Approximation Algorithms for Network Routing and Facility Location Problems

Shi Li

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Moses Charikar

January 2014

Abstract

We study approximation algorithms for two classes of optimization problems.

The first class is network routing problems. These are an important class of optimization problems, among which the edge-disjoint paths (EDP) problem is one of the central and most extensively studied. In the first part of my thesis, I will give a poly-logarithmic approximation for EDP with congestion 2. This culminates a long line of research on the EDP with congestion problem.

The second class is facility location problems. Two important problems in this class are uncapacitated facility location (UFL) and k-median, both having long histories and numerous applications. We give improved approximation ratios for both problems in the second part of my thesis.

For UFL, we present a 1.488-approximation algorithm for the metric uncapacitated facility location (UFL) problem. The previous best algorithm, due to Byrka, gave a 1.5-approximation for UFL. His algorithm is parametrized by γ whose value is set to a fixed number. We show that if γ is randomly selected, the approximation ratio can be improved to 1.488.

For k-median, we present an improved approximation algorithm for k-median. Our algorithm, which gives a $1+\sqrt{3}+\epsilon$-approximation for k-median, is based on two rather surprising components. First, we show that it suffices to find an α-approximate solution that contains $k+O(1)$ medians. Second, we give such a pseudo-approximation algorithm with $\alpha = 1+\sqrt{3}+\epsilon$.

Acknowledgements

Five years have passed and I have had a wonderful life during the five years. I would like to thank the people who have supported and encouraged me along the way, both academically and otherwise.

First of all, I would like to thank my advisor, Professor Moses Charikar, for his excellent guidance during my studies in Princeton University. His persistence in pursuing ultimate answers gives me the confidence and interests to research in theoretical computer science. I learned how to think independently from many discussions with him, where he is always able to express intuitions and complex concepts using simple and clear languages. His ability to suggest the right questions and insights into different kinds of problems are a great source of inspiration to me.

Much of my research work began with my internship at Toyota Technological Institute at Chicago in the summer of 2011. I would like to thank Professor Julia Chuzhoy for mentoring me in and after the internship. Her ability of dealing with complex problems and serious scientific attitude are the qualities I have admired. Next, I would like to thank Matthew Andrews and Gabriel Tucci for mentoring and hosting me at Bell Labs in the summer of 2012. They provided me many opportunities to interact with researchers from different areas in Bell Labs.

I have been very fortunate to collaborate with many people and I thank all of them: Nikhil Bansal, Deeparnab Chakrabarty, Parinya Chalermsook, Moses Charikar, Julia Chuzhoy, Alina Ene, Mohammadtaghi Hajiaghayi, Wei Hu, Ravishankar Krishnaswamy, Tom Leighton, Jian Li, Ankur Moitra, Srivatsan Narayanan, Barna Saha and Ola Svensson. Special thanks to Ola Svensson for writing recommendation letters for my job applications.

I am very grateful to the professors in the department. I have always admired their perspective of theory in science. Thanks to Sanjeev Arora, Bernard Chazelle, Robert Tarjan, Zeev Dvir and Mark Braverman for being the committee members of my general exam and FPO. Thanks to Mark Braverman and Julia Chuzhoy for reading my dissertation and giving useful feedbacks.

My life in the department has been a wonderful experience, due to many of my fellow students. I would like to thank all of them: Rajsekar Manokaran, David Steurer, Moritz Hardt, Wolfgang Mulzer, Sina Jafarpour, Hossein Bateni, Aravindan Vijayaraghaven, Aditya Bhaskara, Sid Sen, Rong Ge, Sushant Sachdeva, Yuri Pritykin, Yiming Liu, Linjie Luo, Huy Nyugen and Siyu Yang.

I sincerely acknowledge the following NSF funds, which have supported my research along the way: CCF-0832797, CCF-0916218, CCF-1218687. My fifth year of study is supported by a Wallace Memorial Fellowship from Princeton University. I would like to take the opportunity to thank the university for its generosity. It is my honor to be a recipient of the fellowship.

Outside the department, I have many great friends who constantly made me engaged and entertained in social activities. I would like to thank Kaiyu Guan for sharing many great moments. I would like to thank Rong Ge, Lanhui Wang, Ke Wang, Bo Yang and several others for so many wonderful gathering and gaming nights.

Finally and importantly, I am deeply grateful to my wife, Wei Shen, whose support, encouragement, patience and love gave me a happy journey in Princeton. I also thank my parents and my brother. Despite the geographical distance, they were always nearby, supporting me.

To my parents
To my wife, Wei Shen
Words can not express how much I love you all

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

We are now living in an era of information technology. More than ever, we need to make fast decisions, from deciding warehouse locations for a company, to scheduling jobs on machines, to displaying ads in empty slots of web pages. Unfortunately, most optimization problems are NP-hard. Unless $\mathsf{NP} = \mathsf{P}$, there are no polynomial time algorithms to find optimal solutions for these optimization problems. The most common approach to overcome this difficulty, known as approximation algorithms, is by relaxing the requirement of finding optimal solutions.

In this thesis, we study approximation algorithms for several classic optimization problems. All these problems have long history and was studied extensively in the literature. In spite of the extensive study, there are still large gaps between the best approximation ratios and the hardness of approximation. These long-standing gaps present big challenges to our algorithmic toolbox. Thus, studying these problems are not only useful for practical applications, but also of theoretical importance.

The problems we study fall into two classes: network routing problems and facility location problems. We now introduce the two classes in details and state our results of the thesis.

1.1.1 Network Routing Problems

Networks are ubiquitous in our modern society. From the World Wide Web to social networks, from economic systems to maps of cities, networks represent a wide range of real-world systems. As networks appear in so many different contexts, optimization problems in networks are increasingly important.

An important class of optimization problems arising in networks is the network routing problems. The common goal of this class is to assign sufficient network resources to satisfy the connection requests from the users. Depending on the objective function, network routing problems are in general one of the following two forms. In the first form, we want to maximize some measure of profit gained by satisfying connection requests, such as the number of accepted requests, the throughput of the connections and the sustainability of the network. In the second form, we want to

minimize some cost function of the assignment, e.g. the traffic congestion of roads, the power consumption of wireless ad-hoc stations, and the delay of transportations from senders to receivers.

The network routing problem we will consider in the thesis is the edge-disjoint paths problem (EDP). This is one of the central and most extensively studied problem in this class. In this problem, we are given an undirected n-vertex graph $G = (V, E)$, and a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k source-sink pairs, that we also call demand pairs. The goal is to find a collection $\mathcal{P}$ of edge-disjoint paths, connecting the maximum possible number of demand pairs.

Study of EDP has its roots in early 1980's in the context of very-large-scale integration (VLSI, [74, 1, 2]). In VLSI design, a single layer of chip is divided into thousands of cells and we want to use wires to connect some pairs of cells so that the wires do not interfere with each other. It is practically useful to route as many connections on a single layer of the chip as possible. In a society of networks, this problem clearly has numerous applications. As an example, consider the situation where many pairs of nodes request to connect through paths in a large-scale communication network.

Besides its practical applications, EDP plays an important role in graph theory. Many special cases of EDP are already interesting classic problems. One special case, where all the (s_i, t_i) pairs are the same, is the network flow problem. The work on this special case has led to the famous minimum-cut problem and many efficient algorithms for the network flow problem. Another special case, where the graph is a star and the pairs are over the leaves of the star, is the maximum matching problem in general graphs. EDP is an important element in the proof of the famous graph-minor theorem by Robertson and Seymour [73]. In their graph minor series, Robertson and Seymour studied algorithms for solving EDP exactly and used the ideas of the algorithms to find graph minors. Moreover, EDP, among others, is one of packing problems in graphs, where the common goal is to find maximum number of edge-disjoint or node-disjoint admissible structures in a graph. While we have good understanding in many other packing problems in graphs ([60], [36]), the approximability of EDP remains mysterious.

Robertson and Seymour have shown that EDP can be solved efficiently, when the number k of the demand pairs is a constant [73]. When the value k is a part of the input, it is NP-hard to even decide whether all pairs can be simultaneously routed via edge-disjoint paths [48]. Good approximation algorithms are known for some special cases of EDP. Rao and Zhou [72] have shown that if the value of the global minimum cut in the input graph G is $\Omega(\log^5 n)$, then there is an efficient randomized poly $\log n$-approximation algorithm for EDP. The EDP problem is also known to have poly-logarithmic approximation algorithms on bounded-degree expander graphs [61, 18, 17, 52, 37], and constant-factor approximation algorithms on trees [40, 28], grids and grid-like graphs [9, 12, 55, 54]. Kleinberg [53] has shown an $O(\log^2 n)$-approximation for Eulerian planar graphs. Kawarabayashi and Kobayashi [49] have recently improved this result to an $O(\log n)$-approximation, for both Eulerian and 4-connected planar graphs.

In spite of these results on special graphs, obtaining good approximation ratio for EDP on general graphs is still elusive. The best current approximation algorithm for

the EDP problem, due to Chekuri, Khanna and Shepherd [27], achieves an $O(\sqrt{n})$-approximation. This is even the best approximation ratio we can achieve for EDP on planar graphs. Indeed, most of the known results are based on the multi-commodity flow relaxation, where instead of connecting the demand pairs with paths, we are only required to send the maximum amount of multi-commodity flow between the demand pairs, with at most one flow unit sent between every pair. Such a fractional solution can be computed efficiently by using the standard multi-commodity flow LP-relaxation, and it can then be rounded to obtain an integral solution. Unfortunately, a simple example by Garg, Vazirani and Yannakakis [40] shows that the integrality gap of the multi-commodity flow relaxation can be as large as $\Omega(\sqrt{n})$, even for planar graphs. This implies that the algorithm of [27] is essentially the best possible for EDP, when using this standard approach. With the current best hardness of approximation factor standing on $\Omega(\log^{1/2-\epsilon} n)$ for any constant ϵ (unless NP is contained in $\mathsf{ZPTIME}(n^{\mathrm{poly}\log n})$ [5, 4]), the approximability of the EDP problem remains one of the central open problems in the area of routing.

A natural question is whether we can obtain better approximation algorithms by slightly relaxing the disjointness requirement, and allowing the paths to share edges. We say that a set $\mathcal{P}$ of paths is an α-approximate solution with congestion c, iff the paths in $\mathcal{P}$ connect at least opt/α of the demand pairs, while every edge of G appears on at most c paths in $\mathcal{P}$. Here, opt is the value of the optimal solution to EDP, where no congestion is allowed. This relaxation of the EDP problem is called EDP with congestion (EDPwC). The EDPwC problem is a natural framework to study the tradeoff between the number of pairs routed and the congestion, and it is useful in scenarios where we can afford a small congestion on edges.

This is also a stream of research papers on the EDPwC problem. The classical randomized rounding technique of Raghavan and Thompson [71] gives a constant factor approximation for EDPwC, when the congestion c is $\Omega(\log n/\log\log n)$. For general congestion value c, factor $O(n^{1/c})$-approximation algorithms are known for EDPwC [13, 15, 56]. Recently, Andrews [3] has shown a randomized $(\mathrm{poly}\log n)$-approximation algorithm with congestion $c = \mathrm{poly}\log\log n$, and Chuzhoy [31] has shown a randomized $(\mathrm{poly}\log k)$-approximation algorithm with congestion 14. For the congestion value $c = 2$, Kawarabayashi and Kobayashi [50] have recently shown an $O(n^{3/7})$-approximation algorithm, thus improving the best previously known $O(\sqrt{n})$-approximation for $c = 2$ [13, 15, 56]. On the negative side, Andrews et al. [4] have shown that the integrality gap of the multicommodity flow relaxation for EDPwC is $\Omega\left(\left(\frac{\log n}{(\log\log n)^2}\right)^{1/(c+1)}\right)$ for any constant congestion c. Andrews et al. [4] have also shown that for any constant ϵ, for any $1 \leq c \leq O\left(\frac{\log\log n}{\log\log\log n}\right)$, there is no $O\left((\log n)^{\frac{1-\epsilon}{c+1}}\right)$-approximation algorithm for EDPwC with congestion c, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathrm{poly}\log n})$.

EDPwC has also been extensively studied on planar graphs. Chekuri, Khanna and Shepherd [23, 26] have shown a poly-logarithmic approximation algorithm for EDPwC with congestion 2 and a constant approximation algorithm with congestion 4. Both

results have recently been improved by Seguin-Charbonneau and Shepherd [75], who showed a constant factor approximation algorithm with congestion 2.

1.1.2 Facility Location Problems

The common goal of facility location problems is to decide optimal locations to build facilities to serve a given set of clients. There are various problems in this class, depending on the constraints and the objective cost function. There might be constraints on the number of facilities we can build, the budget, and the number of clients a facility can serve. The cost function may reflect the cost for building facilities and/or the cost for serving the clients.

The problem with the simplest form in this class is the uncapacitated facility location problem (UFL). In this problem, we are given a set $\mathcal{F}$ of potential facility locations, each location $i \in \mathcal{F}$ with a facility cost f_i, a set $\mathcal{C}$ of clients and connection cost (service cost) function $d : \mathcal{F} \times \mathcal{C} \to \mathbb{R}_+$. The goal is to find a subset $\mathcal{F}' \subseteq \mathcal{F}$ of locations to open facilities, so as to minimize $\sum_{i \in \mathcal{F}'} f_i + \sum_{j \in \mathcal{C}} \min_{i \in \mathcal{F}'} d(i, j)$. The first term in the cost function is the total facility cost, i.e, the cost to build the facilities, while the second term is the cost of connecting clients to facilities. In the literature, UFL usually stands for the *metric* uncapacitated location problem, where clients and facilities lie in a metric space and the cost function $d(i, j)$ represents the distance between facility i and client j. This is common in many applications. The problem is called *uncapacitated* facility location problem since every facility, once open, can serve unlimited number of clients. A more complicated variant of the problem is the capacitated facility location problem, where each facility has an upper bound on the number of clients it can serve.

This UFL problem is well-studied in the operation research literature, dating back to early 1960's ([58, 69, 14]). Several early papers studied the problem under the name warehouse location problem. The problem occurs when a large company is deciding a number of locations for building warehouses to supply its existing stores. Each warehouse built incurs a fixed maintenance cost. Each store will be supplied by its nearest warehouse and supply cost depends on the distance between the store and the warehouse.

A closely related problem to UFL is the classic k-median problem. In this problem, we are not given the facility costs f_i. Instead, we are given an upper bound k on the number of facilities that can be open. The objective function we want to minimize is the connection cost, i.e, the sum of distance from j to the nearest open facility of j, over all clients j. Due to its close connection to UFL, it is reasonable to model the above warehouse location problem as a k-median problem. Assume the large company only has the budget to build k warehouses. The goal is to is to select the k warehouses optimally so as to minimize the total supply cost.

When $\mathcal{F} = \mathcal{C} = X$, a solution $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| = k$ partitions the set of points into what is known as clusters and thus the objective function measures how well the data set X can be partitioned into k clusters. If the dataset is indeed from k classes, then solving the k-median problem will give the desired partition. Due to this nature,

k-median has numerous applications in clustering, data mining and machine learning ([16]).

Both UFL and k-median are NP-hard. Study of them from the perspective of approximation algorithms started in early 1990's. In 1982, Hochbaum [43] presented a greedy algorithm for the non-metric UFL with $O(\log n)$-approximation guarantee. Constant factor approximation algorithms are known for the metric UFL. Shmoys, Tardos and Aardal [77] used the filtering technique of Lin and Vitter [64] to give a 3.16-approximation algorithm, which is the first constant factor approximation for the metric UFL problem. After that, a steady stream of papers giving improved algorithms [29, 57, 21, 45, 46, 68] were proposed. Before our work, the best approximation ratio is 1.50, given by Byrka [19].

On the negative side, Guha and Kuller [41] showed that there is no ρ-approximation for UFL if $\rho < \rho^* \approx 1.463$, unless $\mathbf{NP} \subseteq \mathbf{DTIME}\left(n^{O(\log\log n)}\right)$, where ρ^* is the root of $\gamma = 1 + 2e^{-\gamma}$. Thus, the 1.50-approximation ratio is very close to being optimal. Jain et al. [46] generalized the result to show that no (γ_f, γ_c)-bifactor approximation exists for $\gamma_c < 1 + 2e^{-\gamma_f}$ unless $\mathbf{NP} \subseteq \mathbf{DTIME}\left(n^{O(\log\log n)}\right)$. The exact definition of (γ_f, γ_c)-bifactor approximation is given later in Chapter 5.

In contrast to UFL, current techniques give a relatively worse understanding of the approximability of k-median. The difficulty of k-median lies in the hard constraint that only k facilities are allowed to be opened. Indeed, without such a constraint, we could simply open all facilities. Early approaches [65, 64, 57] overcame this difficulty by giving pseudo-approximations that obtain better guarantees while violating the mentioned constraint by opening $k + \Omega(k)$ facilities. The first constant factor approximation algorithm that opens k facilities is due to Charikar et al. [22]. Based on LP rounding, their algorithm produces a $6\frac{2}{3}$-approximation. Several of the ideas in [22] are inspired from constant factor approximation algorithms obtained for UFL. The best known approximation algorithm is the local search algorithm given by Arya et al. [8]. They showed that if there is a solution $\mathcal{F}'$, where any p swaps of the open facilities cannot improve the solution, then $\mathcal{F}'$ is a $3 + 2/p$ approximation. This leads to a $3 + \epsilon$ approximation that runs in time $n^{2/\epsilon}$. On the negative side, Jain et al. [46] proved that the k-median problem is hard to approximate within a factor $1 + 2/e \approx 1.736$. Moreover, the natural linear programming relaxation of k-median is known to have an integrality gap of at least 2. The best upper bound is by Archer et al. [6], who showed that the integrality gap is at most 3 by giving an exponential time rounding algorithm that requires to solve the maximum independent set problem.

In spite of the apparent similarities between k-median and UFL, they were studied separately in literature, until Jain and Vazirani exploited a deep connection between them in a beautiful paper [47]. The connection is motivated by basic economic theory: if we let the opening costs of facilities be small then a "good" solution to UFL will open many facilities whereas if we let the opening costs of facilities be large then a good solution will only open few facilities. By appropriately selecting the cost of facilities, one can therefore expect that an algorithm for UFL opens close to k facilities and therefore almost also gives a solution to the k-median problem. By exploiting this concept, Jain and Vazirani obtained a 6-approximation algorithm for k-median using

their 3-approximation primal-dual algorithm for UFL. The factor 3 was later improved by Jain et al. [46] to 2 resulting in a 4-approximation algorithm for k-median.

1.2 Our Results

In this thesis, we give our improved approximation algorithms for EDPwC, UFL and k-median. The three results are respectively based on joint work with Chuzhoy [32], the work of [62], and joint work with Svensson [63].

Edge disjoint paths with congestion In Part I of the thesis, we given our randomized poly $\log k$-approximation algorithm for EDPwC with congestion 2. Our result is essentially optimal with respect to this relaxation, both for the congestion and the number of pairs routed, in the following sense. As observed above, if we are interested in obtaining a sub-polynomial approximation for EDP via the multi-commodity flow relaxation, then the best congestion we can hope for is 2. On the other hand, due to the result of Andrews et al. [4], the hardness for congestion $c = 2$ is poly-logarithmic, though the degree of the logarithm is much lower than the degree we obtain in our approximation algorithm. Thus, our result culminates a long line of research on the EDP with congestion problem.

Uncapacitated Facility Location In Chapter 5, we give a 1.488-approximation algorithm for UFL, which is built on the work of Byrka [19]. Byrka presented an algorithm $A_1(\gamma)$ which gives the optimal bifactor approximation $(\gamma, 1 + 2e^{-\gamma})$ for $\gamma \geq \gamma_0 \approx 1.6774$. By either running $A_1(\gamma_0)$ or the $(1.11, 1.78)$-approximation algorithm A_2 proposed by Jain, Mahdian and Saberi [46], Byrka was able to give a 1.5-approximation algorithm. We show that the approximation ratio can be improved to 1.488 if γ is randomly selected.

k-Median In Chapter 6 We present our novel approximation algorithm for k-median that achieves an approximation guarantee of $1 + \sqrt{3} + \epsilon$, improving upon the decade-old ratio of $3 + \epsilon$. Our algorithm is based on two rather surprising components. First, we show that it suffices to find an α-approximate solution that contains $k + O(1)$ medians. Second, we give such a pseudo-approximation algorithm with $\alpha = 1 + \sqrt{(3)} + \epsilon$.

Part I

Poly-logarithmic Approximation Algorithm for Edge Disjoint Path Problem with Congestion 2

Chapter 2

Edge Disjoint Paths: Technical Components

In this part, we give a randomized poly $\log k$-approximation algorithm for EDP with congestion 2. Our main result is summarized in the following theorem.

Theorem 2.1 *There is an efficient randomized algorithm, that, given a graph G, and a collection $\mathcal{M}$ of k source-sink pairs, w.h.p. finds a routing of* $\mathsf{opt}/(\text{poly}\log k)$ *of the pairs in $\mathcal{M}$ with congestion at most* 2*, where* opt *is the maximum number of pairs that can be routed with congestion 2.*

Our result is essentially optimal with respect to this relaxation, both for the congestion and the number of pairs routed, in the following sense. Due to the $\Omega(\sqrt{n})$-integrality gap of Garg, Vazirani and Yannakakis [40], if we are interested in obtaining a sub-polynomial approximation for EDP via the multi-commodity flow relaxation, then the best congestion we can hope for is 2. On the other hand, due to the hardness result of Andrews et al. [4], the hardness for EDP with congestion 2 is poly-logarithmic, though the degree of the logarithm is much lower than the degree we obtain in our approximation algorithm.

While the approximability status of EDP remains open, our results suggests that there is a fundamental difference between routing with congestion 1 and with congestion 2, while there is not too much difference between routing with congestion 2 and with congestion c for any constant $c > 2$. Suppose we are given a solution $\mathcal{P}$ to the EDP problem that connects D of the demand pairs with congestion c. By scaling the paths by a factor of $1/c$, we obtain a solution of value D/c to the multi-commodity flow relaxation. By applying our algorithm, we can connect $D/(c\,\text{poly}\log k)$ pairs with congestion 2. That is, we can lower the congestion to 2 with only a factor $(c\,\text{poly}\log k)$ loss in the number of the demand pairs routed. However, if we are interested in routing with congestion 1, then we have to lose an $\Omega(\sqrt{n})$-factor in the number of pairs routed, due to the integrality gap instance of Garg, Vazirani and Yannakakis [40].

The proof of the Theorem 2.1 is split into 3 chapters. In the remaining part of this chapter, we describe some useful components from previous work. Some of the proofs

are implicit in previous work and we include them for the sake of completeness. Then, in Chapter 3, we give our algorithm for EDP with congestion 2 in [32]. A core component of our algorithm is the construction of what we called *crossbar*. We dedicate the whole Chapter 4 to this construction.

2.1 Preliminaries and Notations

2.1.1 Notations

Given any undirected graph H, we shall use V_H and $V(H)$ to denote the set of vertices in H, E_H and $E(H)$ to denote the set of edges in H, and $d_H(\cdot)$ denote the degree function of H. For any subset $S \subseteq V_H$ of vertices, we use $\text{out}_H(S)$ to denote the set of edges in H with exactly one endpoint in S. Unless otherwise stated, $G = (V, E)$ is the input graph where we try to route the k demand pairs. Then, we shall simply use $\text{out}(\cdot)$ to denote $\text{out}_G(\cdot)$.

Given the routing graph $G = (V, E)$, we say a subset $C \subseteq V$ of vertices is a cluster if $G[C]$ is connected. A clustering of G is a partition of V into disjoint clusters. Throughout this part, we shall fix a parameter $k_1 = k/\text{polylog}(k)$ (the exact order of k_1 is given later). We say that a cluster $C \subseteq V$ is *large* if $|\text{out}(C)| \geq k_1$, and we say that it is *small* otherwise.

Given the set $\mathcal{M} = \{(s_i, t_i) : i \in [k]\}$ of k demand pairs, we use $\mathcal{T} = \bigcup_{i\in[k]} \{s_i, t_i\}$ to denote the vertices participating in the pairs. We call the vertices in $\mathcal{T}$ terminals. For any subset of $\mathcal{M}' \subseteq \mathcal{M}$ of demand pairs, we define $\mathcal{T}(\mathcal{M}') := \bigcup_{(s_i,t_i)\in\mathcal{M}'} \{s_i, t_i\}$ to be the set of terminals participating in $\mathcal{M}'$.

Given any pair (V_1, V_2) of subsets of vertices, we denote by $F : V_1 \rightsquigarrow_\eta V_2$ the flow F where every vertex in V_1 sends one flow unit to some vertex in V_2, and the congestion due to the flow F is at most η. If additionally every vertex in V_2 receives exactly one flow unit, then we denote this flow by $F : V_1 \overset{1:1}{\rightsquigarrow}_\eta V_2$. In this case, we must have $|V_1| = |V_2|$. When the flow F is integral, defined by a set $\mathcal{P}$ of paths, we say $\mathcal{P} : V_1 \rightsquigarrow_\eta V_2$ and $\mathcal{P} : V_1 \overset{1:1}{\rightsquigarrow}_\eta V_2$ if we have $F : V_1 \rightsquigarrow_\eta V_2$ and $F : V_1 \overset{1:1}{\rightsquigarrow}_\eta V_2$ respectively. Notice that if we have $\mathcal{P} : V_1 \overset{1:1}{\rightsquigarrow}_\eta V_2$, then $|V_1| = |V_2| = |\mathcal{P}|$ and $\mathcal{P}$ routes a matching integrally with congestion η between V_1 and V_2.

In the above definitions, either V_1 or V_2 can be replaced by a set of edges. For example, if V_1 is replaced by a set E_1 of edges, then we think of E_1 as a set of vertices that sub-divides edges in E_1.

Given a subset S of vertices and two subsets $E_1, E_2 \subseteq \text{out}(S)$ of edges, we say that the flow $F : E_1 \rightsquigarrow_\eta E_2$ is in S if every flow-path is completely contained in $G[S]$, except for its first and last edges, that belong to $\text{out}(S)$. Similarly, we can define a set $\mathcal{P} : E_1 \rightsquigarrow_\eta E_2$ of paths being in S.

2.1.2 Well-Linkedness

The notion of well-linkedness has been widely used in graph decomposition and routing, see e.g. [25, 72, 3]. While the main ideas are similar, the definition details differ from paper to paper. Our definition of well-linkedness is similar to that of [31].

Definition 2.2 (α-well-linkedness) *Given a graph $G = (V, E)$, a set $\mathcal{T} \subseteq V$ of terminals and a parameter $\alpha > 0$, we say that G is α-well-linked for $\mathcal{T}$, iff for any partition (A, B) of V, $|E(A, B)| \geq \alpha \cdot \min\{|\mathcal{T} \cap A|, |\mathcal{T} \cap B|\}$.*

In many scenarios, we are interested in the following graph. Let $S \subseteq V$ be a cluster of $G = (V, E)$. We subdivide each edge $e \in \text{out}(S)$. Then, the graph $G[S \cup \text{out}(S)]$ is the graph induced by S union the set of newly added vertices that subdivide edges in $\text{out}(S)$. For the sake of simplicity, we identify the vertex that subdivide $e \in \text{out}(S)$ with e itself. That is, an element $e \in \text{out}(S)$ is both an edge in G and a vertex in $G[S \cup \text{out}(S)]$. Given G, $S \subseteq V$, $\Gamma \subseteq \text{out}(S)$, and a parameter $\alpha \in [0, 1]$, we say S is α-well-linked for Γ, if $G[S \cup \text{out}(S)]$ is α-well-linked for Γ. Equivalently, S is α-well-linked for Γ, if for every partition (X, Y) of S, $|E(X, Y)| \geq \alpha \min\{|\text{out}(X) \cap \Gamma|, |\text{out}(Y) \cap \Gamma|\}$. We simply say that the cluster S is α-well-linked if it is α-well-linked for $\text{out}(S)$.

Lemma 2.3 *G is α-well-linked for the set $\mathcal{T} \subseteq V$ of terminals, if and only if for every pair of disjoint subsets $\mathcal{T}_1, \mathcal{T}_2 \subseteq \mathcal{T}$ of equal size, we can find a flow $F : \mathcal{T}_1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} \mathcal{T}_2$ in G.*

Proof: We first assume that for every pair of disjoint subsets $\mathcal{T}_1, \mathcal{T}_2 \subseteq \mathcal{T}$ of equal size, we can find a flow $F : \mathcal{T}_1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} \mathcal{T}_2$. Consider any cut (A, B) of G. W.l.o.g, assume $|A \cap \mathcal{T}| \leq |B \cap \mathcal{T}|$. Then let $\mathcal{T}_1 = A \cap \mathcal{T}$ and $\mathcal{T}_2$ be any subset of $B \cap \mathcal{T}$ of size $|\mathcal{T}_1|$. Since there is a flow $F : \mathcal{T}_1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} \mathcal{T}_2$, we have $|E(A, B)| \geq \alpha|\mathcal{T}_1| = \alpha|A \cap \mathcal{T}|$. Since this is true for every cut (A, B), G is α-well-linked for $\mathcal{T}$.

We then prove the other direction. Focus on two disjoint sets $\mathcal{T}_1, \mathcal{T}_2 \subseteq \mathcal{T}$ such that $|\mathcal{T}_1| = |\mathcal{T}_2|$ and there is no flow $F : \mathcal{T}_1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} \mathcal{T}_2$ in G. Then consider the following network flow problem. Add a super source s^* and super sink t^* to the graph G. Add an edge of capacity 1 between s^* and each terminal $t \in \mathcal{T}_1$; add an edge of capacity 1 between t^* and each terminal $t \in \mathcal{T}_2$. All the edges in G has capacity $1/\alpha$. By the assumption, we can not send $|\mathcal{T}_1|$ flow units from s^* to t^*. By the max-flow min-cut theorem, there is a cut $(A \cup \{s^*\}, B \cup \{t^*\})$ of capacity less than $|\mathcal{T}_1|$ in the network separating s^* and t^*, where (A, B) is a cut in G. That capacity of (A, B) is $|A \cap \mathcal{T}_2| + |B \cap \mathcal{T}_1| + (1/\alpha)|E(A, B)| < |\mathcal{T}_1|$. Thus, $|E(A, B)| < \alpha(|\mathcal{T}_1| - |A \cap \mathcal{T}_2| - |B \cap \mathcal{T}_1|) \leq \alpha \min\{|A \cap \mathcal{T}_1|, |B \cap \mathcal{T}_2|\} \leq \alpha \min\{|A \cap \mathcal{T}|, |B \cap \mathcal{T}|\}$. Then, G is not α-well-linked for $\mathcal{T}$. □

We shall use the following more general definition of well-linkedness:

Definition 2.4 ((k_1, α)-well-linkedness) *Given a graph $G = (V, E)$, a set $\mathcal{T} \subseteq V$ of terminals of degree 1, a parameter $\alpha \in [0, 1]$ and $k_1 \in \mathbb{Z}_+$, we say that G is (k_1, α)-well-linked for $\mathcal{T}$, iff for any partition (A, B) of V, $|E(A, B)| \geq \alpha \cdot \min\{|\mathcal{T} \cap A|, |\mathcal{T} \cap B|, \lfloor k_1/2 \rfloor\}$.*

Though k_1 can be any positive integer in the above definition, we will only be interested in the case where k_1 is the threshold defining small and large clusters. It is easy to see that, G is (k_1, α)-well-linked for $\mathcal{T}$, if and only if G is α-well-linked for every $\mathcal{T}' \subseteq \mathcal{T}$ of size at most k_1. Similarly, given a graph G, we say a cluster S of G is (k_1, α)-well-linked for $\Gamma \subseteq \text{out}(S)$ if $G[S \cup \text{out}(S)]$ is (k_1, α)-well-linked for Γ, and we simply say S is (k_1, α)-well-linked if S is (k_1, α)-well-linked for $\text{out}(S)$.

Definition 2.5 ((k_1, α)-violating cut) *Let S be a cluster of G that is not (k_1, α)-well-linked. We say that a cut (X, Y) of S is a* (k_1, α)-violating cut*, if $|E(X, Y)| < \alpha \cdot \min\{|\text{out}(X) \cap \text{out}(S)|, |\text{out}(Y) \cap \text{out}(S)|, \lfloor k_1/2 \rfloor\}$.*

Lemma 2.6 *Suppose we are given two sets $\Gamma_1, \Gamma_2 \subseteq \text{out}(S)$ such that $\Gamma_1 \cap \Gamma_2 = \emptyset, |\Gamma_1| = |\Gamma_2| \leq k_1/2$ and there is no flow $F : \Gamma_1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} \Gamma_2$ inside S, then we can find a (k_1, α)-violating cut (X, Y) of S.*

Proof: By setting up a max-flow instance and using the maximum-flow-minimum-cut theorem, we can find a cut (X, Y) of S such that $|\text{out}(X) \cap \Gamma_2| + |\text{out}(Y) \cap \Gamma_1| + (1/\alpha)|E(X, Y)| < |\Gamma_1|$. Then

$$\begin{aligned}
E(X, Y) &< \alpha(|\Gamma_1| - |\text{out}(X) \cap \Gamma_2| - |\text{out}(Y) \cap \Gamma_1|) \\
&\leq \alpha \min\{|\Gamma_1| - |\text{out}(Y) \cap \Gamma_1|, |\Gamma_2| - |\text{out}(X) \cap \Gamma_2|\} \\
&\leq \alpha \min\{|(\Gamma_1 \cup \Gamma_2) \cap \text{out}(X)|, |(\Gamma_1 \cup \Gamma_2) \cap \text{out}(Y)|\} \\
&= \alpha \min\{|(\Gamma_1 \cup \Gamma_2) \cap \text{out}(X)|, |(\Gamma_1 \cup \Gamma_2) \cap \text{out}(Y)|, \lfloor k_1/2 \rfloor\} \\
&\leq \alpha \min\{|\text{out}(X)|, |\text{out}(Y)|, \lfloor k_1/2 \rfloor\}.
\end{aligned}$$

Thus, (X, Y) is a (k_1, α)-violating cut. The equation used the fact that $|(\Gamma_1 \cup \Gamma_2) \cap \text{out}(X)| + |(\Gamma_1 \cup \Gamma_2) \cap \text{out}(Y)| = |\Gamma_1 \cap \Gamma_2| \leq k_1$. □

2.1.3 Sparsest Cut and the Flow-Cut Gap

Suppose we are given a graph $G = (V, E)$, and a subset $\mathcal{T} \subseteq V$ of k terminals. The sparsity of a cut $(S, \overline{S})$ in G is $\Phi(S) = \frac{|E(S,\overline{S})|}{\min\{|S \cap \mathcal{T}|, |\overline{S} \cap \mathcal{T}|\}}$, and the value of the sparsest cut in G is defined to be: $\Phi(G) = \min_{S \subset V} \Phi(S)$. The goal of the sparsest cut problem is, given an input graph G and a set $\mathcal{T}$ of terminals, to find a cut of minimum sparsity. Arora, Rao and Vazirani [7] have shown an $O(\sqrt{\log k})$-approximation algorithm for the sparsest cut problem. We denote this algorithm by $\mathcal{A}_{\text{ARV}}$, and its approximation factor by $\alpha_{\text{ARV}}(k) = O(\sqrt{\log k})$.

A problem dual to sparsest cut is the maximum concurrent flow problem. For the above definition of the sparsest cut problem, the corresponding variation of the concurrent flow problem asks to find the maximum value λ, such that every pair of terminals can send λ/k flow units to each other simultaneously with no congestion. The flow-cut gap is the maximum ratio, in any graph, between the value of the minimum sparsest cut and the maximum concurrent flow. The value of the flow-cut gap in undirected graphs, that we denote by $\beta_{\text{FCG}}(k)$ throughout the paper, is

$\Theta(\log k)$ [61, 39, 66, 10]. Therefore, if $\Phi(G) = \alpha$, then every pair of terminals can send $\frac{\alpha}{k\beta_{\text{FCG}}(k)}$ flow units to each other with no congestion. Equivalently, every pair of terminals can send $1/k$ flow units to each other with congestion at most $\beta_{\text{FCG}}(k)/\alpha$. Moreover, any matching on the set $\mathcal{T}$ of terminals can be fractionally routed with congestion at most $2\beta_{\text{FCG}}(k)/\alpha$. To see this, consider any pair (t_1, t_2) in the matching. We can send $1/k$ units flow from t_1 to every vertex in $\mathcal{T}$ and $1/k$ units flow from every vertex in $\mathcal{T}$ to t_2. Thus, we only need to send $2/k$ units flow between every pair of vertices.

Given a graph $G = (V, E)$, a cluster S of G, and a subset $\Gamma \subseteq \text{out}(S)$ of edges, the sparsest-cut instance $\mathsf{SC}(G, S, \Gamma)$ is defined by the graph $G[S \cup \text{out}(S)]$ and the terminals Γ. Observe that for all $\alpha \in [0, 1]$, the sparsity of the sparsest cut in $\mathsf{SC}(G, S, \Gamma)$ is at least α iff set S is α-well-linked with respect to Γ in graph G. If $\Gamma = \text{out}(S)$, then we simply denote instance by $\mathsf{SC}(G, S)$.

2.1.4 Well-linked Decomposition

In the α-well-linked decomposition, we are given a graph $G = (V, E)$, a cluster $C \subseteq V$ such that $|\text{out}(C)| = K$ and the parameter $\alpha \leq 1$. The goal is to partition C into many sub-clusters, such that each sub-cluster is $\alpha_{\text{WL}} := \alpha/\alpha_{\text{ARV}}(K)$-well-linked. In the process, we maintain a partition $\mathcal{W}$ of C, where at the beginning, $\mathcal{W} = \{C\}$. We then perform a number of iterations. In each iteration, we select a cluster $S \in \mathcal{W}$, and set up the sparsest cut instance $\mathsf{SC}(G, S)$. We run the algorithm $\mathcal{A}_{\text{ARV}}$ on the instance. If the sparsity of the cut produced by the algorithm is less than α, then we obtain a partition (X, Y) of S, with $|E(X, Y)| < \alpha \cdot \min\{|\text{out}(S) \cap \text{out}(X)|, |\text{out}(S) \cap \text{out}(Y)|\}$. In this case, we remove S from $\mathcal{W}$, and add X and Y instead. The algorithm ends when for every cluster $S \in \mathcal{W}$, $\mathcal{A}_{\text{ARV}}$ returns a partition of sparsity at least α. We are then guaranteed that every cluster in $\mathcal{W}$ is $\alpha/\alpha_{\text{ARV}}(K) = \alpha_{\text{WL}}$-well-linked. We remark that since $\alpha \leq 1$, every cluster $S \in \mathcal{W}$ in any iteration of the algorithm will have $\text{out}(S) \leq K$.

Theorem 2.7 *Let $\mathcal{W}$ be the partition obtained from the above α-decomposition of C for some $\alpha \leq 1/(8 \log K)$, where $K = |\text{out}(C)|$. Then, $\sum_{S \in \mathcal{W}} |\text{out}(S)| \leq K + O(\alpha K \log K)$.*

Proof: Define a potential function f as $f(i) = i + \beta i \log i$ for some $\beta \geq 0$ to be decided later. We then show that $\sum_{S \in \mathcal{W}} f(|\text{out}(S)|)$ can only decrease. Suppose in some iteration, we partition a cluster S into X and Y. It suffices to show that $f(|\text{out}(X)|) + f(|\text{out}(Y)|) \leq f(\text{out}(S))$. W.l.o.g, we assume $a := |\text{out}(X) \cap \text{out}(S)| \leq b := |\text{out}(Y) \cap \text{out}(S)|$. Thus, $c := |E(X, Y)| < \alpha a$. For our choice of α, we have

$c < a \le b$. Thus,

$$\begin{aligned}
&f(|\operatorname{out}(X)|) + f(|\operatorname{out}(Y)|) = f(a+c) + f(b+c) \\
&= a + c + b + c + \beta((a+c)\log(a+c) + (b+c)\log(b+c)) \\
&\le a + b + 2c\,(1 + \beta\log(a+b)) + \beta\left(a\log\frac{a+b+2c}{2} + b\log(a+b)\right) \\
&\le a + b + 2\alpha a(1 + \beta\log K) + \beta\left((a+b)\log(a+b) - a\log\frac{2(a+b)}{a+b+2c}\right) \\
&\le f(a+b) + 2\alpha a(1+\beta\log K) - \beta a/2.
\end{aligned}$$

If we set $\beta = 8\alpha$, then $2\alpha a(1+\beta\log K) \le 2\alpha a(1+1) \le 4\alpha a = \beta a/2$. The above quantity is at most $f(a+b) = f(|\operatorname{out}(S)|)$.

Thus, we have $\sum_{S\in\mathcal{W}} |\operatorname{out}(S)| \le \sum_{S\in\mathcal{W}} f(|\operatorname{out}(S)|) \le f(|\operatorname{out}(C)|) = K + 4\alpha K\log K = K + O(\alpha K\log K)$. □

Corollary 2.8 *Let $\mathcal{T}$ be a set of K terminals in graph $G = (V, E)$ and $\alpha < 1/(8\log K)$. Then, we can decompose G into many vertex induced sub-graphs such that each sub-graph $G[S]$ is $\alpha_{\text{WL}} = \alpha/\alpha_{\text{ARV}}(K)$-well-linked for the set $\mathcal{T}\cap S$, i.e, the set of terminals in the sub-graph. Moreover, the number of edges across different sub-graphs is at most $O(\alpha K\log K)$.*

Proof: If a terminal $\mathcal{T}$ has degree more than 1, we add a degree-1 vertex connected to the terminal and replace the terminal with the newly added vertex. It is easy to see that if an induced graph is α_{WL}-well-linked for a subset of new terminals, it must be α_{WL}-well-linked for the correspondent subset of original terminals. Thus, we can assume each vertex in $\mathcal{T}$ has degree 1. We apply the α-well-linked-decomposition to the cluster $V\setminus\mathcal{T}$. The decomposition partitions the graph $G\setminus\mathcal{T}$ into many induced sub-graphs. Since terminals have degree 1, we add each terminal to the unique sub-graph containing its neighbour. By doing so, we obtain a partition of G into induced sub-graphs. Notice that each cluster S obtained from the α-well-linked-decomposition is α_{WL}-well-linked (for $\operatorname{out}(S)$). If we let S' denote the union of S and the terminals adjacent to S, then the induced sub-graph $G[S']$ is α_{WL}-well-linked for $S'\cap\mathcal{T}$. The number of edges across different sub-graphs is $O(\alpha K\log K)$, by Theorem 2.7. □

2.1.5 Boosting the Well-Linkedness

Suppose $G = (V, E)$ is α-well-linked for a set $\mathcal{T}\subseteq V$ of terminals for some $\alpha = o(1)$. Can we select a large subset $\mathcal{T}'\subseteq\mathcal{T}$ (say, $|\mathcal{T}'| \ge \Omega(\alpha)|\mathcal{T}|$) such that G is $\Omega(1)$-well-linked for $\mathcal{T}'$? The answer is positive. We call the process of selecting $\mathcal{T}'$ *boosting the well-linkedness of G*.

The first technique to boost the well-linkedness is called *grouping*. It was introduced by Chekuri, Khanna and Shepherd [24], and has since been widely used in algorithms for network routing [25, 72, 3, 31]. [24] proved that we can efficiently compute a partition $\mathcal{U}$ of $\mathcal{T}$ into groups of size $O(1/\alpha)$, such that G is $\Omega(1)$-well-linked

for any $\mathcal{T}' \subseteq \mathcal{T}$ containing at most 1 terminal from each group in $\mathcal{U}$. In particular, this implies that the size of $\mathcal{T}'$ can be as large as $\Omega(\alpha)\,|\mathcal{T}|$.

The well-linkedness factor achieved using the above grouping technique can only be $1-\epsilon$ for any $\epsilon > 0$. In our application, it is essential that we boost the well-linkedness to 1. This can be achieved using a more sophisticated technique, due to Chekuri, Khanna and Shepherd [24]. In order to state the result of [24], we need the following definition.

Definition 2.9 (degenerate graph) *A simple graph H is d-degenerate for some integer $d > 0$, if every induced sub-graph of H has a vertex of degree at most d.*

A d-degenerate graph H has some nice properties. For example, every sub-graph of H has average degree at most $2d$. Also, we can greedily find an independent set of H of size $\lceil |V_H|\,/(d+1) \rceil$.

Ideally, we hope to prove that there is an $O(1/\alpha)$-degenerate graph Z over $\mathcal{T}$, such that G is 1-well-linked for every independent set $\mathcal{T}'$ of Z. However, due to some technical issues, [24] did not give this result. Instead, the subset of terminals for which G is 1-well-linked is selected in a two-step fashion. We construct an $O(1/\alpha)$-degenerate graph $Z = (\mathcal{T}, E_Z)$. Then, given any independent set of $\mathcal{T}'$ of Z, we can construct a $O(1)$-degenerate graph $Y = (\mathcal{T}', E_Y)$ such that G is 1-well-linked for any independent set $\mathcal{T}''$ of Y.

For some technical reason, we need our graphs Z and Y to satisfy the following property :

(A1) Every induced matching of Z (as well as Y) can be routed in G integrally with congestion 1.

An induced matching is an induce sub-graph that is a matching. Now we can formally state the result of [24].

Theorem 2.10 *Given a graph $G = (V, E)$ and a subset $\mathcal{T} \subseteq V$ of terminals such that G is α-well-linked for $\mathcal{T}$, for some $\alpha \in (0, 1]$, we can efficiently construct an $O(1/\alpha)$-degenerate graph $Z = (\mathcal{T}, E_Z)$ satisfying Property (A1), such that the following is true for every independent set $\mathcal{T}' \subseteq \mathcal{T}$ of Z. We can efficiently construct a 4-degenerate graph $Y = (\mathcal{T}', E_Y)$ satisfying Property (A1), such that G is 1-well-linked for every independent set $\mathcal{T}'' \subseteq \mathcal{T}'$ of Y.*

Since the above theorem is only given implicitly in [24], we prove it in Section 2.2 for the sake of completeness.

