that w.r.t. prices $\boldsymbol{p}$ the

tight set in G is empty (since we can always restrict attention to the active subgraph, for the purposes of finding the next tight set). Define

$$x^* = \min_{\emptyset \neq S \subseteq A} \frac{m(\Gamma(S))}{m(S)},$$

the value of x at which a nonempty set goes tight. Let S^* denote the tight set at prices $x^* \cdot \boldsymbol{p}$. If $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ is a cut in the network, we will assume that $A_1, A_2 \subseteq A$ and $B_1, B_2 \subseteq B$.

Lemma 19 *W.r.t. prices $x \cdot \boldsymbol{p}$:*

- *if $x \leq x^*$ then $(s, A \cup B \cup t)$ is a min-cut.*
- *if $x > x^*$ then $(s, A \cup B \cup t)$ is not a min-cut. Moreover, if $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ is a min-cut in $N(x \cdot \boldsymbol{p})$ then $S^* \subseteq A_1$.*

Proof: Suppose $x \leq x^*$. By definition of x^*,

$$\forall S \subseteq A : x \cdot m(S) \leq m(\Gamma(S)).$$

Therefore by Lemma 17, w.r.t. prices $x \cdot \boldsymbol{p}$, the Invariant holds. Hence $(s, A \cup B \cup t)$ is a min-cut.

Next suppose that $x > x^*$. Since $x \cdot m(S^*) > x^* \cdot m(S^*) = m(\Gamma(S^*))$, w.r.t. prices $x \cdot \boldsymbol{p}$, the cut $(s \cup S^* \cup \Gamma(S^*), t)$ has strictly smaller capacity than the cut $(s \cup A \cup B, t)$. Therefore the latter cannot be a min-cut.

Let $S^* \cap A_2 = S_2$ and $S^* - S_2 = S_1$. Suppose $S_2 \neq \emptyset$. Clearly $\Gamma(S_1) \subseteq B_1$ (otherwise the cut will have infinite capacity). If $m(\Gamma(S_2) \cap B_2) < x \cdot m(S_2)$, then by moving S_2 and $\Gamma(S_2)$ to the s side, we can get a smaller cut, contradicting the minimality of the cut picked. In particular, if $S_2 = S^*$, then this inequality must hold, leading to a contradiction. Hence, $S_1 \neq \emptyset$. Furthermore,

$$m(\Gamma(S_2) \cap B_2) \geq x \cdot m(S_2) > x^* m(S_2).$$

On the other hand,

$$m(\Gamma(S_2) \cap B_2) + m(\Gamma(S_1)) \leq x^*(m(S_2) + m(S_1)).$$

The two imply that

$$\frac{m(\Gamma(S_1))}{m(S_1)} < x^*,$$

contradicting the definition of x^*. Hence $S_2 = \emptyset$ and $S^* \subseteq A_1$. □

Remark: A more complete statement for the first part of Lemma 19, which is not essential for our purposes, is: If $x < x^*$, then $(s, A \cup B \cup t)$ is the unique min-cut in $N(x \cdot \boldsymbol{p})$. If $x = x^*$, then the min-cuts are obtained by moving a bunch of connected components of $(S^*, \Gamma(S^*))$ to the s-side of the cut $(s, A \cup B \cup t)$.

Lemma 20 *Let $x = m(B)/m(A)$ and suppose that $x > x^*$. If $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ be a min-cut in $N(x \cdot \boldsymbol{p})$ then A_1 must be a proper subset of A.*

Proof: If $A_1 = A$, then $B_1 = B$ (otherwise this cut has ∞ capacity), and $(s \cup A \cup B, t)$ is a min-cut. But for the chosen value of x, this cut has the same capacity as $(s, A \cup B \cup t)$. Since $x > x^*$, the latter is not a min-cut by Lemma 19. Hence, A_1 is a proper subset of A. □

Lemma 21 *x^* and S^* can be found using n max-flow computations.*

Proof: Let $x = m(B)/m(A)$. Clearly, $x \geq x^*$. If $(s, A \cup B \cup t)$ is a min-cut in $N(x \cdot \boldsymbol{p})$, then by Lemma 19 $x^* = x$. If so, $S^* = A$.

Otherwise, let $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ be a min-cut in $N(x \cdot \boldsymbol{p})$. By Lemmas 19 and 20, $S^* \subseteq A_1 \subset A$. Therefore, it is sufficient to recurse on the smaller graph $(A_1, \Gamma(A_1))$. □

Initialization:

$\forall j \in A, p_j \leftarrow 1/n; \quad \forall i \in B, \alpha_i \leftarrow \min_j u_{ij}/p_j;$

Compute equality subgraph G;

$\forall j \in A$ **if** $degree_G(j) = 0$ **then** $p_j \leftarrow \max_i u_{ij}/\alpha_i$;

Recompute G;

$(F, F') \leftarrow (\emptyset, \emptyset)$ (The frozen subgraph); $(H, H') \leftarrow (A, B)$ (The active subgraph);

while $H \neq \emptyset$ **do**

- $x \leftarrow 1$;
- Define $\forall j \in H$, price of j to be $p_j x$;
- Raise x continuously until one of two events happens:
- **if** $S \subseteq H$ *becomes tight* **then**
  - Move $(S, \Gamma(S))$ from (H, H') to (F, F');
  - Remove all edges from F' to H;
- **if** *an edge* $(i, j), i \in H'$, $j \in F$ *attains equality,* $\alpha_i = u_{ij}/p_j$, **then**
  - Add (i, j) to G;
  - Move connected component of j from (F, F') to (H, H') ;

Algorithm 1: The Basic Algorithm

3.5 Termination with market clearing prices

Let M be the total money possessed by the buyers and let f be the max-flow computed in network $N(\boldsymbol{p})$ at current prices $\boldsymbol{p}$. Thus $M - f$ is the *surplus money* with the buyers. Let us partition the running of the algorithm into *phases*, each phase terminates with the occurrence of Event 1. Each phase is partitioned into *iterations* which conclude with a new edge entering the equality subgraph. We will show that f must be proportional to the number of phases executed so far, hence showing that the surplus must vanish in bounded time.

Let $U = \max_{i \in B, j \in A}\{u_{ij}\}$ and let $\Delta = nU^n$.

Lemma 22 *At the termination of a phase, the prices of goods in the newly tight set must*

be rational numbers with denominator $\leq \Delta$.

Proof: Let S be the newly tight set and consider the equality subgraph induced on the bipartition $(S, \Gamma(S))$. Assume w.l.o.g. that this graph is connected (otherwise we prove the lemma for each connected component of this graph). Let $j \in S$. Pick a subgraph in which j can reach all other vertices $j' \in S$. Clearly, at most $2|S| \leq 2n$ edges suffice. If j reaches j' with a path of length $2l$, then $p_{j'} = ap_j/b$ where a and b are products of l utility parameters (u_{ik}'s) each. Since alternate edges of this path contribute to a and b, we can partition the u_{ik}'s in this subgraph into two sets such that a and b use u_{ik}'s from distinct sets. These considerations lead easily to showing that $m(S) = p_j c/d$ where $c \leq \Delta$. Now,

$$p_j = m(\Gamma(S))d/c,$$

hence proving the lemma. □

Lemma 23 *Each phase consists of at most* n *iterations.*

Proof: Each iteration brings goods from the tight set to the active subgraph. Clearly this cannot happen more than n times without a set going tight. □

Lemma 24 *Consider two phases* P *and* P'*, not necessarily consecutive, such that good* j *lies in the newly tight sets at the end of* P *as well as* P'*. Then the increase in the price of* j*, going from* P *to* P'*, is* $\geq 1/\Delta^2$.

Proof: Let the prices of j at the end of P and P' be p/q and r/s, respectively. Clearly, $r/s > p/q$. By Lemma 22, $q \leq \Delta$ and $r \leq \Delta$. Therefore the increase in price of j,

$$\frac{r}{s} - \frac{p}{q} \geq \frac{1}{\Delta^2}.$$

□

Lemma 25 *After* k *phases,* $f \geq k/\Delta^2$.

