ABSTRACT

Title of dissertation: Online Network Design under Uncertainty

Sina Dehghani, Doctor of Philosophy, 2017

Dissertation directed by: Professor MohammadTaghi Hajiaghayi
Department of Computer Science

Today, computer and information networks play a significant role in the success of businesses, both large and small. Networks provide access to various services and resources to end users and devices. There has been extensive research on designing networks according to numerous criteria such as cost-efficiency, availability, adaptivity, survivability, among others. In this dissertation, we revisit some of the most fundamental network design problems in the presence of uncertainty.

In most realistic models, we are forced to make decisions in the presence of an incomplete input, which is the source of uncertainty for an optimization algorithm. There are different types of uncertainty. For example, in stochastic settings, we may have some random variables derived from some known/unknown distributions. In online settings, the complete input is not known in a-priori and pieces of the input become available sequentially; leaving the algorithm to make decisions only with partial data.

In this dissertation, we consider network design and network optimization problems with uncertainty. In particular, we study online bounded-degree Steiner network design, online survivable network design, and stochastic k-server. We analyze their complexity and design competitive algorithms for them.

ONLINE NETWORK DESIGN UNDER UNCERTAINTY

by

Sina Dehghani

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Advisory Committee:
Prof. MohammadTaghi Hajiaghayi, Chair/Advisor
Prof. Andrew Childs
Prof. Larry S. Davis
Prof. Ali Haghani
Prof. David Mount
Prof. Aravind Srinivasan

© Copyright by
Sina Dehghani
2017

Dedicated to my parents and my brother.

Acknowledgments

I would like to thank my advisor Prof. MohammadTaghi Hajiaghayi for his extraordinary support. Many thanks to Prof. Andrew Childs, Prof. Larry S. Davis, Prof. Ali Haghani, Prof. David Mount, and, Prof. Aravind Srinivasan for serving on my dissertation committee. Specil thanks to my family for their love and support.

Table of Contents

List of Figures

Chapter 1: Introduction

Computer and information networks have become essential in almost any operation. As a result, network design and network optimization has become an active and important line of research. The Steiner network design problem – which asks for a subgraph that connects a given set of vertices – is perhaps one of the most representative problems in this class. There has been many books written on this topic (see e.g. [1–6]). We also study several variants of this problem such as the degree-bounded variant and the survivable network design problem. We also study a variant of the celebrated k-server problems, which is one of the most important online network optimization problems.

In practice, one main challenge to many network processes is uncertainty. Traditionally in computer science problems, an algorithm initially has complete uncertainty about the input instance, and when the input is given, the algorithm receives full information. However in real world, not only the algorithm may have some prior information about the instance (e.g. the input may include random variables drawn from distributions which are known to the algorithm in advance), but also the algorithm may have incomplete information even after the input is given (e.g. jobs can have stochastic running time, online algorithms can only access a fraction of the information at each time).

In this dissertation, we deal with online optimization problems in networks. In an online setting an algorithm sequentially receives only a portion of the input and has to make a decision. In this setting, the classical assumption is that the input is chosen by some adversary and the outcome is compared to some optimal offline algorithm. This evaluates an algorithm in the worst case scenario. In real world, however, adversarial scenarios are sometimes very unlikely. Moreover having no information about the future input is not always reasonable. Therefore, a new line of research is necessary to study these onlint optimization problems in a more realistic setting. An effective and fruitful model to rule out these issues is adding stochastic assumptions. This has been applied to many problems in different areas. Thus, in this dissertation, we revisit two classical online problems in a stochastic setting, where the input is drawn from some known distribution at each online step. In the following we describe each chapter of this dissertation independently.

In Chapter 2, we initiate the study of degree-bounded network design problems in the online setting. The degree-bounded Steiner tree problem asks for a subgraph with minimum degree that connects a given set of vertices. In this chapter, we deal with its well-studied generalization called the degree-bounded Steiner forest problem where the connectivity demands are represented by vertex pairs that need to be individually connected. In the classical online model, the input graph is given offline but the demand pairs arrive sequentially in online steps. The selected subgraph starts off as the empty subgraph, but has to be augmented to satisfy the new connectivity constraint in each online step. The goal is to be competitive against an adversary that knows the input in advance.

The standard techniques for solving degree-bounded problems often fall in the category of *iterative* and *dependent* rounding techniques. Unfortunately, these rounding methods are inherently difficult to adapt to an online settings since the underlying fractional solution may change dramatically in between the rounding steps. Indeed, this might be the very reason that despite many advances in the online network design paradigm in the past two decades, the natural family of degree-bounded problems has remained widely open.

In this chapter, we design an intuitive greedy-like algorithm that achieves a competitive ratio of $O(\log n)$ where n is the number of vertices. We show that no (randomized) algorithm can achieve a (multiplicative) competitive ratio $o(\log n)$; thus our result is asymptotically tight. We further show strong hardness results for the group Steiner tree and the edge-weighted variants of degree-bounded connectivity problems.

Fürer and Raghavachari resolved the offline variant of degree-bounded Steiner forest in their paper in SODA'92. Since then, the family of degree-bounded network design problems has been extensively studied in the literature resulting in the development of many interesting tools and numerous papers on the topic. We hope that our approach and its dual analysis, paves the way for solving the *online* variants of the classical problems in this family of problems.

In Chapter 3, we design the first online algorithm with poly-logarithmic competitive ratio for the *edge-weighted degree-bounded Steiner forest* (EW-DB-ST) problem and its generalized variant. We obtain our result by demonstrating a new generic approach for solving mixed packing/covering integer programs in the online paradigm. In EW-DB-ST, we are given an edge-weighted graph with a degree bound for every vertex. Given a root vertex in advance, we receive a sequence of terminal vertices in an online manner. Upon the arrival of a terminal, we need to augment our solution subgraph to connect the new terminal to the root. The goal is to minimize the total weight of the solution while respecting the degree bounds on the vertices. In the offline setting, *edge-weighted degree-bounded Steiner tree* (EW-DB-ST) and its many variations have been extensively studied since early eighties. Unfortunately, the recent advancements in the online network design problems are inherently difficult to adapt for degree-bounded problems. In particular, it is not known whether the fractional solution obtained by standard primal-dual techniques for mixed packing/covering LPs can be rounded online. In contrast, in this chapter we obtain our result by using structural properties of the optimal solution, and reducing the EW-DB-ST problem to an exponential-size mixed packing/covering integer program in which every variable appears only once in covering constraints. We then design a generic *integral* algorithm for solving this restricted family of IPs.