2.2 Boosting the Well-Linkedness

In this section, we prove Theorem 2.10 that is implicitly proved in [24]. Recall the settings of theorem: we are given a graph $G = (V, E)$, a set $\mathcal{T} \subseteq V$ of terminals and a parameter $\alpha < 1$.

2.2.1 Initial Grouping

Our graph Z is defined by a partition $\mathcal{U}$ of $\mathcal{T}$ into groups of size $\Theta(1/\alpha)$. Z contains an edge between two terminals $t, t' \in \mathcal{T}$ if and only if t and t' are in a same group in $\mathcal{U}$. In other words, Z is the union of disjoint cliques, where each clique is on some $U \in \mathcal{U}$. We give the grouping $\mathcal{U}$ of $\mathcal{T}$ in this section. Let q be the smallest even integer with $q \geq 4/\alpha$, so $4/\alpha \leq q \leq 4/\alpha + 2$. We assume $|\mathcal{T}| \geq q$ since otherwise we can let Z be the complete graph on $\mathcal{T}$ and the theorem is clearly true.

Lemma 2.11 *We can efficient construct a partition $\mathcal{U}$ of the set $\mathcal{T}$ of terminals into groups of size between q and $3q-3$, and associate each group $U \in \mathcal{U}$ with a (connected) cluster S_U of G satisfying $S_U \supseteq U$, such that the following is true. $\bigcup_{U \in \mathcal{U}} S_U = V$ and for any two distinct groups U and U' in $\mathcal{U}$, we have $|S_U \cap S_{U'}| \leq 1$.*

Proof: The algorithm is as follows. Take an arbitrary spanning tree T of G and root it at an arbitrary vertex. Take the deepest vertex v in T such that the sub-tree T_v of T rooted at v contains at least q terminals, breaking ties arbitrarily. Then, for any child u of v in T, T_u contains less than q terminals in $\mathcal{T}$. A group U and its associated cluster S_U is constructed as follows. Initially $U = \{v\} \cap \mathcal{T}$ and $S_U = \{v\}$. Take an arbitrary order for the children of v. For each child u in that order, if $|U| < q$ we add the terminals in T_u to U and vertices in T_u to S_U. Since each T_u contains at most $q-1$ terminals, our final U has $q \leq |U| \leq 2q-2$. We let $\mathcal{U} = \mathcal{U} \cup \{U\}$, $\mathcal{T} = \mathcal{T} \setminus U$ and $T = T \setminus (S_U \setminus v)$. Notice that S_U is indeed a connected cluster containing U, and the resulting T is still a tree. We repeat this process until at most $3q-3$ terminals remain. The final group added to $\mathcal{U}$ will be the set of remaining terminals in $\mathcal{T}$, and the cluster for this group is the set of all the vertices left in T. Thus, $\bigcup_{U \in \mathcal{U}} S_U = V$. For any two different groups $U, U' \in \mathcal{U}$, we have $|S_U \cap S_{U'}| \leq 1$. This is true since whenever we constructed a group U'', we remove all but one vertex in $S_{U''}$ from T. □

With the grouping $\mathcal{U}$, we construct $Z = (\mathcal{T}, E_Z)$ as we mentioned earlier: Z contains an edge between two terminals $t, t' \in \mathcal{T}$ if and only if t and t' are in a same group in $\mathcal{U}$. Since each group $U \in \mathcal{U}$ has size at most $3q-3 = O(1/\alpha)$, the graph Z is $O(1/\alpha)$-degenerate. An induced matching of the graph Z contains 0 or 2 vertices from each group $U \in \mathcal{U}$. If it contains 2 vertices from $U \in \mathcal{U}$, we can connect t and t' in $G[S_U]$. Since the graphs $\{G[S_U] : U \in \mathcal{U}\}$ are edge-disjoint (since every two graph has at most one vertex in common), the induced matching can be routed edge-disjointly in G. Thus, Z satisfies Property (A1).

2.2.2 Centers and Pseudo-Centers

In this section, we define *centers* and *pseudo-centers*, and prove some lemmas that are essential in the algorithm.

Definition 2.12 *Given a sub-graph $G' = (V', E')$ of G and a set $U \subseteq V'$ of at least q terminals in V', we say a vertex $v \in V'$ is a* center *of G' w.r.t U, iff v can send one flow unit to U in graph G', such that every terminal in U receives at most $1/q$ flow units, and the flow on every edge in G' is at most $1/2$.*

We say that v is a pseudo-center *of G' w.r.t U, iff v can send one flow unit to U, with the same restrictions as above, except that the cut edges of G in G' are allowed to carry up to one flow unit.*

Claim 2.13 *v is a center of G' w.r.t U if and only if there is no edge e whose removal separates v and more than $|U| - q/2$ terminals in U in the graph G'.*

Proof: The "only if" direction is easy to see: if there is such an edge e, then e has to carry more than $1/2$ unit flow. Now consider a "if" direction. Consider following the network flow problem. We take the graph G', where each edges in G' has capacity $1/2$. Then add a super sink s and an edge of capacity $1/q$ from each terminal in$U$ to s. If v is not a center of G', we can not send 1 unit flow from v to s in the network. By the maximum-flow-min-cut theorem, there is a cut of capacity less than 1 between s and v. The cut can only contain one edge in G' since each edge in G' has capacity $1/2$. Also, it has to contain an edge in G' since otherwise the cut has to contain all edges between s and U, which have total capacity at least 1. Therefore, the cut contains exactly one edge e in G' and less than $q/2$ edges between U and s. Let U' be the set of terminals whose corresponding edges to s are cut. Then $|U'| < q/2$ and the removal of e in G' separates v and all terminals in $U \setminus U'$. □

Corollary 2.14 *1. If v is a center of G' w.r.t U and u is in the same 2-edge-connected component as v in G', then u is also a center.*

2. The set of centers in G' is connected.

3. G' contains at least a center w.r.t U.

Proof: The first statement holds since no edge separates u and v and no edge separates v from more than $|U| - q/2$ terminals in $|U|$. For the second statement, consider a path from u to v containing w. Then, no edge can separate w from $\{u, v\}$. If u and v are both centers, then w must also be a center. For the third statement, consider the tree of maximal 2-edge-connected components of G' and root the tree arbitrarily. Let C be the lowest component in the tree such that the sub-tree rooted at C contains at least $q/2$ terminals in U. Then, every vertex in C is a center. □

The following lemma is crucial in proving Theorem 2.10.

Lemma 2.15 *Let $\mathcal{T}' \subseteq \mathcal{T}$ be a subset of terminals. Suppose each terminal $t \in \mathcal{T}'$ is associated with a sub-graph G_t of G and a set $W_t \subseteq \mathcal{T}$ of terminals such that*

1. $t \in W_t \subseteq V(G_t)$ and $|W_t| \geq q$ for every $t \in \mathcal{T}'$.

2. For every $t \in \mathcal{T}'$, t is a pseudo-center of G_t w.r.t W_t.

3. The sets $\{W_t : t \in \mathcal{T}'\}$ are disjoint.

4. The sub-graphs $\{G_t : t \in \mathcal{T}'\}$ are edge-disjoint.

Then G is 1-well-linked for $\mathcal{T}'$.

Proof: Let $\mathcal{T}_1', \mathcal{T}_2'$ be any two disjoint subsets of $\mathcal{T}'$ such that $|\mathcal{T}_1'| = |\mathcal{T}_2'|$. It suffices to show that we can find a flow $\mathcal{T}_1' \overset{1:1}{\rightsquigarrow}_1 \mathcal{T}_2'$ in G.

Since each terminal $t \in \mathcal{T}'$ is a pseudo-center of the graph G_t w.r.t W_t, we can send one flow unit from t to W_t in G_t, such that each terminal in W_t receives at most $1/q$ flow units. Moreover, each edge in G_t carries at most $1/2$ flow unit, except for the cut edges of G, which may carry up to 1 flow unit. Since we have selected q to be an even integer, from the integrality of flow, we can assume that the flow is $1/q$-integral. Then, every terminal in W_t receives either 0 or $1/q$ flow units. Let $W_t' \subseteq W_t$ be the set of exactly q vertices that receive $1/q$ flow units and let F_t be the flow of value 1 sent from t to W_t'.

Let $\mathcal{T}_1 = \bigcup_{t \in \mathcal{T}_1'} W_t'$, and let $\mathcal{T}_2 = \bigcup_{t \in \mathcal{T}_2'} W_t'$. Notice that $|\mathcal{T}_1| = q|\mathcal{T}_1'| = q|\mathcal{T}_2'| = |\mathcal{T}_2|$ since the sets $\{W_t' : t \in \mathcal{T}'\}$ are disjoint. Recall that the graph G is α-well-linked for $\mathcal{T}$. Thus there is a flow $\mathcal{T}_1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} \mathcal{T}_2$ in graph G. Scaling this flow down by factor $q \geq 2/\alpha$, we obtain a flow F^*, where every terminal in $\mathcal{T}_1$ sends $1/q$ flow units, every terminal in $\mathcal{T}_2$ receives $1/q$ flow units, and the edge congestion is at most $1/2$. Let flow F' be the concatenation of $\bigcup_{t \in \mathcal{T}_1'} F_t$, the flow F^*, and $\bigcup_{t \in \mathcal{T}_2'} F_t$. Then every terminal in $\mathcal{T}_1'$ sends one flow unit in F', and every terminal in $\mathcal{T}_2'$ receives one flow unit. Every edge of G carries at most one flow unit in F', except for the cut edges of G, which may carry up to 1.5 flow units.

We show that after removing cycles in the flow F', the resulting flow has congestion at most 1 on every edge in G. To see this, we only need to consider the cut edges of G. Since we removed cycles, each cut edge can only carry flow in one direction. For a cut edge whose removal breaks G into $G[X]$ and $G[Y]$, we can assume it carries flow from X to Y w.l.o.g. Then, it must be the case that the amount of flow sent from X to Y is an integer. This holds since every terminal in $\mathcal{T}_1' \cup \mathcal{T}_2'$ sends or receives exactly 1 flow unit. Since the congestion of the cut edge is at most 1.5, it is at most 1. □

Now, we try to construct the graph Y over an independent set $\mathcal{T}'$ of Z. Thus, $\mathcal{T}'$ contains at most one terminal from each group $U \in \mathcal{U}$. If we could claim that every terminal $t \in \mathcal{T}'$ is a pseudo-center of $G[S_U]$ w.r.t U for the group $U \in \mathcal{U}$ containing t, then we would be done by Lemma 2.15. (For every $t \in \mathcal{T}'$, we can let $W_t = U$ and $G_t = G[S_U]$, where U is the unique group in $\mathcal{U}$ containing t.) However this is not always the case. To overcome this difficulty, we use the *tagging* procedure suggested by [24], where some cluster S_U may tag some other cluster $S_{U'}$, such that the unique terminal $t \in U \cap \mathcal{T}'$ is a pseudo-center of a larger cluster formed by merging S_U and $S_{U'}$. This is done in the next subsection.

2.2.3 The Tagging Procedure

Focus on some terminal $t \in \mathcal{T}'$ and its cluster S_U, where $U = U_t$. Assume t is not a pseudo-center of $G[S_U]$ w.r.t U. Then by the three statements of Corollary 2.14, there must be at least one cut edge of $G[S_U]$ that separates t from all centers of $G[S_U]$. Assume there are ℓ edges $e_1, e_2, \cdots, e_\ell$. Let A_i be the connected component of $G[S_U] \setminus e_i$ containing t. We name the edges in a way such that $A_1 \subset A_2 \subset \cdots \subset A_\ell$. Let i be the smallest index such that there is an edge between A_i and $V \setminus S_U$(see

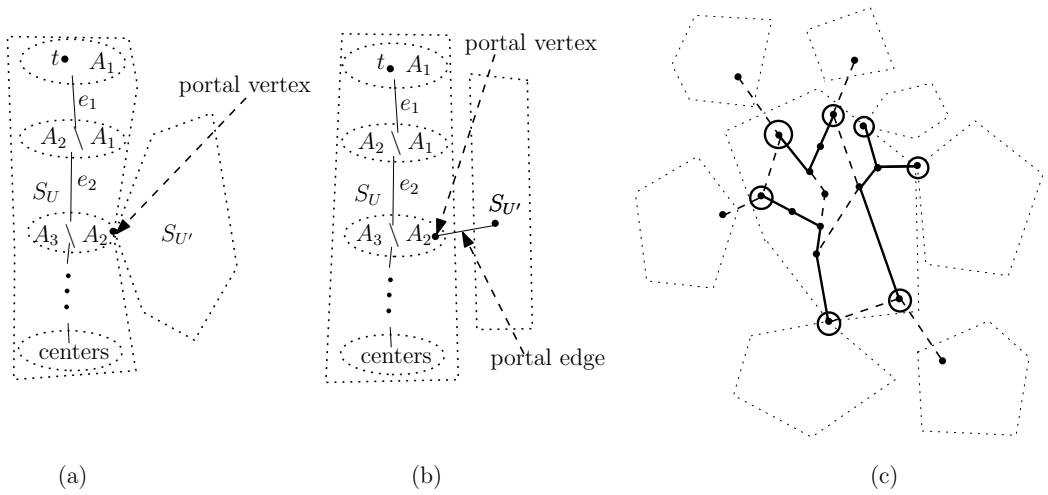

Figure 2.1: Tagging. (a): S_U tags $S_{U'}$ and there is no portal edge for S_U. (b): S_U tags $S_{U'}$ and there is a portal edge for S_U. (c) The partition for portals. The 7 portals marked by empty circles are partitioned in to 3 subsets, and each subset is connected by a sub-graph, denoted by the solid lines.

Figure 2.1). If such i does not exist, then all the edges $e_1, e_2, \cdots, e_\ell$ are cut edges of G and thus t is a pseudo-center of U. Suppose now i exists. If some vertex $v \in A_i \setminus A_{i-1}$ is in some cluster $S_{U'}$ for $U' \in \mathcal{U}$ and $U' \neq U$, then we say S_U tags $S_{U'}$ via the *portal vertex* v(see Figure 2.1(a)). Otherwise there is an edge $(u, v) \in E$ such that $u \in A_i \setminus A_{i-1}$, u is not in any cluster other than S_U, $v \in S_{U'}$ for some $U' \in \mathcal{U}, U' \neq U$ and $v \notin S_U$. In this case, we say S_U tags $S_{U'}$ via the *portal vertex* v and e is the *portal edge* for S_U(see Figure 2.1(b)). In the former case, there is no portal edge for S_U. We remark that if e is a portal edge, then e is not inside any cluster S_U. We also remark that if $S_U, U \in \mathcal{U}$ does not tag any cluster, then either U does not contain a terminal in $\mathcal{T}'$, or the unique terminal in $U \cap \mathcal{T}'$ is a pseudo-center of $G[S_U]$ w.r.t U.

Lemma 2.16 *Suppose S_U tags some cluster and t is the unique terminal in $U \cap \mathcal{T}'$. $S_{U'}$ be some other cluster. Let P be a path connecting S_U and $S_{U'}$ via the portal for S_U(i.e, if there is a portal edge for S_U, then the first edge of P is the portal edge; otherwise, the first vertex of P is the portal vertex for S_U). Moreover, only the first vertex of P is in S_U. Then, t is a pseudo-center of $G' = G[S_U] \cup G[S_{U'}] \cup P$ w.r.t $U \cup U'$.*

Proof: Define $e_1, e_2, \cdots, e_\ell$ and $A_1, A_2, \cdots, A_\ell$ as above. Let i be the smallest index such that there is an edge in G between A_i and $V \setminus S_U$. Notice that $e_1, e_2, \cdots, e_{i-1}$ are cut edges of G. It suffices to show that there is no edge e' of $E_{G'} \setminus \{e_1, e_2, \cdots, e_{i-1}\}$ such that e' separates t from more than $|U \cup U'| - q/2$ terminals in $|U \cup U'|$ in G'. If e' is from the path P or the graph $G[S_{U'}]$, then clearly t is still connected to the at least q terminals of U in $G' \setminus e'$. Thus, we can assume e' is in $G[S_U]$. Noticing that $e_1, e_2, \cdots, e_\ell$ are the only edges whose removal separates t and the centers of $G[S_U]$, and $e' \notin \{e_1, e_2, \cdots, e_{i-1}\}$, we can assume $e' = e_j$ for some $i \leq j \leq \ell$. In this case, t is

connected to all terminals of U' in $G' \setminus e'$, via the path P. Thus, t is a pseudo-center of G' w.r.t $U \cup U'$. □

We now build a graph $Z' = (\mathcal{U}, E_{Z'})$. Then our final graph Z in Theorem 2.10 is obtained from Z' by replacing each group $U \in \mathcal{U}$ that contains a terminal $t \in \mathcal{T}'$ with t and remove the groups $U \in \mathcal{U}$ that do not contain terminals in $\mathcal{T}'$.

If S_U tags $S_{U'}$, we then add an edge (U, U') to Z'. With these edges, we guarantee that U and U' can not both appear in an independent set of Z' if U tags U'. However, if both $S_{U'}$ and $S_{U''}$ tags S_U, U' and U'' may both appear in an independent set. We now deal with this case. Consider a group $U \in \mathcal{U}$ such that S_U is tagged by at least two clusters. Consider the multi-set P_U of portal vertices in S_U via which these clusters tag S_U. We apply the following claim to obtain a partition $\mathcal{B}_U$ of P_U.

Claim 2.17 *We can find a partition $\mathcal{B}_U$ of P_U into subsets of size 2 and 3, each subset associated with a connected sub-graph of $G[S_U]$ containing all portal vertices of the subset, such that the sub-graphs for all subsets in the partition are edge-disjoint. (See Figure 2.1(c).)*

The proof of the above claim is exactly the same as the proof of Lemma 2.11 for $q = 2$ and thus we omit it here. If $S_{U'}$ tags S_U via the portal vertex v and $S_{U''}$ tags S_U via the portal vertex u, and u and v are in the same subset of the partition $\mathcal{B}_U$, then we add an edge (U', U'') in the graph Z'. This finishes the description of Z'.

Consider any independent set $\mathcal{U}' \subseteq \mathcal{U}$ of Z'. Recall that each group $U \in \mathcal{U}'$ contains 0 or 1 terminal in $\mathcal{T}'$. Let $\mathcal{T}'' \subseteq \mathcal{T}'$ be the terminals contained in some group in $\mathcal{U}'$.

Lemma 2.18 *G is 1-well-linked for $\mathcal{T}''$.*

Proof: Consider a terminal $t \in \mathcal{T}''$ and suppose $t \in U \in \mathcal{U}'$. If S_U does not tag any cluster, we let $G_t = G[S_U]$ and $W_t = U$. Notice that in this case, t is a pseudo center of G_t w.r.t W_t. If S_U tags $S_{U'}$ and $S_{U'}$ is only tagged by S_U, we let $W_t = U \cup U'$. Let $G_t = G[S_U] \cup G[S_{U'}]$ if there is no portal edge for S_U and $G_t = G[S_U] \cup G[S_{U'}] \cup e'$ if e' is the portal edge for S_U. By Lemma 2.16, t is a pseudo-center of G_t w.r.t W_t in this case. If S_U tags $S_{U'}$ and $S_{U'}$ is also tagged by some other clusters, then let v be the portal vertex via which S_U tags S'_U. Consider the subset $B \in \mathcal{B}_{U'}$ that contains v and let u be some other portal vertex in B. Suppose u is for the cluster $S_{U''}$; in other words, $S_{U''}$ tags $S_{U'}$ via the portal u. Then we let $W_t = U \cup U''$, and the graph G_t is constructed as follows. It contains the graphs $G[S_U]$, $G[S_{U''}]$ and the sub-graph of $G[S_{U'}]$ for $B \in \mathcal{B}_{U'}$ defined in Claim 2.17. Moreover, we add portal edges for S_U and $S_{U''}$, if they exist. Again, by Lemma 2.16, t is a pseudo-center of G_t w.r.t W_t.

Now we show that the graphs $\{G_t : t \in \mathcal{T}''\}$ are edge-disjoint. First consider the edges in some cluster S_U. If U contains a terminal $t \in \mathcal{T}''$, then $U \in \mathcal{U}'$ and thus for all the clusters $S_{U'}$ that tags S_U, we have $U' \notin \mathcal{U}'$. Thus, $G[S_U]$ is only used in G_t. If U does not contain a terminal in $\mathcal{T}''$ and S_U is not tagged by any clusters, then $G[S_U]$ is not used at all. If U does not contain a terminal in $\mathcal{T}''$ and S_U is tagged by only one cluster $S_{U'}$, then $G[S_U]$ is used by $G_{t'}$ if U' contains a terminal $t' \in \mathcal{T}''$ and

not used otherwise. Now consider the final case that U does not contain a terminal in $\mathcal{T}''$ and S_U is tagged by at least 2 clusters. In this case, we have constructed a partition $\mathcal{B}_U$ for the set P_U of portal vertices via which the clusters tag S_U. Among the 2 or 3 clusters whose correspondent portals are in a same subset $B \in \mathcal{B}_U$, at most 1 cluster $S_{U'}$ can have $U' \in \mathcal{U}'$. Thus, the sub-graph for B defined in Claim 2.17 is used at most once. Notice that the sub-graphs for all subsets $B \in \mathcal{B}_U$ are disjoint. Thus, each edge in $G[S_U]$ is used only once.

Then, we consider the portal edges. Recall that a portal edge is not inside any cluster. Suppose S_U tags $S_{U'}$ via a portal edge $e = (v, u)$, where $v \in S_U$ and $u \in S_{U'}$. Then v is not $S_{U''}$ for any $U'' \in \mathcal{U}$ and $U'' \neq U$. Thus, if e is a portal edge for some other cluster $S_{U''}$, it must be the case that $S_{U''}$ tags S_U via the portal edge e and portal vertex v. If $U'' \neq U'$, then u is in both $S_{U'}$ and $S_{U''}$ and thus $S_{U''}$ should tag $S_{U'}$ (or some other cluster containing u) without portal edges. If $U'' = U'$ then only one group in $\{U, U'\}$ can be in $\mathcal{U}'$ and thus the portal edge is used only once.

By the similar case-by-case analysis, it is easy to see that the sets $\{W_t : t \in \mathcal{T}'\}$ are disjoint. Thus, we have identified a sub-graph G_t of G and a set of terminals $W_t \in V(G_t)$ for each $t \in \mathcal{T}''$. The sub-graphs $\{G_t : t \in \mathcal{T}''\}$ are edge-disjoint and sets $\{W_t : t \in \mathcal{T}''\}$ are disjoint. Moreover, t is an pseudo-center of G_t w.r.t W_t for every $t \in \mathcal{T}'$. Thus, G is 1-well-linked for $\mathcal{T}''$ by Lemma 2.15. □

Recall that Z is obtained from Z' by replacing each group $U \in \mathcal{U}$ containing a terminal $t \in \mathcal{T}'$ with the terminal t and removing groups $U \in \mathcal{U}$ containing no terminals in $\mathcal{T}'$. Thus, G is 1-well-linked for any independent set $\mathcal{T}'' \subseteq \mathcal{T}'$ of Z.

Now we show that Z satisfies Property (A1). Consider any induced-matching of Z' such that every group in the induced matching contains a terminal in $\mathcal{T}'$. Focus on a matching edge (U, U') such that $t \in U \cap \mathcal{T}'$ and $t' \in U' \cap \mathcal{T}'$. If S_U tags $S_{U'}$, we connect t and t' via the concatenation of a path in $G[S_U]$, a path in $G[S_{U'}]$, and possibly the portal edge for S_U. If both S_U and $S_{U'}$ tags $S_{U''}$ and the two portal vertices for S_U and $S_{U'}$ are in the same subset $B \in \mathcal{B}_{U''}$, we connect t and t' via the concatenation of a path in $G[S_U]$, a path $G[S_{U'}]$, a path in the sub-graph of $G[S_{U''}]$ for B defined in Claim 2.17, and possibly the portal edge for S_U and the portal edge for $S_{U'}$. By similar and tedious arguments as above, the routing paths for the terminals are edge-disjoints.

Then we show that Z' is 4-degenerate. In the first step, we add edges (U, U') for all pairs such that S_U tags $S_{U'}$. Notice that each cluster tags at most 1 cluster, the graph we obtained 2-degenerate. In the second step, the degree of each vertex in Z' is increased by at most 2 since the subsets in each $\mathcal{B}_U$ have size at most 3. Thus, Z' is 4-degenerate, implying that Z is 4-degenerate.

2.3 Routing in Crossbar

Chekuri, Khanna and Shepherd [24, 25, 23] have suggested the following high-level approach to solving EDP instances $(G, \mathcal{T}, \mathcal{M})$. They start by defining a graph called a *crossbar*. Roughly speaking, a graph H with a subset $\mathcal{T}' \subseteq V_H$ of vertices is called a crossbar, if given any matching $\mathcal{M}'$ over the vertices of $\mathcal{T}$, we can route a large

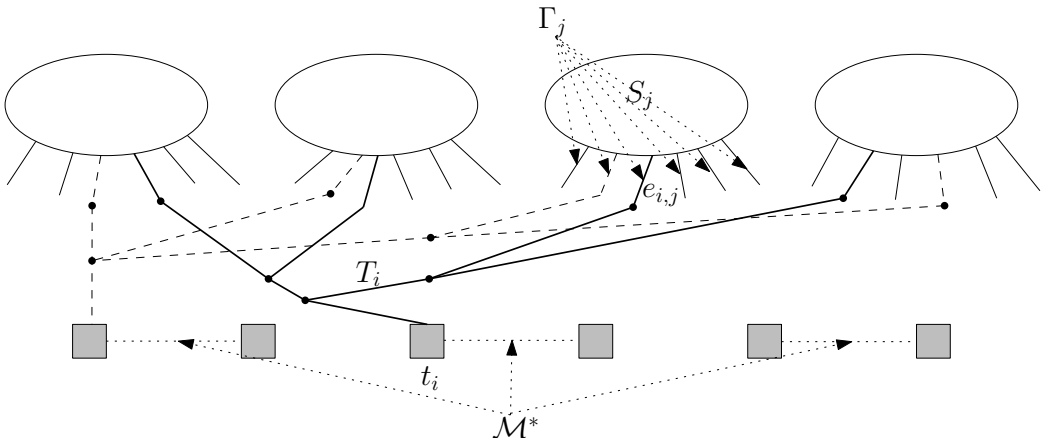

Figure 2.2: (γ^*, k^*)-crossbar. The ellipses denote the γ^* clusters; fat solid lines represent a tree $T_i \in \mathbf{T}^*$; dashed lines represent another tree; $\mathcal{M}^* \subseteq \mathcal{M}$ is a perfect matching on the $2k^*$ terminals.

sub-matching of $\mathcal{M}'$ with small congestion in H. They then note that if we could "embed" a crossbar H in graph G with small congestion, with $\mathcal{T}'$ being a large subset of $\mathcal{T}$, then we can obtain a good approximation to EDPwC with small congestion.

An algorithm for constructing a crossbar with a constant congestion follows from the recent work of Chuzhoy [31]. We follow the approach in [31], except that we construct a crossbar with congestion 2. We now give the exact definition the crossbar used in [31] in Section 2.3.1 and prove that we can route many pairs in the crossbar (Theorem 2.21) in Sections 2.3.2 and 2.3.3.

2.3.1 Crossbar

Definition 2.19 ((γ^*, k^*)-crossbar) *Given a* EDP *instance $(G, \mathcal{T}, \mathcal{M})$ and $\gamma^*, k^* \in \mathbb{Z}_+$, $(\mathcal{S}^*, \mathbf{\Gamma}^*, \mathbf{T}^*, \mathcal{M}^*)$ is called a (γ^*, k^*)-crossbar if the following are true.*

1. *$\mathcal{S}^* = \{S_j : j \in [\gamma^*]\}$ is a family of γ^* disjoint clusters in $G \setminus \mathcal{T}$.*
2. *$\mathbf{\Gamma}^* = \{\Gamma_j : j \in [\gamma^*]\}$. For every $j \in [\gamma^*]$, $\Gamma_j \subseteq \mathrm{out}(S_j)$ is a subset of out-going edges of S_j such that $|\Gamma_j| = 2k^*$.*
3. *For every $j \in [\gamma^*]$, S_j is 1-well-linked for Γ_j.*
4. *$\mathbf{T}^* = \{T_i : i \in [2k^*]\}$. For every $i \in [2k^*]$, T_i is a tree, containing exactly one terminal $t_i \in \mathcal{T}$ and a representative edge $e_{i,j} \in \Gamma_j$ for every $j \in [\gamma^*]$.*
5. *$\Gamma_j = \{e_{i,j} : i \in [2k^*]\}$; that is, every edge in Γ_j is a representative edge for exactly one tree T_i.*
6. *The $2k^*$ terminals $t_1, t_2, \cdots, t_{2k^*}$ are distinct; moreover, $\mathcal{M}^* \subseteq \mathcal{M}$ is a perfect matching on the $2k^*$ terminals.*

In the above crossbar, two main components are the set $\mathcal{S}^*$ of clusters and the set $\mathbf{T}^*$ of trees. The clusters and the trees are well connected in the sense that each tree $T_i \in \mathbf{T}^*$ contains an edge $e_{i,j} \in \text{out}(S_j)$ for every $S_j \in \mathcal{S}^*$. The congestion of the crossbar is then the maximum number of times an edge is used by the clusters and the trees. The formal definition is given below.

Definition 2.20 *Given a crossbar $(\mathcal{S}^*, \mathbf{\Gamma}^*, \mathbf{T}^*, \mathcal{M}^*)$ for an* EDP *instance $(G, \mathcal{T}, \mathcal{M})$, the congestion of the crossbar is the multiplicity of the multi-set $\biguplus_{i\in[2k^*]} E(T_i) \uplus \biguplus_{j\in[\gamma^*]} E(G[S_j])$, where $\uplus$ is the multi-set sum.*

Notice that the clusters $\{S_j : j \in [\gamma^*]\}$ are disjoint. If the congestion of a crossbar $(\mathcal{S}^*, \mathbf{\Gamma}^*, \mathbf{T}^*, \mathcal{M}^*)$ is at most c, then every edge of G is used by at most c trees in $\mathbf{T}^*$, and every edge in any cluster $G[S_j]$ is used by at most $c-1$ trees in $\mathbf{T}^*$.

[31] implicitly proved that in order to give an poly-logarithmic approximation for EDP with congestion c, it suffices to construct (γ^*, k^*)-crossbar of congestion of congestion c with $k^* = k/\text{polylog}(k)$ and a suitable $\gamma^* = \text{polylog}(k)$.

Theorem 2.21 ([31]) *Let $(G, \mathcal{T}, \mathcal{M})$ be an* EDP *instance with $k = |\mathcal{M}|$. Given a (γ^*, k^*)-crossbar $(\mathcal{S}^*, \mathbf{\Gamma}^*, \mathbf{T}^*, \mathcal{M}^*)$ of congestion c for some suitable $\gamma^* = O(\log^2 k)$, we can route $\Omega\left(k^*/\log^4 k^*\right)$ pairs in $\mathcal{M}^*$ with congestion at most c in G.*

In the remaining part of this section, we prove Theorem 2.21. The proof contains two steps. In the first step, we "embed" an expander H in the graph G with congestion c, using the cut-matching game introduced by Khandekar, Rao and Vazirani [51]. In the embedding, each vertex of H corresponds to a connected component in G (to be more specific, a tree), and each edge of H corresponds to a path connecting the two connected components for the two end-points of the edge. Then in the second step, we route many pairs in H *vertex-disjointly*. The routing in H be naturally converted to routing in G, and the vertex-disjointness guarantees that the congestion of the routing in G is at most c.

2.3.2 Embedding an Expander via Cut Matching Game

In this section, we embed an expander in G. Given a graph G', the expansion of $G' = (V', E')$ is defined as $\min_{S\subseteq V', 1\le|S|\le|V'|/2} \frac{|E'(S, V'\setminus S)|}{|S|}$, where $E'(S, V' \setminus S)$ is the set of edges in G' between S and $V' \setminus S$. An α-expander is a graph with expansion at least α.

We use the cut-matching game introduced by Khandekar, Rao and Vazirani [51]. In this game, we are given a set V of N vertices, where N is even. There are two players: a cut player, whose goal is to construct an expander X on the set V of vertices, and a matching player, whose goal is to delay its construction. The game is played in iterations. We start with the graph X containing the set V of vertices, and no edges. In each iteration j, the cut player computes a partition (A_j, B_j) of V into two equal-sized sets, and the matching player returns some perfect matching M_j between the two sets. The edges of M_j are then added to X. Khandekar, Rao and

Vazirani have shown that there is a strategy for the cut player, guaranteeing that after $O(\log^2 N)$ iterations we obtain a $\frac{1}{2}$-expander w.h.p. Subsequently, Orecchia et al. [70] have shown the following improved bound:

Theorem 2.22 ([70]) *There is a probabilistic algorithm for the cut player, such that, no matter how the matching player plays, after $\gamma_{\text{CMG}}(N) = O(\log^2 N)$ iterations, graph X is an $\alpha_{\text{CMG}}(N) = \Omega(\log N)$-expander, with constant probability.*

We now embed an $\Omega(\log k^*)$-expander H in G using Theorem 2.22. The set of vertices of H is $V_H = \mathcal{T}(\mathcal{M}^*)$. The embedding of each vertex $t_i \in V_H$ is T_i, the unique tree in $\mathbf{T}^*$ containing t_i. In order to compute the set of edges of H and their embedding into G, we perform the cut-matching game on the set V_H of $2k^*$ vertices, using Theorem 2.22. Let $\gamma^* = \gamma_{\text{CMG}}(2k^*) = O(\log^2 k^*) = O(\log^2 k)$. Recall that this game consists of γ^* iterations, where in iteration $j \in [\gamma^*]$, the cut player computes a partition (A_j, B_j) of V_H, and the matching player responds with a perfect matching between the vertices of A_j and the vertices of B_j. Given the partition (A_j, B_j) of V_H computed by the cut player, we find a corresponding partition (A'_j, B'_j) of Γ_j, as follows. For each $t_i \in V_H$, if $t_i \in A_j$, then we add $e_{i,j}$ to A'_j, and otherwise we add it to B'_j. Since the set S_j is 1-well-linked for Γ_j, we can find a collection $\mathcal{Q}_j : A'_j \overset{1:1}{\rightsquigarrow}_1 B'_j$ of edge-disjoint paths contained in S_j. These paths define a matching M_j between the edges of A'_j and the edges of B'_j, which in turn defines a matching between the terminals of A_j and the terminals of B_j. We then add the edges of the matching to the graph H. For each such edge $e \in M_j$, its embedding into G is the corresponding path in $\mathcal{Q}_j$. From Theorem 2.22, after γ^* iterations, we obtain a $\Omega(\log k^*)$-expander H, together with its embedding into G. It is easy to see that the embedding causes congestion at most c in graph G, since the multi-set of edges used by the embedding (count multiplicity) is a subset of $\biguplus_{j\in[\gamma^*]} E(G[S_j]) \uplus \biguplus_{i\in[2k^*]} E(T_i)$.

2.3.3 Routing Demands in the Expander

Once we have embedded the expander H into the graph G, we will need to route demand pairs across X via vertex-disjoint paths. There are many algorithms for routing on expanders, e.g. [61, 18, 17, 52, 37], that give different types of guarantees. We use the following theorem, due to Rao and Zhou [72] (see also a proof in [31]).

Theorem 2.23 (Theorem 7.1 in [72]) *Let $G = (V, E)$ be any n-vertex d-regular α-expander. Assume further that n is even, and that the vertices of G are partitioned into $n/2$ disjoint demand pairs $\{(s_1, t_1), \ldots, (s_{n/2}, t_{n/2})\}$. Then there is an efficient algorithm that routes $\Omega\left(\frac{\alpha n}{\log n \cdot d^2}\right)$ of the demand pairs on vertex-disjoint paths in G.*

Notice that $\mathcal{M}^*$ is a perfect matching for V_H. Thus, by applying Theorem 2.23, we can find a routing of a subset $\mathcal{M}' \subseteq \mathcal{M}^*$ of $\Omega\left(\frac{\log k^* k^*}{\log k^* \gamma^{*2}}\right) = \Omega(k^*/\log^4 k^*)$ pairs in the expander H with congestion 1. We can convert the paths in H to paths in G in a natural way. Consider a path $P = (t_{i_0}, t_{i_1}, \cdots, t_{i_\ell})$ in H. Let $e_j = (t_{i_j}, t_{i_{j+1}})$ for every

$j \in \{0, 1, \cdots, \ell - 1\}$. To obtain Q, we replace each edge e_j by its embedding-path Q_{e_j} in G. Recall that Q_{e_j} connects T_{i_j} to $T_{i_{j+1}}$. (Recall that T_{i_j} is the unique tree in $\mathbf{T}^*$ containing t_{i_j}.) Consider an intermediate vertex t_{i_j} in P. We connect the last vertex of $Q_{e_{j-1}}$ to the first vertex of Q_{e_j} using the unique path in T_{i_j}. We also connect t_{i_0} to the first vertex of Q_{e_0} using the unique path in T_{i_0}, and connect the last vertex of $P_{e_{\ell-1}}$ to t_{i_ℓ} using the unique path in Q_{i_ℓ}. This finishes the construction of Q. We remark that the path Q only includes the embedding-path Q_{e_j} of each edge $e_j \in P$, and some portion of each tree T_{i_j}, $j \in \{0, 1, \cdots, \ell\}$.

Now, due to the fact that the routing paths for $\mathcal{M}'$ in H are vertex-disjoint, every embedding-path $Q_e, e \in E_H$ is used at most once, and every tree $T_i, i \in [2k^*]$ is also used at most once. Since the congestion of the crossbar is c, the set of routing paths for $\mathcal{M}'$ in G causes congestion at most c.

Chapter 3

Edge Disjoint Paths: Main Algorithm

3.1 Overview of the Algorithm

In this chapter, we give our algorithm for EDP with congestion 2, by assembling the components described in Chapter 2. We leave the construction of the congestion-2 crossbar, our core component, to Chapter 4.

Our algorithm is based on the natural LP relaxation of the problem. At the very first step, we solve the LP and decompose the input EDP instance into what we called well-linked instances. This is the only step where we use the LP relaxation. After this step, we solve each well-linked instance separately, using a purely combinatorial approach. In a well-linked instance, the graph G is 1-well-linked for the set of $2k$ terminals participating in the k pairs. We shall route $k/\mathrm{polylog}(k)$ pairs in G with congestion 2. This is a clearly a $\mathrm{polylog}(k)$-approximation since the optimum solution can route at most k pairs.

The high-level structure of the combinatorial algorithm for a well-linked instance $(G, \mathcal{T}, \mathcal{M})$ is as follows. Throughout this and the next Chapter, let $\gamma = \Theta(\log^8 k), \alpha = \Theta(1/\log k)$ and $k_1 = \Theta\left(\frac{k}{\gamma^2 \log k}\right)$ be the threshold defining small and large clusters, as in Table 4.1. Suppose we are given γ disjoint large clusters $S_1, S_2, \cdots, S_\gamma$ in $G \setminus \mathcal{T}$ satisfying the following two properties. First, every cluster S_i can send $k_1/2$ units flow to the terminals in $\mathcal{T}$, or equivalently, there is no cut of size less than $k_1/2$ separating S_i and $\mathcal{T}$ for every $i \in [\gamma]$. Secondly, each cluster S_i is (k_1, α)-well-linked (for $\mathrm{out}(S_i)$). Then, we can either route $k/\mathrm{polylog}(k)$ pairs in G integrally with congestion 2, or construct a (γ^*, k^*)-crossbar of congestion 2 in G, for some large enough $\gamma^* = O(\log^2 k)$ and $k^* = k/\mathrm{polylog}(k)$. Then, by Theorem 2.21, we can route $\Omega\left(k^*/\log^4 k^*\right) = k/\mathrm{polylog}(k)$ pairs in G with congestion 2.

Thus, it suffices to construct a family of γ clusters satisfying the two properties. Our algorithm for constructing the clusters is iterative. It maintains a data structure, from which we can derive the family of γ clusters. We then check whether the two properties are satisfied. If either property is not satisfied, given the witness, we update the data structure so that it produces a different family of γ clusters. There

is a potential function defined over all possible contents of the data structure so that each time we update the data structure, its potential goes down by at least 1. The range of the potential function is polynomial in n and k. Thus we can find the family of γ clusters satisfying the two properties in polynomial time.

There is a small caveat with the above high-level approach: we can not check if a cluster S_i is (k_1, α)-well-linked, even if we are willing to lose a factor of polylog(k) in the α-factor. Indeed, we need to lose a polylog(n) factor. Fortunately we can avoid the polylog(n) loss factor by carefully merging the two steps. Given a family of γ disjoint large clusters, we try to construct the crossbar, pretending that the clusters satisfy the two properties. Then, the procedure either succeeds, or outputs a witness that one of the two properties is not satisfied. In the former case, we terminate the algorithm; while in the latter case, we return the witness to the iterative algorithm and ask the algorithm to give a different family of clusters. Thus, a crucial property of our crossbar constructor we used is the following. It not only succeeds given a family of "good" clusters, but also detects bad clusters when it fails.

The remaining part of this section is organized as follows. In Section 3.1.1, we show how to reduce a general EDP instance to well-linked instances. Then in Section 3.1.2, we give the theorem for constructing the crossbar. Since it is the core of the algorithm, we dedicate Chapter 4 to its proof. Finally, we describe the iterative algorithm in Subsection 3.1.3 and leave the proof of an important theorem to Section 3.2.

3.1.1 Preprocessing

Suppose we are given an EDP instance $(G = (V, E), \mathcal{T}, \mathcal{M})$. W.l.o.g, we can assume every terminal in $\mathcal{T}$ has degree 1 in graph G and participates in exactly 1 pair in $\mathcal{M}$. (If a vertex appears in a pairs, we can add a degree-1 terminals attached to this vertex.) Thus, $|\mathcal{T}| = 2|\mathcal{M}|$. We can assume each vertex in G has degree at most 4 via the following transformations. For each vertex $v \in E$ with degree $d > 4$, we replace v with a $d \times d$ grid and connect the d vertices adjacent to v to the d vertices of the first row of the grid. It is easy to verify that these transformations do not change the problem.

We use the standard multicommodity flow LP-relaxation for the EDPwC problem to partition our graph into several disjoint sub-graphs, that are well-linked for their respective sets of terminals. In the standard LP-relaxation for EDPwC, we have an indicator variable x_i for each $1 \leq i \leq k$, for whether the pair (s_i, t_i) is routed. Let $\mathcal{P}_i$ be the set of all paths connecting s_i to t_i in G. For each path P, we have a variable $f(P)$ indicating whether we are using P or not. The LP relaxation is defined as follows.

$$\text{(LP1)} \quad \max \quad \sum_{i=1}^{k} x_i$$

$$\text{s.t.} \sum_{P \in \mathcal{P}_i} f(P) \geq x_i \qquad \forall 1 \leq i \leq k \tag{3.1}$$

$$\sum_{P: e \in P} f(P) \leq 1 \qquad \forall e \in E \tag{3.2}$$

$$0 \leq x_i \leq 1 \qquad \forall 1 \leq i \leq k \tag{3.3}$$

$$f(P) \geq 0 \qquad \forall 1 \leq i \leq k, \forall P \in \mathcal{P}_i \tag{3.4}$$

While this LP has exponentially many variables, it can be efficiently solved using standard techniques, e.g. by using an equivalent polynomial-size LP formulation. We denote by opt the value of the optimal solution to the LP. Clearly, the value of the optimal solution to the EDPwC instance is at most opt.

Notice that if we replace the right-hand-side of Constraint (3.2) with 2, then we obtain an LP relaxation for the EDP with congestion 2 problem. Then, it is clear that the optimum of this new LP is at most $2\mathsf{opt}$. Due to this fact, our algorithm is indeed a true approximation for the EDP with congestion 2 problem.

We now partition the given EDP instance into many instances, using the following theorem whose proof is implicit in [24, 25, 31]. We include the proof for completeness.

Theorem 3.1 *Suppose we are given a graph $G = (V, E)$ and a set $\mathcal{M}$ of k source-sink pairs in G. Then we can either route $\Omega(\mathsf{opt}/\log^{1.5} k)$ source-sink pairs integrally in G, or do the following. Find a collection $G_1, \ldots, G_\ell$ of vertex-disjoint induced sub-graphs, and compute, for each $1 \leq i \leq \ell$, a collection $\mathcal{M}_i \subseteq \mathcal{M}$ of source-sink pairs contained in G_i, such that $\sum_{i=1}^{\ell} |\mathcal{M}_i| = \Omega(\mathsf{opt}/\log^{1.5} k)$, and G_i is 1-well-linked for $\mathcal{T}_i := \mathcal{T}(\mathcal{M}_i)$.*

Proof: We solve (LP1) to obtain opt and the $\{x_i : i \in [k]\}$ values. As a first step, we would like to define a weighted version of well-linkedness and apply a weighted version of the well-linked decomposition. To avoid extra definitions and proofs, we insist on the non-weighted versions of well-linkedness and well-linked decomposition given in Definition 2.2 and Theorem 2.7 via the following trick. First, we round down each x_i to the largest multiple of $1/k$. Then, the sum of x_i is decreased by at most 1. We modify the flows $\{f(P) : P \in \bigcup_i \mathcal{P}_i\}$ accordingly so that each s_i sends x_i units flow to t_i. Then, consider the graph G' obtained from G as follows. For each $(s_i, t_i) \in \mathcal{M}$ and $t \in \{s_i, t_i\}$, we add kx_i terminals connected to t via kx_i edges. We call the kx_i terminals the copies of t. Let $\mathcal{T}'$ be the set of newly added terminals. Thus, $|\mathcal{T}'| = 2k \sum_{i \in [k]} x_i \leq 2k^2$. We also match the kx_i terminals connected to s_i and kx_i terminals connected to t_i in an arbitrary way. By doing so, we obtain a perfect matching $\mathcal{M}'$ over $\mathcal{T}'$. We can route $1/k$ units flow in G' simultaneously for all demand pairs in $\mathcal{M}'$.