Proof: Consider phase P and let j be a good that lies in the newly tight set at the end of this phase. Let P' be the last phase, earlier than P, such that j lies in the newly tight set at the end of P' as well. If there is no such phase (because P is the first phase in which j appears in a tight set), then let P' be the start of the algorithm. Let us charge to P the entire increase in the price of j, going from P' to P (even though this increase takes place gradually over all the intermediate phases). By Lemma 24, this is $\geq 1/\Delta^2$. In this manner, each phase can be charged $1/\Delta^2$. The lemma follows. □

Corollary 26 *Algorithm 1 terminates with market clearing prices in at most $M\Delta^2$ phases, and executes $O(Mn^2\Delta^2)$ max-flow computations.*

Remark: The upper bound given above is quite loose, e.g., it is easy to shave off a factor of n by giving a tighter version of Lemma 23.

3.6 Establishing polynomial running time

For a given flow f in the network $N(\boldsymbol{p})$, define the *surplus* of buyer i, $\gamma_i(\boldsymbol{p}, f)$, to be the residual capacity of the edge (i, t) with respect to f, which is equal to m_i minus the flow sent through the edge (i, t).

In this section we are trying to speed up Algorithm 1 by increasing the prices of goods adjacent only to "high-surplus" buyers. However, the surplus of a buyer might be different for two different maximum flows in the same graph. Therefore, we will restrict ourselves to a specific flow so that the surplus of a buyer is well-defined. The following definition serves this purpose:

Define the surplus vector $\boldsymbol{\gamma}(\boldsymbol{p}, f) := (\gamma_1(\boldsymbol{p}, f), \gamma_2(\boldsymbol{p}, f), \ldots, \gamma_n(\boldsymbol{p}, f))$. Let $\|v\|$ denote the l_2 norm of vector v.

Definition 27 Balanced flow *For any given $\boldsymbol{p}$, a maximum flow that minimizes $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$ over all choices of f is called a* balanced flow.

If $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\| < \|\boldsymbol{\gamma}(\boldsymbol{p}, f')\|$, then we say f is more balanced *than f'.*

For a given $\boldsymbol{p}$ and a flow f in $N(\boldsymbol{p})$, let $R(\boldsymbol{p}, f)$ be the residual network of $N(\boldsymbol{p})$ with respect to the flow f. We will give a characterization of balanced flow via $R(\boldsymbol{p}, f)$

Lemma 28 *Let f and f' be any two maximum flows in $N(\boldsymbol{p})$. If $\gamma_i(\boldsymbol{p}, f') < \gamma_i(\boldsymbol{p}, f)$ for some $i \in B$, then there exist a $j \in B$ such that $\gamma_j(\boldsymbol{p}, f) < \gamma_j(\boldsymbol{p}, f')$ and*

1. *There is a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$.*
2. *There is a path from i to j in $R(\boldsymbol{p}, f') \setminus \{s, t\}$.*

Proof: Consider the flow $f' - f$. It defines a feasible circulation in the network $R(\boldsymbol{p}, f)$. Since $\gamma_i(\boldsymbol{p}, f') < \gamma_i(\boldsymbol{p}, f)$, there is a positive flow along the edge (i, t) in $f' - f$. By following this flow all the way back to t in the circulation, one can find a node j, such that there is a positive flow from t to j and then to i in $f' - f$. Since both flows are maximum, s is an isolated vertex in $f' - f$ and this flow does not go through s. Now, $f' - f$ is a valid flow in $R(\boldsymbol{p}, f)$ and therefore there exists a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. Moreover having a positive flow from t to j implies that $\gamma_j(\boldsymbol{p}, f) < \gamma_j(\boldsymbol{p}, f')$. A similar argument shows that there is also a path from i to j in $R(\boldsymbol{p}, f') \setminus \{s, t\}$. □

Lemma 29 *If $a \geq b_i \geq 0, i = 1, 2, \ldots, n$ and $\delta \geq \sum_{j=1}^{n} \delta_j$ where $\delta, \delta_j \geq 0, j = 1, 2, \ldots, n$, then $\|(a, b_1, b_2, \ldots, b_n)\|^2 \leq \|(a + \delta, b_1 - \delta_1, b_2 - \delta_2, \ldots, b_n - \delta_n)\|^2 - \delta^2$.*

Proof:

$$(a + \delta)^2 + \sum_{i=1}^{n} (b_i - \delta_i)^2 - a^2 - \sum_{i=1}^{n} {b_i}^2 \geq \delta^2 + 2a(\delta - \sum_{i=1}^{n} \delta_i) \geq 0$$

□

The following property characterizes all balanced flows. It defines the flows for which there is no path from a low-surplus node to a high-surplus node in the residual network.

Property 1 There is no path from node $i \in B$ to node $j \in B$ in $R(\boldsymbol{p}, f)$ if surplus of i is more than surplus of j in $N(\boldsymbol{p}, f)$.

Theorem 30 *A maximum-flow f is balanced iff it has Property 1.*

Proof: Suppose f is a balanced flow. Let $\gamma_i(\boldsymbol{p}, f) > \gamma_j(\boldsymbol{p}, f)$ for some i and j, and suppose for the sake of contradiction, that there is a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. Then one can send a circulation of positive value along $t \to j \to i \to t$ in $R(\boldsymbol{p}, f)$, decreasing γ_i and increasing γ_j. ¿From Lemma 29 the resulting flow is more balanced than f, contradicting the fact that f is a balanced flow.

To prove the other direction, suppose that f is not a balanced maximum flow. Let f' be a balanced flow. Since $\|\boldsymbol{\gamma}(\boldsymbol{p}, f')\| < \|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$, there exists $i \in B$ such that $\gamma_i(\boldsymbol{p}, f') < \gamma_i(\boldsymbol{p}, f)$.

By Lemma 28, there exists $j \in B$ such that $\gamma_j(\boldsymbol{p}, f) < \gamma_j(\boldsymbol{p}, f')$ and there is a path from j to i in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. Since f has Property 1 , $\gamma_i(\boldsymbol{p}, f) \leq \gamma_j(\boldsymbol{p}, f)$. The above three inequalities imply $\gamma_i(\boldsymbol{p}, f') < \gamma_j(\boldsymbol{p}, f)$. But again by Lemma 28, there is a path from i to j in $R(\boldsymbol{p}, f') \setminus \{s, t\}$ so f' doesn't have Property 1. This contradicts the assumption that f' is a balanced flow by what we proved in the first half the theorem. □

The following lemma provides our main tool for proving polynomial running time of Algorithm 2. We will use it to prove an upper bound on the l_2-norm of the surplus vector of buyers at the end of every phase.

Lemma 31 *If f and f^* are respectively a feasible and a balanced flow in $N(\boldsymbol{p})$ and for some $i \in B$ and $\delta > 0$ $\gamma_i(f) = \gamma_i(f^*) + \delta$, then there is a flow f' and for some k there is a set of vertices $i_1, i_2, \cdots, i_k$ and values $\delta_1, \delta_2, \cdots, \delta_k$ such that*

- $\sum_{l=1}^{k} \delta_l \leq \delta$
- $\gamma_i(f') = \gamma_i(f) - \delta$
- $\gamma_{i_l}(f') = \gamma_{i_l}(f) + \delta_l$
- $\gamma_i(f') \geq \gamma_{i_l}(f')$.

Proof: Consider $f^* - f$ in $R(\boldsymbol{p}, f)$ and in a similar fashion as in Lemma 28 follow the incoming flow of node i until you reach s or the node i itself. Let f' be the flow

augmented from f by sending back the flow through all these circulations and paths. We will have $\gamma_i(f') = \gamma_i(f) - \delta$ and for a set of vertices $i_1, i_2, \cdots, i_k$ and values $\delta_1, \delta_2, \cdots, \delta_k$ s.t. $\sum_{l=1}^{k} \delta_l \leq \delta$, we have $\gamma_{i_l}(f') = \gamma_{i_l}(f) + \delta_l$. Moreover, since f^* is balanced, $\gamma_i(f') = \gamma_i(f^*) \geq \gamma_{i_l}(f^*) \geq \gamma_{i_l}(f')$. □

Corollary 32 $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|^2 \geq \|\boldsymbol{\gamma}(\boldsymbol{p}, f^*)\|^2 + \delta^2$.