As mentioned above, we demonstrate a new technique for solving mixed packing/covering integer programs. Define the *covering frequency* k of a program as the maximum number of covering constraints in which a variable can participate. Let m denote the number of packing constraints. We design an online *deterministic integral algorithm* with competitive ratio of $O(k \log m)$ for the mixed packing/covering integer programs. We prove the tightness of our result by providing a matching lower bound for any randomized algorithm. We note that our solution solely depends on m and k. Indeed, there can be exponentially many variables. Furthermore, our algorithm directly provides an integral solution, even if the integrality gap of the program is unbounded. We believe this technique can be used as an interesting alternative for the standard primal-dual techniques in solving online problems.

In Chapter 4 we study online survivable network design. In an instance of the *network design* problem, we are given a graph $G = (V, E)$, an edge-cost function $c : E \rightarrow \mathbb{R}^{\geq 0}$, and a connectivity criteria. The goal is to find a minimum-cost subgraph H of G that meets the connectivity requirements. An important family of this class is the *survivable network design problem* (SNDP): Given non-negative integers r_{uv} for each pair $u, v \in V$, the solution subgraph H should contain r_{uv} edge-disjoint paths for each pair u and v.

While this problem is known to admit good approximation algorithms in the offline case, the problem is much harder in the online setting. Gupta, Krishnaswamy, and Ravi [7] (STOC'09) were the first to consider the online survivable network design problem. They demonstrate an elegant algorithm with competitive ratio of $O(k \log^3 n)$, where $k = \max_{u,v} r_{uv}$.

The competitive ratio of the algorithm by Gupta *et al.* grows linearly in k. Indeed, an important open problem in the online community [7, 8] is whether the linear dependency on k can be reduced to a logarithmic dependency. Moreover, the $O(\log^3 n)$ factor is also not plausible in practice. In this chapter, we show that this problem can be circumvented by two different approaches, i.e. considering the stochastic variant of the problem and allowing small congestion on the edges.

We first show that a greedy algorithm does surprisingly well, if we relax the connectivity requirements by a constant factor. In particular we prove the greedy algorithm is $O(\log^2 n \log k)$-competitive if we provide $r_{uv}/2$ edge-disjoint paths between u and v, instead of r_{uv} edge-disjoint paths. Our result is very similar in spirit to the work of Chuzhoy and Li [9] (FOCS'12) in which the authors give a polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2.

Then we study the stochastic version of the problem. We consider the i.i.d. model, where each online demand is drawn from a single probability distribution, the unknown distribution model, where again every demand is drawn from a single but unknown probability distribution, and the prophet setting of the problem, where each online demand is drawn from a (possibly) different probability distribution. We provide constant competitive algorithms for the i.i.d. and the prophet setting of the problem, which surprisingly shows that knowing some stochastic information about the online input can significantly improve the competitive ratio. We also provide $O(\log n)$-competitive algorithm for the unknown distribution model, which is almost tight. To provide a competitive algorithm for the prophet setting, we present a general framework to obtain competitive algorithms for prophet setting, using oblivious algorithms for the i.i.d. model. Interestingly, this technique allows us to obtain competitive algorithms for other fundamental online problems such as set cover, vertex cover, and facility location in the prophet setting.

In Chaapter 5, we study a stochastic variant of the celebrated k-server problem. In the k-server problem, we are required to minimize the total movement of k servers that are serving an online sequence of t requests in a metric. In the stochastic setting we are given t independent distributions $\langle P_1, P_2, \ldots, P_t \rangle$ in advance, and at every time step i a request is drawn from P_i.

Designing the optimal online algorithm in such setting is NP-hard, therefore the emphasis of our work is on designing an approximately optimal online algorithm. We first show a structural characterization for a certain class of *non-adaptive* online algorithms. We prove that in general metrics, the best of such algorithms has a cost of no worse than three times that of the optimal online algorithm. Next, we present an integer program that finds the optimal algorithm of this class for any arbitrary metric. Finally by rounding the solution of the linear relaxation of this program, we present an online algorithm for the stochastic k-server problem with an approximation factor of 3 in the line and circle metrics and factor of $O(\log n)$ in general metrics. In this way, we achieve an approximation factor that is independant of k, the number of servers.

Furthermore, we extend our results to the correlated setting where the probability of a request arriving at a certain point depends not only on the time step but also on the previously arrived requests.

Chapter 2: Online Degree-Bounded Steiner Network Design

2.1 Introduction

The problem of satisfying connectivity demands on a graph while respecting given constraints has been a pillar of the area of network design since the early seventies [10–14]. The problem of DEGREE-BOUNDED SPANNING TREE, introduced in Garey and Johnson's *Black Book* of NP-Completeness [15], was first investigated in the pioneering work of Fürer and Raghavachari [16] (Allerton'90). In the DEGREE-BOUNDED SPANNING TREE problem, the goal is to construct a spanning tree for a graph $G = (V, E)$ with n vertices whose maximal degree is the smallest among all spanning trees. Let b^* denote the maximal degree of an optimal spanning tree. Fürer and Raghavachari [16] give a parallel approximation algorithm which produces a spanning tree of degree at most $O(\log(n)b^*)$.