We apply Corollary 2.8 to G' and $\mathcal{T}'$ of terminals and some suitable $\alpha' = \Theta(1/(k \log k))$ to obtain a partition of G into many induced sub-graphs. Let

$K = |\mathcal{T}'|$. Then, each sub-graph $G'[S]$ is $\alpha'_{\text{WL}} = \alpha'\alpha_{\text{ARV}}(K) = \Omega(1/(k\log^{1.5} k))$-well-linked for the set $S \cap \mathcal{T}'$ of terminals. By Corollary 2.8, there are at most $O(\alpha' K \log K) = O(\alpha' K \log k)$ edges across different sub-graphs. If α' is small enough, the number is at most $K/(4k)$. We remove the demand pairs in $\mathcal{M}'$ whose two end-points lie in different sub-graphs. We also remove the terminals participating in these pairs from $\mathcal{T}'$. Since we can simultaneously send $1/k$ units flow for all pairs in $\mathcal{M}'$, and the number of crossing edges is at most $K/(4k)$, we removed at most $K/(4k)/(1/k) = K/4$ pairs from $\mathcal{M}'$. Thus, the number of remaining pairs is at least $K/2 - K/4 = K/4$. Let $\mathcal{T}''$ be the set of remaining terminals and $\mathcal{M}''$ be the set of remaining pairs.

Now focus on some induced sub-graph $G'[S]$ in the partition. We shall select a large subset of terminals in $\mathcal{T}'' \cap S$ for which $G'[S]$ is 1-well-linked, using the technique of boosting the well-linkedness in Section 2.1.5. We apply Theorem 2.10 to $G'[S]$ to obtain an $O(1/\alpha'_{\text{WL}})$-degenerate graph Z on the set $\mathcal{T}'' \cap S$ of terminals. If both terminals in some pair in $\mathcal{M}''$ are in $\mathcal{T}'' \cap S$, we identify them in Z. The degeneracy of the graph is increased by a factor of at most 4. Thus, the resulting graph Z' is still an $O(1/\alpha'_{\text{WL}})$-degenerate graph. We then can select an independent set of size $\Omega(\alpha'_{\text{WL}}|\mathcal{T}'' \cap S|)$ in Z'. We convert the independent set in Z' back to a set $\widetilde{\mathcal{T}}$ of terminals in Z: for a vertex in the independent set, we include into $\widetilde{\mathcal{T}}$ the pair of terminals identified to the vertex. Thus, the induced graph $Z[\widetilde{\mathcal{T}}]$ forms a partial matching. If the partial matching covers $1/2$ fraction of the vertices in $\widetilde{\mathcal{T}}$, then, by Property (A1), we can route $\Omega(\alpha'_{\text{WL}}|\mathcal{T}'' \cap S|)$ pairs integrally in $G'[S]$. Otherwise, we find a independent set $\widetilde{\mathcal{T}}'$ of size $\Omega(\alpha'_{\text{WL}}|\mathcal{T}'' \cap S|)$ in Z. Moreover, for a pair $(s,t) \in \mathcal{M}''$, either both s and t are $\widetilde{\mathcal{T}}'$, or both are not. By Theorem 2.10, we can obtain an 4-degenerate graph Y over $\widetilde{\mathcal{T}}''$. Again, we identify the pairs of terminals in Y to obtain a 16-degenerate graph Y'. We either find an integral routing of $\Omega(\alpha'_{\text{WL}}|\mathcal{T}'' \cap S|)$ pairs in $G'[S]$, or an independent set of size $\Omega(\alpha'_{\text{WL}}|\mathcal{T}'' \cap S|)$ in Y. In the latter case, the independent set is perfectly matched according to $\mathcal{M}''$, and $G'[S]$ is 1-well-linked for the independent set by Theorem 2.10.

Thus, for each induced sub-graph $G'[S]$ in the partition, we either find an integral routing of $\Omega(\alpha'_{\text{WL}}|\mathcal{T}'' \cap S|)$ pairs or a set of terminals for which $G'[S]$ is 1-well-linked. Recall that the sum of $|\mathcal{T}'' \cap S|$ over all sub-graphs $G'[S]$ is at least $K/2$. Thus, we either find an integral routing of $\Omega(\alpha'_{\text{WL}}K) = \Omega(\mathsf{opt}/\log^{1.5} k)$ pairs, or do the following. Let $G'_1, \ldots, G'_\ell$ be the induced sub-graphs in which we did not find integral routings. For each $1 \le i \le \ell$, let $\mathcal{T}'_i$ be the set of terminals for which G'_i is 1-well-linked. Since $\mathcal{T}'_i$ is perfectly matched, there is a $\mathcal{M}'_i \subseteq \mathcal{M}''$ such that $\mathcal{T}'(\mathcal{M}'_i) = \mathcal{T}'_i$. $\sum_{i=1}^{\ell} |\mathcal{M}'_i| = \Omega(\alpha'_{\text{WL}}K) = \Omega(\mathsf{opt}/\log^{1.5} k)$. The routings in G' naturally gives the routings in G. Also, the sub-graphs G'_i and the sets $\mathcal{T}'_i$ can be easily converted to sub-graphs of G and subsets of the original terminals $\mathcal{T}$. Recall that each original terminal has degree 1 in G, if G'_i is 1-well-linked for $\mathcal{T}'_i$, then only one copy of each original terminal can be in $\mathcal{T}'_i$. This finishes the proof. □

Thus, by losing a $\log^{1.5} k$ factor, it suffices to assume our EDP instance $(G, \mathcal{T}, \mathcal{M})$ has the following properties. We call such an instance satisfying the properties a *well-linked* instance.

(B1) Every non-terminal of G has degree at most 4;

(B2) Every terminal of G has degree exactly 1;

(B3) Every terminal participate in exactly one pair; that is, $|\mathcal{T}| = 2k$;

(B4) G is 1-well-linked for $\mathcal{T}$.

3.1.2 Constructing a Congestion-2 Crossbar

We now state the theorem for constructing the crossbar. Recall that we are given a well-linked EDP instance $(G, \mathcal{T}, \mathcal{M})$. As in Table 4.1, $\alpha = \Theta(1/\log k)$, $\gamma = \Theta(\log^8 k)$,and $k_1 = \Theta\left(\frac{k}{\gamma^2 \log k}\right)$ is the threshold used for defining small and big clusters. Let $\alpha_{\text{WL}} = \alpha/\alpha_{\text{ARV}}(k)$. We let $k^* = \frac{\alpha_{\text{WL}}^2 k}{\gamma^6}$ and $\gamma^* = \Theta(\gamma^{1/4}) = \Theta(\log^2 k)$ be some suitable parameters as in Table 4.1. The proof of the following core theorem is deferred to Chapter 4.

Theorem 3.2 (constructing crossbar) *Given a family of disjoint large clusters $\mathcal{S} = \{S_1, S_2, \cdots, S_\gamma\}$ in $G \setminus \mathcal{T}$, we can efficiently compute one of the following:*

(C1) a (k_1, α)-violating cut (X, Y) of S_j, for some $1 \leq j \leq \gamma$,

(C2) a cut of size less than $k_1/2$ separating some S_j and $\mathcal{T}$, for some $1 \leq j \leq \gamma$,

(C3) an integral routing of $\Omega(k^*)$ pairs in $\mathcal{M}$ with congestion at most 2 in G,

(C4) a (γ^*, k^*) crossbar of congestion 2 in G.

3.1.3 The Iterative Algorithm

Notice that Theorem 3.2 does not always output a routing or a crossbar; it may output a violating cut or a small cut. In this case, we produce an "improved" family $\mathcal{S}'$ of large clusters and apply Theorem 3.2 again on $\mathcal{S}'$. If again the algorithm fails to produce a routing or a crossbar, we produce a further improved set of large clusters. The algorithm is guaranteed to succeed in polynomial number of iterations. To give more details about the iterative algorithm, we need the following two definitions. Recall that a clustering is a partition of G into (connected) clusters.

Definition 3.3 *A clustering $\mathcal{C}$ of G is an* acceptable clustering *iff: (i) every terminal $t \in \mathcal{T}$ is in a separate cluster, that is, $\{t\} \in \mathcal{C}$; (ii) each small cluster $C \in \mathcal{C}$ is α_{WL}-well-linked. An acceptable clustering that contains no large clusters is called a* good clustering.

Given a good clustering $\mathcal{C}$ of G, we use $H_{\mathcal{C}}$ to denote the graph obtained from G by contracting every cluster $C \in \mathcal{C}$ into a super-node v_C. We remove all self-loops, but we do keep parallel edges.

Note that the terminals are not contracted in $H_{\mathcal{C}}$, since each terminal has its own cluster. To simplify the notations, we assume the node $v_{\{t\}}$ is identical to t for any terminal t. Then $\mathcal{T} \subseteq V_{H_{\mathcal{C}}}$ for any good clustering $\mathcal{C}$.

Suppose we are given a potential function ϕ that maps acceptable clusterings to real numbers. We define *a valid clustering family* w.r.t ϕ as follows.

Definition 3.4 *Given a potential function ϕ for acceptable clusterings, we say $\mathcal{F} = (\mathcal{C}, \{X_j : j \in [\gamma]\}, \{\mathcal{C}_j : j \in [\gamma]\})$ is* a valid clustering family *w.r.t ϕ if*

(D1) $\mathcal{C}$ is a good clustering;

(D2) $\{X_j : j \in [\gamma]\}$ forms a partition of $V_{H_{\mathcal{C}}} \setminus \mathcal{T}$;

(D3) each $\mathcal{C}_j$ is an acceptable clustering with $\phi(\mathcal{C}_j) \leq \phi(\mathcal{C}) - 1$ and at least one large cluster;

(D4) if $S \in \mathcal{C}_j$ is a large cluster, then $S \subseteq \bigcup_{v_C \in X_j} C$.

The main theorem we need for the iterative algorithm is the following.

Theorem 3.5 *There is a potential function ϕ such that $\phi(\mathcal{C}) \in [0, 2|E|]$ for every acceptable clustering $\mathcal{C}$, and the following is true. We can efficiently construct a valid clustering family $\mathcal{F} = (\mathcal{C}, \{X_j : j \in [\gamma]\}, \{\mathcal{C}_j : j \in [\gamma]\})$ w.r.t ϕ. Moreover, given such a family $\mathcal{F}$, a large cluster $S \in \mathcal{C}_j$ for some $j \in [\gamma]$, and one of the following:*

(E1) a (k_1, α)-violating cut of S,

(E2) a cut of size less than $k_1/2$ separating S and $\mathcal{T}$,

we can efficiently find another valid clustering family $\mathcal{F}' = \left(\mathcal{C}', \left\{X'_j : j \in [\gamma]\right\}, \left\{\mathcal{C}'_j : j \in [\gamma]\right\}\right)$ w.r.t ϕ such that one of the following is true:

$$\phi(\mathcal{C}') \leq \phi(\mathcal{C}) - 1, \tag{3.5}$$

$$\phi(\mathcal{C}') = \phi(\mathcal{C}) \quad \text{and} \quad \sum_{j \in [\gamma]} \phi(\mathcal{C}'_j) \leq \sum_{j \in [\gamma]} \phi(\mathcal{C}_j) - 1. \tag{3.6}$$

We defer the proof of Theorem 3.5 to Section 3.2. Now, we proceed to describe the iterative algorithm and finish the proof of the Theorem 2.1. We first apply Theorem 3.5 to obtain a valid clustering family $\mathcal{F} = (\mathcal{C}, \{X_j : j \in [\gamma]\}, \{\mathcal{C}_j : j \in [\gamma]\})$ w.r.t ϕ. In each iteration, we do the following. By Property (D3) and (D4), each $\mathcal{C}_j$ contains a large cluster $S_j \subseteq \bigcup_{v_C \in X_j} C$. By Property (D2), the sets $\{X_j : j \in [\gamma]\}$ are disjoint. Thus we have that the set $\mathcal{S} = \{S_1, S_2, \cdots, S_\gamma\}$ of clusters are disjoint. Moreover, they do not contain terminals. We then apply Theorem 3.2 on $\mathcal{S}$. If the algorithm outputs a routing of $\Omega(k^*)$ pairs, then we terminate the algorithm. If the algorithm outputs a (γ^*, k^*)-crossbar of congestion 2, we apply Theorem 2.21 to obtain a routing of $\Omega(k^*/\log^4 k^*)$ pairs with congestion 2 in G, and terminate the algorithm. Otherwise, the algorithm outputs either a (k_1, α)-violating cut of some $S_j \in \mathcal{S}$, or a

cut of size less than $k_1/2$ separating some $S_j \in \mathcal{S}$ and $\mathcal{T}$. In either case, we apply Theorem 3.5 to obtain a valid clustering family $\mathcal{F}' = \left(\mathcal{C}', \left\{X'_j : j \in [\gamma], \{\mathcal{C}_j : j \in [\gamma]\}\right\}\right)$ satisfying either (3.5) or (3.6). We then let $\mathcal{F} = \mathcal{F}'$ and start a new iteration. Since the function ϕ has range $[0, 2|E|]$, in at most $O(|E|) \times O(\gamma|E|) = O(\gamma|E|^2)$ iterations, the algorithm will terminate by returning a routing. The number of pairs routed is always $\Omega(k^*/\log^4 k^*) = \Omega\left(\frac{\alpha_{\text{WL}}^2 k}{\gamma^8 \log k}/\log^4 k^*\right) = \Omega\left(\frac{k}{\log^{72} k}\right)$. Taking the $O(\log^{1.5} k)$-factor lost in Theorem 3.1 into consideration, we obtained an $O(\log^{73.5} k)$-approximation for EDP with congestion 2.

3.2 The Clustering Algorithm

This section is dedicated to the proof of Theorem 3.5.

3.2.1 Defining the Potential Function ϕ

In order to prove Theorem 3.5, we first need to define the potential function ϕ. For convenience, ϕ is defined over all clusterings, not only over acceptable clusterings. On one hand, $\phi(\mathcal{C})$ approximates the number of edges cut by $\mathcal{C}$; on the other hand, edges connecting smaller clusters contribute slightly less than edges connecting larger clusters in the potential function. This will ensure that well-linked decomposition can only decrease the potential function, though it may increase the number of cut edges.

We first define a potential $\phi(h)$ for any integer $h \geq 0$. For $h < k_1$, let $\phi(h) = 4\alpha \log h$. For $h \geq k_1$, let i be the integer such that $(3/2)^{i-1}k_1 \leq h < (3/2)^i k_1$. Then, $\phi(h) = 4\alpha \log k_1 + 4\alpha \sum_{j=0}^{i-1}(2/3)^j$. Notice that ϕ is a non-decreasing function of h. For all h, $\phi(h) \leq 4\alpha \log k_1 + 12\alpha \leq 8\alpha \log k_1 \leq 1/2$.

Consider *any* clustering $\mathcal{C}$ of G and an edge $e \in E$. If both endpoints of e belong to the same cluster of $\mathcal{C}$, then we set $\phi(e) = 0$. Otherwise, if $e = (u, v)$, and $u \in C \in \mathcal{C}$ with $|\text{out}(C)| = h$, while $v \in C' \in \mathcal{C}$ with $|\text{out}(C')| = h'$, then we set $\phi(e) = 1 + \phi(h) + \phi(h')$. We think of the one as the contribution of e, $\phi(h)$ as the contribution of u, and $\phi(h')$ as the contribution of v to $\phi(e)$. The final potential of $\mathcal{C}$ is just $\phi(\mathcal{C}) = \sum_{e \in E} \phi(e)$. Notice that $\phi(e) \leq 2$ for every $e \in E$ and thus $\phi(\mathcal{C}) \in \left[0, 2\,|E|\,\right]$ for any clustering $\mathcal{C}$ as promised.

3.2.2 Operations on Clusterings

In this section, we define some operations on clusterings. All these operations can only decrease the potential function.

Well-linked decomposition We show that the well-linked decomposition of a small cluster will not increase the potential of the clustering:

Lemma 3.6 *Let $\mathcal{C}$ be any clustering of V, and $C \in \mathcal{C}$ be a small cluster. Let $\mathcal{C}'$ be the clustering obtained from $\mathcal{C}$ by replacing C with the clusters obtained from α-well-linked decomposing C (see Section 2.1.4). Then $\phi(\mathcal{C}') \leq \phi(\mathcal{C})$.*

Proof: In each iteration of the well-linked-decomposition, we break one cluster in C into two clusters X and Y. It suffices to show that this single operation does not increase the potential. Thus, we can assume $\mathcal{C}'$ is the partition obtained from $\mathcal{C}$ by replacing $C \in \mathcal{C}$ with two clusters X and Y such that $|E(X,Y)| \leq \alpha \min\{|\mathrm{out}(X) \cap \mathrm{out}(C)|, |\mathrm{out}(Y) \cap \mathrm{out}(C)|\}$.

Assume w.l.o.g. that $|\mathrm{out}(X)| \leq |\mathrm{out}(Y)|$, so $|\mathrm{out}(X)| \leq 2|\mathrm{out}(C)|/3$ (since $\alpha < 1/3$). Let $h = |\mathrm{out}(C)|, h_1 = |\mathrm{out}(X)|, h_2 = |\mathrm{out}(Y)|$, and recall that $h, h_1, h_2 < k_1$. The changes to the potential are the following:

- The potential of the edges in $\mathrm{out}(Y) \cap \mathrm{out}(C)$ only goes down.
- The potential of every edge in $\mathrm{out}(X) \cap \mathrm{out}(C)$ goes down by $\phi(h) - \phi(h_1) = 4\alpha \log h - 4\alpha \log h_1 = 4\alpha \log \frac{h}{h_1} \geq 4\alpha \log 1.5 \geq 2.3\alpha$, since $h_1 \leq 2h/3$. So the total decrease in the potential of the edges in $\mathrm{out}(X) \cap \mathrm{out}(C)$ is at least $2.3\alpha \cdot |\mathrm{out}(X) \cap \mathrm{out}(C)|$.
- The edges in $E(X,Y)$ did not contribute to the potential initially, and now contribute at most $1 + \phi(h_1) + \phi(h_2) \leq 2$ each. Notice that $|E(X,Y)| \leq \alpha \cdot |\mathrm{out}(X) \cap \mathrm{out}(S)|$, and so they contribute at most $2\alpha \cdot |\mathrm{out}(X) \cap \mathrm{out}(S)|$ in total.

Thus, the overall potential can only go down. □

Partitioning a large cluster Suppose we are given a clustering $\mathcal{C}$ of G, a large cluster $C \in \mathcal{C}$, and a (k_1, α)-violating cut (X, Y) of C. The operation $\mathsf{PARTITION}(\mathcal{C}, X, Y)$ replaces $C \in \mathcal{C}$ with connected components of $G[X]$ and $G[Y]$. We prove the following lemma.

Lemma 3.7 *Let $\mathcal{C}'$ be the outcome of* $\mathsf{PARTITION}(\mathcal{C}, X, Y)$. *Then $\phi(\mathcal{C}') \leq \phi(\mathcal{C}) - 1$.*

Proof: Assume w.l.o.g. that $|\mathrm{out}(X)| \leq |\mathrm{out}(Y)|$. Let $h = |\mathrm{out}(C)|$, $h_1 = |\mathrm{out}(X)|$, $h_2 = |\mathrm{out}(Y)|$, so $h_1 \leq 2h/3$. The changes in the potential can be bounded as follows:

- The potential of every edge in $\mathrm{out}(Y) \cap \mathrm{out}(C)$ does not increase.
- The potential of every edge in $\mathrm{out}(X) \cap \mathrm{out}(C)$ goes down by at least $\phi(h) - \phi(h_1)$.
- Every edge in $E(X,Y)$ now contributes $1 + \phi(h_1) + \phi(h_2) < 2$.

If $h_1 < k_1$, then $\phi(h) - \phi(h_1) \geq 4\alpha$ (notice that $h \geq k_1$). Since $|E(X,Y)| \leq \alpha |\mathrm{out}(X) \cap \mathrm{out}(C)|$, the total decrease in the potential function is at least $|E(X,Y)|((\phi(h) - \phi(h_1))/\alpha - 2) \geq |E(X,Y)| \geq 1$(notice that C is connected).

If $h_1 \geq k_1$, then $(3/2)^{i-1}k_1 \leq h_1 < (3/2)^i k_1$ for some $i \geq 1$. Since $h > (3/2)h_1 \geq (3/2)^i k_1$, we have $\phi(h) - \phi(h_1) \geq 4\alpha(2/3)^i$. Notice that $|E(X,Y)| \leq \alpha k_1/2$ in this case, by the definition of (k_1, α)-violating cut. The total decrease in the potential is at least $4\alpha(2/3)^i \times (3/2)^{i-1}k_1 - 2\alpha k_1/2 \geq \alpha k_1 \geq 1$. □

Separating a large cluster Let $\mathcal{C}$ be a clustering of G, and $C \in \mathcal{C}$ be a large cluster in $\mathcal{C}$. Assume further that there is a cut $(A, B = V \setminus A)$ in graph G, with $C \subseteq A$, $\mathcal{T} \subseteq B$, and $|\text{out}(A)| < k_1/2$. We perform the following operation, that we denote by $\mathsf{SEPARATE}(\mathcal{C}, C, A)$.

If for some $C' \in \mathcal{C} \setminus \{C\}$, we have $|\text{out}(C' \setminus A)| > |\text{out}(C')|$, we change A to $A \setminus C'$. Since $|\text{out}(A \setminus C')| + |\text{out}(C' \setminus A)| \leq |\text{out}(A)| + |\text{out}(C')|$, we have $|\text{out}(A \setminus C')| < |\text{out}(A)|$. Thus, $|\text{out}(A)|$ can only decrease and A still satisfies the above properties. Thus, we can assume $|\text{out}(C' \setminus A)| \leq |\text{out}(C')|$ for every $C' \in \mathcal{C}$.

The outcome clustering $\mathcal{C}'$ of $\mathsf{SEPARATE}(\mathcal{C}, C, A)$ is as follows. First, we add every connected component of $G[A]$ to $\mathcal{C}'$. Notice that all these clusters are small, as $|\text{out}(A)| < k_1/2$. Next, for every cluster $C' \in \mathcal{C} \setminus \{C\}$, we add every connected component of $G[C' \setminus A]$ to $\mathcal{C}'$. Since we guaranteed that $|\text{out}(C' \setminus A)| \leq |\text{out}(C')|$ for every $C' \in \mathcal{C}$, the following claim is easy to see:

Claim 3.8 $\bigcup_{S \in \mathcal{C}' : S \text{ is large}} S \subseteq \bigcup_{S \in \mathcal{C} : S \text{ is large}} S$.

We now show that $\mathsf{SEPARATE}$ operation can only decrease the potential:

Lemma 3.9 *Let $\mathcal{C}'$ be the outcome of operation $\mathsf{SEPARATE}(C, A)$. Then $\phi(\mathcal{C}') \leq \phi(\mathcal{C}) - 1$.*

Proof: We can bound the changes in the potential as follows:

- Every edge in $\text{out}(A)$ contributes at most 2 to the potential of $\mathcal{C}'$, and there are at most $\frac{k_1 - 1}{2}$ such edges. These are the only edges whose potential in $\mathcal{C}'$ may be higher than their potential in $\mathcal{C}$.
- Every edge $e \in \text{out}(C)$ contributed at least 1 to the potential of $\mathcal{C}$, and there are at least k_1 such edges, since C is a large cluster. In $\mathcal{C}'$, e contributes either 0, or at most 2, in which case e must be in $\text{out}(C) \cap \text{out}(A)$ and is counted previously.

Therefore, the decrease in the potential is at least $k_1 - \frac{2(k_1 - 1)}{2} = 1$. □

3.2.3 Producing a Valid Clustering Family

We can now proceed to the proof of Theorem 3.5. We start with the first part of the theorem, that is, producing a valid clustering family. The algorithm runs in iterations. In each iteration, it maintains a good clustering $\mathcal{C}$. The initial clustering $\mathcal{C}$ contains all singular clusters, i.e, $\mathcal{C} = \{\{v\} : v \in G\}$. Let $H = H_{\mathcal{C}}$ be the contracted graph correspondent to the current clustering $\mathcal{C}$. Let m be the number of edges in $H \setminus \mathcal{T}$. Recall that terminals in $\mathcal{T}$ are not contracted and thus $\mathcal{T} \subseteq H$.

The proof of the following claim is deferred to Section 3.2.5.

Claim 3.10 $m \geq k/3$.

We randomly partition the vertices in $H \setminus \mathcal{T}$ into γ subsets $X_1, \ldots, X_\gamma$, where each vertex $v \in V(H) \setminus \mathcal{T}$ selects an index $1 \leq j \leq \gamma$ independently uniformly at random, and is then added to X_j. We prove the following lemma in Section 3.2.5.

Lemma 3.11 *With probability at least $\frac{1}{2}$, for each $1 \leq j \leq \gamma$, $|\mathrm{out}_H(X_j)| < \frac{10m}{\gamma}$ and $|E_H(X_j)| \geq \frac{m}{2\gamma^2}$.*

We repeat the random procedure if the sets $X_1, \cdots, X_\gamma$ do not satisfy the condition of lemma 3.11. Assume now they satisfy the condition. Then for each $1 \leq j \leq \gamma$, $|E_H(X_j)| > \frac{|\mathrm{out}_H(X_j)|}{20\gamma}$. Let $X'_j \subseteq V(G) \setminus \mathcal{T}$ be the set obtained from X_j, after we un-contract clusters of C for which $v_C \in X_j$, that is, $X'_j = \bigcup_{C: v_C \in X_j} C$. Notice that $X'_1, X'_2, \cdots, X'_\gamma$ form a partition of $V \setminus \mathcal{T}$.

For each $1 \leq j \leq \gamma$, define $\mathcal{C}_j$ be the following clustering. First, we add all clusters C such that $v_C \notin X_j$ into $\mathcal{C}_j$. Then, we add all connected components of $G[X'_j]$ into $\mathcal{C}_j$. If any of these connected components is a small cluster, we apply α-well-linked decomposition to the cluster. $\mathcal{C}_j$ is clearly an acceptable clustering. Also, if $S \in \mathcal{C}_j$ is a large cluster, then $S \subseteq X'_j$. We then prove

Lemma 3.12 *For any $j \in [\gamma]$, $\phi(\mathcal{C}_j) \leq \phi(\mathcal{C}) - 1$.*

Proof: Let $\mathcal{C}'_j$ be the partition of V, obtained as follows: we add to $\mathcal{C}'_j$ all clusters $C \in \mathcal{C}$ with $v_C \in V_H \setminus X_j$, and we add all connected components of $G[X'_j]$ to $\mathcal{C}'_j$. That is, $\mathcal{C}'_j$ is obtained like $\mathcal{C}_j$, except that we do not perform well-linked decompositions of the small clusters. From Lemma 3.6, it is enough to prove that $\phi(\mathcal{C}'_j) \leq \phi(\mathcal{C}) - 1$. The changes of the potential from $\mathcal{C}$ to $\mathcal{C}'_j$ can be bounded as follows:

- each edge in $E_H(X_j)$ contributes at least 1 to $\phi(\mathcal{C})$ and contributes 0 to $\phi(\mathcal{C}'_j)$.
- The potential of edges in $\mathrm{out}_H(X_j)$ may increase. The increment is at most $\phi(n) \leq \frac{1}{2^8\gamma}$ per edge. So the total increase is at most $\frac{|\mathrm{out}_H(X_j)|}{2^8\gamma} \leq \frac{|E_H(X_j)|}{4}$. These are the only edges whose potential may increase.

Overall, the decrease in the potential is at least $\frac{|E_H(X_j)|}{2} \geq \frac{m}{4\gamma^2} \geq \frac{k}{12\gamma^2} \geq 1$. □

If every $\mathcal{C}_j$ contains at least one large cluster, then $\mathcal{F} = \left(\mathcal{C}, \{X_j : j \in [\gamma]\}, \{\mathcal{C}_j : j \in [\gamma]\}\right)$ is a valid clustering family. Otherwise, some $\mathcal{C}_j$ contains no large clusters and is thus a good clustering. Then, we replace $\mathcal{C}$ with $\mathcal{C}_j$ and start a new iteration. By Lemma 3.12, we will obtain a valid clustering family in polynomial number of iterations.

3.2.4 Improving the Clustering Family

We now proceed to the second part of Theorem 3.5. That is, given a valid clustering family $\mathcal{F}$, a large cluster $S \in \mathcal{C}_j$ for some $j \in [\gamma]$, and either (E1) or (E2), we produce a new $\mathcal{F}'$ satisfying either (3.5) or (3.6). If we are given (E1), i.e, a (k_1, α)-violating cut (X, Y) of S, we apply the operation $\mathsf{PARTITION}(\mathcal{C}_j, X, Y)$ to obtain a new clustering. Applying α-well-linked decomposition to small clusters if necessary, we obtain an acceptable clustering $\mathcal{C}'$. By Lemma 3.6 and 3.7, we have $\phi(\mathcal{C}') \leq \phi(\mathcal{C}_j) - 1$. Also notice that any large cluster of $\mathcal{C}'$ is inside $\bigcup_{v_C \in X_j} C$. If $\mathcal{C}'$ contains at least one large cluster, then we can replace $\mathcal{C}_j$ with $\mathcal{C}'$ to obtain a new clustering

family $\mathcal{F}'$ satisfying (3.6). Otherwise, $\mathcal{C}'$ is a good clustering. We then can produce a new clustering family $\mathcal{F}' = \left(\mathcal{C}'', \left\{X'_j : j \in [\gamma]\right\}, \left\{\mathcal{C}'_j : j \in [\gamma]\right\}\right)$ using the algorithm in Section 3.2.3(the initial good clustering is $\mathcal{C}'$, instead of clustering of singular vertices). $\phi(\mathcal{C}'') \leq \phi(\mathcal{C}') \leq \phi(\mathcal{C}) - 1$ and thus $\mathcal{F}'$ satisfies the (3.5).

Suppose now we are given a cut $(A, B = V \setminus A)$ of size less than $k_1/2$ separating S and $\mathcal{T}$, i.e, $S \subseteq A, \mathcal{T} \subseteq B$ and $|E(A,B)| < k_1/2$. Applying the operation $\mathsf{SEPARATE}(\mathcal{C}_j, S, A)$ and α-well-linked decomposition to small clusters if necessary, we obtain a new acceptable clustering $\mathcal{C}'$. By Lemma 3.6 and 3.9, we have $\phi(\mathcal{C}') \leq \phi(\mathcal{C}_j) - 1$. By Claim 3.8, any large cluster in $\mathcal{C}_j$ is inside $\bigcup_{v_C \in X_j} C$. Using the same argument as above, we can obtain a new clustering family $\mathcal{F}'$ with the desired properties.

3.2.5 Omitted Proofs

Proof of Claim 3.10: For each terminal $t \in \mathcal{T}$, let e_t be the unique edge adjacent to t in H, and let u_t be the other endpoint of e_t. We partition the terminals in $\mathcal{T}$ into groups, where two terminals t, t' belong to the same group iff $u_t = u_{t'}$. Let $\mathcal{U}$ be the resulting partition of the terminals. Since the degree of every vertex in H is at most k_1, each group $U \in \mathcal{U}$ contains at most k_1 terminals. Next, we partition the terminals in $\mathcal{T}$ into two subsets X, Y, where $|X|, |Y| \geq k/3$, and for each group $U \in \mathcal{G}$, either $U \subseteq X$, or $U \subseteq Y$ holds. We can find such a partition by greedily processing each group $U \in \mathcal{U}$, and adding all terminals of U to one of the subsets X or Y, that currently contains fewer terminals. Finally, we remove terminals from X until $|X| = k/3$, and we do the same for Y. Since graph H is 1-well-linked for $\mathcal{T}$ (G is 1-well-linked for $\mathcal{T}$, and H is obtained from G by contractions), it is possible to route $k/3$ flow units from the terminals in X to the terminals in Y. Since no group U is split between the two sets X and Y, each flow-path must contain at least one edge of $H \setminus \mathcal{T}$. Therefore, the number of edges in $H \setminus \mathcal{T}$ is at least $k/3$. □

Proof of Lemma 3.11 : Let $H' = H \setminus \mathcal{T}$ and fix some $1 \leq j \leq \gamma$. Let $\mathcal{E}_1(j)$ be the bad event that $\sum_{v \in X_j} d_H(v) \geq \frac{2(2m+k)}{\gamma}$. In order to bound the probability of $\mathcal{E}_1(j)$, we define, for each vertex $v \in V_{H'}$, a random variable x_v, whose value is $\frac{d_H(v)}{k_1}$ if $v \in X_j$ and 0 otherwise. Notice that $x_v \in [0,1]$, and the random variables $\{x_v\}_{v \in V_{H'}}$ are pairwise independent. Let $B = \sum_{v \in V_{H'}} x_v$. Then the expectation of B is $\mu_1 = \sum_{v \in V_{H'}} \frac{d_H(v)}{\gamma k_1} = \frac{2m+k}{\gamma k_1}$. Using the standard Chernoff bound (see e.g. Theorem 1.1 in [34]),

$$\mathbf{Pr}\left[\mathcal{E}_1(j)\right] = \mathbf{Pr}\left[B > 2\mu_1\right] \leq e^{-\mu_1/3} = e^{-\frac{2m+k}{3\gamma k_1}} < \frac{1}{4\gamma},$$

since $k_1 = \Theta\left(\frac{k}{\gamma^2 \log k}\right)$.

Notice that if events $\mathcal{E}_1(j)$ do not happen, then:

$$|\operatorname{out}_H(X_j)| \leq \sum_{v \in X_j} d_H(v) \leq \frac{2(2m+k)}{\gamma} < \frac{10m}{\gamma},$$

since $m \geq k/3$.

Let $\mathcal{E}_2(j)$ be the bad event that $|E_H(X_j)| < \frac{m}{2\gamma^2}$. We next prove that $\mathbf{Pr}\left[\mathcal{E}_2(j)\right] \leq \frac{1}{4\gamma}$. We say that two edges $e, e' \in E_{H'}$ are *independent* iff they do not share any endpoints. Since the maximum vertex degree in H is at most k_1, each edge in $E_{H'}$ shares end points with at most $2k_1-2$ other edges in $E_{H'}$. Using the Hajnal-Szemerédi Theorem [42], we can compute a partition $U_1, \ldots, U_r$ of the set $E(H \setminus \mathcal{T})$ of edges, where $r \leq 2k_1$, such that for each $1 \leq i \leq r$, $|U_i| \geq \frac{m}{2k_1} - 1 \geq \frac{m}{4k_1}$, and all edges in set U_i are mutually independent. For each $1 \leq i \leq r$, we say that the bad event $\mathcal{E}_2^i(j)$ happens iff $|U_i \cap E(X_j)| < \frac{|U_i|}{2\gamma^2}$. Notice that if $\mathcal{E}_2(j)$ happens, then event $\mathcal{E}_2^i(j)$ must happen for some $1 \leq i \leq r$. Fix some $1 \leq i \leq r$. The expectation of $|U_i \cap E(X_j)|$ is $\mu_2 = \frac{|U_i|}{\gamma^2}$. Since all edges in U_i are independent, we can use the standard Chernoff bound to bound the probability of $\mathcal{E}_2^i(j)$, as follows:

$$\mathbf{Pr}\left[\mathcal{E}_2^i(j)\right] = \mathbf{Pr}\left[|U_i \cap E(X_j)| < \mu_2/2\right] \leq e^{-\mu_2/8} = e^{-\frac{|U_i|}{8\gamma^2}}.$$

Since $|U_i| \geq \frac{m}{4k_1}$, $m \geq k/3$ and $k_1 = \Theta\left(\frac{k}{\gamma^2 \log k}\right)$ is small enough, this is bounded by $\frac{1}{8k_1\gamma}$. We conclude that $\mathbf{Pr}\left[\mathcal{E}_2^i(j)\right] \leq \frac{1}{8k_1\gamma}$, and by using the union bound over all $1 \leq i \leq r$, $\mathbf{Pr}\left[\mathcal{E}_2(j)\right] \leq \frac{1}{4\gamma}$.

Using the union bound over all $1 \leq j \leq \gamma$, with probability at least $\frac{1}{2}$, none of the events $\mathcal{E}_1(j), \mathcal{E}_2(j)$ for $1 \leq j \leq \gamma$ happen, and so for each $1 \leq j \leq \gamma$, $\left|\operatorname{out}_H(X_j)\right| < \frac{10m}{\gamma}$, and $|E_H(X_j)| \geq \frac{m}{2\gamma^2}$ must hold. □

Chapter 4

Edge Disjoint Paths: Constructing Crossbar

4.1 Overview of the Construction

In this chapter, we prove Theorem 3.2, which is repeated below. Recall the settings of the theorem: $(G, \mathcal{T}, \mathcal{M})$ is a well-linked instance with $|\mathcal{M}| = k$. $\gamma = \Theta(\log^9 k), k_1 = \Theta\left(\frac{k}{\gamma^2 \log k}\right), \alpha = \Theta(1/\log k), \alpha_{\text{WL}} = \alpha/\alpha_{\text{ARV}}, \gamma^* = \Theta(\gamma^{1/4})$ and $k^* = \Theta\left(\frac{\alpha_{\text{WL}}^2 k}{\gamma^8 \log k}\right)$ are as in Table 4.1.

Theorem 3.2 (constructing crossbar) *Given a family of disjoint large clusters $\mathcal{S} = \{S_1, S_2, \cdots, S_\gamma\}$ in $G \setminus \mathcal{T}$, we can efficiently compute one of the following:*

- *(C1) a (k_1, α)-violating cut (X, Y) of S_j, for some $1 \leq j \leq \gamma$,*
- *(C2) a cut of size less than $k_1/2$ separating some S_j and $\mathcal{T}$, for some $1 \leq j \leq \gamma$,*
- *(C3) an integral routing of $\Omega(k^*)$ pairs in $\mathcal{M}$ with congestion at most 2 in G,*
- *(C4) a (γ^*, k^*) crossbar of congestion 2 in G.*

To this end, we say a path P avoids a cluster S, or P is S-avoiding, if the path P does not contain any vertex in S as *intermediate vertices*. Notice that the two end-points of P may be inside S. For a family $\mathcal{S}'$ of disjoint clusters, we say a path P avoids $\mathcal{S}'$, or P is $\mathcal{S}'$-avoiding if it avoids every cluster in $\mathcal{S}'$.

We can verify that for each $1 \leq j \leq \gamma$, the vertices of S_j can send $k_1/2$ flow units with no congestion to the terminals. If this is not the case for some set S_j, then there is a cut (A, B) with $S_j \subseteq A$, $\mathcal{T} \subseteq B$ and $|E_G(A, B)| < k_1/2$. We can return the cut (A, B) of G and finish the algorithm. From now on we assume that each cluster $S_j \in \mathcal{S}$ can send $k_1/2$ flow units to $\mathcal{T}$.

We can not verify whether there exists a (k_1, α)-violating cut of some S_j. However, given two subsets $\Gamma, \Gamma' \subseteq \text{out}(S_j)$ of edges, with $|\Gamma| = |\Gamma'| \leq k_1/2$, we can check whether there exists a flow $F : \Gamma \overset{1:1}{\rightsquigarrow}_{1/\alpha} \Gamma'$ inside S_j. If such a flow does not exist, then by Lemma 2.6, we can find a (k_1, α)-violating partition (X, Y) of S_j, return (X, Y) and

Table 4.1: The parameters used in the algorithm

parameters	meanings	values
k	number of pairs in $\mathcal{M}$	$\lvert\mathcal{M}\rvert$
γ	size of the family $\mathcal{S}$	$\Theta(\log^8 k)$
k_1	threshold for defining small and large clusters	$\Theta\left(\frac{k}{\gamma^2 \log k}\right)$
α	parameter for well-linkedness decomposition	$\Theta(1/\log k)$
α_{WL}	well-linkedness guaranteed	$\alpha/\alpha_{\text{ARV}}$
γ'	size of the family $\mathcal{S}'$ constructed in step 1	$\lfloor \gamma^{1/4}/2 \rfloor$
k_2	size of $\mathcal{P}_e, e \in E_T$ constructed in step 1	$\left\lfloor \frac{5k_1}{\gamma^4} \right\rfloor$
γ''	size of the family $\mathcal{S}''$ constructed in step 2	$\geq \gamma'/4$
k_3	size of $\mathcal{M}'$ constructed in step 2	$\Omega(k_2/\gamma^2)$
k_4	size of $\mathcal{M}''$ constructed in step 3	$\Omega(\alpha_{\text{WL}}^2 k_3)$
γ^*	size of $\mathcal{S}^*$ in the crossbar $(\mathcal{S}^*, \mathbf{\Gamma}^*, \mathbf{T}^*, \mathcal{M}^*)$	$\geq \gamma''/2 = \Theta(\gamma^{1/4})$
k^*	size of $\mathcal{M}^*$ in the crossbar $(\mathcal{S}^*, \mathbf{\Gamma}^*, \mathbf{T}^*, \mathcal{M}^*)$	$k_4 = \Omega\left(\frac{\alpha_{\text{WL}}^2 k}{\gamma^8 \log k}\right)$

finish the algorithm. Therefore, whenever the algorithm attempts to find such a flow F, it will either succeed or terminate the algorithm by returning a (k_1, α)-violating cut. From the integrality of flow, if F exists, we can find a set $\mathcal{P} : \Gamma \overset{1:1}{\rightsquigarrow}_{\lceil 1/\alpha \rceil} \Gamma'$ of paths contained in S_j.

The algorithm contains 4 steps. Firstly, we find a subset $\mathcal{S}' \subseteq \mathcal{S}$ of clusters and a degree-3 tree T over $V_T = \{v_j : S_j \in \mathcal{S}'\}$(see Figure 4.1). Every edge $e = (v_j, v_{j'}) \in E_T$ is associated with a collection $\mathcal{P}_e$ of k_2 $\mathcal{S}'$-avoiding paths connecting S_j and $S_{j'}$. The paths $\bigcup_{e \in E_T} \mathcal{P}_e$ cause congestion 2 in G. Secondly, we find a subset $\mathcal{M}' \subseteq \mathcal{M}$ demand pairs and a partition $(\mathcal{T}_1, \mathcal{T}_2)$ of the terminals in $\mathcal{T}(\mathcal{M}')$ that split every pair in $\mathcal{M}'$. We find two collections of $\mathcal{S}'$-avoiding paths $\mathcal{P}_1 : \mathcal{T}_1 \rightsquigarrow S_\ell$ and $\mathcal{P}_2 : \mathcal{T}_2 \rightsquigarrow S_{\ell'}$, for two clusters in $S_\ell, S_{\ell'} \in \mathcal{S}'$ (see Figure 4.2). $\mathcal{P}_1 \cup \mathcal{P}_2$ cause congestion 2. In this step, we trim the tree T a sub-tree T' (for some technical reason) and let $\mathcal{S}'' = \{S_j : v_j \in V_{T'}\}$ is the set of clusters correspondent to T'. Notice that $\bigcup_{e \in E_{T'}} \mathcal{P}_e \cup \mathcal{P}_1 \cup \mathcal{P}_2$ can cause congestion 4 in total. To overcome this, we reroute paths in each collection $\mathcal{P}_e$, $e \in E_{T'}$ to obtain $\mathcal{P}'_e$, so that paths in $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}'_e$ only cause congestion 2 in G. Thirdly, we guarantee that each cluster $S_j \in \mathcal{S}''$ is 1-well-linked for the set of edges in $\text{out}(S_j)$ that are used by $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}'_e$. This is done by sub-sampling paths from the set. Finally, we finish the algorithm by defining the crossbar. The paths $\{\mathcal{P}_e : e \in E_{T'}\}$ and the clusters $\mathcal{S}''$ provide necessary elements for constructing the crossbar(see Figure 4.3).

For the sake of readability, the key parameters used in the algorithm are given in Table 4.1. The 4 steps are described in Section 4.2, 4.3, 4.4 and 4.5 respectively.

4.2 Step 1: Constructing the Tree T

This step is summarized in the following lemma. (See Figure 4.1.)