Proof: By Lemma 29, $\|\boldsymbol{\gamma}(f, \boldsymbol{p})\|^2 \geq \|\boldsymbol{\gamma}(f', \boldsymbol{p})\|^2 + \delta^2$ and since f^* is a balanced flow in $N(\boldsymbol{p})$, $\|\boldsymbol{\gamma}(f', \boldsymbol{p})\|^2 \geq \|\boldsymbol{\gamma}(f^*, \boldsymbol{p})\|^2$. □

Corollary 33 *For any given $\boldsymbol{p}$, all balanced flows in $N(\boldsymbol{p})$ have the same surplus vector.*

As a result, one can define the surplus vector for a given price as $\boldsymbol{\gamma}(\boldsymbol{p}) := \boldsymbol{\gamma}(\boldsymbol{p}, f)$ where f is the balanced flow in $N(\boldsymbol{p})$. This vector can be found by computing a balanced flow in the equality subgraph in the following way:

Corollary 34 *For a given price vector $\boldsymbol{p}$ the balanced flow can be computed by at most n max-flow computation.*

Proof: We will use the divide and conquer method. Let $m_{\text{avg}} := \frac{\sum_{i=1}^{n'} m_i - \sum_{j=1}^{n} p_j}{n'}$. Compute the maximum flow in the equality subgraph after subtracting m_{avg} from the capacity of each edge adjacent t. Let (S, T) be the maximal min-cut in that network. $s \in S, t \in T$. If $A \subset S$ then the current maximum flow is balanced. Otherwise, let N_1 and N_2 be the networks induced by $T \cup \{s\}$ and $S \cup \{t\}$ respectively. Claim that the union of balanced flows in N_1 and N_2 is a balanced flow in N.

In order to prove the claim, it is enough (from Theorem 30) to show that the surplus of all buyers in N_1 (in a balanced flow) is at least m_{avg} and that of all buyers in N_2 is at most m_{avg}. We will prove the former; the proof of the latter is similar. Let L be the set of all buyers in N_1 with the lowest surplus, say s. Suppose $s < m_{avg}$. Let K be the set of goods reachable by L in the residual network of N_1 w.r.t a balanced flow. By Theorem 30 no other buyers are reachable from L in this network. Hence, $\Gamma_{N_1}(K) \subseteq L$. Since the surplus

of all buyers in L is s, $m(K) = m(L) - s|L| > m(L) - m_{\text{avg}}|L|$. This is a contradiction to the fact that (S, T) was a min-cut. □

In a set of feasible vectors, a vector v is called *min-max fair* iff for every feasible vector u and an index i such that $u_i < v_i$ there is a j for which $u_j < v_j$ and $v_j < v_i$. Similarly, v is *max-min fair* iff $u_i > v_i$ implies that there is a j for which $u_j < v_j$ and $v_j > v_i$.

Remark: The surplus vector of a balanced flow is both min-max and max-min fair.

3.6.1 The polynomial time algorithm

The main idea of Algorithm 2 is that it tries to reduce $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$ in every phase. Intuitively, this goal is achieved by finding a set of high-surplus buyers in the balanced flow and increasing the prices of goods in which they are interested. If a subset becomes tight as a result of this increase, we have reduced $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$ because the surplus of a formerly high-surplus buyer is dropped to zero. The other event that can happen is that a new edge is added to the equality subgraph. In that case, this edge will help us to make the surplus vector more balanced: we can reduce the surplus of high-surplus buyers and increase the surplus of low-surplus ones. This operation will result in the reduction of $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|$.

The algorithm starts with finding a price vector that does not violate the invariant. The rest of the algorithm is partitioned into *phases*. In each phase, we have an active graph (H, H') with $H \subset B$ and $H' \subset A$ and we increase the prices of goods in H' like Algorithm 1. Let δ be the maximum surplus in B. The subset H is initially the set of buyers whose surplus is equal to δ. H' is the set of goods adjacent to buyers in H.

Each phase is divided into *iterations*. In each iteration, we increase the prices of goods in H' until either a new edge joins the equality subgraph or a subset becomes tight. If a new edge is added to the equality subgraph, we recompute the balanced flow f. Then we add to H all vertices that can reach a member of H in $R(\boldsymbol{p}, f) \setminus \{s, t\}$. If a subset becomes tight as a result of increase of the prices, then the phase terminates.

Consider a phase in the execution of Algorithm 2. Define $\boldsymbol{p}_i$ and H_i to be the price vector and the set of nodes in H after executing the i'th iteration in that phase. Let H_0

Initialization:
$\forall j \in A, p_j \leftarrow 1/n$;
$\forall i \in B, \alpha_i \leftarrow \min_j u_{ij}/p_j$;
Define $G(A, B, E)$ with $(i, j) \in E$ iff $\alpha_i = u_{ij}/p_j$;
$\forall j \in A$ **if** $degree_G(j) = 0$ **then** $p_j \leftarrow \max_i u_{ij}/\alpha_i$;
Recompute G; $\delta = M$;
repeat
  Compute a balanced flow f in G;
  Define δ to be the maximum surplus in B;
  Define H to be the set of buyers with surplus δ ;
  repeat
    Let H' be the set of neighbors of H in A ;
    Remove all edges from $B \setminus H$ to H';
    $x \leftarrow 1$; Define $\forall j \in H'$, price of j to be $p_j x$;
    Raise x continuously until one of the two events happens:
    Event 1: An edge $(i, j), i \in H, j \in A \setminus H'$ attains equality, $\alpha_i = u_{ij}/p_j$;
      Add (i, j) to G;
      Recompute f;
      In the residual network corresponding to f in G, define I to be the set of buyers that can reach H; $H \leftarrow H \cup I$;
    Event 2: $S \subseteq H$ becomes tight;
  until *some subset* $S \subseteq H$ *is tight*;
until A *is tight* ;

denote the set of nodes in H before the first iteration.

Lemma 35 *The number of iterations executed in a phase is at most n. Moreover, in every phase, there is an iteration in which surplus of at least one of the vertices is reduced by at least $\frac{\delta}{n}$.*

Proof: Let k denote the number of iterations in the phase. Every time an edge is added to the equality subgraph, $|H'|$ is increased by at least one. Therefore k is at most n.

Define $\delta_i = \min_{j \in H_i}(\gamma_j(\boldsymbol{p_i}))$, for $0 \leq i \leq k$. $\delta_0 = \delta$ and the phase ends when the surplus of one buyer in H becomes zero so $\delta_k = 0$. So there is an iteration t in which $\delta_t - \delta_{t-1} \geq \frac{\delta}{n}$.