Agrawal, Klein, and Ravi ([17]) consider the following generalizations of the problem. In the DEGREE-BOUNDED STEINER TREE problem we are only required to connect a given subset $T \subseteq V$. In the even more general DEGREE-BOUNDED STEINER FOREST problem the demands consist of vertex pairs, and the goal is to output a subgraph in which for every demand there is a path connecting the pair. They design an algorithm that obtains a multiplicative approximation factor of $O(\log(n))$. Their main technique is to reduce the problem to minimizing congestion under integral concurrent flow restrictions and to then use the randomized rounding approach due to Raghavan and Thompson ([18]).

Shortly after the work of Agrawal *et al.*, Fürer and Raghavachari [19] significantly improved the result for DEGREE-BOUNDED STEINER FOREST by presenting an algorithm which produces a Steiner forest with maximum degree at most $b^* + 1$. They show that the same guarantee carries over to the *directed* variant of the problem as well. Their result is based on reducing the problem to that of computing a sequence of maximal matchings on certain auxiliary graphs. This result settles the approximability of the problem, as computing an optimal solution is NP-hard even in the spanning tree case.

In this chapter, we study degree-bounded network design problems in an *online* setting, where connectivity demands appear over time and must be immediately satisfied. We first design a deterministic algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST with a logarithmic competitive ratio. Then we show that this competitive ratio is asymptotically best possible by proving a matching lower bound for randomized algorithms that already holds for the Steiner tree variant of the problem.

In the offline scenario, the results of Fürer, Raghavachari [16,19] and Agrawal, Klein, Ravi [17] were the starting point of a very popular line of work on various degree-bounded network design problems [20–25]. We refer the reader to the next sections for a brief summary. One particular variant that has been extensively studied is the *edge-weighted* DEGREE-BOUNDED SPANNING TREE. Initiated by Marathe *et al.* ([20]), in this version, we are given a weight function over the edges and a bound b on the maximum degree of a vertex. The goal is to find a minimum-weight spanning tree with maximum degree at most b. The groundbreaking results obtained by Goemans ([21]) and Singh and Lau ([26]) settle the problem by giving an algorithm that computes a minimum-weight spanning tree with degree at most $b + 1$. A slightly worse result is obtained by Singh and Lau ([23]) for the Steiner tree variant. Unfortunately, in the online setting it is not possible to obtain a comparable result. We show that for any (randomized) algorithm $\mathcal{A}$ there exists a request sequence such that $\mathcal{A}$ outputs a sub-graph that either has weight $\Omega(n) \cdot \mathrm{OPT}_b$ or maximum degree $\Omega(n) \cdot b$.

2.1.1 Our Contributions

In the online variant of DEGREE-BOUNDED STEINER FOREST , we are given the graph G in advance, however, demands arrive in an online fashion. At online step i, a new demand (s_i, t_i) arrives. Starting from an empty subgraph, at each step the online algorithm should augment its solution so that the endpoints of the new demand s_i and t_i are connected. The goal is to minimize the maximum degree of the solution subgraph. In the *non-uniform* variant of the problem, a degree bound $b_v \in \mathbb{R}^+$ is given for every vertex v. For a subgraph H and a vertex v, let $\deg_H(v)$ denote the degree of v in H. The *load* of a vertex is defined as the ratio $\deg_H(v)/b_v$. In the non-uniform variant of ONLINE DEGREE-BOUNDED STEINER FOREST, the goal is to find a subgraph satisfying the demands while minimizing the maximum load of a vertex.

Our algorithm is a simple and intuitive greedy algorithm. Upon the arrival of a new demand (s_i, t_i), the greedy algorithm (GA) satisfies the demand by choosing an (s_i, t_i)-path P_i such that after augmenting the solution with P_i, the maximum load of a vertex in P_i is minimum. A main result of our work is to prove that the maximum load of a vertex in the output of GA is within a logarithmic factor of OPT, the maximum load of a vertex in an optimal offline solution which knows all the demands in advance.

Theorem 2.1. *The algorithm GA produces an output with maximum load at most $O(\log n) \cdot OPT$.*

The crux of our analysis is establishing several structural properties of the solution subgraph. First we group the demands according to the maximum load of the bottleneck vertex at the time of arrival of the demand. We then show that for every threshold $r > 0$, vertices with load at least r at the end of the run of GA, form a cut set that well separates the group of demands with load at least r at their bottleneck vertex. Since the threshold value can be chosen arbitrarily, this leads to a series of cuts that form a chain. The greedy nature of the algorithm indicates that each cut highly disconnects the demands. Intuitively, a cut that highly disconnects the graph (or the demands) implies a lower bound on the number of branches of every feasible solution.

We use a natural dual-fitting argument to show that for every cut set, the ratio between the number of demands in the corresponding group, over the total degree bound of the cut, is a lower bound for OPT. Hence, the problem comes down to showing that one of the cuts in the series has ratio at least $1/O(\log n)$ fraction of the maximum load h of the output of GA. To this end, we first partition the range of $r \in (0, h]$ into $O(\log n)$ layers based on the total degree bound of the corresponding cut. We then show that the required cut can be found in an interval with maximum range of r. We analyze GA formally in Section 2.2.

We complement our first theorem by giving an example for a special case of ONLINE DEGREE-BOUNDED STEINER TREE in which no online (randomized) algorithm can achieve a (multiplicative) competitive ratio $o(\log n)$. This also implies that obtaining (non-trivial) additive competitiveness is not possible in the online setting.

Theorem 2.2. *Any (randomized) online algorithm for the degree bounded online Steiner tree problem has (multiplicative) competitive ratio* $\Omega(\log n)$. *This already holds when* $b_v = 1$ *for every node.*

The previously known techniques. As discussed before, the majority of techniques used for solving the offline variants of degree-bounded problems involve rounding an optimal fractional solution of a relaxed linear program. Since one may need to buy a long path to connect the endpoints of a demand, independent rounding of a fractional solution is hardly efficient. Instead, dependent and iterative rounding methods are usually used for attacking degree-bounded problems. In the online paradigm, one can maintain a competitive fractional solution for these problems, however, it is inherently difficult to apply the aforementioned rounding techniques in an online setting: the underlying online fractional solution changes in between the rounding steps, thus breaking the chain of dependencies.