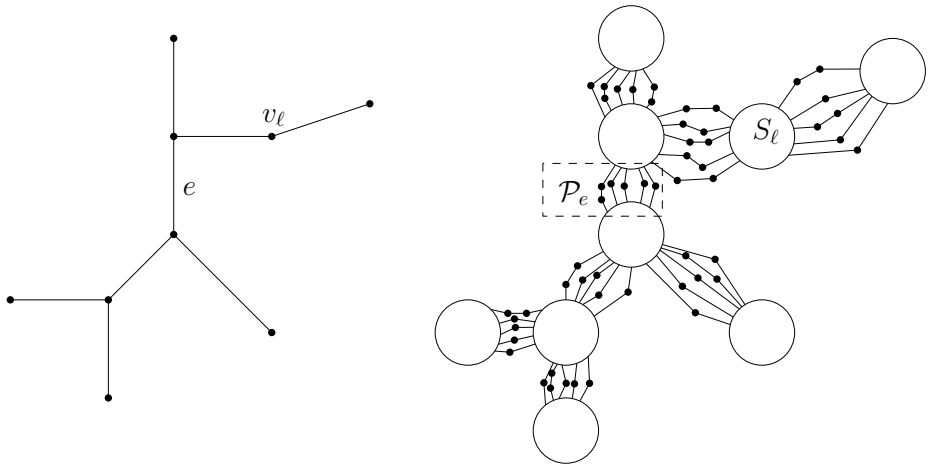

Figure 4.1: Step 1 of crossbar construction – constructing the tree T

Lemma 4.1 *There is an efficient algorithm, that either computes a (k_1, α)-violating cut of some set $S_j \in \mathcal{S}$, or finds $\mathcal{S}' \subseteq \mathcal{S}$, a tree T and $\{\mathcal{P}_e : e \in E_T\}$ such that*

(F1) $\mathcal{S}' \subseteq \mathcal{S}$ is a subset of size $\gamma' = \left\lfloor \gamma^{1/4}/2 \right\rfloor$,

(F2) T is a tree of degree at most 3 with $V_T = \{v_j : S_j \in \mathcal{S}'\}$,

(F3) for every edge $e = (v_j, v_{j'}) \in E_T$, $\mathcal{P}_e$ is a collection of $k_2 = \left\lfloor \frac{5k_1}{\gamma^4} \right\rfloor$ $\mathcal{S}'$-avoiding paths connecting S_j and $S_{j'}$ in G,

(F4) the set $\bigcup_{e \in E_T} \mathcal{P}_e$ of paths causes congestion at most 2 in G.

This section is dedicated to the proof of Lemma 4.1. Since G is 1-well-linked for the set $\mathcal{T}$ of terminals, every pair $(S_j, S_{j'})$ of clusters can send $k_1/2$ flow units to each other with congestion at most 3: concatenate the flows from S_j to a subset $\mathcal{T}_1$ of the terminals, from $S_{j'}$ to a subset $\mathcal{T}_2$ of the terminals, and those between $\mathcal{T}_1$ and $\mathcal{T}_2$. Equivalently, there are at least $k_{11} := \lfloor k_1/6 \rfloor$ edge-disjoint paths connecting S_j and $S_{j'}$ in G.

We build the following graph $Z = (V_Z, E_Z)$. Let $V_Z = \{v_1, \ldots, v_\gamma\}$. For every pair $v_j, v_{j'} \in V_Z$, let $w(v_j, v_{j'})$ be the maximum number of edge-disjoint $\mathcal{S}$-avoiding paths connecting S_j and $S_{j'}$ in G. Then $(v_j, v_{j'}) \in E_Z$ if and only if $w(v_j, v_{j'}) \geq k_{12} := \frac{4k_{11}}{\gamma^2}$. For simplicity, define $w(e) = w(v_j, v_{j'})$ for $e = (v_j, v_{j'}) \in E_Z$. Let $w(T) = \sum_{e \in E(T)} w(e)$.

Consider any non-trivial partition (A, B) of V_Z. Let $v_j \in A$ and $v_{j'} \in B$ be two vertices. Since there are at least k_{11} edge-disjoint paths connecting S_j and $S_{j'}$ in G, there must be at least k_{11} edge-disjoint paths connecting $\bigcup_{v_j \in A} S_j$ and $\bigcup_{v_j \in B} S_j$ that avoid $\mathcal{S}$. Thus, $w(A, B) := \sum_{v_j \in A, v_{j'} \in B} w(v_j, v_{j'}) \geq k_{11}$. We defer the proof of the following lemma to Section 4.2.3.

Lemma 4.2 *We can efficiently find either a spanning tree $\tilde{T}$ of Z containing at least $\gamma' := \left\lfloor \gamma^{1/4}/2 \right\rfloor$ leaves, or an induced path of length $\gamma' - 1$ in Z such that for any edge $e = (v_j, v_{j'})$ in the path, we have $w(v_j, v_{j'}) \geq \frac{k_{11}}{3\gamma'^2\gamma}$.*

In the above lemma, an induced path is a path which is also an induced sub-graph of Z. Apply Lemma 4.2, we obtain either a spanning tree $\tilde{T}$ of Z containing at least γ' leaves, or an induced path of length $\gamma' - 1$. By renaming the vertices in Z and sets in $\mathcal{S}$, we may assume the path is $(v_1, v_2, \cdots, v_{\gamma'})$. By Lemma 4.2, we have:

(G1) for each $j \in [\gamma' - 1]$, $w(v_j, v_{j+1}) \geq \frac{k_{11}}{3\gamma'^2\gamma} = \frac{\lfloor k_1/6 \rfloor}{3\left\lfloor \gamma^{1/4}/2 \right\rfloor^2 \gamma} \geq \frac{k_1}{5\gamma^{1.5}}$;

(G2) for each $j, j' \in [\gamma']$ with $|j - j'| > 1$, $w(v_j, v_{j'}) < k_{12}$.

We deal with the path case and tree case separately.

4.2.1 Path Case

In this case, we have a path $(v_1, v_2, \cdots, v_{\gamma'})$ satisfying Property (G1) and (G2). Let $\mathcal{S}' = \{S_j : j \in [\gamma']\}$. For each $1 \leq j < \gamma'$, let $\widetilde{\mathcal{P}}_j$ be a set of $\left\lfloor \frac{k_1}{5\gamma^{1.5}} \right\rfloor$ edge-disjoint paths connecting S_j to S_{j+1} and avoiding $\mathcal{S}$. While the paths in each set $\widetilde{\mathcal{P}}_j$ are edge-disjoint, the total congestion caused by the set $\bigcup_j \widetilde{\mathcal{P}}_j$ of paths may be as high as $\gamma' - 1$. In order to overcome this difficulty, we perform edge-splitting on an auxiliary graph H, constructed as follows. Start with the graph containing all the vertices and edges in the paths $\widetilde{\mathcal{P}}_1, \widetilde{\mathcal{P}}_2, \cdots, \widetilde{\mathcal{P}}_{\gamma'-1}$. Next, for each $1 \leq j \leq \gamma'$, we identify the vertices in S_j that appear in the graph to a single vertex v_j. Finally, we replace every edge by a pair of bi-directed edges. Let H be the final directed graph. Observe that H is a directed Eulerian graph with the following properties:

(H1) H does not contain any terminals, or vertices from any cluster $S_j \in \mathcal{S}'$(they were identified with v_j);

(H2) For each $1 \leq j < \gamma'$, there are at least $\left\lfloor \frac{k_1}{5\gamma^{1.5}} \right\rfloor$ edge-disjoint paths connecting v_j to v_{j+1} and avoiding $\{v_1, v_2, \cdots, v_{\gamma'}\}$;

(H3) For all $1 \leq j < j' \leq \gamma'$, with $|j - j'| > 1$, the maximum number of edge-disjoint paths connecting v_j and $v_{j'}$ and avoiding $\{v_1, v_2, \cdots, v_{\gamma'}\}$ is less than k_{12}.

We use the following theorem to perform edge-splitting in graph H. Let $D = (V, A)$ be any directed multigraph with no self-loops. For any pair $(v, v') \in V$ of vertices, their connectivity $\lambda(v, v'; D)$ is the maximum number of edge-disjoint paths connecting v to v' in D. Given a pair $a = (u, v)$, $b = (v, w)$ of edges, a splitting-off procedure replaces the two edges a, b by a single edge (u, w). We denote by $D^{a,b}$ the resulting graph. We use the extension of Mader's theorem [67] to directed graphs, due to Frank [35] and Jackson [44]. Following is a simplified version of Theorem 3 from [44]:

Theorem 4.3 *Let $D = (V, A)$ be an Eulerian digraph, $v \in V$ and $a = (u, v) \in A$. Then there is an edge $b = (v, w) \in A$, such that for all $y, y' \in V \setminus \{v\}$: $\lambda(y, y'; D) = \lambda(y, y'; D^{a,b})$.*

We iteratively perform edge-splitting in graph H using Theorem 4.3, by repeatedly choosing vertices $v \notin \{v_1 \ldots, v_{\gamma'}\}$, until all such vertices become isolated. Let $\vec{H}'$ be the underlying resulting graph with all isolated vertices removed. Let H' be the underlying undirected graph of $\vec{H}'$. Then $V_{H'} = \{v_1, \ldots, v_{\gamma'}\}$, and for each $1 \leq j \neq j' \leq \gamma'$, each edge $e = (v_j, v_{j'}) \in E_{H'}$ corresponds to a path P_e in graph G, connecting S_j and $S_{j'}$ and avoiding $\mathcal{S}$. Moreover, the set $\bigcup_{e \in E_{H'}} P_e$ of paths causes congestion at most 2 in G.

Claim 4.4 *For each $1 \leq j, j' \leq \gamma'$ with $|j - j'| > 1$, there are less than $2k_{12}$ parallel edges $(v_j, v_{j'})$ in H'; for each $1 \leq j < \gamma'$, there are at least $\frac{k_1}{10\gamma^{3/2}}$ parallel edges (v_j, v_{j+1}) in H'.*

Proof: The first statement is straightforward. Focus on the directed graph $\vec{H}'$. Each edge from v_j to $v_{j'}$ corresponds to a path from S_j to $S_{j'}$ that avoids $\mathcal{S}'$. W.l.o.g, we can assume the paths correspondent to all parallel edges of $(v_j, v_{j'})$ cause congestion 1 in G (if an edge is used twice, then the two paths go through the edge in different directions and we can remove the edge from both paths). Since $w(v_j, v_{j'}) < k_{12}$, there are less than k_{12} edges from v_j to $v_{j'}$ in $\vec{H}'$. Similarly, there are less than k_{12} edges from $v_{j'}$ to v_j in $\vec{H}'$. Thus, H' contains less than $2k_{12}$ edges between v_j and $v_{j'}$.

We now turn to prove the second statement. Consider some $j \in [\gamma' - 1]$ and the following cut (A, B) of $V_{H'}$: $A = \{v_1, \ldots, v_j\}$, $B = \{v_{j+1}, \ldots, v_{\gamma'}\}$. Recall that graph H contained at least $\left\lfloor \frac{k_1}{5\gamma^{1.5}} \right\rfloor$ edge-disjoint paths connecting v_j to v_{j+1}. By the property of Theorem 4.3, H' contains at least $\left\lfloor \frac{k_1}{5\gamma^{1.5}} \right\rfloor$ edge-disjoint paths connecting v_j to v_{j+1}. In particular, $|E_{H'}(A, B)| \geq \left\lfloor \frac{k_1}{5\gamma^{1.5}} \right\rfloor$. For all pairs $(u, v) \in A \times B$ except (v_j, v_{j+1}), there are less than $2k_{12}$ parallel edges of (u, v), by the first statement. Noticing that there are at most $\frac{\gamma'^2}{4} - 1$ such pairs, the number of edges between v_j and v_{j+1} is at least

$$\left\lfloor \frac{k_1}{5\gamma^{1.5}} \right\rfloor - \left(\frac{\gamma'^2}{4} - 1 \right) \cdot 2k_{12} \geq \left\lfloor \frac{k_1}{5\gamma^{1.5}} \right\rfloor - \frac{\gamma^{1/2}}{16} \cdot \frac{8 \lfloor k_1/6 \rfloor}{\gamma^2} \geq \frac{k_1}{10\gamma^{1.5}}.$$

□

Our final tree T is simply a path connecting the vertices $(v_1, v_2, \cdots, v_{\gamma'})$ in this order. For each edge $e = (v_j, v_{j+1})$ on this path, we let $\mathcal{P}_e$ be the set of paths in graph G' corresponding to the set of parallel edges connecting v_j to v_{j+1} in graph H'. By Claim 4.4, we have $|\mathcal{P}_e| \geq \frac{k_1}{10\gamma^{1.5}} \geq k_2$. We discard paths from each set $\mathcal{P}_e$ until $|\mathcal{P}_e| = k_2$. It is immediate to verify that $\bigcup_{e \in E(T)} \mathcal{P}_e$ cause congestion at most 2 in G, and moreover, every path in the set avoids $\mathcal{S}'$. This finishes the proof of Lemma 4.1 for the path case.

4.2.2 Tree Case

In this case, we have a spanning tree $\tilde{T}$ of Z containing at least γ' leaves. Let $\mathcal{S}' \subseteq \mathcal{S}$ be any set of γ' clusters, corresponding to γ' leaves in tree $\tilde{T}$. For the ease of notations, we assume $\mathcal{S}' = \{S_1, \ldots, S_{\gamma'}\}$. Our first step is summarized in the following claim. The proof is deferred to Section 4.2.3.

Claim 4.5 *There is an efficient algorithm, that either computes a (k_1, α)-violating cut of some set $S_j \in \mathcal{S}$, or computes, for each set $S_j \in \mathcal{S}'$, a subset $\Gamma_j \subseteq \mathrm{out}(S_j)$ of $k_{13} := \left\lfloor \frac{k_{12}}{2\gamma\gamma'} \right\rfloor$ edges, such that for each $S_j, S_{j'} \in \mathcal{S}'$, there is a set $\mathcal{P} : \Gamma_j \overset{1:1}{\rightsquigarrow}_1 \Gamma_{j'}$ of paths in G avoiding $\mathcal{S}'$.*

Our next step is to perform edge splitting, similarly to the path case. We build an auxiliary graph H from G, as follows. We delete from G all terminals, and for each $j \in [\gamma']$, we delete all edges in $\mathrm{out}(S_j) \setminus \Gamma_j$. Next, we contract each set S_j, $j \in [\gamma']$, into a super-node v_j. Finally, we replace every edge in the resulting graph by a pair of bi-directed edges. Let H be the resulting graph. Notice that the in-degree and the out-degree of every super-node v_j in H is exactly k_{13}, and for every pair $v_j, v'_{j'}$ of such super-nodes, there are k_{13} edge-disjoint paths connecting v_j to $v_{j'}$, and k_{13} edge-disjoint paths connecting $v_{j'}$ to v_j in H.

We iteratively perform edge-splitting in graph H using Theorem 4.3, by repeatedly choosing vertices $v \notin \{v_1 \ldots, v_{\gamma'}\}$, until all such vertices become isolated. Let H' be the underlying undirected graph of the resulting graph, with all isolated vertices removed. Then $V_{H'} = \{v_1, \ldots, v_{\gamma'}\}$. Each edge $e = (v_j, v_{j'}) \in E_{H'}$ corresponds to a (directed) path P_e in graph G, connecting S_j to $S_{j'}$ and avoiding $\mathcal{S}'$. Moreover, the set $\bigcup_{e \in E_{H'}} P_e$ of paths causes congestion at most 2 in G.

H' is a $2k_{13}$-regular graph. We show that the connectivity between any two vertices in H' is $2k_{13}$. To see this, focus on the directed version $\vec{H}'$ of H'. That is, $\vec{H}'$ is the resulting directed graph after splitting-off and removing all isolated vertices. Then, by the property of splitting-off, there are k_{13} edge-disjoint paths from v_j to $v_{j'}$, and k_{13} edge-disjoint paths from $v_{j'}$ to v_j in H'. Then, for any cut (A, B) separating v_j and $v_{j'}$ in $\vec{H}'$, there are at least k_{13} edges from A to B and k_{13} edges from B to A. Since H' is the underlying undirected graph of $\vec{H}'$, there are at least $2k_{13}$ edges between A and B in H'.

We prove the following lemma in Section 4.2.3.

Lemma 4.6 *Let K be a sufficiently large integer. Let $H = (V_H, E_H)$ be a K-regular multi-graph on $n_H = |V_H|$ vertices, where the connectivity between any two vertices is K. Then, we can efficiently find a spanning tree T of H of maximum degree 3 such that every edge of T has at least K/n_H^3 parallel edges in H.*

We apply Lemma 4.6 on H' with $K = 2k_{13}$, to obtain a tree T of degree at most 3 such that each edge of T has many parallel edges in H'. We output the tree T as our final tree. Notice that each edge $e = (v_j, v_{j'}) \in E_T$ corresponds to a collection $\mathcal{P}_e$ of

paths in G connecting S_j and $S_{j'}$ and avoiding $\mathcal{S}'$. Moreover, the paths in $\bigcup_{e\in E_T}\mathcal{P}_e$ cause congestion at most 2 in G. The number of paths in each set is:

$$|\mathcal{P}_e| \geq \frac{2k_{13}}{{\gamma'}^3} = \left\lfloor \frac{k_{12}}{2\gamma\gamma'} \right\rfloor / {\gamma'}^3 \geq \left\lfloor \frac{8k_{12}}{\gamma^2} \right\rfloor \geq \left\lfloor 8 \cdot 4 \left\lfloor k_1/6 \right\rfloor / \gamma^4 \right\rfloor \geq \left\lfloor \frac{5k_1}{\gamma^4} \right\rfloor = k_2.$$

We discard the edges in each $\mathcal{P}_e$ until $|\mathcal{P}_e| = k_2$. This finishes the proof of Lemma 4.1.

4.2.3 Omitted Proofs

Proof of Lemma 4.2: It is easy to verify that Z is connected: consider any non-trivial cut (A, B) of V_Z; since $w(A, B) \geq k_{11}$, there must be a pair $(u, v) \in A \times B$ such that $w(u, v) \geq k_{11}/|A||B| \geq 4k_{11}/\gamma^2$ and thus $(u, v) \in E_Z$. Let $\tilde{T}$ be any spanning tree of Z. Root the tree $\tilde{T}$ at any vertex. We now perform a number of iterations. In every iteration, one of the following two operations is performed, whenever possible.

1. Let $(v_j, v_{j'})$ be an edge in $\tilde{T}$ such that v_j is the parent of $v_{j'}$. Suppose v_j is not the root and v_j has degree 2 in $\tilde{T}$. Let $v_{j''}$ be a non-leaf vertex such that $(v_{j'}, v_{j''}) \in E_Z$ and $v_{j''}$ is not a descendant of $v_{j'}$ in $\tilde{T}$. Then we delete $(v_j, v_{j'})$ from $\tilde{T}$, and add $(v_{j'}, v_{j''})$ to it.

2. Let $(v_j, v_{j'})$ be an edge in $\tilde{T}$ such that v_j is the parent of $v_{j'}$. Suppose both v_j and $v_{j'}$ has degree 2 in $\tilde{T}$ and v_j is not the root. Consider the two connected components of $\tilde{T} \setminus (v_j, v_{j'})$, and let $e \in E(Z)$ be the edge connecting these two components, with the maximum $w(e)$. If $w(e) > w(v_j, v_{j'})$, then we delete $(v_j, v_{j'})$ from $\tilde{T}$ and add e to it.

Notice that Operation 1 increases the number of leaves in $\tilde{T}$ by 1, and Operation 2 increases $w(\tilde{T}) := \sum_{e\in\tilde{T}} w(e)$ by at least 1 without decreasing the number of leaves in $\tilde{T}$. Thus, after polynomially many improvement steps, we obtain a final tree $\tilde{T}$, on which none of operations can be performed. Let L denote the set of leaves of $\tilde{T}$.

If $|L| \geq \gamma'$, we can return $\tilde{T}$. We now assume that $|L| < \gamma'$. A path P in the tree $\tilde{T}$ is called a 2-path iff it does not contain the root of $\tilde{T}$, and all its vertices have degree 2 in $\tilde{T}$. A 2-path P is maximal iff it is not a strict sub-path of any other 2-path. Since the number of leaves in $\tilde{T}$ is less than γ', the number of maximal 2-paths in $\tilde{T}$ is at most $2\gamma'$. Therefore, $\tilde{T}$ contains at least one 2-path of length at least $(\gamma - 2\gamma')/2\gamma' \geq \gamma'$. We let P' be a maximal 2-path of length at least γ', that minimizes the size of the sub-tree rooted at the lowest vertex of P'. Let P be the sub-path of P', consisting of the last γ' vertices of P'. Assume w.l.o.g. that $P = (v_1, \ldots, v_{\gamma'})$, where $v_{\gamma'}$ is the vertex that lies deepest in the tree $\tilde{T}$. We now prove that P has the desired properties.

Since the $|L| < \gamma'$, and every 2-path in the sub-tree of $v_{\gamma'}$ has length at most γ' (by the choice of P'), the size of the sub-tree rooted at $v_{\gamma'}$ is bounded by $2\gamma'(\gamma' + 2)$. Fix some $1 \leq j < \gamma'$, and consider the edge $e = (v_j, v_{j+1})$. Notice that the size of the sub-tree rooted at v_{j+1} is bounded by $2\gamma'(\gamma' + 2) + \gamma' \leq 2\gamma'(\gamma' + 3) \leq 3{\gamma'}^2$. Let $\tilde{T}_1, \tilde{T}_2$

be the two connected components of $\tilde{T} \setminus e$. Recall that $w(V_{\tilde{T}_1}, V_{\tilde{T}_2}) \geq k_{11}$. So there is at least one pair $(v_{j'}, v_{j''}) \in V_{\tilde{T}_1} \times V_{\tilde{T}_2}$ such that $w(v_{j'}, v_{j''}) \geq \frac{k_{11}}{|V_{\tilde{T}_1}||V_{\tilde{T}_2}|} \geq \frac{k_{11}}{3\gamma'^2\gamma}$. Since we could not perform Operation 2, we have $w(v_j, v_{j+1}) \geq \frac{k_{11}}{3\gamma'^2\gamma}$.

Now consider some pair $v_j, v_{j'}$ of vertices on path P, with $|j - j'| > 1$, and assume w.l.o.g. that v_j is a descendant of $v_{j'}$. Since we could not perform Operation 1 on edge (v_{j-1}, v_j) and vertex $v_{j'}$, we have $(v_j, v_{j'}) \notin E_Z$. Hence, P is an induced path. □

Proof of Claim 4.5: Take any v_{j^*} as the root of the tree $\tilde{T}$. We claim that for each set $S_j \in \mathcal{S}'$, there is a flow F_j of value k_{12}, connecting S_{j^*} to S_j in G, with congestion at most $\frac{1}{\alpha} + \gamma \leq 2\gamma$, such that the paths in F_j avoid $\mathcal{S}'$. Indeed, consider the root-to-leaf path $(v_j^* = v_{j_0}, v_{j_1}, \ldots, v_{j_h} = v_j)$ in the tree $\tilde{T}$. For each edge $e_z = (v_{j_z}, v_{j_z+1})$ on this path, there is a set $\mathcal{Q}_z$ of k_{12} edge-disjoint paths connecting S_{j_z} to $S_{j_{z+1}}$ and avoiding $\mathcal{S}'$, from the definition of the graph Z. Let $\Gamma_z^2 \subseteq \text{out}(S_{j_z})$ be the set of first edges of paths in $\mathcal{Q}_z$, $\Gamma_{z+1}^1 \subseteq \text{out}(S_{j_{z+1}})$ be the set of last edges of paths in $\mathcal{Q}_z$. For each $1 \leq z \leq h-1$, we now have $\Gamma_z^1, \Gamma_z^2 \subseteq \text{out}(S_{j_z})$, two subsets of edges of size $k_{12} < k_1/2$. We can either find a (k_1, α)-violating cut of S_{j_z}, or find a flow $F_z' : \Gamma_z^1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} \Gamma_z^2$ inside set S_z. Concatenating the flows $(\mathcal{Q}_0, F_1', \mathcal{Q}_1, F_2', \ldots, F_{h-1}', \mathcal{Q}_{h-1})$, we obtain the desired flow F_j of value k_{12}. The total congestion caused by paths in $\bigcup_{z=0}^{h-1} \mathcal{Q}_z$ is at most γ (since $h \leq \gamma$, and the paths in each set $\mathcal{Q}_z$ are edge-disjoint), while each flow F_z' causes congestion at most $1/\alpha$ inside the graph $G[S_{j_z}]$. Therefore, the total congestion due to flow F_j is bounded by $\frac{1}{\alpha} + \gamma \leq 2\gamma$.

Scaling all flows F_j, for $j \in [\gamma']$ by factor $1/2\gamma\gamma'$, we obtain a new flow F, where every set $S_j \in \mathcal{S}'$ sends $\frac{k_{12}}{2\gamma\gamma'}$ flow units to S_{j^*}, and the total congestion due to F is at most 1. From the integrality of flow, there is a collection $\{\mathcal{P}_j : j \in [\gamma']\}$ of path sets, where for each $j \in [\gamma']$, set $\mathcal{P}_j$ contains $\lfloor \frac{k_{12}}{2\gamma\gamma'} \rfloor$ paths connecting S_j to S_{j^*}. The paths in $\bigcup_{j \in [\gamma']} \mathcal{P}_j$ are edge-disjoint and avoid $\mathcal{S}'$. We will also assume w.l.o.g. that the paths avoid S_{j^*}. Let $\Gamma_j \subseteq \text{out}(S_j)$ be the set of first edges of $\mathcal{P}_j$, and $\Gamma_j' \subseteq \text{out}(S_{j^*})$ be the set of last edges of $\mathcal{P}_j$. Consider some pair $j, j' \in [\gamma']$. If we can not find a set of paths $\mathcal{Q}_{j,j'} : \Gamma_j' \overset{1:1}{\rightsquigarrow}_1 \Gamma_{j'}'$, we find a (k_1, α)-violating cut of S_{j^*}. Otherwise, by concatenating $\mathcal{P}_j, \mathcal{Q}_{j,j'}$ and $\mathcal{P}_{j'}$, obtain a set $\Gamma_j \overset{1:1}{\rightsquigarrow}_1 \Gamma_{j'}$ of $\left\lfloor \frac{k_{12}}{2\gamma\gamma'} \right\rfloor = k_{13}$ paths in G avoiding $\mathcal{S}'$. □

Proof of Lemma 4.6: Let $H' = (V_{H'}, E_{H'})$ be any connected graph. It is well-known that the following polytope is the polytope for spanning trees of H':

$$\sum_{e \in E_{H'}} x_e = n_{H'} - 1$$

$$\sum_{e \in E(H'[S])} x_e \leq |S| - 1 \quad \forall S \subseteq V_{H'}, S \neq \emptyset$$

$$0 \leq x_e \leq 1 \qquad \forall e \in E_{H'}$$

We define H' be the following spanning sub-graph of H. An edge $e \in E_H$ is in H' if and only if e has at least K/n_H^3 parallel edges (including e itself). For

our graph H', we let $x_e = \frac{n_H - 1}{|E_{H'}|}$ for every $e \in E_{H'}$. We show that the solution satisfies all the constraints defining the polytope. The first constraint is trivially satisfied. For the third constraint, we count the number of edges in H'. $|E_{H'}| \geq |E_H| - \binom{n_H}{2}\frac{K}{n_H^3} \geq \frac{n_H K}{2} - \frac{K}{2n_H} \geq \frac{K(n_H-1)}{2}$. For $K \geq 2$, we have $x_e \leq 1$ for every $e \in E_{H'}$. It remains to consider the second constraint. Focus on some non-trivial subset $S \subseteq V_H$. Since the connectivity of the graph H is K, then $|E_H(S, V_H \setminus S)| \geq K$, implying $|E_{H'[S]}| \leq |E_{H[S]}| \leq \frac{|S|K-K}{2}$. Then, $\sum_{e \in E_{H'[S]}} \leq \frac{|S|K-K}{2}\frac{2}{K} = |S| - 1$. Thus, $\{x_e : e \in E_{H'}\}$ is a fractional point in the polytope.

Singh and Lau [78] proved the following theorem.

Theorem 4.7 ([78]) *Given any connected graph $H' = (V_{H'}, E_{H'})$ and any fractional point $\{x_e : e \in E_{H'}\}$ in the polytope for spanning trees of H', we can efficiently find a spanning tree T of H', such that $d_T(v) \leq \left\lceil \sum_{e \in \delta_{H'}(v)} x_e \right\rceil + 1$ for every $v \in V_{H'}$, where $d_T(v)$ is the degree of v in T and $\delta_{H'}(v)$ is the set of edges incident to v in H'.*

In our fractional solution $\{x_e : e \in E_{H'}\}$, we have $d_T(v) < K\frac{n_H - 1}{|E_{H'}|} \leq 2$ for every $v \in V_H$. Thus, by applying Theorem 4.7, we obtain a tree T such that $d_T(v) \leq 3$ for every $v \in V_H$. Moreover, since this tree is in the subgraph H', every edge in the tree has at least $\frac{K}{n_H^3}$ parallel edges. □

4.3 Step 2: Connecting terminals

Up to now, we have obtained a subset $\mathcal{S}' = \{S_j : j \in [\gamma']\}$ of clusters, a tree T of maximum degree at most 3 on $V_T = \{v_j : j \in [\gamma']\}$ and a collection $\mathcal{P}_e$ of k_2 paths for every $e \in E_T$ satisfying the conditions of Lemma 4.1.

In this section, we connect a subset of terminals to the clusters in $\mathcal{S}'$, as stated in the following lemma. (See Figure 4.2.)

Lemma 4.8 *There is an efficient algorithm, that either finds a routing of $\Omega(k_1)$ demand pairs in $\mathcal{M}$ via edge-disjoint paths in G, or finds $\mathcal{M}' \subseteq \mathcal{M}$, $\mathcal{T}_1, \mathcal{T}_2 \subseteq \mathcal{T}(\mathcal{M})$, a sub-tree T' of T, $\mathcal{S}'' \subseteq \mathcal{S}'$, $\ell, \ell' \in [\gamma']$, $\mathcal{P}_1, \mathcal{P}_2$ and $\{\mathcal{P}'_e : e \in E_{T'}\}$, such that*

- *(I1) $\mathcal{M}' \subseteq \mathcal{M}$ is a set of $k_3 = \Omega(k_2/\gamma'^2)$ demand pairs;*
- *(I2) $(\mathcal{T}_1, \mathcal{T}_2)$ is a partition of $\mathcal{T}(\mathcal{M}')$ that splits every pair in $\mathcal{M}'$;*
- *(I3) T' is a sub-tree of T of size $\gamma'' \geq \gamma'/4$ and $\mathcal{S}'' = \{S_j : v_j \in V_{T'}\}$;*
- *(I4) $v_\ell \in V_{T'}$ has degree 1 in T', and $v_{\ell'} \in V_{T'}$ has degree at most 2 in T';*
- *(I5) $\mathcal{P}_1 : \mathcal{T}_1 \overset{1:1}{\rightsquigarrow} \mathrm{out}(S_\ell)$, $\mathcal{P}_2 : \mathcal{T}_2 \overset{1:1}{\rightsquigarrow} \mathrm{out}(S_{\ell'})$ are two sets of paths in G;*
- *(I6) for every edge $e \in E_{T'} \subseteq E_T$, $\mathcal{P}'_e \subseteq \mathcal{P}_e$ is a subset of $\Omega(k_2)$ paths;*
- *(I7) paths in $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \left(\bigcup_{e \in E_{T'}} \mathcal{P}'_e\right)$ avoid $\mathcal{S}''$, and cause congestion 2 in G;*

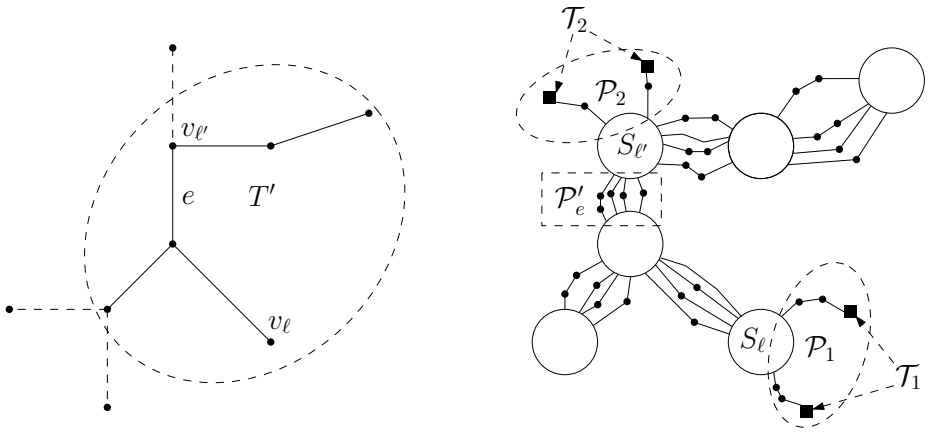

Figure 4.2: Step 2 of crossbar construction – connecting terminals

(I8) every edge in $\bigcup_{v_j \in V_{T'}} \mathrm{out}(S_j)$ *is used by at most one path in the* $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \left(\bigcup_{e \in E_{T'}} \mathcal{P}'_e\right)$.

We prove Lemma 4.8 in this section.

4.3.1 Initial Connection

In this subsection, we shall find a subset $\mathcal{M}_0 \subseteq \mathcal{M}$ of $\lfloor k_2/4 \rfloor$ pairs, and connect the terminals $\mathcal{T}(\mathcal{M}_0)$ to some cluster in $\mathcal{S}'$, say, S_1, via a set $\mathcal{P} : \mathcal{T}(\mathcal{M}_0) \rightsquigarrow_2 \mathrm{out}(S_1)$ of paths in graph G.

Since we assumed there is no cut of size less than $k_1/2$ separating S_1 and $\mathcal{T}$, there is a set $\widetilde{\mathcal{P}}$ of $\lfloor k_1/2 \rfloor$ edge-disjoint paths connecting the terminals in $\mathcal{T}$ to the edges in $\mathrm{out}(S_1)$. Let $\mathcal{T}_0 \subseteq \mathcal{T}$ be the subset of $\lfloor k_1/2 \rfloor$ terminals that serve as endpoints of paths in $\widetilde{\mathcal{P}}$. We partition the remaining terminals into $h-1 = \left\lceil \frac{k - \lfloor k_1/2 \rfloor}{\lfloor k_1/2 \rfloor} \right\rceil$ subsets $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{h-1}$ of at most $\lfloor k_1/2 \rfloor$ terminals each. Then $h = \left\lceil \frac{k}{\lfloor k_1/2 \rfloor} \right\rceil \leq 2.05k/k_1$ for sufficiently large k. Since G is 1-well-linked for $\mathcal{T}$, there is a set $\widetilde{\mathcal{P}}_j : \mathcal{T}_j \rightsquigarrow_1 \mathcal{T}_0$ of paths for every $j \in [h-1]$. Using paths in $\widetilde{\mathcal{P}}$ h times and paths in $\left\{\widetilde{\mathcal{P}}_j : j \in [h-1]\right\}$, we obtain an integral set $\widehat{\mathcal{P}} : \mathcal{T} \rightsquigarrow_{2h} \mathrm{out}(S_1)$ of paths.

Our next step is to perform a grouping of the terminals, using Lemma 2.11, with the parameter $q = 2h$. Let $\mathcal{U}$ be the resulting partition of the terminals. Then each set $U \in \mathcal{U}$ contains at least q and at most $3q-3$ terminals, and is associated with a tree T_U containing the terminals of U. The trees $\{T_U\}_{U \in \mathcal{U}}$ are edge-disjoint.

We partition $\mathcal{M}$ into two subsets: $\mathcal{M}_1$ containing all pairs (s,t) where both s and t belong to the same group $U \in \mathcal{U}$, and $\mathcal{M}_2$ containing all remaining pairs.

Assume first that $|\mathcal{M}_1| \geq k/2$. We then select a subset $\widetilde{\mathcal{M}} \subseteq \mathcal{M}_1$ of demand pairs as follows: for each group $U \in \mathcal{U}$, we select one arbitrary pair $(s,t) \in \mathcal{M}_1$ with $s,t \in U$, if such a pair exists, and add it to $\widetilde{\mathcal{M}}$. Since every group contains at most $3q = 6h$ terminals, $\left|\widetilde{\mathcal{M}}\right| \geq \frac{k}{12h} \geq \frac{k}{12 \times 2.05k/k_1} \geq \frac{k_1}{25} = \Omega(k_1)$. Every pair $(s,t) \in \widetilde{\mathcal{M}}$ can

be connected by some path contained in the tree T_U for the U containing s and t. Since the trees $\{T_U\}_{U\in\mathcal{U}}$ are edge-disjoint, the resulting routing of the $\Omega(k_1)$ pairs in $\widetilde{\mathcal{M}}$ is via edge-disjoint paths.

We assume from now on that $|\mathcal{M}_2| \geq k/2$. We greedily select $\widetilde{\mathcal{M}}$ as follows. Start with $\widetilde{\mathcal{M}} = \emptyset$. While $\mathcal{M}_2 \neq \emptyset$, select any pair $(s,t) \in \mathcal{M}_2$, add it to $\widetilde{\mathcal{M}}$ and remove from $\mathcal{M}_2$ all pairs (s',t') such that one of s' and t' is in the same group as one of s and t. Notice that for each pair added to $\widetilde{\mathcal{M}}$, at most $6q = 12h$ pairs are deleted from $\mathcal{M}_2$. Therefore, at the end of this procedure, $\left|\widetilde{\mathcal{M}}\right| \geq \frac{k}{24h} \geq \frac{k_1}{50}$.

We now show that there is a flow $F^* : \mathcal{T}(\widetilde{\mathcal{M}}) \rightsquigarrow_2 \text{out}(S_1)$ in G. Let F obtained by scaling $\widehat{\mathcal{P}}$ down by a factor of $2h$. Then every terminal in $\mathcal{T}$ sends $1/2h$ flow units to the edges of $\text{out}(S_1)$, and the total congestion caused by F is at most 1. In order to define the flow F^*, consider some terminal $t \in \mathcal{T}(\widetilde{\mathcal{M}})$, and let $U_t \in \mathcal{U}$ be the group to which it belongs. Let $U_t' \subseteq U_t$ be any subset of $2h$ terminals in U_t. Terminal t then sends $1/2h$ flow units to each terminal in U_t', inside the tree T_{U_t}. This flow is then concatenated with the flow that leaves terminals in U_t' in F. Taking the union of this flow for each $t \in \mathcal{T}(\widetilde{\mathcal{M}})$, we obtain a flow F^*, where every terminal in $\mathcal{T}(\widetilde{\mathcal{M}})$ sends one flow unit to the edges of $\text{out}(S_1)$. Since F contributes at most congestion 1 and $\{T_U : U \in \mathcal{G}\}$ are edge-disjoint, the overall congestion caused by the flow F^* is at most 2. From the integrality of flow, we can find a set $\mathcal{T}(\widetilde{\mathcal{M}}) \rightsquigarrow_2 \text{out}(S_1)$ of $2\left|\widetilde{\mathcal{M}}\right| = \Omega(k_1)$ paths in G. Let $\mathcal{M}_0 \subseteq \widetilde{\mathcal{M}}$ be any subset of $\lfloor k_2/4 \rfloor$ pairs, $\mathcal{T}_0 = \mathcal{T}(\mathcal{M}_0)$, and $\mathcal{P}$ be the set of paths connecting $\mathcal{T}_0$ to $\text{out}(S_1)$.

4.3.2 Rerouting Paths in $\mathcal{P}$

Notice that each of the two sets $\mathcal{P}$ and $\bigcup_{e\in E_{T'}} \mathcal{P}_e$ can cause congestion 2 in G and the union of them can cause congestion 4. In order to overcome this, we prove the following lemma.

Lemma 4.9 (Rerouting Lemma) *Let $G = (V, E)$ be a graph. Let $\mathcal{P} : V_1 \rightsquigarrow_1 V_2$ be a set of $|\mathcal{P}| = |V_1|$ paths in G, where $V_1, V_2 \subseteq V$. We are also given a collection $\mathcal{Q}$ of edge-disjoint paths terminating at V_2. Then, we can efficiently find a subset $\mathcal{Q}' \subseteq \mathcal{Q}$ of at least $|\mathcal{Q}| - |\mathcal{P}|$ paths, and a collection $\mathcal{P}' : V_1 \rightsquigarrow_1 V_2$ of $|V_1|$ paths in G, such that $\mathcal{P}' \cup \mathcal{Q}'$ causes congestion 1 in G.*

Proof: The proof is very similar to the arguments used by Conforti et al. [33]. We re-route the paths in $\mathcal{P}$ via segments of paths in $\mathcal{Q}$, by setting up an instance of the stable matching problem. In this instance, we are given two sets A, B of the same size. Each vertex $a \in A$ ($b \in B$, resp.) specifies an preference ordering of vertices in B (A, resp.). A complete matching M between A and B is called stable iff, for every unmatched pair $(a, b) \in A \times B$, either a prefers b' over b, or b prefers a' over a, where a' and b' are the vertices such that $(a', b) \in M$ and $(a, b') \in M$. Conforti et al. [33], generalizing the famous theorem of Gale and Shapley [38], showed an efficient algorithm to find a complete stable matching M for any set of preference orderings.

We set up an instance of the stable matching problem as follows. A contains a vertex v_P for each path $P \in \mathcal{P}$, and set B contains a vertex v_Q for each path $Q \in \mathcal{Q}$. In order to ensure that $|A| = |B|$, we add dummy vertices to A as needed. We define preference lists for vertices in A and B. For each vertex $v_P \in A$, the vertices in B are ordered according to the order of P intersecting their correspondent paths in $\mathcal{Q}$. The vertices v_Q for which P does not intersect Q appear in the end of the preference ordering in arbitrary order. Similarly for each vertex $v_Q \in B$, the vertices of A are ordered according to the order of the *reversal* of Q intersecting the correspondent paths in $\mathcal{P}$. The vertices v_P for which Q does not intersect P and the dummy vertices appear in the end of the preference ordering in arbitrary order.

Let M be a stable matching for the instance. Let $\mathcal{Q}' \subseteq \mathcal{Q}$ be the set of paths Q, whose vertex v_Q is matched to a dummy vertex. This ensures that $|\mathcal{Q}'| \geq |\mathcal{Q}| - |\mathcal{P}|$.

In order to construct the set $\mathcal{P}'$ of paths, consider some path $P \in \mathcal{P}$ with $(v_P, v_Q) \in M$. If P and Q are edge-disjoint, we add P to $\mathcal{P}'$. Otherwise, we add to $\mathcal{P}'$ a path P' consisting of two sub-paths: a segment of P from the beginning of P until it intersect Q, and a segment of Q from the intersecting edge to the end of Q.

This finishes the definitions of $\mathcal{P}'$ and $\mathcal{Q}'$. Observe that the paths in $\mathcal{P}'$ connect every terminal in V_1 to V_2. It remains to show that the set $\mathcal{P}' \cup \mathcal{Q}'$ causes congestion 1 in G.

Clearly, the paths in $\mathcal{Q}'$ are edge-disjoint, since $\mathcal{Q}' \subseteq \mathcal{Q}$. We now show that $\mathcal{P}'$ are edge disjoint. Consider any two paths $P', P'' \in \mathcal{P}'$. Assume w.l.o.g. that P' was obtained from concatenating a segment of some path $P_1 \in \mathcal{P}$ and some path $Q_1 \in \mathcal{Q}$ (possibly Q_1 is empty and $P' = P_1$), and similarly P'' was obtained from concatenating a segment of some path $P_2 \in \mathcal{P}$ and some path $Q_2 \in \mathcal{Q}$ via some edge e_2. The paths P', P'' may share an edge only if the first segment of P' shares an edge with the second segment of P'', or vice versa. Assume w.l.o.g. that it is the former. Then, v_{P_1} prefers v_{Q_2} over v_{Q_1} since P_1 intersects Q_2 before it intersects Q_1. Also, v_{Q_2} prefers v_{P_1} over v_{P_2} since the reversal of Q_2 intersects P_1 first before it intersects P_2. This contradicts the fact that M is a stable matching and thus P' and P'' are edge-disjoint.

Finally, if some pair of paths $P' \in \mathcal{P}'$ and $Q' \in \mathcal{Q}'$ share an edge, we reach a contradiction exactly as before. □

We apply Lemma 4.9 on the graph G with all edges duplicated, $\mathcal{Q} = \bigcup_{e \in E_{T'}} \mathcal{P}_e$, the set $\mathcal{P} : \mathcal{T}_0 \overset{1:1}{\rightsquigarrow}_2 \text{out}(S_1)$ we constructed, $V_1 = \mathcal{T}_0$ and $V_2 = \bigcup_{j \in [\gamma']} \text{out}(S_j)$. We then obtain a set $\mathcal{P}' : \mathcal{T}_0 \to \bigcup_{j \in [\gamma']} \text{out}(S_j)$ and $\mathcal{Q}' \subseteq \mathcal{Q}$ of size at least $|\mathcal{Q}| - |\mathcal{P}| \geq |\mathcal{Q}| - k_2/2$ such that $\mathcal{P}' \cup \mathcal{Q}'$ causes congestion 2 in G. Since there are at most $k_2/2$ less edges in $\mathcal{Q}'$ than in $\mathcal{Q}$, we have $|\mathcal{P}_e \cap \mathcal{Q}'| \geq |\mathcal{P}_e| - k_2/2 \geq k_2/2$. For any $e \in E_{T'}$, let $\mathcal{P}'_e \subseteq \mathcal{P}_e \cap \mathcal{Q}'$ be any subset of size $\lfloor k_2/2 \rfloor$.

For each path $P \in \mathcal{P}'$, we direct the path from the terminal, and truncate it at the first vertex in $\bigcup_{j \in [\gamma']} S_j$. The new collection $\mathcal{P}'$ of paths then connects every terminal in $\mathcal{T}_0$ to some vertex in $\bigcup_{j \in [\gamma']} S_j$ and they avoid $\mathcal{S}'$.

4.3.3 Selecting the Set $\mathcal{M}'$ of Pairs

In the next step, we shall define the subset $\mathcal{M}' \subseteq \mathcal{M}_0$ of k_3 pairs and a partition $(\mathcal{T}_1, \mathcal{T}_2)$ of $\mathcal{T}(\mathcal{M}')$ that split every pair in $\mathcal{M}'$. Moreover, we guarantee that all the paths in $\mathcal{P}'$ originating at $\mathcal{T}_1$ terminate at a same cluster $S_\ell \in \mathcal{S}'$, and all the paths in $\mathcal{P}'$ originating at $\mathcal{T}_2$ terminate a same cluster $S_{\ell'} \in \mathcal{S}'$.