Consider the residual network corresponding to the balanced flow computed at iteration t. In that network, every vertex in $H_t \setminus H_{t-1}$ can reach a vertex in H_{t-1} and therefore, by Theorem 30, its surplus is greater than or equal to the surplus of that vertex. This means that minimum surplus δ_t is achieved by a vertex i in H_{t-1}. Hence, the surplus of vertex i is decreased by at least $\delta_{t-1} - \delta_t$ during iteration t. □

Lemma 36 *If $\boldsymbol{p}_0$ and $\boldsymbol{p}^*$ are price vectors before and after a phase, $\|\boldsymbol{\gamma}(\boldsymbol{p}^*)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2(1-\frac{1}{n^3})$.*

Proof: In every iteration we increase prices of goods in H or add new edges to the equality subgraph. Moreover, all the edges of the network that are deleted in the beginning of a phase have zero flow. Therefore, the balanced flow computed at iteration i is a feasible flow for $N(\boldsymbol{p}_{i+1})$. Therefore by Lemma 32 $\|\boldsymbol{\gamma}(\boldsymbol{p}_0)\| \geq \|\boldsymbol{\gamma}(\boldsymbol{p}_1)\| \geq \|\boldsymbol{\gamma}(\boldsymbol{p}_2)\| \geq \cdots \geq \|\boldsymbol{\gamma}(\boldsymbol{p}_k)\|$. Furthermore, by the previous lemma there is an iteration t and node i such that $\gamma_i(\boldsymbol{p}_{t-1}) - \gamma_i(\boldsymbol{p}_t) \geq \frac{\delta}{n}$. So we have: $\|\boldsymbol{\gamma}(\boldsymbol{p}_t)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_{t-1})\|^2 - (\frac{\delta}{n})^2$ which means that

$$\|\boldsymbol{\gamma}(\boldsymbol{p}^*)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_t)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_{t-1})\|^2 - (\frac{\delta}{n})^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2 - (\frac{\delta}{n})^2.$$

Now $\|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2 \leq \delta^2 n$ so

$$\|\boldsymbol{\gamma}(\boldsymbol{p}^*)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}_0)\|^2(1 - \frac{1}{n^3}).$$

□

Remark: The upper bound given above is quite loose e.g. one can reduce the upper bound to $(1 - \frac{1}{n^2})$ by considering all iterations t in which $\delta_{t-1} - \delta_t > 0$.

By the bound given in the above, it is easy to see that after $O(n^2)$ phases, $\|\gamma(p)\|^2$ is reduced to at most half of its previous value. In the beginning, $\|\gamma(p)\|^2 \leq M^2$. Once the value of $\|\gamma(p)\|^2 \leq \frac{1}{\Delta^4}$, the algorithm takes at most one more step. This is because Lemma 22 and Lemma 24 hold for Algorithm 2 as well. Hence, the number of phases is at most

$$O\left(n^2 \log\left(\Delta^4 M^2\right)\right) = O\left(n^2(\log n + n \log U + \log M)\right)$$

As noted before, the number of iterations in each phase is at most n. Each iteration requires at most $O(n)$ max-flow computations. Hence we get:

Theorem 37 *Algorithm 2 executes at most*

$$O\left(n^4(\log n + n \log U + \log M)\right)$$

max-flow computations and finds market clearing prices.

CHAPTER 4

ON THE CONDUCTANCE OF THE POWER-LAW NETWORKS

4.1 Introduction

The Internet is a computational system of immense complexity that was not designed by a single entity, but emerged from the *ad hoc* interactions of many entities on the basis of ground rules that were deliberately open and minimally restrictive. As a result, it is the first computational artifact that must be studied by observation, measurement, and the development and validation of hypotheses, models and falsifiable theories —in a manner not unlike the one in which other sciences approach the universe, the brain, the cell, and the market. This work aims to contribute to the growing corpus of mathematical results and techniques that are pertinent to this novel, within Computer Science, research mode.

Since connectivity is a network's *raison d' être*, it is no surprise that various aspects of the Internet's connectivity (such as degrees, diameter, cuts, and tolerance to element failures) have been the subject of intense study, measurement, and speculation, see e.g. [9, 28, 7, 44, 43, 18]. Here, we address two sophisticated aspects of connectivity that are particularly relevant to the Internet, namely *conductance* and *frugality*.

As the Internet grows, extensive measurements show a clear congestion increase in the core and relate this to network performance (e.g. see [43, 44, 77, 32]). Therefore, one of the most crucial questions one can ask is, *how does the congestion at the Internet's core scale with the number of nodes?* In other words, if we assume unit traffic between all nodes (more accurately, traffic weighted by some measure of the size of each node, typically captured by its degree), how do the loads on the edges balance? Since the Internet is a very sparse graph (average degree between 3 and 4 [29]), there are two extremes to consider here: In constant degree trees one expects that congestion (traffic in the worst edge) grows as n^2

with the nodes, while in constant-degree expanders this growth is close to the theoretical minimum, $n \log n$.

The observation in [28] that the degree distribution of the Internet has heavy tails, or is "scale-free" (has deviations from the mean that decrease only polynomially, forming a straight line in log-log plot) has brought center-stage several models of random graphs that exhibit such degree distributions; it is thus compelling to estimate the asymptotic growth of congestion in scale-free random graph models. In this work we consider the model of *growth with preferential attachment* in which an arriving node connects with d edges to previously arrived nodes chosen with probability proportional to the degrees of the latter [9, 57, 3, 11]. We show that, for $d \geq 2$, *almost all scale-free graphs* in this model have *constant conductance*; as a corollary, approximate multicommodity flow algorithms imply routing with congestion $O(n \log n)$. An immediate additional implication is constant spectral gap between the first and second eigenvalues of the stochastic normalization of the adjacency matrix of the graph. This is also in accordance with measurement: [31] found the second eigenvalue of the Internet topology between .8 and .9 (and of its core between .6 and .7) for snapshots between 1997 and 2002 during which the network has grown by a factor of 20. Elsewhere, [81] measure a gap for the (symmetrized, degree-homogenized) graph of the world-wide web, again over a long period of observations.

A persistent technical difficulty in treating graphs grown with preferential connectivity arises from the inhomogeneity and dependencies between edges [57, 7, 11]. The crux of our proof is in establishing a bound that is invariant of time (shifting argument in Lemma 39). Prior to our work, [32] and [19] had shown conductance and spectral gap $\Omega(1/\log n)$ and $\Omega(1/\text{poly}\log n)$ respectively, for structural scale-free random graph models (Erdős-Renyi adaptations for skewed degree sequences). Structural scale-free random graph models avoid all dependencies between vertices, and are hence easier to analyze [2, 18, 64, 19]. However, in those models certain bad events occur almost surely and inverse logarithmic factors appear unavoidable. More relevant to this work, [21] had shown conductance $\Omega(1/\log n)$ for the growth with preferential connectivity model considered here, and for constant d much larger than 2. In view of the above, our result (Theorem 38) is the first constant characterization

of these fundamental measures.

4.2 Definitions and Notation

In this section we use the notation $G_{d,n}$ to denote graphs grown with preferential attachment. We will use the following definitions for these random graph processes. $G_{1,n} = T_n$ is a tree grown in n time steps, one vertex at each time step. Its vertices are called *mini-vertices* and they are named after the time that they arrive. At time 1 the tree consists of a single mini-vertex with a self loop. At time t, $2 \leq t \leq n$, mini-vertex t arrives and attaches with a single edge to a mini-vertex t', $t' < t$, chosen among all mini-vertices with probability proportional to their degrees at time $t-1$. We call the mini-vertex t' to which mini-vertex t attached the *father* of t (let the father of 1 be 1, by convention). For $d \geq 2$, the graph $G_{d,n}$ is generated by first growing a tree T_{dn} and then, for $1 \leq \tau \leq n$, contracting mini-vertices $d\tau - i$, for $0 \leq i \leq d-1$. Self-loops and multiple edges are preserved. We call the vertex of $G_{d,n}$ that resulted by contracting mini-vertices $d\tau - i$, $0 \leq i \leq d-1$, vertex τ. Thus, for every $S \subset [n]$, we may associate a subset of vertices of the graph $G_{d,n}$ and a subset of mini-vertices of the tree T_{dn} in the natural way: mini-vertex $d\tau - i$, $0 \leq i \leq d-1$, is associated with S if and only if $\tau \in S$.

Let $G(V, E)$ be an undirected multigraph with self-loops. The degree of a vertex $u \in V$ is denoted by $d_G(u)$, where each self-loop contributes 1 to the degree. For $S \subset V$, the *volume* of S is $\mathrm{vol}_G(S) = \sum_{u \in S} d_G(u)$. For $S \subset V$, the *cutset* of S, $C_G(S, \bar{S})$, is the multiset of edges with one endpoint in S and the other endpoint is $\bar{S}$. The *edge expansion* ρ_G and the *conductance* Φ_G of the graph G are

$$\rho_G = \min_{S \subset V, |S| \leq |V|/2} \frac{|C_G(S, \bar{S})|}{|S|}$$

and

$$\Phi_G = \min_{S \subset V, \mathrm{vol}_G(S) \leq \mathrm{vol}_G(V)/2} \frac{|C_G(S, \bar{S})|}{\mathrm{vol}_G(S)} \quad .$$

4.3 Main Theorem

In Theorem 38 we establish constant conductance. Immediate implications for routing congestion and spectral gap are in Corollaries 40 and 41. The key technical ingredient in the proof of Theorem 38 is the bound of Lemma 39, which is time-invariant. This Lemma is established by a careful shifting argument that makes full use of the structure of the underlying evolutionary process.