In contrast to the works on the offline paradigm, in this study we propose a simple combinatorial algorithm with a dual-fitting analysis. We use the structural properties of the output of our algorithm to show the existence of a chain of cuts that well separates the demand endpoints. When restricted to the case of uniform bounds, these cuts imply an upper bound on the *toughness* of the graph. The toughness of a graph is defined as $\min_{X \subseteq V} \frac{|X|}{|\mathrm{CC}(G \setminus X)|}$; where for a graph H, $\mathrm{CC}(H)$ denotes the collection of connected components of H. It can be shown that the reciprocal of the toughness gives a lower bound for OPT. Therefore we use a combinatorial argument to show that the minimum of this ratio over the cuts in our chain of cuts is within $O(\log(n))$ approximation of the reciprocal of the maximum load of a vertex in our solution.

We would like to emphasize that although the concept of toughness is well-studied in the literature, this line of research is mainly focused on relating toughness conditions to the existence of cycle structures, see for example a comprehensive survey by Bauer et al. [27]. The relation between the graph toughness and degree-bounded problems have been previously observed by Win [10] and Agrawal et al. [17]. However as mentioned in the introduction, Agrawal et al. use a completely different argument for analyzing the problem when reduced to a congestion minimization problem. We hope that the structural properties introduced in this study together with the dual interpretation of our analysis, paves the way for solving the classical problems in the family of degree-bounded problems.

Hardness under more general constraints. We further investigate the following extensions of the online degree bounded Steiner tree problem. First, we consider the edge-weighted variant of the degree-bounded Steiner tree problem. Second, we consider the group Steiner tree version in which each demand consists of a subset of vertices, and the goal is to find a tree that covers at least one vertex of each demand group. The following theorems show that one cannot obtain a non-trivial competitive ratio for these versions in their general form.[1]

Theorem 2.3. *Consider the edge weighted variant of* ONLINE DEGREE-BOUNDED STEINER TREE. *For any (randomized) online algorithm $\mathcal{A}$, there exists an instance and a request sequence such that either* $\mathrm{E}[\mathrm{maxdegree}(\mathcal{A})] \geq \Omega(n) \cdot b$ *or* $\mathrm{E}[\mathrm{weight}(\mathcal{A})] \geq \Omega(n) \cdot OPT_b$*, where OPT_b denotes the minimum weight of a Steiner tree with maximum degree b.*

Theorem 2.4. *There is no deterministic algorithm with competitive ratio $o(n)$ for the* DEGREE-BOUNDED GROUP STEINER TREE *problem.*

2.1.2 Related Degree-Bounded Connectivity Problems

The classical family of degree-bounded network design problems have various applications in broadcasting information, package distribution, decentralized communication networks, etc. (see e.g. [28, 29]). Marathe *et al.* ([20]), first considered the general *edge-weighted* variant of the problem. They give a bi-criteria

[1]Our lower bound results imply that one needs to restrict the input in order to achieve competitiveness. In particular for the edge-weighted variant, our proof does not rule out the existence of a competitive algorithm when the edge weights are polynomially bounded.

$(O(\log n), O(\log n) \cdot b)$-approximation algorithm, i.e., the degree of every node in the output tree is $O(\log n) \cdot b$ while its total weight is $O(\log n)$ times the optimal weight. A long line of work (see e.g. [30] and [31]) was focused on this problem until a groundbreaking breakthrough was obtained by Goemans ([21]); his algorithm computes a minimum-weight spanning tree with degree at most $b + 2$. Later on, Singh and Lau ([26]) improved the degree approximation factor by designing an algorithm that outputs a tree with optimal cost while the maximum degree is at most $b + 1$.

In the *degree-bounded survivable network design* problem, a number d_i is associated with each demand (s_i, t_i). The solution subgraph should contain at least d_i edge-disjoint paths between s_i and t_i. Indeed this problem has been shown to admit bi-criteria approximation algorithms with constant approximation factors (e.g. [23]). We refer the reader to a recent survey in [32]. This problem has been recently considered in the node-weighted variant too (see e.g. [22, 25]). The degree-bounded variant of several other problems such as k-MST and k-arborescence has also been considered in the offline setting for which we refer the reader to [24,33] and references therein.

2.1.3 Related Online Problems

Online network design problems have attracted substantial attention in the last decades. The online edge-weighted Steiner tree problem, in which the goal is to find a minimum-weight subgraph connecting the demand nodes, was first considered by Imase and Waxman ([34]). They showed that a natural greedy algorithm has a competitive ratio of $O(\log n)$, which is optimal up to constants. This result was generalized to the online edge-weighted Steiner forest problem by Awerbuch *et al.* ([35]) and Berman and Coulston ([36]). Later on, Naor, Panigrahi, and Singh ([37])) and Hajiaghayi, Liaghat, and Panigrahi ([38]), designed poly-logarithmic competitive algorithms for the more general *node-weighted* variant of Steiner connectivity problems. This line of work has been further investigated in the prize-collecting version of the problem, in which one can ignore a demand by paying its given penalty. Qian and Williamson ([39]) and Hajiaghayi, Liaghat, and Panigrahi ([40]) develop algorithms with a poly-logarithmic competitive algorithms for these variants.

The online b-matching problem is another related problem in which vertices have degree bounds but the objective is to maximize the size of the solution subgraph. In the worst case model, the celebrated result of Karp *et al.* ([41]) gives a $(1 - 1/e)$-competitive algorithm. Different variants of this problem have been extensively studied in the past decade, e.g., for the random arrival model see [42–45], for the full information model see [46, 47], and for the prophet-inequality model see [48–50]. We also refer the reader to the comprehensive survey by Mehta [51].