Notice that paths in $\mathcal{P}'$ connect $\mathcal{T}_0$ to clusters in $\mathcal{S}'$. Let $S_\ell \in \mathcal{S}'$ be a set such that at least $|\mathcal{T}_0|/\gamma'$ paths in $\mathcal{P}'$ terminate at S_ℓ. Let $\mathcal{T}' \subseteq \mathcal{T}_0$ be the subset of terminals where these paths originate. Thus, we have $|\mathcal{T}'| \geq |\mathcal{T}_0|/\gamma'$.

We first consider the easier case where at least $|\mathcal{T}'|/4$ pairs $(s,t) \in \mathcal{M}_0$ have $\{s,t\} \subseteq \mathcal{T}'$. We can let $\mathcal{M}' \subseteq \mathcal{M}_0$ be a subset of $k_3 := \lfloor |\mathcal{T}'|/4 \rfloor = \Omega(k_2/\gamma')$ such pairs. Let $\mathcal{T}_1(\mathcal{T}_2$, resp.) contain the first (second, resp.) terminals of pairs in $\mathcal{M}'$. Let $\ell' = \ell$ in this case. Then paths in $\mathcal{P}'$ connect $\mathcal{T}(\mathcal{M}')$ to $S_\ell = S_{\ell'}$. Let $\mathcal{P}_1 \subseteq \mathcal{P}'$ be the paths originating at $\mathcal{T}_1$ and $\mathcal{P}_2 \subseteq \mathcal{P}'$ be the paths originating at $\mathcal{T}_2$.

Now consider the harder case where less than $|\mathcal{T}'|/4$ pairs $(s,t) \in \mathcal{M}_0$ have $\{s,t\} \subseteq \mathcal{T}'$. So, at least $|\mathcal{T}'|/2$ pairs $(s,t) \in \mathcal{M}_0$ have $|\{s,t\} \cap \mathcal{T}'| = 1$. Let $\mathcal{T}_1'$ be the set of terminals in $\mathcal{T}'$ whose partners are not in $\mathcal{T}'$ and $\mathcal{T}_2'$ be the partners of $\mathcal{T}_1'$. Thus, $|\mathcal{T}_1'| = |\mathcal{T}_2'| \geq |\mathcal{T}'|/2, \mathcal{T}_1' \subseteq \mathcal{T}'$ and $\mathcal{T}_2' \cap \mathcal{T}' = \emptyset$. Focus on the $|\mathcal{T}_2'|$ paths in $\mathcal{P}'$ originating at $\mathcal{T}_2'$. There is a $S_{\ell'} \in \mathcal{S}'$ such that at least $|\mathcal{T}_2'|/\gamma'$ of these paths terminate at $\text{out}(S_{\ell'})$.(Notice that $\ell' \neq \ell$ since none of the $|\mathcal{T}_2'|$ paths terminate at S_ℓ.) Let $\mathcal{T}_2 \subseteq \mathcal{T}_2'$ be a subset of $\lceil |\mathcal{T}_2'|/\gamma' \rceil$ terminals whose correspondent paths in $\mathcal{P}'$ terminate at $S_{\ell'}$. Let $\mathcal{T}_1 \subseteq \mathcal{T}_1'$ be the partners of terminals in $\mathcal{T}_2$ and $\mathcal{M}' \subseteq \mathcal{M}_0$ be the set of pairs between $\mathcal{T}_1$ and $\mathcal{T}_2$. Then, $k_3 := |\mathcal{M}'| \geq |\mathcal{T}_2'|/\gamma' \geq |\mathcal{T}_0|/\gamma'^2 = \Omega(k_2/\gamma'^2)$. Let $\mathcal{P}_1$ ($\mathcal{P}_2$, resp.) be the set of paths in $\mathcal{P}'$ originating at $\mathcal{T}_1(\mathcal{T}_2$, resp.).

We require that each edge in $\text{out}(S_\ell) \cup \text{out}(S_{\ell'})$ is used by at most 1 path in $\mathcal{P}_1 \cup \mathcal{P}_2$, as in the theorem statements. This is not hard to guarantee. We say two terminals $s, s' \in \mathcal{T}_1 \cup \mathcal{T}_2$ conflict if the two paths in $\mathcal{P}_1 \cup \mathcal{P}_2$ originating at s and s' terminate at the same edge in $\text{out}(S_\ell) \cup \text{out}(S_{\ell'}$. Then, two pairs (s,t) and (s',t') in $\mathcal{T}_1 \times \mathcal{T}_2$ conflict if some terminal in $\{s,t\}$ and some terminal in $\{s',t'\}$ conflict. Notice that each pair conflicts with at most 2 other pairs, we can select $|\mathcal{M}'|/3$ pairs without conflicts. For notational simplicity, we assume there are no conflicts in $\mathcal{M}'$.

4.3.4 Truncating the Tree T

Our final step is to truncate the tree T to a tree T'. Assume first that $\ell \neq \ell'$. Since T has degree at most 3, we obtain at most 5 sub-trees by removing v_ℓ and $v_{\ell'}$ from T. Let T_0 denote the sub-tree neighbouring both v_ℓ and $v_{\ell'}$, T_1, T_2 the sub-trees neighbouring v_ℓ, and T_3, T_4 the sub-trees neighbouring only $v_{\ell'}$ (possibly some of the sub-trees are empty). W.l.o.g, assuming T_4 has the most number of vertices among T_1, T_2, T_3, T_4. Then T' is obtained from T by deleting T_1, T_2, T_3. It is easy to see that T' must contain at least $\gamma'/4$ vertices, and it contains both vertices v and v'. Moreover, the degree of v_ℓ is 1, and the degree of $v_{\ell'}$ is at most 2 in T'. If $\ell = \ell'$, removing v_ℓ from T will result in at most 3 sub-trees. Similarly, we only keep the largest sub-tree. The resulting tree T' contains at least $\gamma'/3$ vertices and v_ℓ has degree 1 in T'.

We define $\mathcal{S}'' = \{S_j \in \mathcal{S}' : v_j \in V_{T'}\}$. The only thing left is to guarantee Property (I8). This is not hard to guarantee by randomly sampling : if an edge $e \in \text{out}(S_j)$ is used by two paths in $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}'_e$, we randomly one of the two paths. We remove a pair from $\mathcal{M}'$ if one of the two paths correspondent to the two terminals are removed. Notice each path in $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}'_e$ remains with probability at least $1/4$, and each pair in $\mathcal{M}'$ remains with probability at least $1/4$. By Chernoff bound and union bound, with high probability, every $\mathcal{P}'_e$ still contains $\Omega(k_2)$ edges and $\mathcal{M}'$ contains $\Omega(k_3)$ pairs. Property (I1) can be satisfied by redefining k_3. By redefining $\mathcal{T}_1, \mathcal{T}_2, \mathcal{P}_1, \mathcal{P}_2$, all the other properties are still satisfied.

4.4 Step 3: Ensuring Well-Linkedness for Tails

After Step 2, we have constructed the structure in Lemma 4.8 that satisfy Property (I1) to (I8). W.l.o.g. we assume that $\mathcal{S}'' = \{S_1, \ldots, S_{\gamma''}\}$, where $\gamma'' \geq \gamma'/4$. $\mathcal{S}''$ will serve as the clusters in the cross-bar. Notice that each edge $e = (v_j, v_{j'}) \in T'$ corresponds to many paths connecting S_j and $S_{j'}$. These set of paths for all edges $e \in T'$ provides necessary elements to construct many parallel trees in G. In order to complete these trees, we need to connect the tails of the paths inside clusters. This requires that clusters are well-linked for the set of tails : the edges via which the paths in $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}'_e$ connecting to clusters in $\mathcal{S}''$. This is guaranteed by selecting subsets of these paths, as stated in the following lemma. To the end of this section, we shall use $\Gamma_j(\mathcal{P})$ to denote the set of edges in $\text{out}(S_j)$ that are used by $\mathcal{P}$, for some cluster $S_j \in \mathcal{S}''$ and some set $\mathcal{P}$ of paths avoiding $\mathcal{S}''$. Notice that if an edge in $\text{out}(S_j)$ is used by $\mathcal{P}$, it must be the first or the last edge of some path $P \in \mathcal{P}$, since paths in $\mathcal{P}$ avoid S_j. Moreover, if $\mathcal{P}$ is a subset of $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}'_e$, every edge in $\Gamma_j(\mathcal{P})$ is used by exactly one path in $\mathcal{P}$ by Property (I8).

Lemma 4.10 *There is an efficient algorithm, that either computes a routing of $\Omega(\alpha_{\text{WL}} k_3)$ pairs in $\mathcal{M}'$ with congestion at most 2 in G, or a (k_1, α)-violating partition of some set $S_j \in \mathcal{S}''$, or $\{\mathcal{P}''_e : e \in E_{T'}\}, \mathcal{M}'' \subset \mathcal{M}', \mathcal{T}'_1, \mathcal{T}'_2, \mathcal{P}'_1$ and $\mathcal{P}'_2$ such that*

- *(J1) for every $e \in E_{T'}$, $\mathcal{P}''_e \subseteq \mathcal{P}'_e$ is a subset of $2k_4 = \Omega(\alpha^2_{\text{WL}} k_3)$ paths in $\mathcal{P}_e$,*
- *(J2) $\mathcal{M}'' \subseteq \mathcal{M}'$ is set of size k_4,*
- *(J3) $\mathcal{T}'_1 := \mathcal{T}_1 \cap \mathcal{T}(\mathcal{M}'')$ and $\mathcal{T}'_2 := \mathcal{T}_2 \cap \mathcal{T}(\mathcal{M}'')$ are respectively the subset of terminals in $\mathcal{T}'_1$ and $\mathcal{T}'_2$ involved in $\mathcal{M}''$,*
- *(J4) $\mathcal{P}'_1 \subseteq \mathcal{P}_1$ ($\mathcal{P}'_2 \subseteq \mathcal{P}_2$, resp.) is the set of paths in $\mathcal{P}_1$ ($\mathcal{P}_2$, resp.) connecting $\mathcal{T}'_1$ ($\mathcal{T}'_2$, resp.) to S_ℓ ($S_{\ell'}$, resp.),*
- *(J5) every $S_j \in \mathcal{S}''$ is 1-well-linked for the set $\Gamma_j\left(\mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}''_e\right)$ of edges.*

The proof of this lemma is by applying Theorem 2.10 repeatedly. Initially, let $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \bigcup_{e \in E_{T'}} \mathcal{P}'_e$. We apply a sampling procedure for each $j \in [\gamma'']$ in arbitrary order. After the procedure for j, we guarantee that S_j is 1-well-linked for $\Gamma_j(\mathcal{P})$. The procedures for $j \notin \{\ell, \ell'\}$ and for $j \in \{\ell, \ell'\}$ are slightly different and we describe them separately.

Before describing the procedure, we prove the following useful lemma:

Lemma 4.11 *Let $H = (V_H, E_H)$ be a d-degenerate graph for some integer $d > 0$. Assume we are given r disjoint subsets $V_1, V_2, \cdots, V_r$ of V_H. Then, we can find an independent set $V' \subseteq V_H$ of H such that $|V' \cap V_i| \geq \lfloor |V_i| / 2(d+1)r \rfloor$ for every $i \in [r]$.*

Proof: We describe the algorithm for constructing V'. Let $U = V_i$ and $V' = \emptyset$ initially. Repeat the following procedure until U becomes empty. Since H is d-degenerate, $H[U]$ contains a vertex v of degree at most d. Suppose $v \in V_i$ for some $i \in [\gamma']$. If $|V' \cap V_i| < \lfloor s/(d+1)\gamma' \rfloor$, we add v to V' and remove v and all neighbours of v from U. Otherwise, we simply remove v from U.

Now, we prove that $|V' \cap V_i| \geq \lfloor s/(d+1)\gamma' \rfloor$ for every $i \in [\gamma']$. Assume towards contradiction that $|V' \cap V_i| < \lfloor s/(d+1)\gamma' \rfloor$ for some i. Consider the an iteration where we removed some vertices of V_i from U. If in the iteration we added some v to V', then we removed at most $d+1$ vertices from U. Otherwise, $v \notin V_i$ since we have $|V' \cap V_i| < \lfloor s/(d+1)\gamma' \rfloor$. In this case we did not remove any vertex from U. The former case can happen at most $\gamma' \lfloor s/(d+1)\gamma' \rfloor - 1 \leq s/(d+1) - 1$ times. Thus, we removed at most $(s/(d+1) - 1)(d+1) < s$ vertices of V_i from U, contradicting the fact that U becomes empty at the end. □

4.4.1 Sampling Procedure for $j \notin \{\ell, \ell'\}$

We now describe the sampling procedure for $j \in [\gamma''] \setminus \{\ell, \ell'\}$. Focus on the graph $G' = G[S_j \cup \mathrm{out}(S_j)]$ and terminals $\Gamma = \Gamma_j(\mathcal{P})$. We run $\mathcal{A}_{\mathrm{ARV}}$ on G' with terminal set Γ, to obtain a cut (A, B). If the sparsity of the cut (A, B) is less than α, then (A, B) defines a (k_1, α)-violating cut of S_j. We then stop the algorithm by returning the cut (A, B). Otherwise, we are guaranteed that set S_j is $\alpha/\alpha_{\mathrm{ARV}} = \alpha_{\mathrm{WL}}$-well-linked for Γ. We then apply Theorem 2.10 on the graph G' and terminal set Γ to obtain an $O(1/\alpha_{\mathrm{WL}})$-degenerate graph $Z = (\Gamma, E_Z)$. Notice that v_j has degree at most 3 in T'. Then the set Γ of edges is naturally divided into at most 3 subsets, one for each of the edges incident to v_j. That is, a subset for an edge e is the set of all tails correspondent to $\Gamma_j(\mathcal{P} \cap \mathcal{P}'_e)$. For simplicity, assume there are exactly 3 subsets and they are Γ_1, Γ_2 and Γ_3. Thus, we can apply Lemma 4.11 to obtain a independent set $\Gamma' \subseteq \Gamma$ of Z such that $|\Gamma' \cap \Gamma_i| \geq \lfloor |\Gamma_i| / O(1/\alpha_{\mathrm{WL}}) \rfloor = \Omega(\alpha_{\mathrm{WL}}) |\Gamma_i|$ for every $i \in [3]$. Define $\Gamma'_i = \Gamma' \cap \Gamma_i$ for $i \in [3]$. By Theorem 2.10, we can construct a 4-degenerate graph $Y = (\Gamma', E_Y)$ such that G' is 1-well-linked for every independent set Γ'' of Y. We apply Lemma 4.11 again to obtain an independent set $\Gamma'' \subseteq \Gamma'$ of H such that $|\Gamma'' \cap \Gamma'_i| \geq \Omega(|\Gamma'_i|) = \Omega(\alpha_{\mathrm{WL}}) |\Gamma_i|$ for every $i \in [3]$. Now, we update the set $\mathcal{P}$ of paths so that $\Gamma_j(\mathcal{P}) = \Gamma''$ as follows: we remove a path P from $\mathcal{P}$ if one of its tails is in $\Gamma \setminus \Gamma''$.

4.4.2 Sampling Procedure for $j \in \{\ell, \ell'\}, \ell \neq \ell'$

We proceed to deal with the case $j \in \{\ell, \ell'\}$. First consider the simpler case that $\ell \neq \ell'$. We apply the above procedure for $j = \ell$ and $j = \ell'$ respectively. Define $G' = G[S_j \cap \text{out}(S_j)]$ and $\Gamma = \Gamma_j(\mathcal{P})$. Γ is divided into at most 3 subsets : one subset for tails of $\mathcal{P} \cap \mathcal{P}'_e$, for each edge $e \in E_{T'}$ incident to v_j (there are 1 or 2 such edges) and one subset for tails of $\mathcal{P} \cap \mathcal{P}_1$ or $\mathcal{P} \cap \mathcal{P}_2$ (depending on whether $j = \ell$ or $j = \ell'$). Then we can then apply Theorem 2.10 to obtain a subset $\Gamma'' \subseteq \Gamma$, and update the set $\mathcal{P}$ of paths as before, using the same procedure as we did for $j \notin \{\ell, \ell'\}$. In order to make sure that the paths in $\mathcal{P}'_1$ and $\mathcal{P}'_2$ are matched according to $\mathcal{M}'$, when removing a path $P \in \mathcal{P}_1$ (resp., $\mathcal{P}_2$) from $\mathcal{P}$, we remove its matched path from $\mathcal{P}_2$(resp., $\mathcal{P}_1$). The matched paths are naturally defined: a path $P \in \mathcal{P}_1$ and a path in $P' \in \mathcal{P}_2$ are matched if the two terminals contained in P and P' are matched in $\mathcal{M}'$.

4.4.3 Sampling Procedure for $j = \ell = \ell'$

Now we consider the more complicated case $j = \ell = \ell'$. Then v_j has degree 1 in T'. Focus on the graph $G' = G[S_j \cup \text{out}(S_j)]$ and terminals $\Gamma = \Gamma_j(\mathcal{P})$. Γ is naturally divided into 3 subsets: $\Gamma_1 = \Gamma_j(\mathcal{P} \cap \mathcal{P}_1)$, $\Gamma_2 = \Gamma_j(\mathcal{P} \cap \mathcal{P}_2)$, and $\Gamma_3 = \Gamma_j(\mathcal{P} \cap \mathcal{P}'_e)$, where e is the unique edge in T' incident to v_j. The complication of this case comes from the requirement that $\mathcal{P} \cap \mathcal{P}_1$ and $\mathcal{P} \cap \mathcal{P}_2$ are matched. As before, we guarantee that G' is α_{WL}-well-linked for Γ, and find a $O(1/\alpha_{\text{WL}})$-degenerate graph Z. In the degenerate graph, if $e \in \Gamma_1$ and $e' \in \Gamma_2$ are matched according to $\mathcal{M}'$(i.e, e and e' are tails of two matched paths), we identify e and e'. The resulting graph Z' has degeneracy at most 4 times the degeneracy of Z and is thus $O(1/\alpha_{\text{WL}})$-degenerate. We can then apply Lemma 4.11 to obtain an independent set $\Gamma' \subseteq \Gamma_1 \cap \Gamma_3$ (notice that vertices in Γ_2 are identified with vertices in Γ_1). Let $\Gamma'_1 = \Gamma' \cap \Gamma_1$, $\Gamma'_3 = \Gamma' \cap \Gamma_3$ and Γ'_2 be the vertices in Γ_2 identified with Γ'_1 (thus, $|\Gamma'_1| = |\Gamma'_2|$). Redefine Γ' as $\Gamma'_1 \cup \Gamma'_2 \cup \Gamma'_3$. By the lemma, $|\Gamma'_i| \geq \Omega(\alpha_{\text{WL}}) |\Gamma_i|$ for every $i \in [3]$.

Focus on the graph $Z[\Gamma']$. The edges in $Z[\Gamma']$ form a partial matching: the only edges are of the form (e, e') for which e and e' are matched. If $Z[\Gamma']$ contains at least $|\Gamma'_1|/2$ edges, we can route $|\Gamma'_1|/2$ matched pairs in $\Gamma'_1 \times \Gamma'_2$, by Theorem 2.10, since the edges form an induced matching. By concatenating these paths with correspondent paths in $\mathcal{P}_1$ and $\mathcal{P}_2$, we obtain a routing of $|\Gamma'_1|/2$ pairs in G with congestion 2 (since $\mathcal{P}_1 \cup \mathcal{P}_2$ causes congestion 2). Thus, we assume $Z[\Gamma']$ contains less than $|\Gamma'_1|/2$ edges. We remove the elements incident to the edges in $Z[\Gamma']$ from Γ', Γ'_1 and Γ'_2. Then, the resulting set Γ' is an independent set of Z. By Theorem 2.10, we can then construct a 4-degenerate graph Y on Γ' such that G' is 1-well-linked for any independent set $\Gamma'' \subseteq \Gamma'$ of Y. Repeating the same process for Y, Γ'_1, Γ'_2 and Γ'_3(that is, identifying Γ'_2 with Γ'_1 to obtain Y', applying Lemma 4.11, etc.), we either find a routing of $\Omega(\Gamma'_1)$ pairs in $\mathcal{M}'$ with congestion 2, or a subset $\Gamma'' \subseteq \Gamma'$ such that $|\Gamma'' \cap \Gamma'_i| = \Omega(|\Gamma'_i|)$ for every $i \in [3]$. Moreover, $\Gamma'' \cap \Gamma'_1$ and $\Gamma'' \cap \Gamma'_2$ are perfectly matched. Finally we update $\mathcal{P}$ by removing paths with one of its tails in $\Gamma \setminus \Gamma''$. This finishes the sampling procedure.

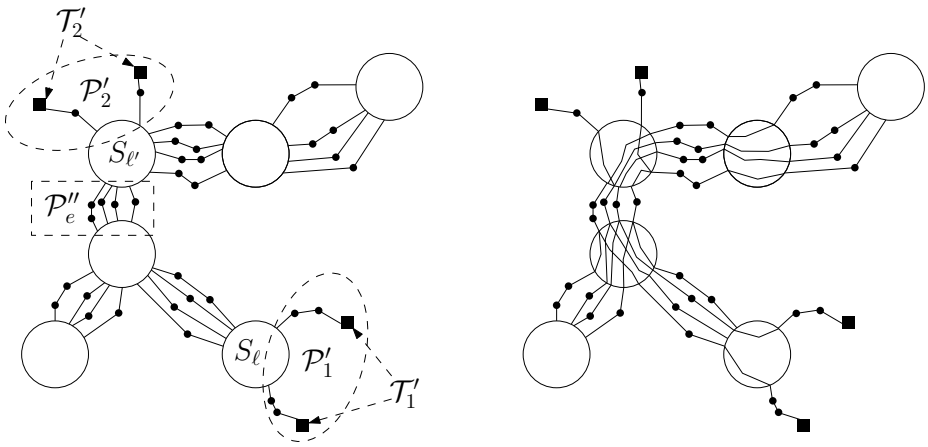

Figure 4.3: Step 4 of crossbar construction – finishing up

4.4.4 Finishing Proving Lemma 4.10

We then let $\mathcal{P}''_e = \mathcal{P} \cap \mathcal{P}'_e$ for every $e \in E_{T'}$, $\mathcal{P}'_1 = \mathcal{P} \cap \mathcal{P}_1$ and $\mathcal{P}'_2 = \mathcal{P} \cap \mathcal{P}_2$. Define $\mathcal{T}'_1(\mathcal{T}'_2$, resp.) to be the set of terminals $\mathcal{P}'_1$ ($\mathcal{P}'_2$, resp.) originating at. Our sampling procedure guarantees that each S_j is 1-well-linked for $\Gamma_j(\mathcal{P})$. In order to prove Lemma 4.10, it now remains show that the sizes of these sets are large. For each edge $e = (v_j, v_{j'}) \in E_{T'}$, $|\mathcal{P} \cap \mathcal{P}'_e|$ is decreased twice in the iterative process: one time in the iteration for j and the other in the iteration for j'. Since each time the fraction of elements remaining is at least $\Omega(\alpha_{\text{WL}})$, we have that the $|\mathcal{P}''_e| \cap \Omega(\alpha^2_{\text{WL}})\,|\mathcal{P}'_e|$ for every $e \in E_{T'}$. Now consider $\mathcal{P}'_1$ and $\mathcal{P}'_2$. If $\ell \neq \ell'$, $|\mathcal{P} \cap \mathcal{P}_1|$ is decreased twice in the process: one time in the iteration for ℓ and the other in the iteration for ℓ'. Also, in each time the fraction of elements remaining is at least $\Omega(\alpha_{\text{WL}})$. Thus, $|\mathcal{P}'_1| \geq \Omega(\alpha^2_{\text{WL}})\,|\mathcal{P}_1|$. By the same argument, $|\mathcal{P}'_2| \geq \Omega(\alpha^2_{\text{WL}})\,|\mathcal{P}_2|$. For the case $\ell = \ell'$, $|\mathcal{P} \cap \mathcal{P}_1|$ is decreased once. Thus, $|\mathcal{P}'_1| \geq \Omega(\alpha_{\text{WL}})\,|\mathcal{P}_1|$. Therefore, $|\mathcal{P}''_e| \geq \Omega(\alpha^2_{\text{WL}}\,|\mathcal{P}'_e|) = \Omega(\alpha^2_{\text{WL}} k_2)$ for every $e \in E_{T'}$, and $|\mathcal{P}'_1| = |\mathcal{P}'_2| = \Omega(\alpha^2_{\text{WL}} k_3)$. If the procedure for $j = \ell = \ell'$ returned a routing, the number of pairs routed is at least $\Omega(\alpha_{\text{WL}}|\mathcal{M}'|) = \Omega(\alpha_{\text{WL}} k_2/\gamma'^2)$. By discarding some paths, we can assume $|\mathcal{P}'_1| = |\mathcal{P}'_2| = k_4 = \Omega(\alpha^2_{\text{WL}} k_3)$ and $|\mathcal{P}''_e| = 2k_4$ for every $e \in E_{T'}$.

4.5 Step 4: Finishing Up

With the structure guaranteed by Lemma 4.10, it is straightforward to construct our crossbar (see Figure 4.3).

We now describe how we construct the $2k_4$ trees in the crossbar. Instead of giving the set of vertices and edges for these trees, we construct a set $\mathcal{P}$ of paths. If we replace each path with a fake edge connecting the two points, it is straightforward to check that the set of fake edges form $2k_4$ trees.

First, we include all paths in $\bigcup_{e \in E_{T'}} \mathcal{P}''_e$ in the set $\mathcal{P}$. Now, consider any set $S_j \in \mathcal{S}''$. If the degree of v_j in tree T' is 2, then let e, e' be the two edges incident to v_j in T'. Since S_j is 1-well-linked for $\Gamma_j(\mathcal{P}''_e \cap \mathcal{P}''_{e'})$, we can find a set $\mathcal{Q}_j : \Gamma_j(\mathcal{P}''_e) \overset{1:1}{\rightsquigarrow}_1$

$\Gamma_j(\mathcal{P}''_{e'})$ of paths contained in S_j. We include $\mathcal{Q}_j$ in $\mathcal{P}$. If the degree of v_j in tree T' is 3, let e, e' and e'' be the three edges incident to v_j in T'. We can find two sets $\mathcal{Q}^1_j : \Gamma_j(\mathcal{P}''_e) \overset{1:1}{\rightsquigarrow}_1 \Gamma_j(\mathcal{P}''_{e'})$ and $\mathcal{Q}^2_j : \Gamma_j(\mathcal{P}''_e) \overset{1:1}{\rightsquigarrow}_1 \Gamma_j(\mathcal{P}''_{e''})$ of paths inside S_j. We include $\mathcal{Q}^1_j$ and $\mathcal{Q}^2_j$ in $\mathcal{P}$.

The set of paths in $\mathcal{P}$ form $2k_4$ trees. We now need to add more edges to $\mathcal{P}$ so that each tree contains one terminal. To do this, we add $\mathcal{P}'_1$ and $\mathcal{P}'_2$ to $\mathcal{P}$. If $\ell = \ell'$, let e be the unique edge incident to v_ℓ in T'. We can find a set of paths $\mathcal{Q}' : \Gamma_\ell(\mathcal{P}''_e) \overset{1:1}{\rightsquigarrow}_1 \Gamma_\ell(\mathcal{P}'_1 \cup \mathcal{P}'_2)$. Then we add $\mathcal{Q}'$ to $\mathcal{P}$. Now consider the case $\ell \neq \ell'$. Let e be some edge incident to v_ℓ in T', and e' be some edge incident to v'_ℓ in T'. Construct arbitrarily two subsets $\Gamma \subseteq \Gamma_\ell(\mathcal{P}''_e)$ and $\Gamma' \subseteq \Gamma_{\ell'}(\mathcal{P}''_{e'})$ of tails of size k_4 each, with the constraint that Γ and Γ' appear in $2k_4$ different trees. Then, we can find two sets of paths $\mathcal{Q}' : \Gamma \overset{1:1}{\rightsquigarrow}_1 \Gamma_\ell(\mathcal{P}'_1)$ and $\mathcal{Q}'' : \Gamma' \overset{1:1}{\rightsquigarrow}_1 \Gamma_{\ell'}(\mathcal{P}'_2)$. We add $\mathcal{Q}'$ and $\mathcal{Q}''$ to $\mathcal{P}$. It is easy to see that we constructed a set of $2k_4$ trees in either case, each containing one of the $2k_4$ terminals in $\mathcal{T}'_1 \cup \mathcal{T}'_2$.

We now analyze the congestion caused by the trees. The paths in $\bigcup_{e \in E_{T'}} \mathcal{P}''_e \cup \mathcal{P}'_1 \cup \mathcal{P}'_2$ cause congestion 2 and avoid $\mathcal{S}''$. Each of the other paths added to $\mathcal{P}$ is inside some cluster $S \in \mathcal{S}''$. If $v_j, j \in [\gamma''] \setminus \{\ell, \ell'\}$ has degree d in T', then each edge in S_j is used at most $d-1$ times. Each edge in S_ℓ is used at most once.

Thus, we let $\mathcal{S}^*$ contain all sets S_j, for all $j \in [\gamma''] \setminus \{\ell'\}$ such that the degree of vertex v_j in tree T' is either 1 or 2. Then, each edge inside any cluster in S^* is used by at most one tree. Notice that at least half the vertices of T' have this property, and therefore, $|\mathcal{S}^*| \geq \gamma''/2$. By discarding clusters, we can assume $|\mathcal{S}^*| = \gamma^*$. W.l.o.g, assume $S^* = S_1, S_2, \cdots, S_{\gamma^*}$ Let $k^* = k_4$ and name the terminals in $\mathcal{T}'_1 \cup \mathcal{T}'_2$ $t_1, t_2, \cdots, t_{2k^*}$. For a terminal $t_i \in \mathcal{T}'_1 \cup \mathcal{T}'_2$, let T_i be the unique tree containing t_i. For $i \in [2k^*]$ and a cluster $S_j \in \mathcal{S}^*$, we can define $e_{i,j}$ to be the unique edge in $E_{T_i} \cap \Gamma_j(\mathcal{P}''_e)$, for an arbitrary edge $e \in E_{T'}$ incident to v_j. Then, S_j is 1-well-linked for the set $\Gamma_j := \bigcup_{i \in [2k^*]} \{e_{i,j}\}$. Let $\mathbf{\Gamma} = \{\Gamma_j : j \in [\gamma']\}$, $\mathbf{T} = \{T_i : i \in [2k^*]\}$ and $\mathcal{M}^* = \mathcal{M}''$, we obtained a (γ^*, k^*)-crossbar $(\mathcal{S}^*, \mathbf{\Gamma}^*, \mathbf{T}^*, \mathcal{M}^*)$ of congestion 2 in G.

Part II

Approximation Algorithms for Facility Location Problems

Chapter 5

Uncapacitated Facility Location Problem

In this chapter, we give our 1.488-approximation algorithm for the UFL problem. As mentioned in the introduction, our algorithm was built on the work of Byrka [19]. An algorithm is a (γ_f, γ_c)-bifactor approximation algorithm if the solution given by the algorithm has expected total cost at most $\gamma_f F^* + \gamma_c C^*$, where F^* and C^* are respectively the facility and the connection cost of an optimal solution for the linear programming relaxation of the UFL problem, which is described later. Byrka presented an algorithm $A_1(\gamma)$ which gives the optimal bifactor approximation $(\gamma, 1 + 2e^{-\gamma})$ for $\gamma \geq \gamma_0 \approx 1.6774$. By either running $A_1(\gamma_0)$ or the $(1.11, 1.78)$-approximation algorithm A_2 proposed by Jain, Mahdian and Saberi [46], Byrka was able to give a 1.5-approximation algorithm. We show that the approximation ratio can be improved to 1.488 if γ is randomly selected. To be more specific, we show

Theorem 5.1 *There is a distribution over $(1, \infty) \cup \{\perp\}$ such that the following random algorithm for the* UFL *problem gives a solution whose expected cost is at most 1.488 times the cost of the optimal solution : we randomly choose a γ from the distribution; if $\gamma = \perp$, return the solution given by A_2; otherwise, return the solution given by $A_1(\gamma)$.*

Due to the $(\gamma, 1 + 2e^{-\gamma} - \epsilon)$-hardness result given by [46], there is a hard instance for the algorithm $A_1(\gamma)$ for every γ. Roughly speaking, we show that a fixed instance can not be hard for two different γ's. Guided by this fact, we first give a bifactor approximation ratio for $A_1(\gamma)$ that depends on the input instance and then introduce a 0-sum game that characterizes the approximation ratio of our algorithm. The game is between an algorithm designer and an adversary. The algorithm designer plays either $A_1(\gamma)$ for some $\gamma > 1$ or A_2, while the adversary plays an input instance for the UFL problem. By giving an explicit (mixed) strategy for the algorithm designer, we show that the value of the game is at most 1.488.

The remaining part of the paper is organized as follows. In Section 5.1, we review the approximation algorithm $A_1(\gamma), \gamma > 1$ in [19], which gives a $(\gamma, 1 + 2e^{-\gamma})$-bifactor approximation for $\gamma \geq \gamma_0 \approx 1.67736$, and then we give our algorithm in Section 5.2.

5.1 Review of the Algorithm $A_1(\gamma)$ in [19]

In $A_1(\gamma), \gamma > 1$, we first solve the following natural linear programming relaxation for the UFL problem.

$$\min \quad \sum_{i\in\mathcal{F}, j\in\mathcal{C}} d(i,j)x_{i,j} + \sum_{i\in\mathcal{F}} f_i y_i \quad \text{s.t.}$$

$$\sum_{i\in\mathcal{F}} x_{i,j} = 1 \quad \forall j \in \mathcal{C} \tag{5.1}$$

$$x_{i,j} - y_i \le 0 \quad \forall i \in \mathcal{F}, j \in \mathcal{C} \tag{5.2}$$

$$x_{i,j}, y_i \ge 0 \quad \forall i \in \mathcal{F}, j \in \mathcal{C} \tag{5.3}$$

In the integer programming correspondent to the above LP relaxation, we have additional constraint that $x_{i,j}, y_i \in \{0,1\}$ for every $i \in \mathcal{F}$ and $j \in \mathcal{C}$. y_i indicates if the facility i is open and $x_{i,j}$ indicates if the client j is connected to the facility i. Equation (5.1) says that the client j must be connected to some facility and Inequality (5.2) says that a client j can be connected to a facility i only if i is open.

If the y-variables are fixed, x-variables can be assigned greedily in the following way. Initially, $x_{i,j} = 0$. For each client $j \in \mathcal{C}$, execute the following steps. Sort facilities by their distances to j; then for each facility i in the order, assign $x_{ij} = y_i$ if $\sum_{i'\in\mathcal{F}} x_{i',j} + y_i \le 1$ and $x_{i,j} = 1 - \sum_{i'} x_{i',j}$ otherwise.

After obtaining a solution (x, y), we modify it by scaling the y-variables up by γ. Let $\overline{y}$ be the scaled vector of y-variables. We reassign x-variables using the above greedy process to obtain a new solution $(\overline{x}, \overline{y})$.

Without loss of generality, we can assume that the following conditions hold for every $i \in \mathcal{C}$ and $j \in \mathcal{F}$:

1. $x_{i,j} \in \{0, y_i\}$;
2. $\overline{x}_{i,j} \in \{0, \overline{y}_i\}$;
3. $0 < \gamma y_i = \overline{y}_i \le 1$.

Indeed, the above conditions can be guaranteed by splitting facilities. To guarantee the first condition, we split i into 2 co-located facilities i' and i'' and let $x_{i',j} = y_{i'} = x_{i,j}, y_{i''} = y_i - x_{i,j}$ and $x_{i'',j} = 0$, if we find some facility $i \in \mathcal{F}$ and client $j \in \mathcal{C}$ with $0 < x_{i,j} < y_i$. The other x variables associated with i' and i'' can be assigned naturally. We update $\overline{x}$ and $\overline{y}$ variables accordingly. Similarly, we can guarantee the second condition. To guarantee the third condition, we can remove the facilities i with $y_i = 0$; we can split a facility i into 2 co-located facilities i' and i'' with $\overline{y}_{i'} = 1$ and $\overline{y}_{i''} = \overline{y}_i - 1$, if we find some facility $i \in \mathcal{F}$ with $\overline{y}_i > 1$.

Definition 5.2 (volume) *For some subset $\mathcal{F}' \subseteq \mathcal{F}$ of facilities, define the* volume *of $\mathcal{F}'$, denoted by $\mathsf{vol}(\mathcal{F}')$, to be the sum of $\overline{y}_i$ over all facilities $i \in \mathcal{F}'$. i.e, $\mathsf{vol}(\mathcal{F}') = \sum_{i\in\mathcal{F}'} \overline{y}_i$.*

Definition 5.3 (close and distant facilities) *For a client $j \in \mathcal{C}$, we say a facility i is one of its* close facilities *if $\overline{x}_{i,j} > 0$. If $\overline{x}_{i,j} = 0$, but $x_{i,j} > 0$, then we say i is a* distant facility *of client j. Let $\mathcal{F}_j^C$ and $\mathcal{F}_j^D$ be the set of close and distant facilities of j, respectively. Let $\mathcal{F}_j = \mathcal{F}_j^C \cup \mathcal{F}_j^D$.*

Note that if $\overline{x}_{i,j} > 0$, then $x_{i,j} > 0$, due to the greedy assignment of x and $\overline{x}$ variables.

Definition 5.4 *For a client $j \in \mathcal{C}$ and a subset $\mathcal{F}' \subseteq \mathcal{F}$ of facilities such that $\mathsf{vol}(\mathcal{F}') > 0$, define $d(j, \mathcal{F}')$ to be the* average distance *of j to facilities in $\mathcal{F}'$, with respect to the weights $\overline{y}$. Recalling that $\overline{y}$ is a scaled vector of y, the average distance is also with respect to the weights y. i.e,*

$$d(j, \mathcal{F}') = \frac{\sum_{i \in \mathcal{F}'} \overline{y}_i d(j,i)}{\sum_{i \in \mathcal{F}'} \overline{y}_i} = \frac{\sum_{i \in \mathcal{F}'} y_i d(j,i)}{\sum_{i \in \mathcal{F}'} y_i}.$$

Definition 5.5 ($d_{\mathsf{ave}}^C(j), d_{\mathsf{ave}}^D(j), d_{\mathsf{ave}}(j)$ and $d_{\max}^C(j)$) *For a client $j \in \mathcal{C}$, define $d_{\mathsf{ave}}^C(j), d_{\mathsf{ave}}^D(j)$ and $d_{\mathsf{ave}}(j)$ to be the average distance from j to $\mathcal{F}_j^C, \mathcal{F}_j^D$ and $\mathcal{F}_j$ respectively, i.e, $d_{\mathsf{ave}}^C(j) = d(j, \mathcal{F}_j^C), d_{\mathsf{ave}}^D(j) = d(j, \mathcal{F}_j^D)$ and $d_{\mathsf{ave}}(j) = d(j, \mathcal{F}_j)$. Define $d_{\max}^C(j)$ to be the maximum distance from j to a facility in $\mathcal{F}_j^C$.*

By definition, $d_{\mathsf{ave}}(j)$ is the connection cost of j in the optimal fractional solution. See Figure 5.1 for an illustration of the above 3 definitions. The following claims hold by the definitions of the corresponding quantities, and will be used repeatedly later :

Claim 5.6 *$d_{\mathsf{ave}}^C(j) \le d_{\max}^C(j) \le d_{\mathsf{ave}}^D(j)$ and $d_{\mathsf{ave}}^C(j) \le d_{\mathsf{ave}}(j) \le d_{\mathsf{ave}}^D(j)$, for every client $j \in \mathcal{C}$.*

Claim 5.7 *$d_{\mathsf{ave}}(j) = \frac{1}{\gamma} d_{\mathsf{ave}}^C(j) + \frac{\gamma - 1}{\gamma} d_{\mathsf{ave}}^D(j)$, for every client $j \in \mathcal{C}$.*

Claim 5.8 *$\mathsf{vol}(\mathcal{F}_j^C) = 1, \mathsf{vol}(\mathcal{F}_j^D) = \gamma - 1$ and $\mathsf{vol}(\mathcal{F}_j) = \gamma$, for every client $j \in \mathcal{C}$.*

Recall that y_i indicates if the facility i is open. If we are aiming at a (γ, γ)-bifactor approximation, we can open i with probability $\gamma y_i = \overline{y}_i$. Then, the expected opening cost is exactly γ times that of the optimal fractional solution. If the sets $\mathcal{F}_j^C, j \in \mathcal{C}$ are disjoint, the following simple algorithm gives a $(\gamma, 1)$-bifactor approximation. Open exactly 1 facility in $\mathcal{F}_j^C$, with $\overline{y}_i$ being the probability of opening i. (Recall that $\sum_{i \in \mathcal{F}_j^C} \overline{y}_i = \mathsf{vol}(\mathcal{F}_j^C) = 1$.) Connect each client j to its closest open facility. This is indeed a $(\gamma, 1)$-bifactor approximation, since the expected connection cost of j given by the algorithm is $d_{\mathsf{ave}}^C(j) \le d_{\mathsf{ave}}(j)$.

In general, the sets $\mathcal{F}_j^C, j \in \mathcal{C}$ may not be disjoint. In this case, we can not randomly select 1 open facility from every $\mathcal{F}_j^C$, since a facility i belonging to two different $\mathcal{F}_j^C$'s would be open with probability more than $\overline{y}_i$. To overcome this problem, we shall select a subset $\mathcal{C}' \subseteq \mathcal{C}$ of clients such that the sets $\mathcal{F}_j^C, j \in \mathcal{C}'$ are disjoint. We randomly open a facility in $\mathcal{F}_j^C$ only for facilities $j \in \mathcal{C}'$. To allow us to bound the

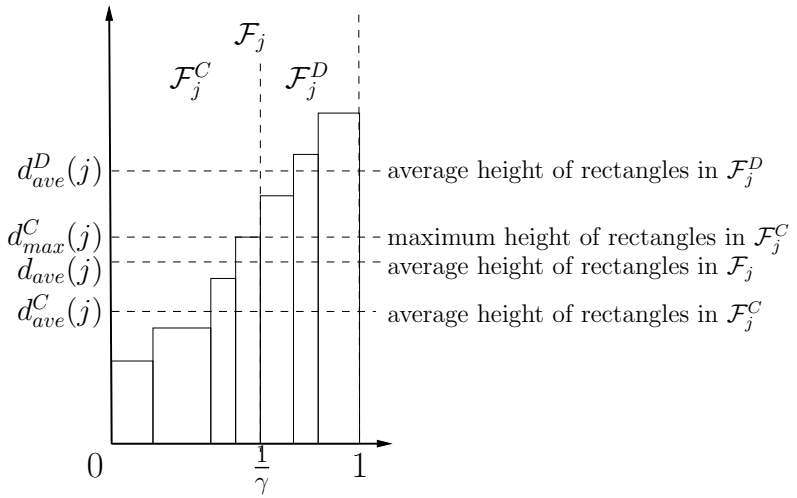

Figure 5.1: Illustration for $d^C_{\mathsf{ave}}(j), d^D_{\mathsf{ave}}(j), d_{\mathsf{ave}}(j), d^C_{\max}(j), \mathcal{F}_j, \mathcal{F}_j^C$ and $\mathcal{F}_j^D$. Each rectangle is a facility : its width corresponds to y value and its height corresponds to its distance to j. The heights of the rectangles are non-decreasing from the left to the right. The average heights shown in the figure are weighted with respect to the widths of rectangles.

expected connection cost of clients not in $\mathcal{C}'$, $\mathcal{C}'$ also has the property that every client $j \in \mathcal{C} \setminus \mathcal{C}'$ is close to some client in $\mathcal{C}'$. The exact definition of being close is given later in Claim 5.10.

The subset of clients $\mathcal{C}'$ is selected greedily in the following way. Initially let $\mathcal{C}'' = \mathcal{C}$ and $\mathcal{C}' = \emptyset$. While $\mathcal{C}''$ is not empty, select the client j in $\mathcal{C}''$ with the minimum $d^C_{\mathsf{ave}}(j) + d^C_{\max}(j)$, add j to $\mathcal{C}'$ and remove j and all clients j' satisfying $\mathcal{F}_j^C \cap \mathcal{F}_{j'}^C \neq \emptyset$ from $\mathcal{C}''$. The following two claims hold for $\mathcal{C}'$.

Claim 5.9 *$\mathcal{F}_j^C \cap \mathcal{F}_{j'}^C = \emptyset$, for two distinct clients $j, j' \in \mathcal{C}'$.*

As already mentioned, this claim allows us to randomly open 1 facility from each set $\mathcal{F}_j^C, j \in \mathcal{C}'$.

Claim 5.10 *For every client $j \notin \mathcal{C}'$, there exists a client $j' \in \mathcal{C}'$ such that $\mathsf{vol}(\mathcal{F}_j^C \cap \mathcal{F}_{j'}^C) > 0$ and $d^C_{\mathsf{ave}}(j') + d^C_{\max}(j') \leq d^C_{\mathsf{ave}}(j) + d^C_{\max}(j)$.*

Notice that $\mathsf{vol}(\mathcal{F}_j^C \cap \mathcal{F}_{j'}^C) > 0$ implies $d(j, j') \leq d^C_{\max}(j) + d^C_{\max}(j')$. This property and that $d^C_{\mathsf{ave}}(j') + d^C_{\max}(j') \leq d^C_{\mathsf{ave}}(j) + d^C_{\max}(j)$ will be used to bound the expected connection cost of j.