Theorem 38 *There is a positive constant α such that, for any constant $d \geq 2$, the random graph $G_{d,n}$ has edge expansion α and conductance $\frac{\alpha}{d+\alpha}$, almost surely. In particular, for any non-negative constant $c < 2(d-1) - 4\alpha - 1$*

$$\Pr\left[\rho_{G_{d,n}} < \alpha\right] \leq o(n^{-c})$$

and

$$\Pr\left[\Phi_{G_{d,n}} \leq \frac{\alpha}{d+\alpha}\right] \leq o(n^{-c}) \quad .$$

Proof: Let us first bound conductance in terms of edge expansion. Let $S \subset [n]$ be a set with $\mathrm{vol}_{G_{d,n}}(S) \leq dn/2$. Since, by construction, every vertex associated with S contributes d to the total degree of S, we have $d|S| \leq \mathrm{vol}(S) \leq d|S| + C_{G_{d,n}}(S, \bar{S})$. The left hand side of this inequality implies $|S| \leq n/2$. Now the right hand side can be used to bound conductance by

$$\begin{aligned}
\Phi_{G_{d,n}} = & \min_{\substack{S \subset V \\ \mathrm{vol}_{G_{d,n}}(S) \leq dn/2}} \frac{C_{G_{d,n}}(S, \bar{S})}{\mathrm{vol}_{G_{d,n}}(S)} \\
\geq & \min_{\substack{S \subset V \\ \mathrm{vol}_{G_{d,n}}(S) \leq dn/2}} \frac{C_{G_{d,n}}(S, \bar{S})}{d|S| + C_{G_{d,n}}} \\
\geq & \frac{\rho}{d+\rho}
\end{aligned}$$

Now let us bound edge expansion. We will use a counting argument. Let us fix $k \leq n/2$ and let us fix a set $S \subset [n]$ with $|S| = k$. Let T_{dn} be the tree from which $G_{n,d}$ was generated.

Say that a mini-vertex t, $1 \leq t \leq dn$ is GOOD if and only if either t is associated with S and the father of t is associated with $\bar{S}$, or t is associated with $\bar{S}$ and the father of t is associated with S. Say that a mini-vertex is BAD if and only if it is not GOOD. Realize that mini-vertex 1 is BAD, by convention. Realize also that if 1 belongs to S (resp. $\bar{S}$) then the first mini-vertex in $\bar{S}$ (resp. S) is always GOOD, by construction. Now let us fix the set $A \subset [dn]$ of GOOD mini-vertices, so that $|A| \leq \alpha k$. By Lemma 39

$$\Pr[\bigwedge_{\substack{t \in [dn] \\ t \notin A}} t \text{ is BAD}] \leq \frac{\binom{dk}{\alpha k}}{\binom{dn-\alpha k}{dk-\alpha k}} .$$

There are $\binom{n}{k}$ choices for S. Once S is fixed, there are at most $\alpha k \binom{dn}{\alpha k}$ choices for A. Finally, because of the way we construct the graph we do not need to argue about singletons, therefore we need to consider $2 \leq k \leq n/2$. The above imply

$$\begin{aligned}
\Pr[\rho_{G_{d,n}} < \alpha] &= \Pr[\exists \text{ a BAD SET } S] \\
&\leq \sum_{k=2}^{n/2} \binom{n}{k} \alpha k \binom{dn}{\alpha k} \frac{\binom{dk}{\alpha k}}{\binom{dn-\alpha k}{dk-\alpha k}} \\
&\leq \sum_{k=2}^{n/2} \alpha k \binom{dn}{\alpha k}\binom{dk}{\alpha k}\binom{(d-1)n-\alpha k}{(d-1)k-\alpha k}^{-1} && \text{, using } \binom{n}{k}\binom{(d-1)n-\alpha k}{(d-1)k-\alpha k} \leq \binom{dn-\alpha k}{dk-\alpha k} \\
&\leq \sum_{k=2}^{n/2} \alpha k \left(\frac{n}{k}\right)^{\alpha k} \left(\frac{ed}{\alpha}\right)^{2\alpha k} \left(\frac{(d-1)k-\alpha k}{(d-1)n-\alpha k}\right)^{(d-1)k-\alpha k} && \text{, using the bound } \left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k \\
&\leq \sum_{k=2}^{n/2} \alpha k \left(\frac{n}{k}\right)^{\alpha k} \left(\frac{ed}{\alpha}\right)^{2\alpha k} \left(\frac{k}{n}\right)^{(d-1)k-\alpha k} \\
&\leq \sum_{k=2}^{n/2} \alpha k \left(\frac{ed}{\alpha}\right)^{2\alpha k} \left(\frac{k}{n}\right)^{(d-1-2\alpha)k} .
\end{aligned}$$

There are $O(n)$ terms in the above summation. So we can bound the sum by $o(n^{-c})$, if we bound the leading term by $o(n^{-(c+1)})$. It can be seen that, for α small enough, all terms are smaller than the term for $k=2$, provided $d-1-2\alpha > 0$, which is true for α small enough

and $d \geq 2$. Hence we need to bound $n^{-(2d-1-4\alpha)}$. This can be bounded by $n^{-(c+1)}$, for c as in the statement of the theorem. $\square$

Lemma 39 *For a fixed subset $S \subset [n]$, $|S| = k$, and for a fixed subset $A \subset [dn]$, $|A| \leq \alpha k$, the probability that all mini-vertices associated with $[dn] \setminus A$ are* BAD *in $G_{d,n}$ is at most $\binom{dk}{\alpha k} / \binom{dn-\alpha k}{dk-\alpha k}$.*

Proof: Let A_1 be the mini-vertices in A associated with S and A_2 be the mini-vertices in A associated with $\bar{S}$. Let $|A_1| = k_1$ and $|A_2| = k_2$, with $k_1 + k_2 = |A|$. Let $x_1 < x_2 < \ldots < x_{dk-k_1}$ be the mini-vertices associated with S that do not belong to A. We may write $x_i = y_i + z_i + 1$, where y_i is the total number of mini-vertices that arrived prior to x_i and belong to A and z_i is the total number of mini-vertices that arrived prior to x_i and belong to $[dn] \setminus A$. Let $\bar{x}_1 < \bar{x}_2 < \ldots < \bar{x}_{dn-dk-k_2}$ be the mini-vertices associated with $\bar{S}$ that do not belong to A. We may write $\bar{x}_i = \bar{y}_i + \bar{z}_i + 1$, where $\bar{y}_i$ is the total number of mini-vertices that arrived prior to $\bar{x}_i$ and belong to A and $\bar{z}_i$ is the total number of mini-vertices that arrived prior to $\bar{x}_i$ and belong to $[dn] \setminus A$.