Many of the aforementioned problems can be characterized as an online packing or covering linear program. Initiated by work of Alon *et al.* [52] on online set cover, Buchbinder and Naor developed a strong framework for solving packing/covering LPs fractionally online. For the applications of their general framework in solving numerous online problems, we refer the reader to the survey in [53]. Azar *et al.* [54] generalize this method for the fractional *mixed* packing and covering

LPs. In particular, they show an application of their method for integrally solving a generalization of capacitated set cover. Their result is a bi-criteria competitive algorithm that violates the capacities by at most an $O(\log^2 n)$ factor while the cost of the ouput is within $O(\log^2 n)$ factor of optimum. We note that although the fractional variant of our problem is a special case of mixed packing/covering LPs, we do not know of any online rounding method for Steiner connectivity problems.

2.1.4 Preliminaries

Let $G = (V, E)$ denote an undirected graph of size n ($|V| = n$). For every vertex $v \in V$, let $b_v \in \mathbb{R}^+$ denote the *degree bound* of v. In the DEGREE-BOUNDED STEINER FOREST problem, we are given a sequence of connectivity *demands*. The i^{th} demand is a pair of vertices (s_i, t_i) which we call the *endpoints* of the demand. An algorithm outputs a subgraph $H \subseteq G$ in which for every demand its endpoints are connected. The *load* of a vertex v w.r.t. H is defined as $\ell_H(v) = \deg_H(v)/b_v$. We may drop the subscript H when it is clear from the context. The goal is to find a subgraph H that minimizes the maximum load of a vertex. Observe that one can always find an optimal solution without a cycle, hence the name Steiner *forest*. Furthermore, without loss of generality[2], we assume that the endpoints of the demands are distinct vertices with degree one in G and degree

[2]One can add a dummy vertex for every vertex $v \in V$ connected to v. We then interpret a demand between two vertices as a demand between the corresponding dummy vertices. The degree bound of a dummy vertex can be set to infinity. This transformation can increase the degree of any node by at most a factor of 2, which does not affect our asymptotic results.

bound infinity. We denote the maximum load of a vertex in an optimal subgraph by $OPT = \min_H \max_v \ell_H(v)$.

The following mixed packing/covering linear program ($\mathbb{P}$) is a relaxation for the natural integer program for DEGREE-BOUNDED STEINER FOREST. Let $\mathcal{S}$ denote the collection of subsets of vertices that separate the endpoints of at least one demand. For a set of vertices S, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. In $\mathbb{P}$, for an edge e, $\mathbf{x}(e) = 1$ indicates that we include e in the solution while $\mathbf{x}(e) = 0$ indicates otherwise. The variable α indicates an upper bound on the load of every vertex. The first set of constraints ensures that the endpoints of every demand are connected. The second set of constraints ensures that the load of a vertex is upper bounded by α. The program $\mathbb{D}$ is the dual of the LP relaxation $\mathbb{P}$.

$$\begin{array}{lll} \text{minimize} & \alpha & (\mathbb{P}) \\ \forall S \subseteq \mathcal{S} & \sum_{e \in \delta(S)} \mathbf{x}(e) \geq 1 & (\mathbf{y}(S)) \\ \forall v \in V & \sum_{e \in \delta(\{v\})} \mathbf{x}(e) \leq \alpha \cdot b_v & (\mathbf{z}(v)) \\ & \mathbf{x}(e), \alpha \in \mathbb{R}_{\geq 0} & \end{array}$$

$$\begin{array}{lll} \text{maximize} & \sum_{S \in \mathcal{S}} \mathbf{y}(S) & (\mathbb{D}) \\ \forall e = (u, v) & \sum_{S : e \in \delta(S)} \mathbf{y}(S) \leq \mathbf{z}(u) + \mathbf{z}(v) & (\mathbf{x}(e)) \\ & \sum_{v} \mathbf{z}(v) b_v \leq 1 & (\alpha) \\ & \mathbf{z}(v), \mathbf{y}(S) \in \mathbb{R}_{\geq 0} & \end{array}$$

In the online variant of the problem, G and the degree bounds are known in advance, however, the demands are given one by one. Upon the arrival of the i^{th} demand, the online algorithm needs to output a subgraph H_i that satisfies all the demands seen so far. The output subgraph can only be augmented, i.e., for every $j < i$, $H_j \subseteq H_i$. The *competitive ratio* of an online algorithm is then defined as the worst case ratio of $\max_v \ell_H(v)/OPT$ over all possible demand sequences where H is the final output of the algorithm.

2.2 Online Degree-Bounded Steiner Forest

Consider an arbitrary online step where a new demand (s, t) arrived. Let H denote the online output of the previous step. In order to augment H for connecting s and t, one can shortcut through the connected components of H. We say an edge $e = (u, v)$ is an *extension edge* w.r.t. H, if u and v are not connected in H. Let P denote an (s, t)-path in G. The extension part P^* of P is defined as the set of extension edges of P. Observe that augmenting H with P^* satisfies the demand (s, t). For a vertex v, we define $\ell^+_H(v) = \ell_H(v) + 2/b_v$ to be the *uptick load of* v. We slightly abuse the notation by defining $\ell^+_H(P^*) = \max_{v \in V(P^*)} \ell^+_H(v)$ as the uptick load of P^*, where $V(P^*)$ is the set of endpoints of edges in P^*.

The *greedy algorithm* (GA) starts with an empty subset H. Upon arrival of the i-th demand (s_i, t_i), we find a path P_i with smallest uptick load $\ell^+_H(P^*_i)$ where P^*_i is the extension part of P_i. We break ties in favor of the path with a smaller number of edges. Note that the path P_i can be found efficiently using Dijkstra-like algorithms. We define $\tau_i = \ell^+_H(P^*_i)$ to be the *arrival threshold* of the i-th demand. We satisfy the new demand by adding P^*_i to the edge set of H (see Algorithm 1).

```
Algorithm 1 Online Degree-Bounded Steiner Forest
Input: A graph $G$, and an online stream of demands $(s_1, t_1), (s_2, t_2), \ldots$.
Output: A set $H$ of edges such that every given demand $(s_i, t_i)$ is connected via $H$.
Offline Process:
1: Initialize $H = \varnothing$.
Online Scheme; assuming a demand $(s_i, t_i)$ is arrived:
1: Compute $P_i$, a $(s_i, t_i)$-path with the smallest uptick load $\ell^+_H(P^*_i)$ in its extension part.
     • Shortcut the connected components of $H$ by replacing the edges of a component with that of a clique.
     • In the resulting graph, define the distance of a vertex $v$ from $s_i$ as the minimum uptick load of a $(s_i, v)$-path.
     • Find $P_i$ by evoking a Dijkstra-like algorithm according to this notion of distance.
2: Set $H = H \cup P^*_i$.
```