Definition 5.11 (cluster center) *For a client $j \notin \mathcal{C}'$, we select arbitrarily a client $j' \in \mathcal{C}$ that makes Claim 5.10 hold and call it the* cluster center *of j.*

We now randomly round the fractional solution $(\overline{x}, \overline{y})$. As we already mentioned, for each $j \in \mathcal{C}'$, we open exactly one of its close facilities randomly with probabilities $\overline{y}_i$. For each facility i that is not a close facility of any client in $\mathcal{C}'$, we open it

independently with probability $\overline{y}_i$. Connect each client j to its closest open facility and let C_j be its connection cost.

Since i is open with probability exactly $\overline{y}_i$ for every facility $i \in \mathcal{F}$, the expected opening cost of the solution given by the above algorithm is exactly γ times that of the optimal fractional solution. If $j \in \mathcal{C}'$ then $\mathbf{E}[C_j] = d^C_{\mathsf{ave}}(j) \leq d_{\mathsf{ave}}(j)$.

Byrka in [19] showed that for every client $j \notin \mathcal{C}'$,

1. The probability that some facility in $\mathcal{F}^C_j$ (resp., $\mathcal{F}^D_j$) is open is at least $1 - e^{-\mathsf{vol}(\mathcal{F}^C_j)} = 1 - e^{-1}$(resp., $1 - e^{-\mathsf{vol}(\mathcal{F}^D_j)} = 1 - e^{-(\gamma-1)}$), and under the condition that the event happens, the expected distance between j and the closest open facility in $\mathcal{F}^C_j$ (resp., $\mathcal{F}^D_j$) is at most $d^C_{\mathsf{ave}}(j)$ (resp., $d^D_{\mathsf{ave}}(j)$); [1]

2. $d(j, \mathcal{F}^C_{j'} \setminus \mathcal{F}_j) \leq d^D_{\mathsf{ave}}(j) + d^C_{\max}(j) + d^C_{\mathsf{ave}}(j)$(recall that $d(j, \mathcal{F}^C_{j'} \setminus \mathcal{F}_j)$ is the weighted average distance from j to facilities in $\mathcal{F}^C_{j'} \setminus \mathcal{F}_j$), where j' is the cluster center of j; or equivalently, under the condition that there is no open facility in $\mathcal{F}_j$, the expected distance between j and the unique open facility in $\mathcal{F}^C_{j'}$ is at most $d^D_{\mathsf{ave}}(j) + d^C_{\max}(j) + d^C_{\mathsf{ave}}(j)$.

Combining the above 2 facts, we have

$$\begin{aligned}
\mathbf{E}[C_j] &\leq (1 - e^{-1}) d^C_{\mathsf{ave}}(j) + e^{-1}(1 - e^{-(\gamma-1)}) d^D_{\mathsf{ave}}(j) \\
&\qquad + e^{-1} e^{-(\gamma-1)} (d^D_{\mathsf{ave}}(j) + d^C_{\max}(j) + d^C_{\mathsf{ave}}(j)) \\
&= (1 - e^{-1} + e^{-\gamma}) d^C_{\mathsf{ave}}(j) + e^{-1} d^D_{\mathsf{ave}}(j) + e^{-\gamma} d^C_{\max}(j) \\
&\leq (1 - e^{-1} + e^{-\gamma}) d^C_{\mathsf{ave}}(j) + (e^{-1} + e^{-\gamma}) d^D_{\mathsf{ave}}(j). \qquad (5.4)
\end{aligned}$$

Notice that the connection cost of j in the optimal fractional solution is $d_{\mathsf{ave}}(j) = \frac{1}{\gamma} d^C_{\mathsf{ave}}(j) + \frac{\gamma - 1}{\gamma} d^D_{\mathsf{ave}}(j)$. We compute the maximum ratio between $(1 - e^{-1} + e^{-\gamma}) d^C_{\mathsf{ave}}(j) + (e^{-1} + e^{-\gamma}) d^D_{\mathsf{ave}}(j)$ and $\frac{1}{\gamma} d^C_{\mathsf{ave}}(j) + \frac{\gamma - 1}{\gamma} d^D_{\mathsf{ave}}(j)$. Since $d^C_{\mathsf{ave}}(j) \leq d^D_{\mathsf{ave}}(j)$, the ratio is maximized when $d^C_{\mathsf{ave}}(j) = d^D_{\mathsf{ave}}(j) > 0$ or $d^D_{\mathsf{ave}}(j) > d^C_{\mathsf{ave}}(j) = 0$. For $\gamma \geq \gamma_0$, the maximum ratio is achieved when $d^C_{\mathsf{ave}}(j) = d^D_{\mathsf{ave}}(j) > 0$, in which case the maximum is $1 + 2e^{-\gamma}$. Thus, the algorithm $A_1(\gamma_0)$ gives a $(\gamma_0 \approx 1.67736, 1 + 2e^{-\gamma_0} \approx 1.37374)$-bifactor approximation.[2]

5.2 A 1.488 Approximation Algorithm for the UFL Problem

In this section, we give our 1.488-approximation algorithm for the UFL problem. Our algorithm is also based on the combination of $A_1(\gamma)$ and A_2. However, instead of

[1] There is a minor difference between what is explained here and what is in [19]: [19] used a bound on the probability that some facility in $\mathcal{F}_j$ (instead of $\mathcal{F}^D_j$) is open.

[2] Byrka's analysis in [19] was slightly different; it used some variables from the dual LP. Later, Byrka et al. [20] gave an analysis without using the dual LP, which is the one we explain in our paper.

running $A_1(\gamma)$ for a fixed γ, we randomly select γ from some distribution described later.

To understand why this approach can reduce the approximation ratio, we list some necessary conditions that the upper bound in (5.4) is tight.

1. The facilities in $\mathcal{F}_j$ have tiny weights. In other words, $\max_{i\in\mathcal{F}_j} \overline{y}_i$ tends to 0. Moreover, all these facilities were independently sampled in the algorithm. These conditions are necessary to tighten the $1-e^{-1}$ (resp., $1-e^{-(\gamma-1)}$) upper bound for the probability that at least 1 facility in $\mathcal{F}_j^C$ (resp., $\mathcal{F}_j^D$) is open.

2. The distances from j to all the facilities in $\mathcal{F}_j^C$ (resp., $\mathcal{F}_j^D$) are the same. Otherwise, the expected distance from j to the closest open facility in $\mathcal{F}_j^C$ (resp., $\mathcal{F}_j^D$), under the condition that it exists, is strictly smaller than $d^C_{\mathsf{ave}}(j)$ (resp., $d^D_{\mathsf{ave}}(j)$).

3. $d^C_{\max}(j) = d^D_{\mathsf{ave}}(j)$. This is also required since we used $d^D_{\mathsf{ave}}(j)$ as an upper bound of $d^C_{\max}(j)$ in (5.4).

To satisfy all the above conditions, the distances from j to $\mathcal{F}_j$ must be distributed as follows. $1/(\gamma+\epsilon)$ fraction of facilities in $\mathcal{F}_j$ (the fraction is with respect to the weights y_i) have distances a to j, and the other $1-1/(\gamma+\epsilon)$ fraction have distances $b \geq a$ to j. For ϵ tending to 0, $d^C_{\mathsf{ave}}(j) = a$ and $d^C_{\max}(j) = d^D_{\mathsf{ave}}(j) = b$.

As discussed earlier, if $a = b$, then $\mathbf{E}[C_j]/d_{\mathsf{ave}}(j) \leq 1 + 2e^{-\gamma}$. Intuitively, the bad cases (cases with large $\mathbf{E}[C_j]/d_{\mathsf{ave}}(j)$) should have $a \ll b$. However, if we replace γ with $\gamma + 1.01\epsilon$ (say), then $d^C_{\max}(j)$ will equal $d^C_{\mathsf{ave}}(j) = a$, instead of $d^D_{\mathsf{ave}}(j) = b$. Thus, we can greatly reduce the approximation ratio if the distributions of weights for all j's are of the above form.

Hence, using only two different γ's, we are already able to make an improvement. To give a better analysis, we first give in Section 5.2.1 an upper bound on $\mathbf{E}[C_j]$, in terms of the distribution of distances from j to $\mathcal{F}_j$, not just $d^C_{\mathsf{ave}}(j)$ and $d^D_{\mathsf{ave}}(j)$, and then give in Section 5.2.2 an explicit distribution for γ by introducing a 0-sum game.

5.2.1 Upper-Bounding the Expected Connection Cost of a Client

We bound $\mathbf{E}[C_j]$ in this subsection. It suffices to assume $j \notin \mathcal{C}'$, since we can think of a client $j \in \mathcal{C}'$ as a client $j \notin \mathcal{C}'$ which has a co-located client $j' \in \mathcal{C}'$. Similar to [19], we first give an upper bound on $d(j, \mathcal{F}^C_{j'} \setminus \mathcal{F}_j)$ in Lemma 5.12, where j' is the cluster center of j. The bound and the proof are the same as the counterparts in [19], except that we made a slight improvement. The improvement is not essential to the final approximation ratio; however, it will simplify the analytical proof in Section 5.2.2.

Lemma 5.12 *For some client $j \notin \mathcal{C}'$, let j' be the cluster center of j. So $j' \in \mathcal{C}'$ and $\mathsf{vol}(\mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) > 0$. Assuming $\mathsf{vol}(\mathcal{F}^C_{j'} \setminus \mathcal{F}_j) > 0$, we have,*

$$d(j, \mathcal{F}^C_{j'} \setminus \mathcal{F}_j) \leq (2-\gamma) d^C_{\max}(j) + (\gamma-1) d^D_{\mathsf{ave}}(j) + d^C_{\max}(j') + d^C_{\mathsf{ave}}(j'). \tag{5.5}$$

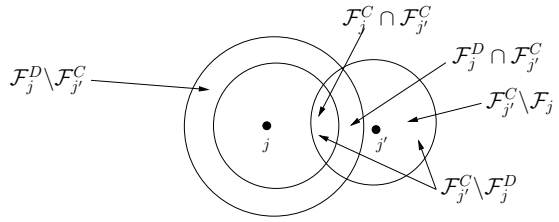

Figure 5.2: Sets of facilities used in the proof of Lemma 5.12.

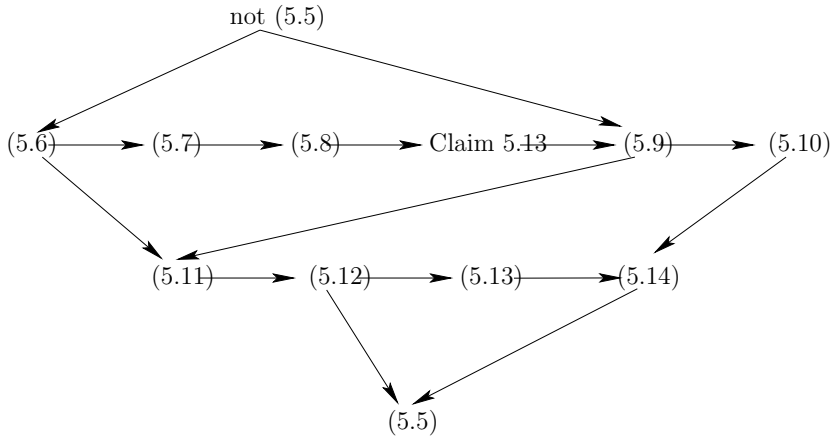

Figure 5.3: The dependence graph for the equations in the proof of Lemma 5.12. An equation is implied by its predecessor-equations.

Proof: Figure 5.2 illustrates the sets of facilities we are going to use. For the flow of the proof, we shall first assume that all the 5 marked sets in the figure have positive volumes so that the average distances from j or j' to them are well-defined. Later we deal with the other cases. Figure 5.3 shows the dependence of equations we shall prove and can be viewed as the outline of the proof.

If $d(j,j') \leq (2-\gamma)d^C_{\max}(j) + (\gamma-1)d^D_{\mathsf{ave}}(j) + d^C_{\mathsf{ave}}(j')$, the remaining $d^C_{\max}(j')$ is enough for the distance between j' and any facility in $\mathcal{F}^C_{j'}$. So, we will assume

$$d(j,j') > (2-\gamma)d^C_{\max}(j) + (\gamma-1)d^D_{\mathsf{ave}}(j) + d^C_{\mathsf{ave}}(j'). \tag{5.6}$$

Since $d^D_{\mathsf{ave}}(j) \geq d^C_{\max}(j)$ and $\gamma - 1 > 0$, (5.6) implies

$$d(j,j') > d^C_{\max}(j) + d^C_{\mathsf{ave}}(j'). \tag{5.7}$$

By triangle inequality,

$$d(j', \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) \geq d(j,j') - d(j, \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}),$$

and by (5.7) and $d(j, \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) \leq d^C_{\max}(j)$,

$$d(j', \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) > d^C_{\max}(j) + d^C_{\mathsf{ave}}(j') - d^C_{\max}(j) = d^C_{\mathsf{ave}}(j'). \tag{5.8}$$

Claim 5.13 *If* $d(j', \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) \geq d_{\mathsf{ave}}^C(j')$, *then* (5.5) *holds.*

Proof: Notice that $d_{\mathsf{ave}}^C(j') = d(j', \mathcal{F}_{j'}^C)$ and $\mathcal{F}_{j'}^C$ is the union of the following 3 disjoint sets: $\mathcal{F}_j^C \cap \mathcal{F}_{j'}^C, \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C, \mathcal{F}_{j'}^C \setminus \mathcal{F}_j$. If $d(j', \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) \geq d_{\mathsf{ave}}^C(j')$, then by (5.8), we have $d(j', \mathcal{F}_{j'}^C \setminus \mathcal{F}_j) < d_{\mathsf{ave}}^C(j')$. Then, by triangle inequality,

$$d(j, \mathcal{F}_{j'}^C \setminus \mathcal{F}_j) \leq d(j, j') + d(j', \mathcal{F}_{j'}^C \setminus \mathcal{F}_j) < d(j, j') + d_{\mathsf{ave}}^C(j').$$

Since $\mathsf{vol}(\mathcal{F}_j^C \cap \mathcal{F}_{j'}^C) > 0$, we have $d(j, j') \leq d_{\max}^C(j) + d_{\max}^C(j')$. Since $d_{\max}^C(j) \leq d_{\mathsf{ave}}^D(j)$, we have

$$\begin{aligned} d(j, \mathcal{F}_{j'}^C \setminus \mathcal{F}_j) &\leq d_{\max}^C(j) + d_{\max}^C(j') + d_{\mathsf{ave}}^C(j') \\ &\leq (2-\gamma) d_{\max}^C(j) + (\gamma - 1) d_{\mathsf{ave}}^D(j) + d_{\max}^C(j') + d_{\mathsf{ave}}^C(j'). \end{aligned}$$

This proves the claim. □

So, by Claim 5.13, we can assume that

$$d(j', \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) = d_{\mathsf{ave}}^C(j') - z \tag{5.9}$$

for some positive z. Let $\hat{y} = \mathsf{vol}(\mathcal{F}_j^D \cap \mathcal{F}_{j'}^C)$. Notice that $0 < \hat{y} < \min\{\gamma - 1, 1\}$, since we assumed that all the 5 sets marked in Figure 5.2 have positive volumes. Since $d_{\mathsf{ave}}^C(j') = \hat{y} d(j', \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) + (1 - \hat{y}) d(j', \mathcal{F}_{j'}^C \setminus \mathcal{F}_j^D)$, we have

$$d(j', \mathcal{F}_{j'}^C \setminus \mathcal{F}_j^D) = d_{\mathsf{ave}}^C(j') + \frac{\hat{y}}{1 - \hat{y}} z. \tag{5.10}$$

By (5.6), (5.9) and triangle inequality, we have

$$\begin{aligned} d(j, \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) &\geq d(j, j') - d(j', \mathcal{F}_{j'}^C \cap \mathcal{F}_j^D) \\ &> (2-\gamma) d_{\max}^C(j) + (\gamma - 1) d_{\mathsf{ave}}^D(j) + d_{\mathsf{ave}}^C(j') - (d_{\mathsf{ave}}^C(j') - z) \\ &= d_{\mathsf{ave}}^D(j) - (2-\gamma)(d_{\mathsf{ave}}^D(j) - d_{\max}^C(j)) + z. \end{aligned} \tag{5.11}$$

Noticing that $d_{\mathsf{ave}}^D(j) = \dfrac{\hat{y}}{\gamma - 1} d(j, \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) + \dfrac{\gamma - 1 - \hat{y}}{\gamma - 1} d(j, \mathcal{F}_j^D \setminus \mathcal{F}_{j'}^C)$, we have

$$d(j, \mathcal{F}_j^D \setminus \mathcal{F}_{j'}^C) = d_{\mathsf{ave}}^D(j) - \frac{\hat{y}}{\gamma - 1 - \hat{y}} \left(d(j, \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) - d_{\mathsf{ave}}^D(j) \right).$$

Then, by $d_{\max}^C(j) \leq d(j, \mathcal{F}_j^D \setminus \mathcal{F}_{j'}^C)$ and (5.11),

$$d_{\max}^C(j) \leq d(j, \mathcal{F}_j^D \setminus \mathcal{F}_{j'}^C) < d_{\mathsf{ave}}^D(j) - \frac{\hat{y}}{\gamma - 1 - \hat{y}} \left(z - (2-\gamma)(d_{\mathsf{ave}}^D(j) - d_{\max}^C(j)) \right).$$

So,

$$d_{\mathsf{ave}}^D(j) - d_{\max}^C(j) > \frac{\hat{y}}{\gamma - 1 - \hat{y}} \left(z - (2-\gamma)(d_{\mathsf{ave}}^D(j) - d_{\max}^C(j)) \right),$$

and since $1 + \dfrac{(2-\gamma)\hat{y}}{\gamma - 1 - \hat{y}} = \dfrac{(\gamma-1)(1-\hat{y})}{\gamma-1-\hat{y}} \geq 0$,

$$d^D_{\mathsf{ave}}(j) - d^C_{\max}(j) > \frac{\hat{y}}{\gamma-1-\hat{y}} z \Big/ \left(1 + \frac{(2-\gamma)\hat{y}}{\gamma-1-\hat{y}}\right) = \frac{\hat{y}}{(\gamma-1)(1-\hat{y})} z. \tag{5.12}$$

By triangle inequality,

$$d(j', \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) \geq d(j, j') - d(j, \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}),$$

and by (5.6) and $d(j, \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) \leq d^C_{\max}(j)$,

$$\begin{aligned} d(j', \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) &> (2-\gamma) d^C_{\max}(j) + (\gamma-1) d^D_{\mathsf{ave}}(j) + d^C_{\mathsf{ave}}(j') - d^C_{\max}(j) \\ &= (\gamma-1)(d^D_{\mathsf{ave}}(j) - d^C_{\max}(j)) + d^C_{\mathsf{ave}}(j'). \end{aligned}$$

Then by (5.12), we have

$$d(j', \mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) > \frac{\hat{y}}{1-\hat{y}} z + d^C_{\mathsf{ave}}(j'). \tag{5.13}$$

Notice that $\mathcal{F}^C_{j'} \setminus \mathcal{F}^D_j$ is the union of the following two sets : $\mathcal{F}^C_j \cap \mathcal{F}^C_{j'}$ and $\mathcal{F}^C_{j'} \setminus \mathcal{F}_j$. Combining (5.10) and (5.13), we have

$$d(j', \mathcal{F}^C_{j'} \setminus \mathcal{F}_j) < d^C_{\mathsf{ave}}(j') + \frac{\hat{y}}{1-\hat{y}} z. \tag{5.14}$$

So,

$$\begin{aligned} d(j, \mathcal{F}^C_{j'} \setminus \mathcal{F}_j) &\leq d^C_{\max}(j) + d^C_{\max}(j') + d(j', \mathcal{F}^C_{j'} \setminus \mathcal{F}_j) \\ &= (2-\gamma) d^C_{\max}(j) + (\gamma-1) d^C_{\max}(j) + d^C_{\max}(j') + d(j', \mathcal{F}^C_{j'} \setminus \mathcal{F}_j) \end{aligned}$$

by (5.12) and (5.14),

$$\begin{aligned} &< (2-\gamma) d^C_{\max}(j) + (\gamma-1) \left(d^D_{\mathsf{ave}}(j) - \frac{\hat{y} z}{(\gamma-1)(1-\hat{y})} \right) \\ &\quad + d^C_{\max}(j') + d^C_{\mathsf{ave}}(j') + \frac{\hat{y} z}{1-\hat{y}} \\ &= (2-\gamma) d^C_{\max}(j) + (\gamma-1) d^D_{\mathsf{ave}}(j) + d^C_{\max}(j') + d^C_{\mathsf{ave}}(j'). \end{aligned}$$

This finishes the proof of Lemma 5.12 for the case where all the 5 sets marked in Figure 5.2 have positive volumes. By the conditions of the lemma, we have $\mathsf{vol}(\mathcal{F}^C_j \cap \mathcal{F}^C_{j'}) > 0$ and $\mathsf{vol}(\mathcal{F}^C_{j'} \setminus \mathcal{F}^D_j) \geq \mathsf{vol}(\mathcal{F}^C_{j'} \setminus \mathcal{F}_j) > 0$. Thus, it remains to consider the case where $\mathsf{vol}(\mathcal{F}^D_j \setminus \mathcal{F}^C_{j'}) = 0$ or $\mathsf{vol}(\mathcal{F}^D_j \cap \mathcal{F}^C_{j'}) = 0$. If $\mathsf{vol}(\mathcal{F}^D_j \cap \mathcal{F}^C_{j'}) = 0$, then the

proof of Claim 5.13 can be repeated to yield Inequality (5.5). If $\mathsf{vol}(\mathcal{F}_j^D \setminus \mathcal{F}_{j'}^C) = 0$, then $\hat{y} = \gamma - 1$ and $d(j, \mathcal{F}_j^D \cap \mathcal{F}_{j'}^C) = d_{\mathsf{ave}}^D(j)$. Thus, Inequality (5.11) implies $d_{\mathsf{ave}}^D(j) - d_{\max}^C(j) < \frac{z}{2-\gamma} = \frac{\hat{y}z}{(\gamma-1)(1-\hat{y})}$, which is exactly Inequality (5.12). □

Lemma 5.14

$$d(j, \mathcal{F}_{j'}^C \setminus \mathcal{F}_j) \leq \gamma d_{\mathsf{ave}}(j) + (3-\gamma) d_{\max}^C(j). \tag{5.15}$$

Proof: Noticing that $d_{\max}^C(j') + d_{\mathsf{ave}}^C(j') \leq d_{\max}^C(j) + d_{\mathsf{ave}}^C(j)$, the proof is straightforward from Lemma 5.12.

$$\begin{aligned} d(j, \mathcal{F}_{j'}^C \setminus \mathcal{F}_j) &\leq (2-\gamma) d_{\max}^C(j) + (\gamma-1) d_{\mathsf{ave}}^D(j) + d_{\max}^C(j') + d_{\mathsf{ave}}^C(j') \\ &\leq (2-\gamma) d_{\max}^C(j) + (\gamma-1) d_{\mathsf{ave}}^D(j) + d_{\max}^C(j) + d_{\mathsf{ave}}^C(j) \\ &= \gamma \left(\frac{1}{\gamma} d_{\mathsf{ave}}^C(j) + \frac{\gamma-1}{\gamma} d_{\mathsf{ave}}^D(j) \right) + (3-\gamma) d_{\max}^C(j) \\ &= \gamma d_{\mathsf{ave}}(j) + (3-\gamma) d_{\max}^C(j). \end{aligned}$$

□

Definition 5.15 (characteristic function) *Given a* UFL *instance and its optimal fractional solution* (x, y)*, the* characteristic function $h_j : (0, 1] \to \mathbb{R}$ *of some client* $j \in \mathcal{C}$ *is defined as follows. Let* $i_1, i_2, \cdots, i_k$ *be the facilities in* $\mathcal{F}_j$*, in the non-decreasing order of distances to* j*. Then* $h_j(p) = d(i_t, j)$*, where* t *is the minimum number such that* $\sum_{s=1}^{t} y_{i_s} \geq p$*. The* characteristic function *of the instance is defined as* $h = \sum_{j \in \mathcal{C}} h_j$*.*

Notice that $h_j, j \in \mathcal{C}$ and h are defined using the y vector, not the $\overline{y}$ vector, and is thus independent of γ.

Claim 5.16 $h : (0, 1] \to \mathbb{R}$ *is a non-decreasing piece-wise constant function. That is, there exists points* $0 < p_1 < p_2 < \cdots < p_m = 1$ *and values* $0 \leq c_1 < c_2 < \cdots < c_m$ *such that* $h(p) = c_t$ *where* t *is the minimum integer such that* $p \leq p_t$*.*

The above claim will be used in the proof of Lemma 5.23. (Although the monotonicity of h is enough to prove Lemma 5.23, an argument for such general statement requires the use of measure theory. Using the piece-wise-constant property will simplify the proof.)

Claim 5.17

$$d_{\mathsf{ave}}(j) = \int_0^1 h_j(p) \mathbf{d}p, \qquad d_{\max}^C(j) = h_j(1/\gamma).$$

This claim together with Lemma 5.14 implies

$$d(j, \mathcal{F}_{j'}^C \setminus \mathcal{F}_j) \leq \gamma \int_0^1 h_j(p) \mathbf{d}p + (3-\gamma) h_j\left(\frac{1}{\gamma}\right). \tag{5.16}$$

Lemma 5.18 *For any client j,*

$$\mathbf{E}[C_j] \leq \int_0^1 h_j(p)e^{-\gamma p}\gamma \mathbf{d}p + e^{-\gamma}\left(\gamma \int_0^1 h_j(p)\mathbf{d}p + (3-\gamma)h_j\left(\frac{1}{\gamma}\right)\right). \tag{5.17}$$

Proof:

Let $j' \in \mathcal{C}'$ be the cluster center of j. We connect j to the closest open facility in $\mathcal{F}_j \cup \mathcal{F}_{j'}^C$. The proof is outlined as follows: we first prove the lemma for a special case; then we show that any general case can be converted to the special case by a sequence of operations which can only make the instance worse.

We first consider the special case where all the facilities in $\mathcal{F}_j$ have infinitely small $\overline{y}$ values (say, $\overline{y}_i = \epsilon$ and ϵ tends to 0) and they were independently sampled in the algorithm. Let $i_1, i_2, \cdots, i_M$ be the facilities in $\mathcal{F}_j$, in the order of increasing distances to j. Notice that $M\epsilon = \gamma$. Then, the probability we connect j to i_t is $(1-\epsilon)^{t-1}\epsilon$. Under the condition that no facilities in $\mathcal{F}_j$ is open, the expected connection cost of j is at most $D = \gamma \int_0^1 h_j(p)\mathbf{d}p + (3-\gamma)h_j\left(\frac{1}{\gamma}\right)$ by (5.16).

$$\begin{aligned}\mathbf{E}[C_j] &\leq \sum_{t=1}^{M} \epsilon(1-\epsilon)^{t-1} d(j,i_t) + (1-\epsilon)^M D \\ &= \sum_{t=1}^{M} \frac{\gamma}{M}\left(1-\frac{\gamma}{M}\right)^{t-1} h_j(t/M) + \left(1-\frac{\gamma}{M}\right)^M D.\end{aligned}$$

If we let ϵ tend to 0 (i.e, let M tend to ∞), then the upper bound becomes

$$\begin{aligned}&\int_0^1 \gamma e^{-\gamma p} h_j(p)\mathbf{d}p + e^{-\gamma} D \\ &= \int_0^1 h_j(p)e^{-\gamma p}\gamma \mathbf{d}p + e^{-\gamma}\left(\gamma \int_0^1 h_j(p)\mathbf{d}p + (3-\gamma)h_j\left(\frac{1}{\gamma}\right)\right).\end{aligned}$$

Now, we deal with a general case. We only handle the case $\mathsf{vol}(\mathcal{F}_{j'}^C \setminus \mathcal{F}_j) > 0$. Handling the case $\mathsf{vol}(\mathcal{F}_{j'}^C \setminus \mathcal{F}_j) = 0$ only requires a slight modification to the proof. Given a partition $\mathcal{P}$ of $\mathcal{F}_j \cup \mathcal{F}_{j'}^C$ into sets of volumes at most 1, we define a distribution $g_{\mathcal{P}}$ of sets of opening facilities in $\mathcal{F}_j \cup \mathcal{F}_{j'}^C$ using the following process: for every $\mathcal{F}' \in \mathcal{P}$, open 1 facility in $\mathcal{F}'$ with probability $\mathsf{vol}(\mathcal{F}')$ and open no facility with probability $1 - \mathsf{vol}(\mathcal{F}')$; the probability of opening $i \in \mathcal{F}'$ is $\overline{y}_i$; the sets in $\mathcal{P}$ are handled independently. Let $W_{\mathcal{P}}$ be the expected distance between j and its closest open facility in $\mathcal{F}_j \cup \mathcal{F}_{j'}^C$($\infty$ if no facility is open in $\mathcal{F}_j \cup \mathcal{F}_{j'}^C$), according to the distribution $g_{\mathcal{P}}$.

The distribution of sets of opening facilities in $\mathcal{F}_j \cup \mathcal{F}_{j'}^C$ in $A_1(\gamma)$ is $g_{\tilde{\mathcal{P}}}$, where $\tilde{\mathcal{P}}$ is defined as follows. For every $j'' \in \mathcal{C}'$ such that $\left(\mathcal{F}_j \cup \mathcal{F}_{j'}^C\right) \cap \mathcal{F}_{j''}^C \neq \emptyset$, $\left(\mathcal{F}_j \cup \mathcal{F}_{j'}^C\right) \cap \mathcal{F}_{j''}^C \in \tilde{\mathcal{P}}$. For every facility $i \in \mathcal{F}_j \cup \mathcal{F}_{j'}^C$ that is not inside any $\mathcal{F}_{j''}^C, j'' \in \mathcal{C}'$, we have $\{i\} \in \tilde{\mathcal{P}}$. Since the sets $\mathcal{F}_{j''}^C, j'' \in \mathcal{C}'$ are disjoint, $\tilde{\mathcal{P}}$ is a partition. Moreover, $\mathcal{F}_{j'}^C \in \tilde{\mathcal{P}}$. Since

$\mathsf{vol}(\mathcal{F}_{j'}^C) = 1$, there is always an open facility in the distribution $g_{\tilde{\mathcal{P}}}$. Notice that $\mathbf{E}[C_j] \leq W_{\tilde{\mathcal{P}}}$.

Claim 5.19 *If some subset $\mathcal{F}' \subseteq \mathcal{F}_j \setminus \mathcal{F}_{j'}^C, \mathcal{F}' \in \mathcal{P}$ has $|\mathcal{F}'| \geq 2$, removing $\mathcal{F}'$ from $\mathcal{P}$ and adding $|\mathcal{F}'|$ singular sets, each containing one facility in $\mathcal{F}'$, to $\mathcal{P}$ can only increase $W_{\mathcal{P}}$.*

Proof:

For the sake of description, we only consider the case where $|\mathcal{F}'| = 2$. The proof can be easily extended to the case where $|\mathcal{F}'| \geq 3$.

Assume $\mathcal{F}' = \{i, i'\}$, where i and i' are two distinct facilities in $\mathcal{F}_j \setminus \mathcal{F}_{j'}^C$ and $d(j,i) \leq d(j,i')$. Focus on the two distributions of the distance D between j and its closest open facility in $\{i, i'\}$ ($D = \infty$ if it does not exist), with respect to the old $g_{\mathcal{P}}$ and the new $g_{\mathcal{P}}$.

For the old $g_{\mathcal{P}}$, the distribution is: with probability $\overline{y}_i$, $D = d(j,i)$, with probability $\overline{y}_{i'}$, $D = d(j,i')$, and with probability $1 - \overline{y}_i - \overline{y}_{i'}$, $D = \infty$. For the new $g_{\mathcal{P}}$, the distribution is: with probability $\overline{y}_i$, $D = d(j,i)$, with probability $(1 - \overline{y}_i)\overline{y}_{i'}$, $D = d(j,i')$ and with the probability $(1 - \overline{y}_i)(1 - \overline{y}_{i'})$, $D = \infty$. So, the former distribution strictly dominates the latter one. Thus, splitting $\{i, i'\}$ can only increase $W_{\mathcal{P}}$. □

After performing the splitting operation for each set $\mathcal{F}' \in \tilde{\mathcal{P}}$ with $|\mathcal{F}'| \geq 2$, we obtain a new partition $\tilde{\mathcal{P}}' = \left\{\mathcal{F}_{j'}^C\right\} \bigcup \left\{\{i\} : i \in \mathcal{F}_j \setminus \mathcal{F}_{j'}^C\right\}$. By Lemma 5.19, $W_{\tilde{\mathcal{P}}} \leq W_{\tilde{\mathcal{P}}'}$. Let $\mathcal{P}'$ be the new partition obtained by performing the following sequence of operations to $\tilde{\mathcal{P}}'$.

1. Split the set $\mathcal{F}_{j'}^C \in \tilde{\mathcal{P}}'$ into two subsets: $\mathcal{F}_{j'}^C \cap \mathcal{F}_j$, and $\mathcal{F}_{j'}^C \setminus \mathcal{F}_j$;

2. Scale up $\bar{y}$ values in $\mathcal{F}_{j'}^C \setminus \mathcal{F}_j$ so that the volume of $\mathcal{F}_{j'}^C \setminus \mathcal{F}_j$ becomes 1. (Notice that the distributions $g_{\tilde{\mathcal{P}}'}$ and $g_{\mathcal{P}'}$ also depend on the vector $\overline{y}$. For notational purpose, we omitted the subscript $\overline{y}$. This operation means that $g_{\tilde{\mathcal{P}}'}$ is with respect to the old $\overline{y}$, while $g_{\mathcal{P}'}$ is with respect to the new $\overline{y}$.)

We show that $W_{\mathcal{P}'} = W_{\tilde{\mathcal{P}}'}$. Consider the two distributions, with respect to $g_{\tilde{\mathcal{P}}'}$ and $g_{\mathcal{P}'}$, of the distance between j and its closest open facility in $\mathcal{F}_{j'}^C$. The two distributions are actually the same, since $d_{\max}(j, \mathcal{F}_{j'}^C \cap \mathcal{F}_j) \leq d_{\min}(j, \mathcal{F}_{j'}^C \setminus \mathcal{F}_j)$, where $d_{\max}$ and $d_{\min}$ denotes the maximum and the minimum distance from a client to a set of facilities, respectively. To be more specific, both distributions are generated by the following process: with probability $\mathsf{vol}(\mathcal{F}_{j'}^C \cap \mathcal{F}_j)$, output the distance between j and a random facility (with respect to the weights $\overline{y}$) in $\mathcal{F}_{j'}^C \cap \mathcal{F}_j$; with the remaining $1 - \mathsf{vol}(\mathcal{F}_{j'}^C \cap \mathcal{F}_j)$ probability, output the distance between j and a random facility in $\mathcal{F}_{j'}^C \setminus \mathcal{F}_j$.

Again, by Claim 5.19, we can split $\mathcal{F}_{j'}^C \cap \mathcal{F}_j$ into singular sets. By the same argument as in the proof of Claim 5.19, splitting a facility $i \in \mathcal{F}_j$ into 2 facilities i' and i'' with $\overline{y}_i = \overline{y}_{i'} + \overline{y}_{i''}$ can only increase $W_{\mathcal{P}}$. Now, we get from $g_{\mathcal{P}'}$ a distribution $g_{\mathcal{P}}$ where, facilities in $\mathcal{F}_j$ are independently sampled, each facility $i \in \mathcal{F}_j$ has $\overline{y}_i = \epsilon$ with $\epsilon \to 0$, and we open exactly 1 facility in $\mathcal{F}_{j'}^C \setminus \mathcal{F}_j$. This is exactly the special case defined at the beginning of the proof. Thus, (5.17) holds. □

Lemma 5.20 *The expected connection cost of the integral solution given by $A_1(\gamma)$ is*

$$\mathbf{E}[C] \leq \int_0^1 h(p)e^{-\gamma p}\gamma \mathbf{d}p + e^{-\gamma}\left(\gamma \int_0^1 h(p)\mathbf{d}p + (3-\gamma)h\left(\frac{1}{\gamma}\right)\right). \tag{5.18}$$

Proof: Summing up (5.17) over all clients j will give us the lemma. □

5.2.2 An Explicit Distribution for γ

In this subsection, we give an explicit distribution for γ by introducing a 0-sum game.

Definition 5.21 *Let $h : (0,1] \to \mathbb{R}$ be the characteristic function of some* UFL *instance and $\gamma > 1$. Define*

$$\alpha(\gamma, h) = \int_0^1 h(p)e^{-\gamma p}\gamma \mathbf{d}p + e^{-\gamma}\left(\gamma \int_0^1 h(p)\mathbf{d}p + (3-\gamma)h\left(\frac{1}{\gamma}\right)\right). \tag{5.19}$$

By Lemma 5.20, $\alpha(\gamma, h)$ is an upper bound for the connection cost of the solution given by $A_1(\gamma)$ when the characteristic function of the input instance is h.

We can scale the distances as well as the opening costs of the input instance so that $\int_0^1 h(p)\mathbf{d}p = 1$. Then,

$$\alpha(\gamma, h) = \int_0^1 h(p)e^{-\gamma p}\gamma \mathbf{d}p + e^{-\gamma}\left(\gamma + (3-\gamma)h\left(\frac{1}{\gamma}\right)\right). \tag{5.20}$$

We consider a 0-sum game between an algorithm designer A and an adversary B. The strategy of A is a pair (μ, θ), where $0 \leq \theta \leq 1$ and μ is $1-\theta$ times a probability density function for γ. i.e,

$$\theta + \int_1^\infty \mu(\gamma)\mathbf{d}\gamma = 1. \tag{5.21}$$

The pair (μ, θ) corresponds to running A_2 with probability θ and running $A_1(\gamma)$ with probability $1-\theta$, for γ randomly selected according to the density function $\mu/(1-\theta)$. The strategy for B is a non-decreasing piece-wise constant function $h : (0,1] \to \mathbb{R}$ such that $\int_0^1 h(p)\mathbf{d}p = 1$.

Definition 5.22 *The value of the game, when A plays (μ, θ) and B plays h, is defined as*

$$\nu(\mu, \theta, h) = \max\left\{\int_1^\infty \gamma\mu(\gamma)\mathbf{d}\gamma + 1.11\theta, \int_1^\infty \alpha(\gamma, h)\mu(\gamma)\mathbf{d}\gamma + 1.78\theta\right\}. \tag{5.22}$$

Let $h_q : (0,1] \to \mathbb{R}, 0 \le q < 1$ be a threshold function defined as follows :

$$h_q(p) = \begin{cases} 0 & p \le q \\ \frac{1}{1-q} & p > q \end{cases}. \tag{5.23}$$

Lemma 5.23 *For a fixed strategy (θ, μ) for A, there is a best response of B that is a threshold function h_q.*

Proof: Let $h^* : (0,1] \to \mathbb{R}$ be a best response of B. Notice that h^* is a piece-wise constant function. Let $0 < p_1 < p_2 < \cdots < p_m = 1, 0 \le c_1 < c_2 < \cdots < c_m$ be the values that makes Claim 5.16 true for h^*. Then, (assuming $c_0 = p_0 = 0$)

$$h^* = \sum_{i=0}^{m-1} (c_{i+1} - c_i)(1 - p_i) h_{p_i}, \tag{5.24}$$

and

$$\int_1^\infty \alpha(\gamma, h^*)\mu(\gamma)\mathbf{d}\gamma = \int_1^\infty \alpha\left(\gamma, \sum_{i=0}^{m-1}(c_{i+1} - c_i)(1 - p_i)h_{p_i}\right)\mu(\gamma)\mathbf{d}\gamma$$

by the linearity of α

$$\begin{aligned} &= \int_1^\infty \sum_{i=0}^{m-1}(c_{i+1} - c_i)(1 - p_i)\alpha\left(\gamma, h_{p_i}\right)\mu(\gamma)\mathbf{d}\gamma \\ &= \sum_{i=0}^{m-1}(c_{i+1} - c_i)(1 - p_i)\int_1^\infty \alpha\left(\gamma, h_{p_i}\right)\mu(\gamma)\mathbf{d}\gamma. \end{aligned}$$

Since $\sum_{i=0}^{m-1}(c_{i+1} - c_i)(1 - p_i) = 1$, for some $q = p_i, 0 \le i \le m-1$,

$$\int_1^\infty \alpha(\gamma, h_q)\mu(\gamma)\mathbf{d}\gamma \ge \int_1^\infty \alpha(\gamma, h^*)\mu(\gamma)\mathbf{d}\gamma.$$

Thus,

$$\begin{aligned} \nu(\mu, \theta, h_q) &= \max\left\{\int_1^\infty \gamma\mu(\gamma)\mathbf{d}\gamma + 1.11\theta, \int_1^\infty \alpha(\gamma, h_q)\mu(\gamma)\mathbf{d}\gamma + 1.78\theta\right\} \\ &\ge \max\left\{\int_1^\infty \gamma\mu(\gamma)\mathbf{d}\gamma + 1.11\theta, \int_1^\infty \alpha(\gamma, h^*)\mu(\gamma)\mathbf{d}\gamma + 1.78\theta\right\} \\ &= \nu(\mu, \theta, h^*). \end{aligned} \tag{5.25}$$

This finishes the proof. □

Now, our goal becomes finding a strategy (θ, μ) for A such that $\sup_{q\in[0,1)} \nu(\mu, \theta, h_q)$ is minimized. With the help of a computer program, we obtain a strategy for A. We first restrict the support of μ to (1,3). Then, we discretize the domain (1,3) into $2n$

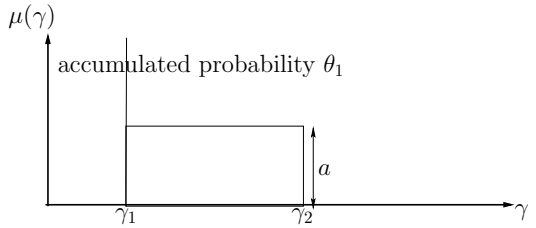

Figure 5.4: The distribution of γ. With probability θ_1, we run $A_1(\gamma_1)$; with probability $a(\gamma_2 - \gamma_1)$, we run $A_1(\gamma)$ with γ randomly selected from $[\gamma_1, \gamma_2]$; with probability $\theta_2 = 1 - \theta_1 - a(\gamma_2 - \gamma_1)$, we run A_2.

small intervals divided by points $\{r_i = 1 + i/n : 0 \le i \le 2n\}$. The value of the game is approximated by the following LP.

$$\min \quad \beta \qquad \text{s.t}$$

$$\frac{1}{2n}\sum_{i=1}^{2n} x_i + \theta = 1 \tag{5.26}$$

$$\frac{1}{2n}\sum_{i=1}^{2n} \frac{r_{i-1}+r_i}{2} x_i + 1.11\theta \le \beta \tag{5.27}$$

$$\frac{1}{2n}\sum_{i=1}^{2n} \alpha\left(\frac{r_{i-1}+r_i}{2}, h_q\right) x_i + 1.78\theta \le \beta \quad \forall q \in [0,1) \tag{5.28}$$

$$x_1, x_2, \cdots, x_{2n}, \theta \ge 0$$

In the above LP, x_i is $2n$ times the probability that we run $A_1\left(\frac{r_{i-1}+r_i}{2}\right)$, θ is the probability that we run A_2 and β is approximation ratio we can get by using the strategy specified by $x_i, 1 \le i \le 2n$ and θ. Equation (5.26) requires that with probability 1 we run either A_1 or A_2. Inequality (5.27) and (5.28) together say that the value of the game is at most β, no matter what B plays: Inequality (5.27) bounds the scaling factor of the facility cost, while Inequality (5.28) bounds the factor of the connection cost.

We solve the above LP for $n = 500$ using Matlab. Since we can only handle a finite number of constraints using Matlab, we only require that the constraints (5.28) hold for $q = i/n, i = 0, 1, \cdots, n-1$. The value of the LP is at most 1.4879 and the correspondent strategy (μ, θ) for A is roughly the following. With probability $\theta \approx 0.2$, run A_2; with probability about 0.5, run $A_1(\gamma_1)$ for $\gamma_1 \approx 1.5$; with the remaining 0.3 probability, run $A_1(\gamma)$ for γ selected from $(\gamma_1, \gamma_2 \approx 2)$ almost uniformly.

In light of the program generated solution, we give a purely analytical strategy for A and show that the value of the game is at most 1.488. The strategy (μ, θ) is defined as follows. With probability $\theta = \theta_2$, we run A_2; with probability θ_1, we run $A_1(\gamma)$ with $\gamma = \gamma_1$; with probability $1 - \theta_2 - \theta_1$, we run $A_1(\gamma)$ with γ randomly chosen

between γ_1 and γ_2. Thus, the function μ is

$$\mu(\gamma) = aI_{\gamma_1,\gamma_2}(\gamma) + \theta_1\delta(\gamma - \gamma_1), \tag{5.29}$$

where δ is the Dirac-Delta function, $a = \dfrac{1-\theta_1-\theta_2}{\gamma_2-\gamma_1}$, and $I_{\gamma_1,\gamma_2}(\gamma)$ is 1 if $\gamma_1 < \gamma < \gamma_2$ and 0 otherwise(See Figure 5.4). The values of $\theta_1, \theta_2, \gamma_1, \gamma_2, a$ are given later.

The remaining part of the paper is devoted to prove the following lemma.

Lemma 5.24 *There exists some* $\theta_1 \geq 0, \theta_2 \geq 0, 1 < \gamma_1 < \gamma_2$ *and* $a \geq 0$ *such that* $\theta_1 + \theta_2 + a(\gamma_2 - \gamma_1) = 1$ *and* $\sup_{q\in[0,1)} \nu(\mu, \theta_2, h_q) \leq 1.488$.