Now let us assume that the only GOOD mini-vertices are the ones belonging to A. Thus all mini-vertices associated with $[dn] \setminus A$ are BAD, and hence $x_1, \ldots, x_{dk-k1}$ as well as $\bar{x}_1, \ldots, \bar{x}_{dn-dk-k_2}$ are BAD. Recall also that the first mini-vertex 1 is not associated with A, since, by definition, 1 is BAD. Now realize that $\bigcup_{i=1}^{dk-k_1} \{z_i\} \cup \bigcup_{i=1}^{dn-dk-k_2} \{\bar{z}_i\} = \{0, 1, \ldots, dn - |A| - 1\}$, and, equivalently,

$$\bigcup_{i=1}^{dk-k_1} \{z_i + 1\} \cup \bigcup_{i=1}^{dn-dk-k_2} \{\bar{z}_i + 1\} = [dn - |A|]. \tag{27}$$

Let us now proceed to bound the probability that all mini-vertices associated with $[dn] \setminus A$ are BAD, given that all mini-vertices in A are GOOD. First realize that the total volume of the graph when mini-vertex t arrives is $2(t-1) - 1$, for $t \geq 2$. If $t = x_i$ (resp. $t = \bar{x}_i$), we can write this as

$$2(z_i + y_i) - 1 \qquad \text{resp.} \quad 2(\bar{z}_i + \bar{y}_i) - 1 \ . \tag{28}$$

We shall bound the probability that a mini-vertex in $[dn] \setminus A \setminus S$ is BAD and a mini-vertex in $[dn] \setminus A \setminus \bar{S}$ is BAD separately. Assume, without loss of generality, that $1 \notin S$,

otherwise rename S and $\bar{S}$ (this lemma does not require $k < n/2$). It now follows that, by connectivity, the first mini-vertex in S is necessarily GOOD and thus this mini-vertex belongs to A. When x_i arrives, the total volume of S is contributed by: (a)All BAD mini-vertices that arrived prior to x_i and are associated with S, where each such mini-vertex contributes degree 2 and there are $i-1$ such mini-vertices. (b)All GOOD mini-vertices that arrived prior to x_i, where each such mini-vertex contributes 1 to the degree and there are y_i such mini-vertices; notice $y_i \geq 1$ since we argued above that the first mini-vertex in S belongs to A. This gives that the total degree of S when x_i arrives is

$$2(i-1) + y_i \tag{29}$$

Now (28) and (29) bound the probability that x_i attaches to S and is hence BAD, given that all mini-verties that arrived prior to x_i and belong to A are GOOD, while those belonging to $\bar{A}$ are BAD by

$$\begin{aligned}
\tfrac{2(i-1)+y_i}{2(z_i+y_i)-1} &\leq \tfrac{2(i-1)+y_i}{2z_i+y_i} && \text{by subtracting } y_i-1 \geq 0 \text{ from the denomenator,} \\
&&& \text{which is possible since } y_i \geq 1, \\
&\leq \tfrac{2(i-1)+|A|}{2z_i+|A|} && \text{by adding } |A|-y_i \geq 0 \\
&&& \text{to the numerator and the denomenator,} \\
&&& \text{which is possible since } y_i \leq |A|, \\
&\leq \tfrac{2i+|A|}{2z_i+2+|A|} && \text{by adding 2 to the numerator} \\
&&& \text{and the denomenator,} \\
&= \tfrac{i+|A|/2}{z_i+1+|A|/2} \\
&= \tfrac{i+|A|}{z_i+1+|A|} && \text{by adding } |A|/2 \text{ to the numerator} \\
&&& \text{and the denomenator.}
\end{aligned} \tag{30}$$

When $\bar{x}_i$ arrives, the total volume of $\bar{S}$ is contributed by: (a)All BAD mini-vertices that arrived prior to $\bar{x}_i$ and are associated with $\bar{S}$, where there are $i-1$ such mini-vertices and

each one contributes degree 2 to $\bar{S}$, except for mini-vertex 1 which contributes degree 1. (b)All GOOD mini-vertices that arrived prior to $\bar{x}_i$, where each such mini-vertex contributes 1 to the degree and there are $\bar{y}_i$ such mini-vertices. This gives that the total degree of $\bar{S}$ when $\bar{x}_i$ arrives is

$$2(i-1)-1+\bar{y}_i \tag{31}$$

Now (28) and (31) bound the probability that $\bar{x}_i$ attaches to $\bar{S}$ and is hence BAD, given that all mini-verties that arrived prior to $\bar{x}_i$ and belong to A are GOOD, while those belonging to $\bar{A}$ are BAD by

$$\begin{aligned}
\tfrac{2(i-1)-1+\bar{y}_i}{2(\bar{z}_i+\bar{y}_i)-1} &\leq \tfrac{2(i-1)-1+\bar{y}_i}{2\bar{z}_i+\bar{y}_i-1} && \text{by subtracting } \bar{y}_i \geq 0 \text{ from the denomenator,} \\
&\leq \tfrac{2(i-1)-1+|A|}{2\bar{z}_i+|A|-1} && \begin{array}{c}\text{by adding } |A|-\bar{y}_i \geq 0 \\ \text{to the numerator and the denomenator,} \\ \text{which is possible since } \bar{y}_i \leq |A|,\end{array} \\
&\leq \tfrac{2i+|A|}{2\bar{z}_i+2+|A|} && \begin{array}{c}\text{by adding 3 to the numerator} \\ \text{and the denomenator,}\end{array} \\
&= \tfrac{i+|A|/2}{\bar{z}_i+1+|A|/2} && \\
&= \tfrac{i+|A|}{\bar{z}_i+1+|A|} && \begin{array}{c}\text{by adding } |A|/2 \text{ to the numerator} \\ \text{and the denomenator.}\end{array}
\end{aligned} \tag{32}$$

Now (30) and (32) imply that the probability that all mini-vertices not belonging to S are BAD is at most

$$\prod_{i=1}^{dk-k_1} \frac{i+|A|}{z_i+1+|A|} \prod_{i=1}^{dn-dk-k_2} \frac{i+|A|}{\bar{z}_i+1+|A|}$$

which, by using (27) and multiplying numerator and denomenator with $(|A|!)^2$, becomes

$$\frac{(dk+k_2)!\,(dn-dk+k_1)!}{|A|!\,(dn)!}.$$

Which is clearly

$$= \frac{(dk)!\,(dn-dk)!}{(dn-|A|)!\,(|A|)!} \qquad \text{using (27)}$$
$$= \frac{(dk-|A|)!\,(dn-dk)!}{(dn-|A|)!} \cdot \frac{(dk)!}{(|A|)!\,(dk-|A|)!}$$
$$= \binom{dk}{|A|}\binom{dn-|A|}{dk-|A|}^{-1}$$
$$\leq \binom{dk}{\alpha k}\binom{dn-\alpha k}{dk-\alpha k}^{-1}.$$

□

4.4 Corollaries and Applications

We may now quote approximation techniques for multicommodity flow [58, 79] and claim:

Corollary 40 *Let $G_{d,n}$ be a random graph as in Theorem 38. There is a polynomial time algorithm that routes $d_{G_{d,n}}(u) \cdot d_{G_{d,n}}(v)$ units of flow between every pair of vertices u and v, with maximum link congestion $O(n \log n)$.*

The reason that we insist of $d_{G_{d,n}}(u) \cdot d_{G_{d,n}}(v)$ units of flow is that, in general (e.g. for large d), the random graph may model the core of the entire network. In that case, every node in the core has to serve a number of customers that tends to be proportional to its degree in the core, hence the demand between two nodes in the core becomes proportional to the product of their degrees (we refer the reader to [32] for further explanation of the assumptions on uniform demand and capacities, and the implications of Corollary 40 in routing congestion on the Internet).

Most of the routing on the Internet is done along integral shortest paths [35]. Leighton and Rao have already observed that randomized rounding applies to their algorithm, hence Corollary 40 can be restated for integral paths. We can also apply the techniques of disjoint paths for constant-degree expanders and for routing along short paths [53] through the following simple construction: Every vertex u in $G_{d,n}$ of degree $d_{G_{d,n}}(u)$ is replaced with $d_{G_{d,n}}(u)$ mini-vertices. Each mini-vertex is connected to the corresponding edge of $G_{d,n}$, and within the $d_{G_{d,n}}(u)$ mini-vertices we put a constant-degree expander. It can be argued routinely that the resulting graph is a constant degree expander.