2.2.1 Analysis

We now use a dual-fitting approach to show that GA has an asymptotically tight competitive ratio of $O(\log(n))$. In the following we use GA to also refer to the *final output* of our greedy algorithm. We first show several structural properties of GA. We then use these combinatorial properties to construct a family of feasible dual vectors for $\mathbb{D}$. We finally show that there always exists a member of this family with an objective value of at least a $1/O(\log(n))$ fraction of the maximum load of a vertex in GA. This in turn proves the desired competitive ratio by using weak duality.

For a real value $r \geq 0$, let $\Gamma(r)$ denote the set of vertices with $\ell^+_{\mathsf{GA}}(v) \geq r$. Let $D(r)$ denote the indices of demands i for which the arrival threshold τ_i is at least r. For a subgraph X, let $\mathrm{CC}(X)$ denote the collection of connected components of X. For ease of notation, we may use the name of a subgraph to denote the set of vertices of that subgraph as well. Furthermore, for a graph X and a subgraph $Y \subseteq X$, let $X \setminus Y$ denote the graph obtained from X by removing the vertices of Y.

Recall that $\mathcal{S}$ is the collection of subsets that separate the endpoints of at least one demand. The following lemma, intuitively speaking, implies that $\Gamma(r)$ well-separates the endpoints of $D(r)$.

Lemma 2.1. *For any threshold $r > 0$, we have $|\mathrm{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}| \geq |D(r)| + 1$.*

Proof. For a vertex $v \in G \setminus \Gamma(r)$ we use $\mathrm{CC}(v)$ to denote its connected component. Observe that, since the endpoints of demands are nodes with infinite degree bound, the endpoints are *not* contained in $\Gamma(r)$, and, hence, are in $G \setminus \Gamma(r)$.

We construct a graph F that has one node for every connected component in $G \setminus \Gamma(r)$ that contains an endpoint of a demand in $D(r)$. Edges in F are defined as follows. For every demand $i \in D(r)$ between s_i and t_i, we add an edge that connects the components $\mathrm{CC}(s_i)$ and $\mathrm{CC}(t_i)$. In the following we argue that F does not contain cycles. This gives the lemma since in a forest $|D(r)| + 1 = |E_F| + 1 \leq |V_F| \leq |\operatorname{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}|$.

Assume for contradiction that the sequence $(e_{i_0}, \ldots, e_{i_{k-1}})$, $k \geq 2$ forms a (minimal) cycle in F, where e_{i_j} corresponds to the demand between vertices s_{i_j} and t_{i_j} (see Figure 2.1). Assume wlog. that $e_{i_{k-1}}$ is the edge of the cycle for which the corresponding demand appears last, i.e., $i_{k-1} \geq i_j$ for every $j < k$. Let H denote the online solution at the time of arrival of the demand i_{k-1}. We can augment H to connect each t_{i_j} to $s_{i_{j+1 \bmod k}}$, $0 \leq j \leq k-1$ without using any node from $\Gamma(r)$, since these nodes are in the same component in $G \setminus \Gamma(r)$. But then we have a path P between $s_{i_{k-1}}$ and $t_{i_{k-1}}$ and the extension part P^* does not contain any vertex from $\Gamma(r)$. This is a contradiction since the arrival threshold for the demand i_{k-1} is at least r. □

Lemma 2.1 shows that cutting $\Gamma(r)$ highly disconnects the demands in $D(r)$. Indeed this implies a bound on the *toughness* of the graph. Toughness, first defined by Chvátal [11] and later generalized by Agrawal *et al.* [17], is a measure of how easy it is to disconnect a graph into many components by removing a few vertices. More formally, the toughness of a graph is defined as $\min_{X \subseteq V} \frac{|X|}{|\operatorname{CC}(G \setminus X)|}$. For the spanning tree variant of the problem, it is not hard to see that OPT is at least the reciprocal of toughness. Although it is more involved, we can still establish a similar relation for the non-uniform Steiner forest problem (see Lemma 2.3). However, first we need to show a lower bound on the number of demands separated by $\Gamma(r)$.

We establish a lower bound for $|D(r)|$ with respect to the load of vertices in

$\Gamma(r)$. For any $r, b > 0$ we define $\Gamma_b(r) := \{\ell^+_{\mathsf{GA}}(v) \geq r \wedge b_v \geq b\}$, as the set of nodes with degree bound at least b that have uptick load at least r in the final online solution. We further define

$$\text{excess}(r, b) = \sum_{v \in \Gamma_b(r)} \Big(\deg_{\mathsf{GA}}(v) - \lceil r b_v \rceil + 2 \Big) \ ,$$

which sums the (absolute) load that nodes in $\Gamma_b(r)$ receive *after* their *uptick load* became r or larger. The following lemma relates $|D(r)|$ to $\text{excess}(r, b)$.

Lemma 2.2. *For any $r, b > 0$,* $\text{excess}(r, b) \leq 2|D(r)| + 3|\Gamma_b(v)|$.

Proof. Consider some online step i. Let H denote the output of the previous step. Let P^*_i be the extension part of the path selected by GA and let $V(P^*_i)$ denote the endpoints of edges of P^*_i. Since in GA we break ties in favor of the path with the smaller number of edges, we can assume that the path does not go through a connected component of H more than once, i.e., for every $C \in \text{CC}(H)$, $|V(P^*_i) \cap C| \leq 2$.