Proof: The scaling factor for the facility costs is

$$\gamma_f = \theta_1\gamma_1 + a(\gamma_2 - \gamma_1)\frac{\gamma_1+\gamma_2}{2} + 1.11\theta_2. \tag{5.30}$$

Now, we consider the scaling factor γ_c for the connection costs when $h = h_q$. By Lemma 5.20,

$$\gamma_c(q) \leq \int_1^\infty \left(\int_0^1 e^{-\gamma p}\gamma h_q(p)\mathbf{d}p + e^{-\gamma}(\gamma + (3-\gamma)h_q(1/\gamma)) \right) \mu(\gamma)\mathbf{d}\gamma + 1.78\theta_2$$

replacing μ with $aI_{\gamma_1,\gamma_2}(\gamma) + \theta_1\delta(\gamma - \gamma_1)$ and $h_q(p)$ with 0 or $1/(1-q)$ depending on whether $p \leq q$,

$$\begin{aligned}
&= \int_{\gamma_1}^{\gamma_2} \left(\int_q^1 e^{-\gamma p}\gamma\frac{1}{1-q}\mathbf{d}p + e^{-\gamma}\gamma + e^{-\gamma}(3-\gamma)h_q(1/\gamma) \right) a\mathbf{d}\gamma \\
&\quad + \theta_1 \left(\int_q^1 e^{-\gamma_1 p}\gamma_1\frac{1}{1-q}\mathbf{d}p + e^{-\gamma_1}(\gamma_1 + (3-\gamma_1)h_q(1/\gamma_1)) \right) + 1.78\theta_2 \\
&= B_1(q) + B_2(q) + B_3(q) + 1.78\theta_2,
\end{aligned} \tag{5.31}$$

where

$$\begin{aligned}
B_1(q) &= \int_{\gamma_1}^{\gamma_2}\int_q^1 e^{-\gamma p}\gamma\frac{1}{1-q}\mathbf{d}pa\mathbf{d}\gamma + \int_{\gamma_1}^{\gamma_2} e^{-\gamma}\gamma a\mathbf{d}\gamma, \\
B_2(q) &= \int_{\gamma_1}^{\gamma_2} e^{-\gamma}(3-\gamma)h_q(1/\gamma)a\mathbf{d}\gamma,
\end{aligned}$$

and

$$B_3(q) = \theta_1\int_q^1 e^{-\gamma_1 p}\gamma_1\frac{1}{1-q}\mathbf{d}p + \theta_1 e^{-\gamma_1}(\gamma_1 + (3-\gamma_1)h_q(1/\gamma_1)).$$

Then, we calculate $B_1(q), B_2(q)$ and $B_3(q)$ separately.

$$\begin{aligned} B_1(q) &= \int_{\gamma_1}^{\gamma_2}\int_q^1 e^{-\gamma p}\gamma\frac{1}{1-q}\mathbf{d}pa\mathbf{d}\gamma + \int_{\gamma_1}^{\gamma_2} e^{-\gamma}\gamma a\mathbf{d}\gamma \\ &= \frac{a}{1-q}\int_{\gamma_1}^{\gamma_2}(e^{-\gamma q}-e^{-\gamma})\mathbf{d}\gamma - a(\gamma+1)e^{-\gamma}\Big|_{\gamma_1}^{\gamma_2} \\ &= \frac{a}{(1-q)q}(e^{-\gamma_1 q}-e^{-\gamma_2 q}) - \frac{a}{1-q}(e^{-\gamma_1}-e^{-\gamma_2}) \\ &\quad + a((\gamma_1+1)e^{-\gamma_1}-(\gamma_2+1)e^{-\gamma_2}). \end{aligned} \tag{5.32}$$

$$\begin{aligned} B_2(q) &= \int_{\gamma_1}^{\gamma_2} e^{-\gamma}(3-\gamma)h_q(1/\gamma)a\mathbf{d}\gamma \\ &= \begin{cases} \frac{a}{1-q}\left((2-\gamma_1)e^{-\gamma_1}-(2-\gamma_2)e^{-\gamma_2}\right) & 0\le q<1/\gamma_2 \\ \frac{a}{1-q}\left((2-\gamma_1)e^{-\gamma_1}-(2-1/q)e^{-1/q}\right) & 1/\gamma_2\le q\le 1/\gamma_1 \\ 0 & 1/\gamma_1<q<1 \end{cases}. \end{aligned} \tag{5.33}$$

$$\begin{aligned} B_3(q) &= \theta_1\int_q^1 e^{-\gamma_1 p}\gamma_1\frac{1}{1-q}\mathbf{d}p + \theta_1 e^{-\gamma_1}(\gamma_1+(3-\gamma_1)h_q(1/\gamma_1)) \\ &= \theta_1\left(\frac{1}{1-q}(e^{-\gamma_1 q}-e^{-\gamma_1}) + e^{-\gamma_1}\gamma_1 + e^{-\gamma_1}(3-\gamma_1)h_q(1/\gamma_1)\right) \\ &= \begin{cases} \theta_1\left(\frac{1}{1-q}(e^{-\gamma_1 q}-e^{-\gamma_1}) + e^{-\gamma_1}\gamma_1 + \frac{e^{-\gamma_1}(3-\gamma_1)}{1-q}\right) & 0\le q\le 1/\gamma_1 \\ \theta_1\left(\frac{1}{1-q}(e^{-\gamma_1 q}-e^{-\gamma_1}) + e^{-\gamma_1}\gamma_1\right) & 1/\gamma_1<q<1 \end{cases}. \end{aligned} \tag{5.34}$$

So, we have 3 cases :

1. $0\le q<1/\gamma_2$.

$$\begin{aligned} \gamma_c(q) &\le B_1(q)+B_2(q)+B_3(q)+1.78\theta_2 \\ &= \frac{a}{(1-q)q}(e^{-\gamma_1 q}-e^{-\gamma_2 q}) - \frac{a}{1-q}(e^{-\gamma_1}-e^{-\gamma_2}) \\ &\quad + a((\gamma_1+1)e^{-\gamma_1}-(\gamma_2+1)e^{-\gamma_2}) + \frac{a}{1-q}\left((2-\gamma_1)e^{-\gamma_1}-(2-\gamma_2)e^{-\gamma_2}\right) \\ &\quad + \theta_1\left(\frac{1}{1-q}(e^{-\gamma_1 q}-e^{-\gamma_1}) + e^{-\gamma_1}\gamma_1 + \frac{1}{1-q}e^{-\gamma_1}(3-\gamma_1)\right) + 1.78\theta_2 \\ &= \frac{a}{(1-q)q}(e^{-\gamma_1 q}-e^{-\gamma_2 q}) + \frac{A_1}{1-q} + \theta_1\frac{e^{-\gamma_1 q}}{1-q} + A_2, \\ \text{where } A_1 &= a(e^{-\gamma_1}-\gamma_1 e^{-\gamma_1}-e^{-\gamma_2}+\gamma_2 e^{-\gamma_2}) + 2\theta_1 e^{-\gamma_1} - \theta_1 e^{-\gamma_1}\gamma_1 \\ \text{and } A_2 &= a((\gamma_1+1)e^{-\gamma_1}-(\gamma_2+1)e^{-\gamma_2}) + \theta_1 e^{-\gamma_1}\gamma_1 + 1.78\theta_2. \end{aligned}$$

2. $1/\gamma_2\le q\le 1/\gamma_1$.

The only difference between this case and the first case is the definition of $B_2(q)$. Comparing the definition of $B_2(q)$ for the case $0 \le q < 1/\gamma_2$ and the case $1/\gamma_2 \le q \le 1/\gamma_1$, we can get

$$\gamma_c(q) = \frac{a}{(1-q)q}(e^{-\gamma_1 q} - e^{-\gamma_2 q}) + \frac{A_1}{1-q} + \theta_1 \frac{e^{-\gamma_1 q}}{1-q} + A_2 + \frac{a}{1-q}\left((2-\gamma_2)e^{-\gamma_2} - (2-1/q)e^{-1/q}\right). \quad (5.35)$$

3. $1/\gamma_1 < q < 1$

$$\begin{aligned}\gamma_c(q) &\le \frac{a}{(1-q)q}(e^{-\gamma_1 q} - e^{-\gamma_2 q}) - \frac{a}{1-q}(e^{-\gamma_1} - e^{-\gamma_2}) \\ &\quad + a((\gamma_1+1)e^{-\gamma_1} - (\gamma_2+1)e^{-\gamma_2}) \\ &\quad + \theta_1\left(\frac{1}{1-q}(e^{-\gamma_1 q} - e^{-\gamma_1}) + e^{-\gamma_1}\gamma_1\right) + 1.78\theta_2 \\ &= \frac{a}{(1-q)q}(e^{-\gamma_1 q} - e^{-\gamma_2 q}) + \frac{A_3}{1-q} + \theta_1\frac{e^{-\gamma_1 q}}{1-q} + A_2, \quad (5.36)\end{aligned}$$

where $A_3 = a(-e^{-\gamma_1} + e^{-\gamma_2}) - \theta_1 e^{-\gamma_1}$.

We set $\gamma_1 = 1.479311, \gamma_2 = 2.016569, \theta_1 = 0.503357, a = 0.560365$ and $\theta_2 = 1 - \theta_1 - a(\gamma_2 - \gamma_1) \approx 0.195583$. Then,

$$\gamma_f = \theta_1\gamma_1 + a(\gamma_2 - \gamma_1)\frac{\gamma_1+\gamma_2}{2} + 1.11\theta_2 \approx 1.487954. \quad (5.37)$$

$A_1 \approx 0.074347, A_2 \approx 0.609228$ and $A_3 \approx -0.167720$. $\gamma_c(q)$ has the maximum value about 1.487989, achieved at $q = 0$(see Figure 5.5). This finishes the proof of Lemma 5.24. □

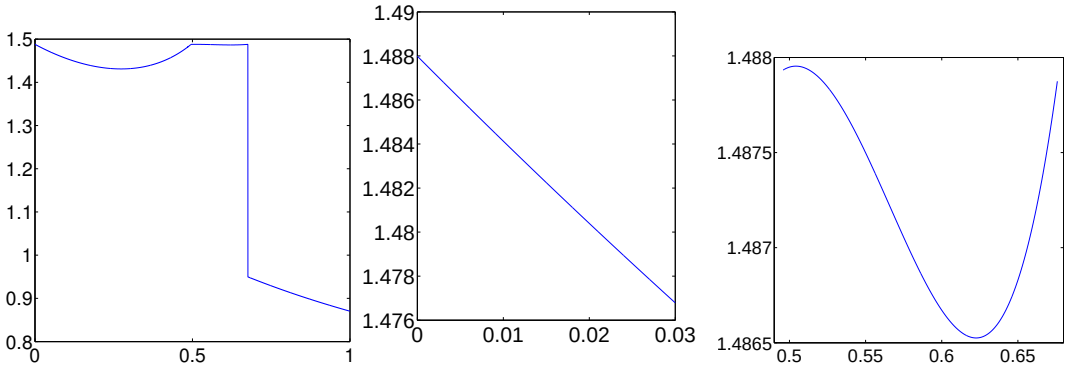

Figure 5.5: The function $\gamma_c(q)$. The curve in the middle is the function restricted to the interval $[0, 0.03]$ and the curve on the right is the function restricted to the interval $(1/\gamma_2, 1/\gamma_1)$. The maximum value of the function is achieved at $q = 0$.

Thus, Theorem 5.1 follows immediately from Lemma 5.23 and Lemma 5.24.

Chapter 6

Approximating k-Median

6.1 Introduction

In this section, we give our $1 + \sqrt{3} + \epsilon$ approximation for k-median. The main difficulty of the k-median problem is the hard constraint that we can open at most k facilities. We take a different approach that allows us to relax this constraint and thereby addressing the problem from a novel point of view using what we call a pseudo-approximation algorithm. This leads to the improved approximation algorithm, which can be stated as follows.

Theorem 6.1 *There is an algorithm which, given a k-median instance $\mathcal{I}$ and a number $\epsilon > 0$, produces a $1 + \sqrt{3} + \epsilon$-approximate solution to $\mathcal{I}$ in running time $O\left(n^{O(1/\epsilon^2)}\right)$.*

Our algorithm contains two main components, each of which, we believe, is of independent interest. First, we show that in order to give an approximation algorithm for k-median, it suffices to give a *pseudo-approximation algorithm* $\mathcal{A}$ which, given a k-median instance $\mathcal{I}$, outputs a set $\mathcal{S} \subseteq \mathcal{F}$ of $k + c$ facilities with $\mathsf{cost}_{\mathcal{I}}(\mathcal{S}) \leq \alpha \mathsf{opt}_{\mathcal{I}}$, where $\mathsf{opt}_{\mathcal{I}}$ is the cost of optimum solution for $\mathcal{I}$. Given such an algorithm $\mathcal{A}$ as a black box, we can design an $\alpha + \epsilon$-approximation algorithm $\mathcal{A}'$ whose running time is $n^{O(c/\epsilon)}$ times that of $\mathcal{A}$. Interestingly, the instance (see Figure 6.1) that gives the integrality gap of 2 for the natural LP relaxation of k-median vanishes if we allow the integral solution to open $k + 1$ facilities. This suggests that our reduction may bring in new avenues for approximating k-median. In particular, we find the following open problem interesting: given a k-median instance $\mathcal{I}$, what is the maximum ratio between the cost of the optimum integral solution of $\mathcal{I}$ with $k+1$ open facilities, and the LP value (with k open facilities)?

To complement the first component, we give the aforementioned pseudo-approximation algorithm $\mathcal{A}$ with $\alpha = 1 + \sqrt{3} + \epsilon$. Prior to our work, it was not even known whether opening $k + o(k)$ facilities would help improve the approximation ratio; all known pseudo-approximation algorithms require $k + \Omega(k)$ open facilities. In contrast, our algorithm only opens $k + O(1/\epsilon)$ facilities. The algorithm $\mathcal{A}$ contains 2 steps. We obtain a *bi-point solution* for k-median using the algorithm

of [46]. We lose a factor of 2 in this step. Then, we convert the bi-point solution into an integral solution with $k + O(1/\epsilon)$ open facilities, losing another factor of $\frac{1+\sqrt{3}+\epsilon}{2}$ in the approximation ratio. We remark that if we had insisted on opening k facilities, then a factor of 2 has to be lost in the last step as the instance achieving an integrality gap of 2 has a bi-point solution.

Theorem 6.1 does not give a better upper bound on the integrality gap of the natural LP due to the following reason: instead of running the pseudo-approximation algorithm $\mathcal{A}$ on the input instance $\mathcal{I}$, we run it on a residual instance $\mathcal{I}'$ obtained from $\mathcal{I}$ by removing a subset of facilities that the optimal solution does not open. The way we obtain $\mathcal{I}'$ is to guess $O(1/\epsilon^2)$ "events" and let $\mathcal{I}'$ be the instance conditioned on these events. Due to this nature, our algorithm can be converted to a rounding algorithm based on solving an $O(1/\epsilon^2)$-level LP in the Sherali-Adams hierarchy. Instead of guessing the $O(1/\epsilon^2)$ events, we can now find these events explicitly by looking at the LP solution. Conditioning on these events, we obtain a fractional solution of the basic LP. By rounding this LP, we obtain a $1 + \sqrt{3} + \epsilon$-approximate solution. Thus, our approach can be seen to give an $1 + \sqrt{3} + \epsilon$-upper bound on the integrality gap of the $O(1/\epsilon^2)$-level LP in the Sherali-Adams hierarchy. Our result was in fact first obtained by studying the power of the Sherali-Adams hierarchy for the k-median problem. However, as it can also be obtained using a combinatorial approach with less cumbersome notation, we have chosen to present that approach.

We also remark that if $\mathcal{F} = \mathcal{C}$, the proof of our first component can be simplified, by using a recent related result. Awasthi et al. [11] considered k-median clustering under the stability assumption: they obtained a PTAS for what they called *stable instances*. To be more specific, if the given k-median instance has the property that the optimum cost with $k - 1$ medians is at least a factor $(1 + \delta)$ larger than the optimum cost with k medians (this is called stable instance), then their algorithm finds a $(1 + \epsilon)$-approximate solution to the instance in $n^{O(1/(\delta\epsilon))}$ time. Using their result as a blackbox, we can convert a pseudo-approximation algorithm $\mathcal{A}$ to a true approximation algorithm $\mathcal{A}'$ easily.[1] However, as we mentioned, one caveat with this approach is that their algorithm is for the case $\mathcal{F} = \mathcal{C}$. Extending their algorithm to the general case is not immediate and requires reproving all the lemmas. For the completeness of the paper, we have therefore chosen to present our own approach. Another difference between the two approaches is that we have a weaker notion of stability, called *sparse instances* (defined in Section 6.2) that can be found in polynomial time (and can be made LP based using the Sherali-Adams hierarchy). This weaker notion does not imply a PTAS for the problem (assuming P≠NP) but it is still sufficient for our purposes. Specifically, the sparsity condition implies that, given

[1] To see how [11] implies our first component, consider the following algorithm. Given a k-median instance $(k, \mathcal{F}, \mathcal{C}, d)$, we apply our pseudo-approximation algorithm $\mathcal{A}$ to the instance $(k-c, \mathcal{F}, \mathcal{C}, d)$ to obtain a set $\mathcal{T} \subset \mathcal{F}$ of $k - c + c = k$ facilities, such that $\mathsf{cost}(\mathcal{T})$ is at most α times opt_{k-c}, the optimum cost with $k-c$ open facilities. If $\mathsf{opt}_{k-c} \leq (1+\epsilon)\mathsf{opt}_k$, then we get a $\alpha(1+\epsilon)$-approximation. Otherwise, there must be a $i \in \{0, 1, \cdots, c-1\}$ such that $\mathsf{opt}_{k-i-1} \geq (1+\epsilon)^{1/c}\mathsf{opt}_{k-i}$. Consider the smallest such i. Then applying the algorithm of [11] to $(k-i, \mathcal{F}, \mathcal{C}, d)$ with $\delta = (1+\epsilon)^{1/c} - 1 \approx \epsilon/c$ (for small ϵ) will give a solution of cost at most $(1+\epsilon)\mathsf{opt}_{k-i} \leq (1+\epsilon)^2\mathsf{opt}_k$. Although we do not know i, we can try all i's and output the best solution.

any pseudo-solution to k-median, we can either (i) remove one of the facilities without increasing the cost too much or (ii) we can similarly to the result in [11] find a $(1+\epsilon)$-approximate solution. This becomes explicit in the proof of Lemma 6.8.

6.1.1 Preliminaries

Given a k-median instance $\mathcal{I} = (k, \mathcal{F}, \mathcal{C}, d)$, a *pseudo-solution* to $\mathcal{I}$ is a set $\mathcal{S} \subseteq \mathcal{F}$. A pseudo-solution $\mathcal{S}$ satisfying $|\mathcal{S}| \leq k$ is a *solution* to $\mathcal{I}$; a pseudo-solution $\mathcal{S}$ with $|\mathcal{S}| \leq k + c$, for some number $c \geq 0$, is called a *c-additive (pseudo-)solution*. The cost of a pseudo-solution $\mathcal{S}$ to $\mathcal{I}$ is defined as $\mathsf{cost}_{\mathcal{I}}(\mathcal{S}) = \sum_{j \in \mathcal{C}} d(j, \mathcal{S})$, where $d(j, \mathcal{S})$ denotes the distance from j to its closest facility in $\mathcal{S}$. We let $\mathsf{OPT}_{\mathcal{I}}$ denote an optimal solution to $\mathcal{I}$, i.e., one of minimum cost, and we let $\mathsf{opt}_{\mathcal{I}} = \mathsf{cost}_{\mathcal{I}}(\mathsf{OPT}_{\mathcal{I}})$. To avoid confusion we will throughout the paper assume that the optimal solution is unique and that the concept of closest facility (or client) is also uniquely defined. This can be achieved either by slightly perturbing the metric or by simply breaking ties in an arbitrary but fixed way.

When considering a client or facility, it shall be convenient to argue about close clients or facilities. For any $p \in \mathcal{F} \cup \mathcal{C}$ and $r \geq 0$, we therefore define $\mathsf{FBall}_{\mathcal{I}}(p, r) = \{i \in \mathcal{F} : d(p, i) < r\}$ and $\mathsf{CBall}_{\mathcal{I}}(p, r) = \{j \in \mathcal{C} : d(p, j) < r\}$ to be the set of facilities and clients within distance less than r from p, respectively. When $\mathcal{I}$ is clear from the context, we omit the subscripts in $\mathsf{cost}_{\mathcal{I}}$, $\mathsf{OPT}_{\mathcal{I}}$, $\mathsf{opt}_{\mathcal{I}}$, $\mathsf{FBall}_{\mathcal{I}}$, and $\mathsf{CBall}_{\mathcal{I}}$.

The standard linear programming relaxation for the k-median problem is formulated as follows.

$$\begin{aligned}
&\text{minimize } \textstyle\sum_{i \in \mathcal{F}, j \in \mathcal{C}} d(i,j) x_{ij} \\
&\text{subject to} \qquad \textstyle\sum_{i \in \mathcal{F}} y_i \leq k && && \text{(6.1a)} \\
&\qquad\qquad \textstyle\sum_{i \in \mathcal{F}} x_{ij} = 1 && j \in \mathcal{C} && \text{(6.1b)} \\
&\qquad\qquad x_{ij} \leq y_i && i \in \mathcal{F}, j \in \mathcal{C} && \text{(6.1c)} \\
&\qquad\qquad x_{ij}, y_i \in [0,1] && i \in \mathcal{F}, j \in \mathcal{C} && \text{(6.1d)}
\end{aligned}$$

Constraint (6.1a) says that we are allowed to open at most k facilities, Constraint (6.1b) says that we must connect each client, and Constraint (6.1c) says that if we connect a client to a facility then that facility has to be opened.

As mentioned earlier, the above linear programming has an integrality gap of 2, even when the underlying metric is a tree. The instance that gives the integrality gap of 2 is depicted in Figure 6.1. It is a star with $k+1$ leaves. The center of the star is a facility and the leaves are both facilities and clients. Note that a pseudo-solution that opens all leaves, i.e., $k+1$ facilities, has cost 0 whereas any solution that opens only k facilities has cost 2. The solution to the linear program obtained by a linear combination of the pseudo-solution that opens all leaves and the solution that only opens the center of the star has cost $1 + 1/k$ yielding the integrality gap of 2 when k tends to infinity. In general, a solution that is a linear combination of two pseudo-

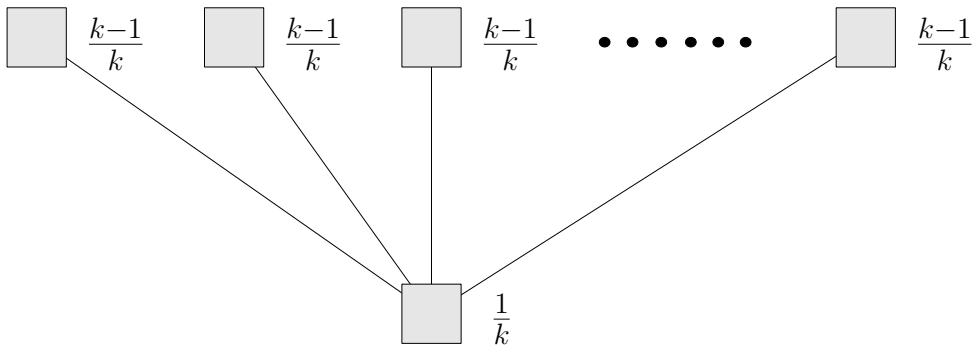

Figure 6.1: Instance that gives integrality gap $2/(1+1/k)$ and the optimal fractional solution. We have $k+2$ facilities and $k+1$ clients co-located with the top $k+1$ facilities. All edges in the graph have length 1. The optimal integral solution has cost 2, while the optimal fractional solution has cost $(k+1)\left(\frac{k-1}{k}\cdot 0+\frac{1}{k}\cdot 1\right)=1+1/k$.

solutions is called a *bi-point (fractional) solution*. As this concept is important for our pseudo-approximation algorithm, we state its formal definition.

Definition 6.2 (bi-point (fractional) solution) *Let $\mathcal{I}=(k,\mathcal{F},\mathcal{C},d)$ be a k-median instance. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two pseudo-solutions to $\mathcal{I}$ such that $|\mathcal{S}_1|\le k<|\mathcal{S}_2|$. Let $a\ge 0, b\ge 0$ be the real numbers such that $a+b=1$ and $a|\mathcal{S}_1|+b|\mathcal{S}_2|=k$. Then, the following fractional solution to $\mathcal{I}$, denoted by $a\mathcal{S}_1+b\mathcal{S}_2$, is called a bi-point (fractional) solution:*

1. $y_i=a1_{i\in\mathcal{S}_1}+b1_{i\in\mathcal{S}_2}$;
2. $x_{i,j}=a1_{\mathsf{clst}(i,\mathcal{S}_1,j)}+b1_{\mathsf{clst}(i,\mathcal{S}_2,j)}$, *where* $\mathsf{clst}(i,\mathcal{S},j)$ *denotes the event that i is the closest facility in $\mathcal{S}$ to j.*

It is easy to see that the cost of the fractional solution $a\mathcal{S}_1+b\mathcal{S}_2$ is exactly $a\mathsf{cost}_{\mathcal{I}}(\mathcal{S}_1)+b\mathsf{cost}_{\mathcal{I}}(\mathcal{S}_2)$. Jain and Vazirani [47] gave a *Lagrangian multiplier preserving* 3-approximation for UFL, which immediately yields an algorithm which produces a bi-point solution whose cost is at most 3 times the optimum. Together with an algorithm which converts a bi-point solution to an integral solution at the cost of a factor 2, [47] gave a 6-approximation for k-median. Later, the factor 3 was improved by Jain et al. [46] to 2. We now formally state the result of [46].

Theorem 6.3 ([46]) *Given a k-median instance $\mathcal{I}$, we can find in polynomial time a bi-point solution $a\mathcal{S}_1+b\mathcal{S}_2$ to $\mathcal{I}$ whose cost is at most 2 times the cost of an optimal solution to $\mathcal{I}$.*

6.1.2 Overview of the Algorithm

The two components of our algorithm are formally stated in Theorem 6.4 and Theorem 6.5, whose proofs will be given in Sections 6.2 and 6.3, respectively. Together they immediately imply Theorem 6.1.

Theorem 6.4 *Let $\mathcal{A}$ be a c-additive α-approximation algorithm for k-median, for some $\alpha > 1$. Then, for every $\epsilon > 0$ there is a $\alpha + \epsilon$-approximation algorithm $\mathcal{A}'$ for k-median whose running time is $O\left(n^{O(c/\epsilon)}\right)$ times the running time of $\mathcal{A}$.*

Theorem 6.5 *There exists a polynomial time algorithm which, given a k-median instance $\mathcal{I} = (k, \mathcal{F}, \mathcal{C}, d)$ and $\epsilon > 0$, produces an $O(1/\epsilon)$-additive $1+\sqrt{3}+\epsilon$-approximate solution to $\mathcal{I}$.*

We now provide more details about the proof of the two theorems. At first glance, it seems that the transformation from a pseudo-approximation to a real approximation stated in Theorem 6.4 is impossible, since there are cases where allowing $k+1$ open facilities would give much smaller cost than only allowing k open facilities. However, we show that we can pre-process the input instance so as to avoid these problematic instances. Roughly speaking, we say that a facility i is dense if the clients in a small ball around i contribute a lot to the cost of the optimum solution OPT (see Definition 6.6). We guess the $O(1/\epsilon)$ densest facilities and their respective nearest open facilities in OPT. Then for each such dense facility i whose nearest open facility in OPT is i', we remove all facilities that are closer to i than i' (including the dense facility i). Then we get a residual instance in which the gap between the costs of opening $k + O(1)$ and k facilities is small. The pseudo-approximation algorithm is then applied to this residual instance.

For example, consider the integrality gap instance depicted in Figure 6.1 and let OPT be the optimal solution that opens the center and $k-1$ leaves. Then the two leaves that were not opened contribute a large fraction of the total cost (each contributes $\mathsf{opt}/2$ to be precise) and the two corresponding facilities are dense. By removing these dense facilities in a preprocessing step, the gap between the costs of opening $k + O(1)$ facilities and k facilities for the residual instance becomes small (actually 0 in this example).

Regarding the proof of Theorem 6.5, we first use Theorem 6.3 to obtain a bi-point solution for k-median whose cost is at most twice the optimum cost. Jain and Vazirani [47] showed how to convert a bi-point solution to an integral solution, losing a multiplicative factor of 2 in the approximation. As we previously mentioned, this factor of 2 is tight, as the fractional solution for the gap instance in Figure 6.1 is a bi-point solution. Thus, this approach can only yield a 4-approximation.

This is where the c-additive pseudo-approximation is used and again the integrality gap instance depicted in Figure 6.1 inspired our approach. Recall that if we open the $k+1$ leaves of that instance, then we get a solution of cost 0. In other words, by opening 1 additional facility, we can do better than the fractional solution. One may argue that this trick is too weak to handle more sophisticated cases and try to enhance the gap instance. A natural way to enhance it is to make many separate copies of the instance to obtain several "stars". One might expect that the fractional cost in each copy is 1, the integral cost in each copy is 2 and opening 1 more facility can only improve the integral solution of one copy and thus does not improve the overall ratio by too much. However, the integral solution can do much better since one cannot restrict the integral solution to open k facilities in each star. As an example, consider

the case where we have 2 copies. The integral solution can open $k-1$ facilities in the first star, and $k+1$ facility in the second star. Then, the cost of this solution is 3, as opposed to 4 achieved by opening k facilities in each star. The gap is already reduced to 1.5, without opening additional facilities. Thus, this simple way to enhance the instance failed.

Our pseudo-approximation algorithm is based on this intuition. From the bi-point solution $a\mathcal{F}_1+b\mathcal{F}_2$, we obtain copies of "stars" (similar to the integrality gap instance). Then for each star we (basically) open either its center with probability a or all its leaves with probability b. Note that since either the center or all leaves of a star is open we have that a client always has a "close" facility opened. With this intuition we prove in Section 6.3 that the expected cost of the obtained pseudo-solution is at most $\frac{1+\sqrt{3}+\epsilon}{2}$ times the cost of the bi-fractional solution if we open $O(1/\epsilon)$ additional facilities. The $O(1/\epsilon)$ additional facilities (and the case distinction in Section 6.3) comes from the difficulty of handling stars of different sizes. If all stars are of the same size the pseudo-approximation algorithm becomes easier (run the algorithm in Section 6.3.2 with one group of stars) and one obtains a $\frac{1+\sqrt{3}}{2}$-approximate solution that opens at most $k+3$ facilities.

6.2 Obtaining Solutions from Additive Pseudo-Solutions

In this section, we prove Theorem 6.4. As we mentioned earlier, there are instances where pseudo-solutions opening $k+1$ facilities may have much smaller cost than solutions opening k facilities. A key concept to overcome this issue is the notion of sparse instances:

Definition 6.6 *For $A>0$, an instance $\mathcal{I}=(k,\mathcal{F},\mathcal{C},d)$ is A-sparse if for each facility $i\in\mathcal{F}$,*

$$(1-\xi)d(i,\mathsf{OPT}_{\mathcal{I}})\cdot|\mathsf{CBall}_{\mathcal{I}}(i,\xi d(i,\mathsf{OPT}_{\mathcal{I}}))|\leq A, \tag{6.2}$$

where $\xi:=1/3$. We shall also say that a facility i is A-dense if it violates (6.2).

Recall that $d(i,\mathsf{OPT}_{\mathcal{I}})$ is the distance from i to its nearest facility in $\mathsf{OPT}_{\mathcal{I}}$.

The idea of the above definition is to avoid instances where we can significantly reduce the cost by opening $O(1)$ additional facilities. Consider the gap instance $\mathcal{I}$ in Figure 6.1 and suppose $\mathsf{OPT}_{\mathcal{I}}$ opens the center and the first $k-1$ leaf-facilities. Then $\mathcal{I}$ is not A-sparse for $A<\mathsf{opt}_{\mathcal{I}}/2$ since the last two leaf-facilities are A-dense.

The usefulness of the definition is twofold. On the one hand, we show that we can concentrate on very sparse instances without loss of generality. On the other hand, we show that any c-additive pseudo-solution to a sparse instance can be turned into a solution that opens k facilities by only increasing the cost slightly. The intuition behind the result that we can only concentrate on sparse instances is the following. Consider an instance $\mathcal{I}$ that is not $\mathsf{opt}_{\mathcal{I}}/t$-sparse for some constant t. If we consider a facility i that is $\mathsf{opt}_{\mathcal{I}}/t$-dense then the connection cost of the

clients contained in $\mathsf{CBall}(i, \xi d(i, \mathsf{OPT}_{\mathcal{I}}))$ in the optimal solution $\mathsf{OPT}_{\mathcal{I}}$ is at least $(1-\xi)d(i, \mathsf{OPT}_{\mathcal{I}})|\mathsf{CBall}(i, \xi d(i, \mathsf{OPT}_{\mathcal{I}}))| > \mathsf{opt}_{\mathcal{I}}/t$. So, there can essentially (assuming disjointedness of the balls of clients) only be a constant t number of facilities that violate the sparsity condition. We can guess this set of dense facilities, as well as their nearest facility in $\mathsf{OPT}_{\mathcal{I}}$ in time $n^{O(t)}$.

This is the intuition of Algorithm 1 (that tries to guess and remove opt/t-dense facilities) and the proof of the following lemma which is given in Section 6.2.1.

Lemma 6.7 *Given a k-median instance $\mathcal{I} = (k, \mathcal{F}, \mathcal{C}, d)$ and a positive integer t, Algorithm 1 outputs in time $n^{O(t)}$ many k-median instances obtained by removing facilities from $\mathcal{I}$ so that at least one, say $\mathcal{I}' = (k, \mathcal{F}' \subseteq \mathcal{F}, \mathcal{C}, d)$, satisfies*

1. *the optimal solution $\mathsf{OPT}_{\mathcal{I}}$ to $\mathcal{I}$ is also an optimal solution to $\mathcal{I}'$; and*
2. *$\mathcal{I}'$ is $\mathsf{opt}_{\mathcal{I}}/t$-sparse.*

Note that $\mathcal{I}'$ is obtained by removing facilities from $\mathcal{I}$. Therefore any solution to $\mathcal{I}'$ defines a solution to $\mathcal{I}$ of the same cost and we can thus restrict our attention to sparse instances. The next lemma shows the advantage of considering such instances. Assume we now have a c-additive solution $\mathcal{T}$ to a sparse instance $\mathcal{I}$. Algorithm 2 tries first in Lines 2-3 to identify facilities in $\mathcal{T}$ whose removal does not increase the cost by too much. If the removal results in a set of at most k facilities, we have obtained a "good" solution returned at Step 4 of the algorithm. Otherwise, as we prove in Section 6.2.2 using sparsity, more than $k - t$ of the facilities of the solution $\mathcal{T}$ are very close to facilities in $\mathsf{OPT}_{\mathcal{I}}$. Algorithm 2 therefore tries to guess these facilities (the set $\mathcal{D}$) and the remaining facilities of $\mathsf{OPT}_{\mathcal{I}}$ (the set $\mathcal{V}$). The obtained bounds are given in the following lemma.

Lemma 6.8 *Given an A-sparse instance $\mathcal{I} = (k, \mathcal{F}, \mathcal{C}, d)$, a c-additive pseudo-solution $\mathcal{T}$, $\delta \in (0, 1/8)$, and an integer $t \geq 2c/(\delta\xi)$, Algorithm 2 finds in time $n^{O(t)}$ a set $\mathcal{S} \subseteq \mathcal{F}$ such that:*

1. *$\mathcal{S}$ is a solution to $\mathcal{I}$, i.e, $|\mathcal{S}| \leq k$; and*
2. *$\mathsf{cost}_{\mathcal{I}}(\mathcal{S}) \leq \max\left\{\mathsf{cost}_{\mathcal{I}}(\mathcal{T}) + cB, \frac{1+3\delta}{1-3\delta} \cdot \mathsf{opt}_{\mathcal{I}}\right\}$, where $B := 2 \cdot \frac{A + \mathsf{cost}_{\mathcal{I}}(\mathcal{T})/t}{\xi\delta}$.*

Before giving the proofs of Lemmas 6.7 and 6.8 let us see how they imply the main result of this section.

Proof of Theorem 6.4. Select the largest $\delta \in (0, 1/8)$ such that $(1+3\delta)/(1-3\delta) \leq \alpha$ and $t := \frac{4}{\epsilon} \cdot \frac{\alpha c}{\xi \cdot \delta} = O(c/\epsilon)$. Given a k-median instance $\mathcal{I}$, use Algorithm 1 to obtain a set of k-median instances such that at least one of these instances, say $\mathcal{I}'$, satisfies the properties of Lemma 6.7. In particular, $\mathcal{I}'$ is $\mathsf{opt}_{\mathcal{I}}/t$-sparse. Now use algorithm $\mathcal{A}$ to obtain c-additive pseudo-solutions to each of these instances. Note that when we apply $\mathcal{A}$ to $\mathcal{I}'$, we obtain a solution $\mathcal{T}$ such that $\mathsf{cost}_{\mathcal{I}}(\mathcal{T}) = \mathsf{cost}_{\mathcal{I}'}(\mathcal{T}) \leq \alpha \cdot \mathsf{opt}_{\mathcal{I}'} = \alpha \cdot \mathsf{opt}_{\mathcal{I}}$. Finally, use Algorithm 2 (with the same t and δ selected as above) to transform the pseudo-solutions into real solutions and return the solution to $\mathcal{I}$ of minimum cost. The

Input: a k-median instance $\mathcal{I} = (k, \mathcal{F}, \mathcal{C}, d)$ and a positive integer t
Output: a set of k-median instances so that at least one satisfies the properties of Lemma 6.7

for all $t' \leq t$ facility-pairs $(i_1, i'_1), (i_2, i'_2), \ldots, (i_{t'}, i'_{t'})$ **output** $(k, \mathcal{F}', \mathcal{C}, d)$, where

$$\mathcal{F}' = \mathcal{F} \setminus \bigcup_{z=1}^{t'} \mathsf{FBall}(i_z, d(i_z, i'_z)) \qquad \triangleright$$

the facilities that are closer to i_z than i'_z is to i_z are removed

Algorithm 1: Enumeration of k-median instances.

cost of the returned solution is at most the cost of $\mathcal{S}$ where $\mathcal{S}$ is the solution obtained by transforming $\mathcal{T}$. By Lemmas 6.7 and 6.8, we have that $\mathsf{cost}_{\mathcal{I}}(S) = \mathsf{cost}_{\mathcal{I}'}(S)$ is at most

$$\max\left\{\mathsf{cost}_{\mathcal{I}}(\mathcal{T}) + c \cdot 2\frac{\mathsf{opt}_{\mathcal{I}} + \mathsf{cost}_{\mathcal{I}}(\mathcal{T})}{t\xi\delta}, \frac{1+3\delta}{1-3\delta}\mathsf{opt}_{\mathcal{I}}\right\},$$

which in turn, by the selection of δ, ξ, and t, is at most $\alpha\mathsf{opt}_{\mathcal{I}} + c \cdot \frac{4\alpha\mathsf{opt}_{\mathcal{I}}}{t\xi\delta} \leq (\alpha+\epsilon)\mathsf{opt}_{\mathcal{I}}$.

We conclude the proof of Theorem 6.4 by observing that the runtime of the algorithm is $n^{O(t)} = n^{O(c/\epsilon)}$ times the runtime of $\mathcal{A}$.

6.2.1 Proof of Lemma 6.7: Obtaining a Sparse Instance

First note that Algorithm 1 selects $n^{O(t)}$ facility-pairs and can be implemented to run in time $n^{O(t)}$. We proceed by showing that for one selection of facility-pairs the obtained instance satisfies the properties of Lemma 6.7. Consider a maximal-length sequence $(i_1, i'_1), (i_2, i'_2), \ldots, (i_\ell, i'_\ell)$ of facility-pairs satisfying: for every $b = 1, \ldots, \ell$,

- $i_b \in \mathcal{F} \setminus \bigcup_{z=1}^{b-1} \mathsf{FBall}(i_z, d(i_z, i'_z))$ is an $\mathsf{opt}_{\mathcal{I}}/t$-dense facility; and
- i'_b is the closest facility to i_b in $\mathsf{OPT}_{\mathcal{I}}$.

Note that the instance $\mathcal{I}' := (k, \mathcal{F}', \mathcal{C}, d)$ with $\mathcal{F}' = \mathcal{F} \setminus \bigcup_{z=1}^{\ell} \mathsf{FBall}(i_z, d(i_z, i'_z))$ is $\mathsf{opt}_{\mathcal{I}}/t$-sparse since otherwise the sequence $(i_1, i'_1), (i_2, i'_2), \ldots, (i_\ell, i'_\ell)$ would not be of maximal length. Moreover, since we do not remove any facilities in $\mathsf{OPT}_{\mathcal{I}}$, i.e., $(\mathcal{F} \setminus \mathcal{F}') \cap \mathsf{OPT}_{\mathcal{I}} = \emptyset$, we have that $\mathsf{OPT}_{\mathcal{I}}$ is also an optimal solution to $\mathcal{I}'$. In other words, $\mathcal{I}'$ satisfies the properties of Lemma 6.7.

We complete the proof by showing that Algorithm 1 enumerates $\mathcal{I}'$, i.e., that $\ell \leq t$. For the sake of notation let $\mathcal{B}_z := \mathsf{CBall}(i, \xi d(i_z, i'_z))$. First, note that the client-balls $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\ell$ are disjoint. Indeed, if a ball $\mathcal{B}_z$ overlaps a ball $\mathcal{B}_w$ with $1 \leq z < w \leq \ell$ then $d(i_z, i_w) < \xi d(i_z, i'_z) + \xi d(i_w, i'_w)$. However, since i_w must be in $\mathcal{F} \setminus \mathsf{FBall}(i_z, d(i_z, i'_z))$, we have $d(i_z, i_w) \geq d(i_z, i'_z)$. Since i'_w is the closest facility in $\mathsf{OPT}_{\mathcal{I}}$ to i_w, we have $d(i_w, i'_w) \leq d(i_w, i'_z)$, which, by triangle inequalities, is at most $d(i_z, i_w) + d(i_z, i'_z) \leq 2d(i_z, i_w)$. Hence (using that $\xi = 1/3$),

$$\xi(d(i_z, i'_z) + d(i_w, i'_w)) \leq 3\xi d(i_z, i_w) \leq d(i_z, i_w),$$

Input: an A-sparse instance $\mathcal{I} = (k, \mathcal{F}, \mathcal{C}, d)$, a c-additive pseudo-solution $\mathcal{T}$, an integer $t \geq c$ and $\delta \in (0, 1/8)$

Output: A solution $\mathcal{S}$ satisfying the properties of Lemma 6.8

1: $\mathcal{T}' := \mathcal{T}$ and $B := 2 \cdot \frac{A + \mathsf{cost}_{\mathcal{I}}(\mathcal{T})/t}{\delta \xi}$

2: **while** $|\mathcal{T}'| > k$ and there is a facility $i \in \mathcal{T}'$ such that $\mathsf{cost}_{\mathcal{I}}(\mathcal{T}' \setminus \{i\}) \leq \mathsf{cost}_{\mathcal{I}}(\mathcal{T}') + B$ **do**

3: Remove i from $\mathcal{T}'$;

4: **return** $\mathcal{S} := \mathcal{T}'$ if $|\mathcal{T}'| \leq k$;

5: **for all** $\mathcal{D} \subseteq \mathcal{T}'$ and $\mathcal{V} \subseteq \mathcal{F}$ such that $|\mathcal{D}| + |\mathcal{V}| = k$ and $|\mathcal{V}| < t$ **do**

6: For $i \in \mathcal{D}$, let $L_i = d(i, \mathcal{T}' \setminus \{i\})$ and f_i be the facility in $\mathsf{FBall}(i, \delta L_i)$ that minimizes

$$\sum_{j \in \mathsf{CBall}(i, L_i/3)} \min\{d(f_i, j), d(j, \mathcal{V})\}$$

7: Let $\mathcal{S}_{\mathcal{D},\mathcal{V}} := \mathcal{V} \cup \{f_i : i \in \mathcal{D}\}$

8: **return** $\mathcal{S} := \arg\min_{S_{\mathcal{D},\mathcal{V}}} \mathsf{cost}_{\mathcal{I}}(S_{\mathcal{D},\mathcal{V}})$

Algorithm 2: Obtaining a solution from a c-additive pseudo-solution.

which implies that the balls do not overlap.

Second, note that the connection cost of a client in $\mathcal{B}_z$ is, by triangle inequalities, at least $(1-\xi)d(i_z, i'_z) = (1-\xi)d(i_z, \mathsf{OPT}_{\mathcal{I}})$. We thus have (using that the client-balls are disjoint) that $\mathsf{opt}_{\mathcal{I}} \geq \sum_{z=1}^{\ell}(1-\xi)d(i_z, \mathsf{OPT}_{\mathcal{I}})|\mathcal{B}_z|$. As we only selected $\mathsf{opt}_{\mathcal{I}}/t$-dense facilities, $(1-\xi)d(i_z, \mathsf{OPT}_{\mathcal{I}})|\mathcal{B}_z| \geq \mathsf{opt}_{\mathcal{I}}/t$ and hence $\mathsf{opt}_{\mathcal{I}} \geq \ell \mathsf{opt}_{\mathcal{I}}/t$. It follows that $t \geq \ell$ which completes the proof of Lemma 6.7.