Another notable implication of Theorem 38 concerns the spectral gap of the stochastic normalization of the adjacency matrix of the graph[1]. In particular, by using $\lambda_2 < 1 - \frac{\Phi^2}{2}$ we get:

Corollary 41 *Let $G_{d,n}$ be a random graph as in Theorem 38. Let A be the adjacency matrix of $G_{d,n}$. Let P be the stochastic matrix corresponding to a random walk in $G_{d,n}$. The largest eigenvalue of P is $\lambda_1 = 1$. Let λ_2 be the second largest eigenvalue. Then, for some positive constant c, the second eigenvalue $\lambda_2 < 1 - c$, almost surely.*

It is known that the cover time of a graph is bounded by $O(\frac{n \log n}{1-\lambda_2})$ —e.g. see [12]. Then Corollary 41 gives cover time $O(n \log n)$. We note that the cover time of scale free graphs has been associated with crawling and searching on the world-wide web and P2P networks [21, 1].

Constant-degree expander graphs have played a central role in algorithms and complexity over the last thirty years. In a rather strong sense, Theorem 38 and Corollary 41 suggest analogies between constant-degree constant expanders and constant average degree scale free graphs. It is reasonable to expect that analogies will find many further applications.

[1]This is not to be confused with the spectrum of the adjacency matrix prior to stochastic normalization, considered elsewhere [28, 64, 19]. The eigenvalues of the matrix prior to normalization are a restatement of skewed statistics in the large degrees, and are hence of no particular content or algorithmic significance [64].

CHAPTER 5

DISCUSSION AND OPEN QUESTIONS

The fusion of algorithmic ideas with concepts and techniques from game theory has been suggested (see [67]) for understanding and analyzing the Internet and other systems involving strategic agents.

However, we believe that the applications of this line of research could go far beyond those contexts. In particular, the methodology of computer science can ultimately shed light on the issue of complexity in game theory, a long standing problem in this field.

One avenue of research is to study the the game theoretic concepts whose computational complexity remains open. Some of the candidates for algorithmic analysis include market equilibria with non-linear utilities, Shapley value and Nash equilibria. It would be also very interesting to study algorithmic aspects of mechanism design, specially in the context of routing and congestion control in the Internet.

The algorithm that we presented in the second chapter for facility location problem is improved by [61]. However, there is still a small gap between the lower and upper bounds. Some of the other interesting problems in approximation algorithms are finding a constant-factor approximation algorithm for asymmetric TSP, a combinatorial constant-factor approximation algorithm for Steiner network (network survivability), and an algorithm for uniform generation and approximate counting of graphs with a given degree sequence using Markov chain Monte Carlo methods. One of the applications of the last problem is in generating random graphs with skewed degree distribution which is important for modeling the topology of the Internet and other complex networks.

As a final note, skewed statistics arise also in biological networks (for example in genetic networks where nodes represent genes and proteins). It would be interesting to apply methods discussed in our work in this context as well.

REFERENCES

[1] Adamic, L., Lukose, R., Puniyani, A., and Huberman, B., "Search in power law networks," *Physical review E*, vol. 64.

[2] Aiello, W., Chung, F., and Lu, L., "A random graph model for massive graphs," pp. 171–180, 2000.

[3] Aiello, W., Chung, F. R. K., and Lu, L., "Random evolution in massive graphs," in *IEEE Symposium on Foundations of Computer Science*, pp. 510–519, 2001.

[4] Andrews, M. and Zhang, L., "The access network design problem," in *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 1998.

[5] Arrow, K. and Debreu, G., "Existence of an equilibrium for a competitive economy," *Econometrica*, vol. 22, pp. 265–290, 1954.

[6] Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V., "Local search heuristics for k-median and facility location problems," in *Proceedings of 33rd ACM Symposium on Theory of Computing*, 2001.

[7] B., B. and Riordan, O., "The diameter of scale-free graphs," to appear in Combinatorica.

[8] Balinski, M., "On finding integer solutions to linear programs," in *Proc. IBM Scientific Computing Symposium on Combinatorial Problems*, pp. 225–248, 1966.

[9] Barabasi, A.-L. and Albert, R., "Emergence of scaling in random graphs," *Science*, vol. 286, 1999.

[10] Beasley, J. E., "Operations research library," 2002.

[11] Bollobas, B., Riordan, O., Spencer, J., and Tusnady, G., "The degree sequence of a scale-free random graph process," *Random Structures and Algorithms*, vol. 18, 2001.

[12] Broder, A. and Karlin, A., "Bounds on the cover time," *J. Theoretical Probability*, vol. 2, 1989.

[13] Cameron, C. W., Low, S. H., and Wei, D. X., "High-density model for server allocation and placement." unpublished manuscript, 2002.

[14] Charikar, M. and Guha, S., "Improved combinatorial algorithms for facility location and k-median problems," in *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pp. 378–388, October 1999.

[15] Charikar, M., Guha, S., Tardos, E., and Shmoys, D., "A constant-factor approximation algorithm for the k-median problem," in *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pp. 1–10, May 1999.

[16] Charikar, M., Khuller, S., Mount, D., and Narasimhan, G., "Facility location with outliers," in *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, (Washington DC), January 2001.

[17] Chudak, F. and Shmoys, D., "Improved approximation algorithms for the uncapacitated facility location problem." unpublished manuscript, 1998.

[18] Chung, F. and Lu, L., "The average distance in a random graph with given expected degrees," *Proceedings of the National Academy of Sciences*, vol. 99, 2002.

[19] Chung, F., Lu, L., and V., V., "The spectra of random graphs with given expected degrees," *Proceedings of the National Academy of Sciences*. to appear.

[20] Chvatal, V., "A greedy heuristic for the set covering problem," *Math. Oper. Res.*, vol. 4, pp. 233–235, 1979.

[21] Cooper, C. and Frieze, A., "Crawling on web graphs," in *STOC*, 2002.

[22] Cornuejols, G., Nemhauser, G., and Wolsey, L., "The uncapacitated facility location problem," in *Discrete Location Theory* (Mirchandani, P. and Francis, R., eds.), pp. 119–171, John Wiley and Sons Inc., 1990.

[23] Debreu, G., "Economic theory in a mathematical mode: the Nobel Lecture," 1984. AER.

[24] Deng, X., Papadimitriou, C., and Safra, S., "On the complexity of equilibria," in *Proceedings of ACM Symposium on Theory of Computing*, 2002.

[25] Devanur, N. and Vazirani, V. V., "The spending constraint model for market equilibrium: Algorithmic, existence and uniqueness results," to appear in STOC 2004.

[26] Devanur, N. R. and Vazirani, V. V., "An improved approximation scheme for computing arrow-debreu prices in the linear case," in *Proc. of Foundations of Software Tecnology and Theoretical Computer Science*, 2003.

[27] Eisenberg, E. and Gale, D., "Consensus of subjective probabilities: The pari-mutuel method," *Annals Of Mathematical Statistics*, vol. 30, pp. 165–168, 1959.

[28] Faloutsos, M., Faloutsos, P., and Faloutsos, C., "On power-law relationships of the internet topology," in *Sigcomm*, 1999.

[29] for Applied Networking Research, N. L., "Routing data."

[30] Gale, D., *Theory of Linear Economic Models*. N.Y.: McGraw Hill, 1960.

[31] Gkantsidis, C., Mihail, M., and Zegura, E., "Spectral analysis of internet topologies," in *INFOCOM*, 2003.

[32] Gkantsidis, C., Mihail, M., and Saberi, A., "Conductance and congestion in power law graphs," 2003.

[33] Goemans, M. and Kleinberg, J., "An improved approximation ratio for the minimum latency problem," *Mathematical Programming*, vol. 82, pp. 111–124, 1998.

[34] Goemans, M. and Skutella, M., "Cooperative facility location games," in *Symposium on Discrete Algorithms*, pp. 76–85, 2000.

[35] Griffin, T., "An introduction to interdomain routing and bgp." SIGCOMM 2001 Tutorial.

[36] Guha, S., *Approximation algorithms for facility location problems.* PhD thesis, Stanford University, 2000.

[37] Guha, S. and Khuller, S., "Greedy strikes back: Improved facility location algorithms," *Journal of Algorithms*, vol. 31, pp. 228–248, 1999.