Consider the variable quantity $\delta(r, b) := \sum_{v : \ell^+_H(v) \geq r \wedge b_v \geq b} (\deg_H(v) - \lceil r b_v \rceil + 2)$ throughout the steps of GA where H denotes the output of the algorithm at every step. Intuitively, at each step $\delta(r, b)$ denotes the total increment in degrees of those vertices in $\Gamma_b(r)$ that already reached uptick load r. In particular, at the end of the run of GA, $\delta(r, b) = \text{excess}(r, b)$.

Now suppose at step i adding the edges P^*_i to H increases $\delta(r, b)$ by q_i. There are two reasons for such an increase. On the one hand, if the demand i is from $D(r)$ it may simply increase $\deg_H(v)$ for some node with uptick load at least r. On the other hand also if the demand is not from $D(r)$ it may cause a node to increase its uptick load to r in which case it could contribute to the above sum with at most 1 (in a step the degree may increase by 2; the first increase by 1 could get the uptick load to $\geq r$ and the second increase contributes to the sum).

Clearly increases of the second type can accumulate to at most $|\Gamma_b(r)|$. In order to derive a bound on the first type of increases recall that we assume that endpoints of demands are distinct vertices with degree one and thus s_i and t_i are outside any connected component of H. Since $V(P_i^*)$ contains at most two vertices of a connected component of H, we can assert that the path selected by GA is passing through at least $q_i/2$ components of H that contain some vertices of $\Gamma_b(r)$. Hence, after adding P_i^* to the solution, the number of connected components of the solution that contain some vertices of $\Gamma_b(r)$ decreases by at least $(q_i - 2)/2$. However, throughout all the steps, the total amount of such decrements cannot be more than the number of vertices in $\Gamma_b(r)$. Therefore

$$\begin{aligned}
\text{excess}(r, b) &= \sum_{i \in D(r)} q_i + \sum_{i \notin D(r)} q_i \\
&= 2|D(r)| + \sum_{i \in D(r)} (q_i - 2) + |\Gamma_b(r)| \\
&\leq 2|D(r)| + 3|\Gamma_b(r)| \ ,
\end{aligned}$$

and the lemma follows. □

Let $\Delta > 0$ denote the minimum degree bound of a vertex with non-zero degree in an optimal solution. Note that this definition implies that $OPT \geq 1/\Delta$. Indeed, replacing Δ with one in the following would make the arguments slightly simpler. However, by doing so we incur an additive $\log(n)$ term in the final result which can be arbitrary far from OPT. Therefore it is necessary to employ Δ in the analysis. For a set of vertices Γ, let $b(\Gamma) = \sum_{v \in \Gamma} b_v$. We now describe the main dual-fitting argument for bounding the maximum load in GA.

Lemma 2.3. *For any $r > 0$, $|D(r)|/b(\Gamma_\Delta(r)) \leq OPT$.*

Proof. Let G_Δ denote the graph obtained from G by removing vertices with degree bound below Δ. Similarly, let $\mathcal{S}_\Delta$ denote the collection of sets that separate a demand in G_Δ. Consider a slightly modified dual program $\mathbb{D}_\Delta$ defined on G_Δ and $\mathcal{S}_\Delta$, i.e., we obtain $\mathbb{D}_\Delta$ from the original dual program by removing all vertices with degree bound below Δ. We note that the objective value of a dual feasible solution for $\mathbb{D}_\Delta$ is still a lower bound for OPT. In the remainder of the proof, we may refer to $\mathbb{D}_\Delta$ as the dual program. Recall that in a dual solution, we need to define a dual value $\mathbf{y}(S)$ for every cut $S \in \mathcal{S}_\Delta$ such that for every edge $e = (u, v)$ the total value for cuts containing e does not exceed $\mathbf{z}(u) + \mathbf{z}(v)$. On the other hand, $\sum_v \mathbf{z}(v) b_v$ cannot be more than one.

For a real value $r > 0$, we construct a dual vector $(\mathbf{y}_r, \mathbf{z}_r)$ as follows. For every $v \in \Gamma_\Delta(r)$, set $\mathbf{z}_r(v) = 1/b(\Gamma_\Delta(r))$; for other vertices set $\mathbf{z}_r(v) = 0$. For every component $S \in \mathrm{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta$, set $\mathbf{y}_r(S) = 1/b(\Gamma_\Delta(r))$; for all other sets set $\mathbf{y}_r(S) = 0$. We prove the lemma by showing the feasibility of the dual vector $(\mathbf{y}_r, \mathbf{z}_r)$ for $\mathbb{D}_\Delta$.

Consider an arbitrary component $C \in \mathrm{CC}(G \backslash \Gamma(r))$. By definition, C separates at least one demand. Let i be such a demand and let $t_i \in C$ denote an endpoint of it. Removing vertices with degree bound below $1/\Delta$ from C may break C into multiple smaller components. However, an endpoint of a demand has degree bound infinity, and, hence, the component that contains t_i belongs to $\mathcal{S}_\Delta$. Therefore $|\operatorname{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}| \leq |\operatorname{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta|$; which by Lemma 2.1 leads to $|\operatorname{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta| \geq |D(r)| + 1$. Therefore the dual objective for $(\mathbf{y}_r, \mathbf{z}_r)$ is at least

$$\sum_{S \in \mathcal{S}_\Delta} \mathbf{y}_r(S) = \sum_{S \in \operatorname{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta} \frac{1}{b(\Gamma_\Delta(r))} \geq \frac{|D(r)| + 1}{b(\Gamma_\Delta(r))}$$

Thus we only need to show that $(\mathbf{y}_r, \mathbf{z}_r)$ is feasible for Program $\mathbb{D}_\Delta$. First, by construction we have

$$\sum_v \mathbf{z}_r(v) b_v = \sum_{v \in \Gamma_\Delta(r)} \frac{1}{b(\Gamma_\Delta(r))} \cdot b_v = 1 \ .$$

Now consider an arbitrary edge $e = (u, v)$. If e does not cross any of the components in $\operatorname{CC}(G_\Delta \setminus \Gamma_\Delta(r))$, then $\sum_{S : e \in \delta(S)} \mathbf{y}_r(S) = 0$ and we are done. Otherwise, $\sum_{S : e \in \delta(S)} \mathbf{y}_r(S) = 1/b(\Gamma_\Delta(r))$. However, exactly one endpoint of e is in $\Gamma_\Delta(r)$. Thus $\mathbf{z}_r(u) + \mathbf{z}_r(v) = 1/b(\Gamma_\Delta(r))$, which implies that the dual vector is feasible. $\square$

We now have all the ingredients to prove the main theorem.