6.2.2 Proof of Lemma 6.8: Obtaining Solution to Sparse Instance from a Pseudo-Solution

We start by analyzing the running time of Algorithm 2. Clearly the while loop can run at most c iterations (a constant). The number of different pairs $(\mathcal{D}, \mathcal{V})$ in the for loop is at most

$$\sum_{\ell=0}^{t} \binom{|\mathcal{T}'|}{k-\ell} \binom{|\mathcal{F}|}{\ell}.$$

Notice that $|\mathcal{T}'| \leq k + c$ and $c \leq t$. For sufficiently large k and $|\mathcal{F}|$, the above quantity is at most $\binom{|\mathcal{F}|}{t} \sum_{\ell=0}^{t} \binom{k+c}{c+\ell} = n^{O(t)}$. Algorithm 2 can thus be implemented to run in time $n^{O(t)}$ as required. Moreover, it is clear from its definition that it always returns a solution $\mathcal{S}$, i.e., $|\mathcal{S}| \leq k$.

We proceed by proving that $\mathcal{S}$ satisfies (6.8b) of Lemma 6.8. Suppose first that the algorithm returns at Line 4. By the condition of the while loop from Line 2 to 3, we increase $\mathsf{cost}_{\mathcal{I}}(\mathcal{T}')$ by at most B each time we remove an element from $\mathcal{T}'$. We

remove at most c elements and thus we increase the total cost by at most cB. It follows that (6.8b) is immediately satisfied in this case.

From now on suppose instead that we reached Line 5 of Algorithm 2 and thus $|\mathcal{T}'| > k$. We shall exhibit sets $\mathcal{D}_0$ and $\mathcal{V}_0$ such that $|\mathcal{D}_0| + |\mathcal{V}_0| = k$, $|\mathcal{V}_0| < t$ and $\mathsf{cost}(\mathcal{S}_{\mathcal{D}_0,\mathcal{V}_0}) \leq \frac{1+3\delta}{1-3\delta}\mathsf{opt}_{\mathcal{I}}$. As Algorithm 2 selects $\mathcal{D}_0$ and $\mathcal{V}_0$ in one iteration and it returns the minimum cost solution, this concludes the proof of Lemma 6.8. In order to define the sets $\mathcal{D}_0$ and $\mathcal{V}_0$ it shall be convenient to use the following definitions.

Definition 6.9 *For every facility $i \in \mathcal{T}'$, let $L_i = d(i, \mathcal{T}' \setminus \{i\})$ be the distance from i to its nearest neighbor in $\mathcal{T}'$, and let $\ell_i = d(i, \mathsf{OPT}_{\mathcal{I}})$ be the minimum distance from i to any facility in $\mathsf{OPT}_{\mathcal{I}}$.*

For a facility $i \in \mathcal{T}'$, we say i is determined *if $\ell_i < \delta L_i$. Otherwise, we say i is* undetermined*.*

The sets $\mathcal{D}_0$ and $\mathcal{V}_0$ are now defined as follows. Set $\mathcal{D}_0$ contain all facilities in $i \in \mathcal{T}'$ that are determined. If we let f_i^* for $i \in \mathcal{D}_0$ be the facility in $\mathsf{OPT}_{\mathcal{I}}$ that is closest to i, then set $\mathcal{V}_0 := \mathsf{OPT}_{\mathcal{I}} \setminus \{f_i^* : i \in \mathcal{D}_0\}$. The intuition of $\mathcal{D}_0$ and $\mathcal{V}_0$ is that the solution $\mathcal{S}_{\mathcal{D}_0,\mathcal{V}_0}$ is very close to $\mathsf{OPT}_{\mathcal{I}}$: the only difference is the selection of f_i at Line 6 of Algorithm 2 instead of f_i^*. Since each $i \in \mathcal{D}_0$ is determined, selecting f_i greedily using a "locally" optimal strategy gives a good solution.

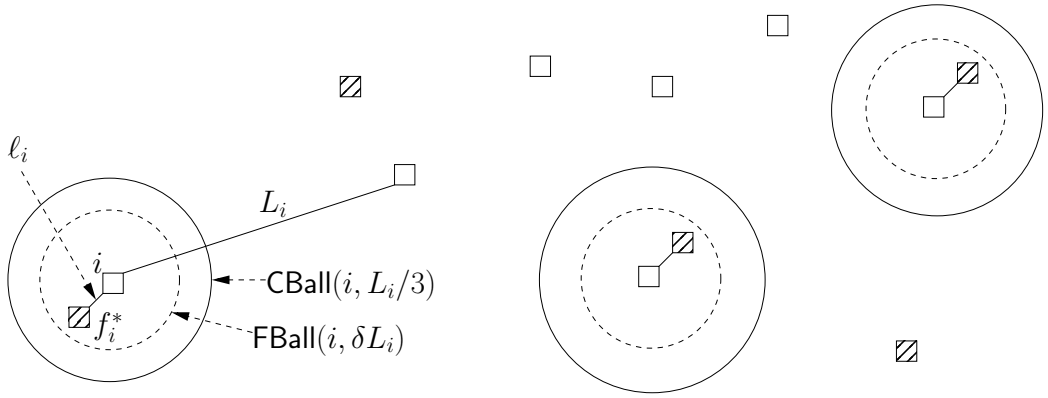

Figure 6.2: Definitions of $\mathcal{D}_0, \mathcal{V}_0$ and $\mathcal{U}_0$. Dashed and empty squares represent facilities in $\mathsf{OPT}_{\mathcal{I}}$ and $\mathcal{T}'$ respectively. $\mathcal{D}_0$ is the set of empty squares circles. A dashed circle represents $\mathsf{FBall}(i, \delta L_i)$ for a determined facility $i \in \mathcal{D}_0$. Thus, f_i^* is in the ball since $\ell_i < \delta L_i$. $\mathcal{U}_0(\mathcal{V}_0$, resp.) is the sets of empty (dashed, resp.) squares that are not inside any circle. A solid circle for $i \in \mathcal{D}_0$ represents the "care-set" of i.

We first show that sets $\mathcal{D}_0$ and $\mathcal{V}_0$ are indeed selected by Algorithm 2 and then we conclude the proof of the lemma by bounding the cost of $\mathcal{S}_{\mathcal{D}_0,\mathcal{V}_0}$.

Claim 6.10 *$|\mathcal{D}_0| + |\mathcal{V}_0| = k$ and $|\mathcal{V}_0| < t$.*

Proof of Claim. We start by proving that $|\mathcal{D}_0| + |\mathcal{V}_0| = k$. Recall that $\mathcal{V}_0 = \mathsf{OPT}_{\mathcal{I}} \setminus \{f_i^* : i \in \mathcal{D}_0\}$. It is not hard to see that $f_i^* \neq f_{i'}^*$ for two distinct facilities in $\mathcal{D}_0$. This is indeed true since $d(i, i') \geq \max(L_i, L_{i'}), d(i, f_i^*) \leq \delta L_i, d(i', f_{i'}^*) \leq \delta L_{i'}$ and $\delta \leq 1/8$.

Thus, $f^*(\mathcal{D}_0) := \{f_i^* : i \in \mathcal{D}_0\}$ has size $|\mathcal{D}_0|$, which in turn implies that (to simplify calculations we assume w.l.o.g. that $|\mathsf{OPT}_{\mathcal{I}}| = k$)

$$|\mathcal{V}_0| = |\mathsf{OPT}_{\mathcal{I}}| - |\mathcal{D}_0| = k - |\mathcal{D}_0|.$$

We proceed by proving $|\mathcal{V}_0| < t$. Note that the sets of determined and undetermined facilities partition $\mathcal{T}'$. Therefore, if we let $\mathcal{U}_0$ be the set of undetermined facilities, we have that $|\mathcal{D}_0| = |\mathcal{T}'| - |\mathcal{U}_0|$. Combining this with the above expression for $|\mathcal{V}_0|$ gives us

$$|\mathcal{V}_0| = k - |\mathcal{T}'| + |\mathcal{U}_0| \leq |\mathcal{U}_0|.$$

We complete the proof of the claim by showing that $|\mathcal{U}_0| < t$.

By the assumption that we reached Line 5 of Algorithm 2, we have $|\mathcal{T}'| > k$ and $\mathsf{cost}_{\mathcal{I}}(\mathcal{T}' \setminus \{i\}) > \mathsf{cost}_{\mathcal{I}}(\mathcal{T}') + B$ for every $i \in \mathcal{T}'$. Assume towards contradiction that $|\mathcal{U}_0| \geq t$. For every $i \in \mathcal{T}'$, let $\mathcal{C}_i$ be the set of clients in $\mathcal{C}$ connected to i in the solution $\mathcal{T}'$ and C_i be the total connection cost of these clients. Thus, $\mathsf{cost}_{\mathcal{I}}(\mathcal{T}') = \sum_{i \in \mathcal{T}'} C_i$. Take the facility $i \in \mathcal{U}_0$ with the minimum C_i. Then, we have $C_i \leq \mathsf{cost}_{\mathcal{I}}(\mathcal{T}')/t$. Let i' be the nearest neighbor of i in $\mathcal{T}'$; thus $d(i, i') = L_i$.

We shall remove the facility i from $\mathcal{T}'$ and connect the clients in $\mathcal{C}_i$ to i'. In order to consider incremental connection cost incurred by the operation, we divide $\mathcal{C}_i$ into two parts.

$\mathcal{C}_i \cap \mathsf{CBall}(i, \delta\xi L_i)$. Since i is undetermined, we have $\delta L_i \leq \ell_i$ and $\mathsf{CBall}(i, \delta\xi L_i) \subseteq \mathsf{CBall}(i, \xi\ell_i)$. As $\mathcal{I}$ is an A-sparse instance, i is not an A-dense facility. That is $(1-\xi)\,|\mathsf{CBall}(i, \xi\ell_i)|\,\ell_i \leq A$, implying

$$(1+\delta\xi)\,|\mathcal{C}_i \cap \mathsf{CBall}(i, \delta\xi L_i)|\,L_i \leq \frac{(1+\delta\xi)}{\delta(1-\xi)} A \leq A/(\delta\xi).$$

Then, as each client in $\mathcal{C}_i \cap \mathsf{CBall}(i, \delta\xi L_i)$ has distance at most $(1+\delta\xi)L_i$ to i' (by triangle inequalities), connecting all clients in $\mathcal{C}_i \cap \mathsf{CBall}(i, \delta\xi L_i)$ to i' can cost at most $A/(\delta\xi)$.

$\mathcal{C}_i \setminus \mathsf{CBall}(i, \delta\xi L_i)$. Consider any client j in this set. Since $d(j, i') \leq d(j, i) + L_i$ and $d(j, i) \geq \delta\xi L_i$, we have $\frac{d(j,i') - d(j,i)}{d(j,i)} \leq \frac{L_i}{\delta\xi L_i} = 1/(\delta\xi)$. Hence, the connection cost of a single client is increased by at most a factor $1/(\delta\xi)$. Therefore, the total connection cost increases by at most $C_i/(\delta\xi)$, which by the selection of i is at most $\mathsf{cost}_{\mathcal{I}}(\mathcal{T}')/(\delta\xi t)$.

Summing up the two quantities, removing i from $\mathcal{T}'$ can only increase the connection cost by at most $\frac{A + \mathsf{cost}_{\mathcal{I}}(\mathcal{T}')/t}{\delta\xi}$. As the while loop of Algorithm 2 ran for less than c iterations, $\mathsf{cost}_{\mathcal{I}}(\mathcal{T}') < \mathsf{cost}_{\mathcal{I}}(\mathcal{T}) + cB$. Therefore, $\frac{A + \mathsf{cost}_{\mathcal{I}}(\mathcal{T}')/t}{\delta\xi} < \frac{A + (\mathsf{cost}_{\mathcal{I}}(\mathcal{T}) + cB)/t}{\delta\xi}$ which since $t \geq 2c/(\delta\xi)$ is at most $\frac{A + \mathsf{cost}_{\mathcal{I}}(\mathcal{T})/t}{\delta\xi} + B/2 = B$ leading to a contradiction. Hence, $|\mathcal{U}_0| < t$ which concludes the proof of the claim. □

Having proved that the instance $\mathcal{S}_{\mathcal{D}_0, \mathcal{V}_0}$ is selected by Algorithm 2, we conclude the proof of Lemma 6.8 by bounding the cost of $\mathcal{S}_{\mathcal{D}_0, \mathcal{V}_0}$.

Recall that, for every $i \in \mathcal{D}_0$, Algorithm 2 opens one facility f_i in the ball $\mathsf{FBall}(i, \delta L_i)$. We know we can do this so that the connection cost of $\mathcal{C}$ is $\mathsf{opt}_{\mathcal{I}}$. We show that we can approximate this instance within a factor of $1 + O(\delta)$. Roughly speaking, if a client is far away from any of these balls, then it does not care which facilities to open inside the balls, up to a factor $1 + O(\delta)$. If a client is close to one of these balls, say $\mathsf{FBall}(i, \delta L_i)$, then we put the client into the "care-set" of i. For each i, we open a facility in the ball that is best for its care-set.

To be more specific, let the care-set of i be $\mathsf{CBall}(i, L_i/3)$ for any $i \in \mathcal{D}_0$. Clearly, the balls $\mathsf{CBall}(i, L_i/3), i \in \mathcal{D}_0$ are disjoint. As stated in Line 6 of Algorithm 2, we open a facility f_i in $\mathsf{FBall}(i, \delta L_i)$ that minimizes

$$\sum_{j \in \mathsf{CBall}(i, L_i/3)} \min \{d(f_i, j), d(j, \mathcal{V}_0)\}.$$

Claim 6.11 $\mathsf{cost}_{\mathcal{I}}(\mathcal{S}_{\mathcal{D}_0, \mathcal{V}_0}) \leq \frac{1+3\delta}{1-3\delta} \mathsf{opt}_{\mathcal{I}}$.

Proof of Claim. We compare $\mathsf{OPT}_{\mathcal{I}}$ and $\mathcal{S}_{\mathcal{D}_0, \mathcal{V}_0}$. Consider a client $j \in \mathsf{CBall}(i, L_i/3)$ for some $i \in \mathcal{D}_0$. The distance from j to any facility in $\mathsf{FBall}(i, \delta L_i)$ is at most $(1/3+\delta)L_i$. For any distinct facility $i' \in \mathcal{D}_0$, the distance from j to any facility in $\mathsf{FBall}(i', \delta L_{i'})$ is at least $d(i, i') - L_i/3 - \delta L_{i'} \geq d(i, i') - d(i, i')/3 - \delta d(i, i') = (2/3 - \delta) d(i, i') \geq (2/3 - \delta) L_i$. For $\delta \leq 1/8$, $1/3 + \delta < 2/3 - \delta$. Thus, j is either connected to f_i^* or some facility in $\mathcal{V}_0$ in the solution $\mathsf{OPT}_{\mathcal{I}}$. Noticing that we are selecting the best f_i for every $i \in \mathcal{D}_0$, the total connection cost of $\bigcup_{i \in \mathcal{D}_0} \mathsf{CBall}(i, L_i/3)$ in the solution $\mathcal{S}_{\mathcal{D}_0, \mathcal{V}_0}$ is at most that in $\mathsf{OPT}_{\mathcal{I}}$.

Now, consider a client j that is not in $\bigcup_{i \in \mathcal{D}_0} \mathsf{CBall}(i, L_i/3)$. If it is connected to some facility in $\mathcal{V}_0$ in the solution $\mathsf{OPT}_{\mathcal{I}}$, then the connection cost of j in the solution $\mathcal{S}_{\mathcal{D}_0, \mathcal{V}_0}$ can not be larger, since $\mathcal{V}_0 \subseteq \mathcal{S}$. Assume j is connected to $f_i^* \in \mathsf{CBall}(i, L_i/3)$ for some $i \in \mathcal{D}_0$. We compare $d(j, f_i^*)$ to $d(j, f_i)$:

$$\frac{d(j, f_i)}{d(j, f_i^*)} \leq \frac{d(j, i) + \delta L_i}{d(j, i) - \delta L_i} \leq \frac{L_i/3 + \delta L_i}{L_i/3 - \delta L_i} = \frac{1 + 3\delta}{1 - 3\delta}.$$

Thus, $\mathcal{S}_{\mathcal{D}_0, \mathcal{V}_0}$ has connection cost at most $\frac{1+3\delta}{1-3\delta} \mathsf{opt}_{\mathcal{I}}$. □

6.3 A Pseudo-Approximation Algorithm for k-Median

This section is dedicated to prove Theorem 6.5. Given a k-median instance $\mathcal{I} = (k, \mathcal{F}, \mathcal{C}, d)$, we first use Theorem 6.3 to obtain a bi-point solution $a\mathcal{S}_1 + b\mathcal{S}_2$ whose cost is at most 2 times the optimum cost of $\mathcal{I}$. Then it suffices to convert $a\mathcal{S}_1 + b\mathcal{S}_2$ into an $O(1/\epsilon)$-additive solution, whose cost is at most $\frac{1+\sqrt{3}+\epsilon}{2}$ times that of $a\mathcal{S}_1 + b\mathcal{S}_2$.

By the definition of bi-point solutions, we have $a + b = 1, |\mathcal{F}_1| \leq k < |\mathcal{F}_2|$ and $a|\mathcal{F}_1| + b|\mathcal{F}_2| = k$. It shall be convenient to think of $a\mathcal{F}_1 + b\mathcal{F}_2$ as a bipartite graph (see Figure 6.3) with vertex sets $\mathcal{F}_1$ and $\mathcal{F}_2$ and an edge for each client $j \in \mathcal{C}$ that is

incident to its closest facilities in $\mathcal{F}_1$ and $\mathcal{F}_2$ denoted by $i_1(j)$ and $i_2(j)$, respectively. Moreover, let $d_1(j) := d(j, i_1(j))$ and $d_2(j) := d(j, i_2(j))$. Then, the (fractional) connection cost of j in the bi-point solution is $ad_1(j) + bd_2(j)$. Similarly, if we let $d_1 := \mathsf{cost}(\mathcal{F}_1) = \sum_{j \in \mathcal{C}} d_1(j)$ and $d_2 := \mathsf{cost}(\mathcal{F}_2) = \sum_{j \in \mathcal{C}} d_2(j)$ then the bi-point solution has cost $ad_1 + bd_2$.

We shall prove Theorem 6.5 by exhibiting different algorithms based on the value of a. Specifically, we shall distinguish between the cases when a is in $\left(0, \frac{\sqrt{3}-1}{4}\right]$, $\left(\frac{\sqrt{3}-1}{4}, \frac{2}{1+\sqrt{3}}\right]$, and $\left(\frac{2}{1+\sqrt{3}}, 1\right]$. The simplest case is when $a \in \left(\frac{2}{1+\sqrt{3}}, 1\right]$: the solution where we open all facilities in $\mathcal{F}_1$ is then a $\frac{d_1}{ad_1+bd_2} \leq 1/a = (1+\sqrt{3})/2$ approximation.

For the two remaining cases, we will use the concept of *stars*. For each facility $i \in \mathcal{F}_2$ define $\pi(i)$ to be the facility in $\mathcal{F}_1$ that is closest to i. For a facility $i \in \mathcal{F}_1$, let $S_i = \{i' \in \mathcal{F}_2 : \pi(i') = i\}$. We think of S_i as a star with *center* i and *leaves* S_i. Note that by the definition of stars, we have that any client j with $i_2(j) \in S_i$ has $d(i_2(j), i) \leq d(i_2(j), i_1(j)) = d_2(j) + d_1(j)$ and therefore $d(j, i) \leq d(j, i_2(j)) + d(i_2(j), i) \leq 2d_2 + d_1$. Our algorithms will ensure that there is an open facility "close" to every client by always opening i if not all facilities in S_i are opened. The strategy for either opening the center of a star or its leaves (or sometimes both) depends on the value of a. We start in Section 6.3.1 by explaining the simpler case when $a \in \left(0, \frac{\sqrt{3}-1}{4}\right]$ and then complete the proof of Theorem 6.5 by considering the final case in Section 6.3.2.

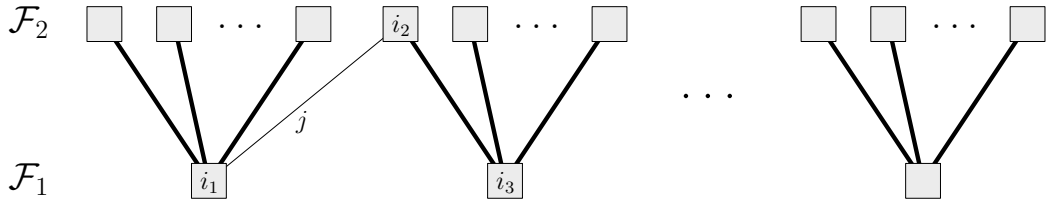

Figure 6.3: Depiction of the bipartite graph associated to a bi-point solution. The fat edges are the clients that form the edges of the stars. For clarity, we only depicted one client j that is not part of a star. Client j has distances $d(j, i_1) = d_1(j), d(j, i_2) = d_2$ and $d(j, i_3) \leq 2d_2(j) + d_1(j)$.

6.3.1 Algorithm for $a \in \left(0, \frac{\sqrt{3}-1}{4}\right]$

The idea behind our algorithm is that when a is small then we can open most facilities in $\mathcal{F}_2$. We shall do so by starting with the trivial solution $\mathcal{F}_1$ that we will improve by almost greedily selecting stars and open all their leaves while closing their centers.

As we will maintain the property that i is open if not all facilities in S_i are open, we have that the connection cost of a client j is $d_2(j)$ if $i_2(j)$ is open and at most $d_1(j) + 2d_2(j)$ otherwise. Consider the trivial solution where we open all facilities in $\mathcal{F}_1$. Then the total connection cost is upper-bounded by $\sum_j (d_1(j) + 2d_2(j))$. If we open the facilities in S_i instead of i this will save us the cost $\sum_{j \in \delta(S_i)} (d_1(j) + d_2(j))$,

where $\delta(S_i)$ denotes the clients that are incident to the facilities in S_i. This motivates the following linear program that maximizes the cost we will save compared to the trivial solution:

$$\max \sum_{i \in \mathcal{F}_1} \sum_{j \in \delta(S_i)} (d_1(j) + d_2(j)) x_i \qquad \text{subject to}$$

$$\sum_{i \in \mathcal{F}_1} x_i(|S_i| - 1) \leq k - |\mathcal{F}_1|$$

$$0 \leq x_i \leq 1, \quad \forall i \in \mathcal{F}_1$$

Intuitively, x_i takes value 1 if we open all the facilities in S_i and 0 if we open i. If $x_i = 1$, we need to open $|S_i| - 1$ more facilities (i.e, close i and open all facilities in S_i). Thus, the constraint says that we can only open k facilities. Note that this is a Knapsack LP and hence it is easy to see that an optimal solution has at most one fractional variable. Furthermore, $x_i = b$ is a feasible solution since $b\,|\mathcal{F}_2| - b\,|\mathcal{F}_1| = k - |\mathcal{F}_1|$. Therefore, the optimal solution to the LP has value at least $b(d_1 + d_2)$.

Consider an optimal solution to the Knapsack LP with at most 1 fractional variable. Then, we open all the facilities in S_i with $x_i = 1$, all the facilities $i \in \mathcal{F}_1$ with $x_i = 0$, and for the i with fractional x_i we open i and $\lceil x_i|S_i| \rceil$ facilities in S_i uniformly at random. (This step can easily be derandomized by greedily selecting the $\lceil x_i|S_i| \rceil$ facilities in S_i that maximizes the reduced cost.)

Note that we opened the facilities so that the (expected) saved cost compared to the trivial solution is at least the value of the optimal solution to the linear program. Therefore, this gives us a solution of (expected) cost at most $2d_2 + d_1 - b(d_2 + d_1) = (1 + a)d_2 + ad_1$. Also, the solution opens at most $k + 2$ facilities, where the additive term 2 comes from the star S_i with fractional x_i value.

Since we can assume that $d_2 \leq d_1$ (otherwise we can simply open all facilities in $\mathcal{F}_1$), the algorithm has an approximation guarantee of

$$\frac{(1 + a)d_2 + ad_1}{(1 - a)d_2 + ad_1} \leq (1 + 2a),$$

which is at most $\frac{1+\sqrt{3}}{2}$ if $a \leq \frac{\sqrt{3}-1}{4}$.

6.3.2 Algorithm for $a \in \left(\frac{\sqrt{3}-1}{4}, \frac{2}{1+\sqrt{3}}\right]$

In this subsection, we give the algorithm for the most complex case. To simplify the arguments, we give a randomized algorithm that can easily be derandomized using the standard method of conditional probabilities. The idea is that we wish to describe a randomized rounding that opens a facility in $\mathcal{F}_1$ with probability $\approx a$ and a facility in $\mathcal{F}_2$ with probability $\approx b$ and at the same time ensuring that there always is an open facility "close" to a client by maintaining the property: if i is not open then all facilities in S_i are open for all stars.

We now describe such a randomized rounding that takes a parameter $\eta > 0$ that balances the achieved approximation guarantee with the amount of additional facilities we open: the achieved approximation ratio is $(1+\eta)\frac{1+\sqrt{3}}{2}$ while we open at most $k + O(1/\eta)$ facilities. It shall be convenient to distinguish between large and small stars. We say that a star S_i is *large* if $|S_i| \geq 2/(ab\eta)$ and *small* otherwise. Moreover, we partition the small stars into $\lceil 2/(ab\eta) \rceil$ groups according to their sizes:

$$\mathcal{U}_h = \{i \in \mathcal{F}_1 : |S_i| = h\}, \qquad \text{for } h = 0, 1, \ldots, \lceil 2/(ab\eta) \rceil - 1.$$

The randomized algorithm can now be described as follows:

1: For each large star S_i: open i and open $\lfloor b(|S_i| - 1) \rfloor$ facilities in S_i uniformly at random.

2: For each group $\mathcal{U}_h$ of small stars: take a random permutation of the stars in $\mathcal{U}_h$, open the centers of the first $\lceil a|\mathcal{U}_h| \rceil + 1$ stars, and open all leaves of the remaining stars. In addition, if we let L be the number of already opened leaves subtracted from $bh|\mathcal{U}_h|$, then with probability $\lceil L \rceil - L$ open $\lfloor L \rfloor$ and with remaining probability open $\lceil L \rceil$ randomly picked leaves in the first $\lceil a|\mathcal{U}_h| \rceil + 1$ stars.

Note that for a large star the algorithm always opens its center and (almost) a b fraction of its leaves. For a group $\mathcal{U}_h$ of small stars, note that we open either the center (with probability at least a) or all leaves of a star. Moreover, we open the additional leaves so that in expectation exactly a b fraction of the leaves of the stars in $\mathcal{U}_h$ are opened.

We start by showing that the algorithm does not open too many facilities; we then continue by bounding the expected cost of the obtained solution.

Claim 6.12 *The algorithm opens at most $k + 3\lceil 2/(ab\eta) \rceil$ facilities.*

Proof of Claim. Recall that we have that $a|\mathcal{F}_1| + b|\mathcal{F}_2| = k$ and therefore

$$\sum_{i \in \mathcal{F}_1} (a + b|S_i|) = k. \tag{6.3}$$

First, consider a large star $i \in \mathcal{F}_1$, i.e., $a|S_i| \geq 1/(b\eta) \geq 1/\eta$. For such a star, the algorithm opens $1 + \lfloor b(|S_i| - 1) \rfloor \leq 1 + b(|S_i| - 1) = a + b|S_i|$ facilities, which is the contribution of star i to (6.3).

Second, consider a group $\mathcal{U}_h$ of small stars and let $m := |\mathcal{U}_h|$. When considering this group, the algorithm opens $\lceil am \rceil + 1 \leq am + 2$ facilities in $\mathcal{F}_1$, and at most

$$(m - \lceil am \rceil - 1)h + \lceil bhm - (m - \lceil am \rceil - 1)h \rceil \leq bhm + 1$$

facilities in $\mathcal{F}_2$. Thus, the total number of facilities open from the group $\mathcal{U}_h$ of small stars is at most $m(a+bh)+3$. As m is the size of $\mathcal{U}_h$ and $a+bh$ is the contribution of each star in $\mathcal{U}_h$ to (6.3), the statement follows from that we have at most $\lceil 2/(ab\eta) \rceil$ groups. □

We proceed by bounding the expected cost of the obtained solution. The intuition behind the following claim is that we have designed a randomized algorithm that

opens a facility in $\mathcal{F}_2$ with probability $\approx b$ and a facility in $\mathcal{F}_1$ with probability $\approx a$. Therefore, if we connect a client j to $i_2(j)$ with connection cost $d_2(j)$ if that facility is open, to $i_1(j)$ with connection cost $d_1(j)$ if that facility but not $i_2(j)$ is open, and to the center i of the star $S_i : i_2(j) \in S_i$ with connection cost at most $2d_2(j) + d_1(j)$ if neither $i_1(j)$ or $i_2(j)$ are opened (recall that i is open if not all facilities in S_i are open), then the expected connection cost of client j is at most

$$b \cdot d_2(j) + (1-b)a \cdot d_1(j) + ab(2d_2(j) + d_1(j)) = ad_1(j) + b(1+2a)d_2(j).$$

The following claim then follows by linearity of expectation.

Claim 6.13 *The algorithm returns a solution with expected cost at most*

$$(1+\eta)\left(ad_1 + b(1+2a)d_2\right).$$

Proof of Claim. Focus on a client j with $i_1(j) = i_1$ and $i_2(j) = i_2$ as depicted in Figure 6.3. Let $i_3 = \pi(i_2)$ be the closest facility in $\mathcal{F}_1$ to i_2, i.e., i_3 is the center of the star S_{i_3} with $i_2 \in S_{i_3}$. Notice that $d(i_3, i_2) \le d(i_1, i_2) \le d_1(j) + d_2(j)$ by the definition of π. Thus, $d(j, i_3) \le d_2(j) + d(i_3, i_2) \le d_1(j) + 2d_2(j)$. We connect j to i_2, if i_2 is open; otherwise, we connect j to i_1 if i_1 is open. We connect j to i_3 if both i_1 and i_2 are not open. (Notice that for a star S_i, if i is not open, then all facilities in S_i are open. Thus, either i_2 or i_3 is open.) Connecting j to the nearest open facility can only give smaller connection cost. By abusing notations we let i_1 (i_2, resp.) denote the event that i_1 (i_2, resp.) is open and $\bar{i}_1$ ($\bar{i}_2$, resp.) denote the event that i_1 (i_2, resp.) is not open. Then, we can upper bound the expected connection cost of j by

$$\Pr[i_2] \cdot d_2(j) + \Pr\left[i_1\bar{i}_2\right] \cdot d_1(j) + \Pr\left[\bar{i}_1\bar{i}_2\right] \cdot (2d_2(j) + d_1(j)),$$

which, by substituting $\Pr\left[\bar{i}_1\bar{i}_2\right] = 1 - \Pr[i_2] - \Pr\left[i_1\bar{i}_2\right]$, equals

$$\left(2 - \Pr\left[i_2\right] - 2\Pr\left[i_1\bar{i}_2\right]\right) d_2(j) + \left(1 - \Pr\left[i_2\right]\right) d_1(j). \tag{6.4}$$

We upper bound this expression by analyzing these probabilities.

Let us start with $\Pr\left[i_1\bar{i}_2\right]$. If $i_2 \in S_{i_1}$ (i.e., $i_1 = i_3$) then i_1 is always open if i_2 is closed and thus we have $\Pr\left[i_1\bar{i}_2\right] = \Pr\left[\bar{i}_2\right]$. If S_{i_1} is a large star, then i_1 is always open and we also have $\Pr\left[i_1\bar{i}_2\right] = \Pr\left[\bar{i}_2\right]$. In both cases, we have $\Pr\left[i_1\bar{i}_2\right] = 1 - \Pr[i_2]$.

We now consider the case where S_{i_1} is a small star in a group $\mathcal{U}_h$ with $m := |\mathcal{U}_h|$ and $i_1 \neq i_3$. Note that if S_{i_3} is either a large star or a small star not in $\mathcal{U}_h$ then the events i_1 and $\bar{i}_2$ are independent. We have thus in this case that

$$\begin{aligned}\Pr\left[i_1\bar{i}_2\right] &= \Pr[i_1] \cdot (1 - \Pr[i_2]) \\ &= \frac{\lceil am \rceil + 1}{m} \cdot (1 - \Pr[i_2])\end{aligned}$$

It remains to consider the case when S_{i_3} is a star in $\mathcal{U}_h$. Notice that the dependence between i_1 and i_2 comes from that if i_2 is closed then i_3 is opened. Therefore, we

have

$$\begin{aligned}\Pr\left[i_1\bar{i}_2\right] &= \Pr\left[i_1|\bar{i}_2\right]\cdot(1-\Pr[i_2]) \\ &= \frac{\lceil am\rceil+1-1}{m}\cdot(1-\Pr[i_2]).\end{aligned}$$

We have thus showed that $\Pr\left[i_1\bar{i}_2\right]$ is always at least $a\cdot(1-\Pr\left[i_2\right])$. Substituting in this bound in (6.4) allows us to upper bound the connection cost of j by

$$(2b+(2a-1)\Pr\left[i_2\right])\,d_2(j)+(1-\Pr\left[i_2\right])\,d_1(j).$$

We proceed by analyzing $\Pr\left[i_2\right]$. On the one hand, if i_2 is a leaf of some big star S_i with $s=|S_i|\geq 2/(ba\eta)$ then $\Pr[i_2]=\frac{\lfloor b(s-1)\rfloor}{s}$ is greater than $b-2/s\geq b(1-a\eta)$ and smaller than b. On the other hand, if i_2 is a leaf of a small star S_i in group $\mathcal{U}_h$ with $m:=|\mathcal{U}_h|$ then in expectation we open exactly a b fraction of the leaves so $\Pr\left[i_2\right]=b$. We have thus that $b(1-a\eta)\leq\Pr\left[i_2\right]\leq b$. Since $(1+\eta)\cdot(1-a\eta)\geq 1$ we have that the expected connection cost of facility j is at most $(1+\eta)$ times

$$(2b+(2a-1)b)d_2(j)+(1-b)d_1(j)=b(1+2a)d_2(j)+ad_1(j).$$

The claim now follows by summing up the expected connection cost of all clients. □

We complete the analysis by balancing the solution obtained by running our algorithm with the trivial solution of cost d_1 that opens all facilities in $\mathcal{F}_1$.

Claim 6.14 *We have that* $\min\{d_1, ad_1+b(1+2a)d_2\}\leq\frac{1+\sqrt{3}}{2}(ad_1+bd_2)$.

Proof of Claim. We change d_1 and d_2 slightly so that ad_1+bd_2 does not change. Apply the operation to the direction that increases the left-hand-side of the inequality. This operation can be applied until one of the 3 conditions is true: (1) $d_1=0$; (2) $d_2=0$ or (3) $d_1=ad_1+b(1+2a)d_2$.

For the first two cases, the inequality holds. In the third case, we have $d_1=(1+2a)d_2$. Then $\frac{d_1}{ad_1+bd_2}=\frac{1+2a}{a(1+2a)+1-a}=\frac{1+2a}{1+2a^2}$. The maximum value of the quantity is $\frac{1+\sqrt{3}}{2}$, achieved when $a=\frac{\sqrt{3}-1}{2}$. □

We have shown that, by letting $\eta=\epsilon/(1+\sqrt{3})$, we can efficiently obtain a $O(1/\epsilon)$-additive $\frac{1+\sqrt{3}+\epsilon}{2}$-approximation to a bi-point solution with constant a and b, which proves Theorem 6.5 when $a\in\left(\frac{\sqrt{3}-1}{4},\frac{2}{1+\sqrt{3}}\right]$.

6.4 Discussion

We have given a $1+\sqrt{3}+\epsilon$-approximation algorithm for k-median, improving upon the previous best $3+\epsilon$-approximation algorithm. Besides the improved approximation guarantee, we believe that the most interesting technical contribution is Theorem 6.4,

namely that we can approximate k in k-median without loss of generality. More specifically, any pseudo-approximation algorithm which outputs a solution that opens $k+O(1)$ facilities can be turned into an approximation algorithm with essentially the same approximation guarantee but that only opens k facilities.

For k-median this new point of view has the potential to overcome a known barrier for obtaining an approximation algorithm that matches the $1+2/e$ hardness of approximation result: the lower bound of 2 on the integrality gap of the natural LP for k-median. In particular, the known instances that give the integrality gap of 2 vanish if we allow $k+1$ open facilities in the integral solution. Following our work, we therefore find it important to further understand the following open question: what is the maximum ratio between the cost of the optimum solution with $k+O(1)$ open facilities, and the value of the LP with k open facilities? One can note that the hardness of approximation reduction in [46] implies that the integrality gap is at least $1+2/e$ even if we open $k+o(k)$ facilities. Moreover our $O(1/\epsilon)$-additive approximation for bi-point solutions achieving a guarantee of $\frac{1+\sqrt{3}+\epsilon}{2} < 1+2/e$ shows that the worst case integrality gap instances are not of this type when pseudo-approximation is allowed.

Finally, we would like to mention that Theorem 6.4 naturally motivates the question if other hard constraints can be relaxed to soft constraints with a "violation-dependent" increase in the runtime. Soft constraints often greatly help when designing algorithms. For example, the capacitated versions of facility location and k-median are notorious problems when the capacities are hard constraints but better approximation algorithms are known if the capacities are allowed to be slightly violated (see e.g. [30]). As our approach was inspired by studying the power of the Sherali-Adams hierarchy [76] for the k-median problem, we believe that a promising research direction is to understand the power of that hierarchy and the stronger Lasserre hierarchy [59] when applied to these kinds of problems.

Bibliography

[1] Alok Aggarwal, Maria Klawe, David Lichtentein, Nathan Linial, and Avi Wigderson. Multi-layer grid embeddings. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, pages 186–196, Washington, DC, USA, 1985. IEEE Computer Society.

[2] Alok Aggarwal, Jon Kleinberg, and David P. Williamson. Node-disjoint paths on the mesh and a new trade-off in vlsi layout. *SIAM J. Comput.*, 29(4):1321–1333, February 2000.

[3] Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 277–286, Washington, DC, USA, 2010. IEEE Computer Society.

[4] Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Combinatorica*, 30(5):485–520, 2010.

[5] Matthew Andrews and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 276–283. ACM, 2005.

[6] A. Archer, R. Rajagopalan, and D. B. Shmoys. Lagrangian relaxation for the k-median problem: new insights and continuity properties. In *In Proceedings of the 11th Annual European Symposium on Algorithms*, pages 31–42, 2003.

[7] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[8] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC '01, pages 21–29, New York, NY, USA, 2001. ACM.

[9] Yonatan Aumann and Yuval Rabani. Improved bounds for all optical routing. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pages 567–576, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[10] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[11] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a ptas for k-median and k-means clustering. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 309–318, Washington, DC, USA, 2010. IEEE Computer Society.

[12] Baruch Awerbuch, Rainer Gawlick, Tom Leighton, and Yuval Rabani. On-line admission control and circuit routing for high performance computing and communication. In *Proc. 35th IEEE Symp. on Foundations of Computer Science*, pages 412–423, 1994.

[13] Yossi Azar and Oded Regev. Strongly polynomial algorithms for the unsplittable flow problem. In *In Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 15–29, 2001.

[14] M. L. Balinksi. On finding integer solutions to linear programs. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, pages 225–248, 1966.

[15] Alok Baveja and Aravind Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *Mathematics of Operations Research*, 25:2000, 2000.

[16] P. S. Bradley, Fayyad U. M., and O. L. Mangasarian. Mathematical programming for data mining: Formulations and challenges. *INFORMS Journal on Computing*, 11:217–238, 1998.

[17] Andrei Z. Broder, Alan M. Frieze, Stephen Suen, and Eli Upfal. Optimal construction of edge-disjoint paths in random graphs. In *Proc. 5th ACM-SIAM SODA*, pages 603–612, 1994.

[18] Andrei Z. Broder, Alan M. Frieze, and Eli Upfal. Existence and construction of edge-disjoint paths on expander graphs. *SIAM J. Comput.*, pages 976–989, 1994.

[19] J. Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *APPROX '07/RANDOM '07: Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 29–43, Berlin, Heidelberg, 2007. Springer-Verlag.

[20] J. Byrka, M. Ghodsi, and A. Srinivasan. Lp-rounding algorithms for facility-location problems. *in arxiv1007.3611*, 2010.

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

[22] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, STOC '99, pages 1–10, New York, NY, USA, 1999. ACM.

[23] C. Chekuri, S. Khanna, and F. B. Shepherd. Edge-disjoint paths in planar graphs. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 71–80, Washington, DC, USA, 2004. IEEE Computer Society.

[24] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. The all-or-nothing multicommodity flow problem. In László Babai, editor, *STOC*, pages 156–165. ACM, 2004. A full version at http://www.math.mcgill.ca/~bshepherd/PS/all.pdf.

[25] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 183–192, New York, NY, USA, 2005. ACM.

[26] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Edge-disjoint paths in planar graphs with constant congestion. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 757–766, New York, NY, USA, 2006. ACM.

[27] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(1):137–146, 2006.

[28] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3, August 2007.

[29] F. A. Chudak and D. B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2004.

[30] J. Chuzhoy and Y. Rabani. Approximating k-median with non-uniform capacities. In *SODA*, pages 952–958, 2005.

[31] Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *arXiv:1107.2554v1*, 2011.

[32] Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 233–242, Washington, DC, USA, 2012. IEEE Computer Society.

[33] M. Conforti, R. Hassin, and R. Ravi. Reconstructing flow paths. *Operations Research Letters*, 31:273–276, 2003.

[34] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[35] A. Frank. On connectivity properties of Eulerian digraphs. *Ann. Discrete Math.*, 41, 1989.

[36] Andras Frank. *Paths, Flows, and VLSI Layout*, chapter Packing Paths, Circuits, and Cuts—A Survey. Springer-Verlag.

[37] Alan M. Frieze. Edge-disjoint paths in expander graphs. *SIAM Journal On Computing*, 30:2001, 2000.

[38] David Gale and Lloyd Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 1:9–14, 1962.

[39] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)-cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1995.

[40] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees, with applications to matching and set cover. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *ICALP*, volume 700 of *Lecture Notes in Computer Science*, pages 64–75. Springer, 1993.

[41] S Guha and S Khuller. Greedy strikes back: Improved facility location algorithms. In *Journal of Algorithms*, pages 649–657, 1998.

[42] A. Hajnal and E. Szemerédi. Proof of a conjecture of P. Erdos. *Combinatorial Theory and its Applications*, pages 601–623, 1970.

[43] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.

[44] Bill Jackson. Some remarks on arc-connectivity, vertex splitting, and orientation in graphs and digraphs. *J. Graph Theory*, 12:429–436, 1998.

[45] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50:795–824, November 2003.

[46] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 731–740, New York, NY, USA, 2002. ACM.

[47] K Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[48] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[49] Ken-ichi Kawarabayashi and Yusuke Kobayashi. An $O(\log n)$-approximation algorithm for the disjoint paths problem in Eulerian planar graphs and 4-edge-connected planar graphs. In *APPROX-RANDOM'10*, pages 274–286, 2010.

[50] Ken-ichi Kawarabayashi and Yusuke Kobayashi. Breaking $O(n^{1/2})$-approximation algorithms for the edge-disjoint paths problem with congestion two. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 81–88. ACM, 2011.

[51] Rohit Khandekar, Satish Rao, and Umesh V. Vazirani. Graph partitioning using single commodity flows. In Jon M. Kleinberg, editor, *STOC*, pages 385–390. ACM, 2006.

[52] Jon Kleinberg and Ronitt Rubinfeld. Short paths in expander graphs. In *In Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 86–95, 1996.

[53] Jon M. Kleinberg. An approximation algorithm for the disjoint paths problem in even-degree planar graphs. In *FOCS'05*, pages 627–636, 2005.

[54] Jon M. Kleinberg and Éva Tardos. Disjoint paths in densely embedded graphs. In *FOCS*, pages 52–61. IEEE Computer Society, 1995.

[55] Jon M. Kleinberg and Éva Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *J. Comput. Syst. Sci.*, 57(1):61–73, 1998.

[56] Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99:63–87, 2004.

[57] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, SODA '98, pages 1–10, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[58] A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Mathematical Models in Marketing*, 9(9):643–666, July 1963.

[59] J. B. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM Journal on Optimization*, 12(3):756–769, 2002.

[60] Lap Chi Lau. An approximate max-steiner-tree-packing min-steiner-cut theorem. *Combinatorica*, 27(1):71–90, February 2007.

[61] F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46:787–832, 1999.

[62] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Automata, Languages and Programming - 38th International Colloquium (ICALP)*, pages 77–88, 2011.

[63] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 901–910, New York, NY, USA, 2013. ACM.

[64] J. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Inf. Process. Lett.*, 44:245–249, December 1992.

[65] J. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC), Victoria, British Columbia, Canada*, pages 771–782, 1992.

[66] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Proceedings of 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 577–591, 1994.

[67] W. Mader. A reduction method for edge connectivity in graphs. *Ann. Discrete Math.*, 3:145–164, 1978.

[68] M. Mahdian, Y. Ye, and J. Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.

[69] A. S. Manne. Plant location under economies-of-scale-decentralization and computation. In *Managment Science*, 1964.

[70] Lorenzo Orecchia, Leonard J. Schulman, Umesh V. Vazirani, and Nisheeth K. Vishnoi. On partitioning graphs via single commodity flows. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 461–470, New York, NY, USA, 2008. ACM.

[71] Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, December 1987.

[72] Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

[73] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In *Paths, Flows and VLSI-Layout*. Springer-Verlag, 1990.

[74] R.Rivest. The placement and interconnect system. In *Proceedings of the 19th Design and Automation Conference*, 1982.

[75] Loïc Seguin-Charbonneau and F. Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. In Rafail Ostrovsky, editor, *FOCS*, pages 200–209. IEEE, 2011.

[76] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.*, 3(3):411–430, 1990.

[77] D. B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274, New York, NY, USA, 1997. ACM.

[78] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 661–670. ACM, 2007.