[38] Guha, S., Meyerson, A., and Munagala, K., "Hierarchical placement and network design problems," in *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, 2000.

[39] Guha, S., Meyerson, A., and Munagala, K., "Improved combinatorial algorithms for single sink edge installation problems," Tech. Rep. STAN-CS-TN00 -96, Stanford University, 2000.

[40] Guha, S., Meyerson, A., and Munagala, K., "Improved algorithms for fault tolerant facility location," in *Symposium on Discrete Algorithms*, pp. 636–641, 2001.

[41] Hajiaghayi, M., Mahdian, M., and Mirrokni, M., "Facility location problem with concave cost functions." unpublished manuscript, 2002.

[42] Hochbaum, D. S., "Heuristics for the fixed cost median problem," *Mathematical Programming*, vol. 22, no. 2, pp. 148–162, 1982.

[43] Holme, P., "Edge overload breakdown in evolving networks," *Physical Review E*, vol. 66, 2002.

[44] Holme, P. and Kim, B., "Attack vulnerability of complex networks," vol. 65, 2002.

[45] Jain, K., "A polynomial time algorithm for computing the arrow-debreu market equilibrium for linear utilities." manuscript, 2004.

[46] Jain, K., Mahdian, M., and Saberi, A., "Approximating market equilibria," in *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2003.

[47] Jain, K. and Vazirani, V., "Primal-dual approximation algorithms for metric facility location and k-median problems," in *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pp. 2–13, October 1999.

[48] Jain, K. and Vazirani, V., "An approximation algorithm for the fault tolerant metric facility location problem," in *Approximation Algorithms for Combinatorial Optimization, Proceedings of APPROX 2000* (Jansen, K. and Khuller, S., eds.), vol. 1913 of *Lecture Notes in Computer Science*, pp. 177–183, September 2000.

[49] Jain, K. and Vazirani, V., "Applications of approximation algorithms to cooperative games," in *ACM Symposium on Theory of Computing*, pp. 364–372, 2001.

[50] Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L., "On the placement of internet instrumentations," in *Proceedings of IEEE INFOCOM'00*, pp. 26–30, March 2000.

[51] Kapoor, S. and Garg, R., "Auction algorithms for market equilibrium," to appear in STOC 2004.

[52] Kaufman, L., Eede, M. v., and Hansen, P., "A plant and warehouse location problem," *Operational Research Quarterly*, vol. 28, pp. 547–557, 1977.

[53] Kleinberg, J. and Rubinfeld, R., "Short paths in expander graphs," in *FOCS*, vol. 96.

[54] Korupolu, M., Plaxton, C., and Rajaraman, R., "Analysis of a local search heuristic for facility location problems," in *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1–10, January 1998.

[55] Kuehn, A. and Hamburger, M., "A heuristic program for locating warehouses," *Management Science*, vol. 9, pp. 643–666, 1963.

[56] Kuhn, H. W., "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1995.

[57] Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., and Upfal, E., "Stochastic models for the web graph," in *FOCS*, 2000.

[58] Leighton, F. T. and Rao, S., "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms," *Journal of the ACM*, vol. 46, 1999.

[59] Li, B., Golin, M., Italiano, G., X., D., and Sohraby, K., "On the optimal placement of web proxies in the internet," in *Proceedings of IEEE INFOCOM'99*, pp. 1282–1290, 1999.

[60] Lovasz, L., "On the ratio of optimal integral and fractional covers," *Discrete Mathematics*, vol. 13, pp. 383–390, 1975.

[61] Mahdian, M., Ye, Y., and Zhang, J., "Improved approximation algorithms for metric facility location problems." Unpublished manuscript, 2002.

[62] McEliece, R., Rodemich, E., Rumsey Jr., H., and Welch, L., "New upper bounds on the rate of a code via the delsarte-macwilliams inequalities," *IEEE Transactions on Information Theory*, vol. 23, pp. 157–166, 1977.

[63] Mettu, R. and Plaxton, G., "The online median problem," in *IEEE Symposium on Foundations of Computer Science*, pp. 339–348, 2000.

[64] Mihail, M. and Papadimitriou, C., "On the eigenvalue power law," in *RANDOM*, 2002.

[65] Nemhauser, G. and Wolsey, L., *Integer and Combinatorial Optimization*. John Wiley and Sons, 1990.

[66] Papadimitriou, C. H., "On the complexity of the parity argument and other inefficient proofs of existence," *JCSS*, vol. 48(3), pp. 498–532.

[67] Papadimitriou, C. H., "Algorithms, games, and the internet," *Lecture Notes in Computer Science*, vol. 2076, 2001.

[68] Qiu, L., Padmanabhan, V., and Voelker, G., "On the placement of web server replicas," in *Proceedings of IEEE INFOCOM'01*, (Anchorage, AK, USA), April 2001.

[69] Qiu, L., Padmanabhan, V. N., and Voelker, G. M., "On the placement of web server replicas," in *INFOCOM*, pp. 1587–1596, 2001.

[70] Rajagopalan, S. and V.V., V., "Primal-dual RNC approximation algorithms for set cover and covering integer programs," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 525–540, 1999.

[71] Scarf, H., *The Computation of Economic Equilibria (with collaboration of T. Hansen).* New Haven: Yale University Press: Cowles Foundation Monograph No. 24., 1973.

[72] Shmoys, D., Tardos, E., and Aardal, K., "Approximation algorithms for facility location problems," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pp. 265–274, 1997.

[73] Stollsteimer, J., *The effect of technical change and output expansion on the optimum number, size and location of pear marketing facilities in a California pear producing region.* PhD thesis, University of California at Berkeley, 1961.

[74] Stollsteimer, J., "A working model for plant numbers and locations," *J. Farm Econom.*, vol. 45, pp. 631–645, 1963.

[75] Sviridenko, M., "An 1.582-approximation algorithm for the metric uncapacitated facility location problem." to appear in the Ninth Conference on Integer Programming and Combinatorial Optimization, 2002.

[76] Thorup, M., "Quick k-median, k-center, and facility location for sparse graphs," in *Automata, Languages and Programming, 28th International Colloquium, Crete, Greece*, vol. 2076 of *Lecture Notes in Computer Science*, pp. 249–260, 2001.

[77] Towsley, D., "Modeling the internet: Seeing the forest through the trees," 2002. Keynote Address, Sigmetrics.

[78] Vazirani, V. V., "Market equilibrium when buyers have spending constriants." submitted, 2004.

[79] Vazirani, V. V., *Approximation Algorithms.* Berlin: Springer-Verlag, 2001.

[80] Walras, L., *Éléments d'économie politique pure ou théorie de la richesse sociale (Elements of Pure Economics, or the theory of social wealth).* 1874. (1899, 4th ed.; 1926, rev ed., 1954, Engl. transl.).

[81] Z., B.-Y., Berg, A., Chien, S., Fakcharoenphol, J., and Weitz, D., "Approximating aggregate queries about web pages via random walks," VLDB 2000.

[82] Zegura, E. W., Calvert, K. L., and Bhattacharjee, S., "How to model an internetwork," in *IEEE Infocom*, vol. 2, (San Francisco, CA), pp. 594–602, IEEE, March 1996.

VITA

Contact Information

Home Address: 1401 West Paces Ferry Road, Apt No.: 4302, Atlanta, GA 30327, USA

E-mail: saberi@cc.gatech.edu

Homepage: http://www.cc.gatech.edu/~saberi

Research Interests

Algorithms, computational issues in game theory, approximation algorithms

Networking: models and algorithms for WWW, Internet, and peer to peer networks

Graph theory and combinatorics

Education

Georgia Institute of Technology, Atlanta, Georgia, USA

Ph.D. in Computer Science (Algorithms, Combinatorics, and Optimization Program)

August 2000 - June 2004

Advisors: Vijay V. Vazirani and Milena Mihail

Sharif Institute of Technology, Tehran, IRAN

Graduated with B.S. in Computer Science (1996-2000)

Thesis: Design and analysis of approximation algorithms using linear programming

Advisor: Mohammad Ghodsi