Proof of Theorem 2.1: Let GA denote the final output of the greedy algorithm. Let h denote the maximum load of a vertex in GA, i.e., $h = \max_v \ell_{\mathsf{GA}}(v)$. Furthermore, let $h_\Delta = \max_{v : b_v \geq \Delta} \ell_{\mathsf{GA}}(v)$. In the following we first show that $h_\Delta \leq O(\log n) \cdot OPT$. For this we use the folowing claim.

Claim 2.1. *There exists an* $r > 0$ *such that* $\operatorname{excess}(r, \Delta) \geq \frac{h_\Delta - 1/\Delta}{4 \log_2 n + 6} \cdot b(\Gamma_\Delta(r)) - |\Gamma_\Delta(r)|$.

Proof. Recall that $\Gamma_\Delta(r)$ is the set of vertices with uptick load at least r in GA and degree bound at least Δ. The function $b(\Gamma_\Delta(r))$ is non-increasing with r since for every $r_1 < r_2$, $\Gamma_\Delta(r_2) \subseteq \Gamma_\Delta(r_1)$. For $r = 1/\Delta$, $b(\Gamma_\Delta(r)) \leq n(n+1)\Delta$ as a node with degree bound $b_v > (n+1)\Delta$ will have uptick load at most $(n+1)/b_v < 1/\Delta$, and, hence, will not be in $\Gamma_\Delta(r)$. Also, $r < h_\Delta$ implies $b(\Gamma_\Delta(r)) \geq \Delta$ as there exists a node with load h_Δ in $\Gamma_\Delta(r)$ and this node has degree bound at least Δ.

We now partition the range of r (from $1/\Delta$ to h_Δ) into logarithmically many intervals. We define the q-th interval by

$$U(q) =$$

$$\left\{r \mid 1/\Delta \leq r < h_\Delta \ \wedge \ 2^q/\Delta \leq b(\Gamma_\Delta(r)) < 2^{q+1}/\Delta\right\},$$

for $q \in \{0, \ldots, \lceil \log_2((n+1)n) \rceil\}$. We further set $\overline{r}(q) = \max U(q)$ and $\underline{r}(q) = \min U(q)$, and call $\overline{r}(q) - \underline{r}(q)$ the length of the q-th interval. Since there are only $2\log_2 n + 3$ possible choices for q there must exist an interval of length at least $(h_\Delta - 1/\Delta)/(2\log_2 n + 3)$.

Consider a node v that is contained in $\Gamma_\Delta(\overline{r})$ and, hence, also in $\Gamma_\Delta(\underline{r})$. This node starts contributing to $\text{excess}(\underline{r}, \Delta)$, once its uptick load reached $\underline{r}$ and contributes at least until its uptick load reaches $\overline{r}$. Hence, the total contribution is at least $\deg(\overline{r}) - \deg(\underline{r})$, where $\deg(\overline{r})$ and $\deg(\underline{r})$ denotes the degree of node v when reaching uptick load $\overline{r}$ and $\underline{r}$, respectively. We have

$$\deg(\overline{r}) - \deg(\underline{r}) = (\lceil \overline{r} b_v \rceil - 2) - (\lceil \underline{r} b_v \rceil - 2) \geq (\overline{r} - \underline{r}) b_v - 1.$$

Summing this over all nodes in $\Gamma_\Delta(\overline{r})$ gives

$$\begin{aligned}
\text{excess}(\underline{r}, \Delta) &\geq (\overline{r} - \underline{r}) \cdot b(\Gamma_\Delta(\overline{r})) - |\Gamma_\Delta(\overline{r})| \\
&\geq \frac{1}{2}(\overline{r} - \underline{r}) \cdot b(\Gamma_\Delta(\underline{r})) - |\Gamma_\Delta(\underline{r})| \\
&\geq \frac{h_\Delta - 1/\Delta}{4\log_2 n + 6} \cdot b(\Gamma_\Delta(\underline{r})) - |\Gamma_\Delta(\underline{r})| \ ,
\end{aligned}$$

where the second inequality uses the fact that $|\Gamma_\Delta(\overline{r})| \leq |\Gamma_\Delta(\underline{r})| \leq |\Gamma_\Delta(\overline{r})|/2$. □

Claim 2.2. *$h_\Delta \leq (24\log_2 n + 37)OPT$.*

Proof. If we use the r from the previous claim and solve for h_Δ we obtain

$$\begin{aligned}
h_\Delta &\leq (4\log_2 n + 6) \cdot \frac{\text{excess}(r, \Delta) + \Gamma_\Delta(r)}{b(\Gamma_\Delta(r))} + \frac{1}{\Delta} \\
&\leq (4\log_2 n + 6) \cdot \frac{2D(r) + 4\Gamma_\Delta(r)}{b(\Gamma_\Delta(r))} + \frac{1}{\Delta} \\
&\qquad\qquad \textit{Lemma 2.2} \\
&\leq (4\log_2 n + 6) \cdot \left(2OPT + 4\frac{1}{\Delta}\right) + \frac{1}{\Delta} \\
&\qquad\qquad \textit{Lemma 2.3 and } b(\Gamma_\Delta(r)) \geq \Delta|\Gamma_\Delta(r)| \\
&\leq (24\log_2 n + 37) \cdot OPT \\
&\qquad\qquad OPT \geq 1/\Delta \ .
\end{aligned}$$

□

We now bound the maximum load h in terms of the restricted maximum load h_Δ. Consider a vertex v^* with maximum load $\ell_{\mathsf{GA}}(v^*) = h$. If $b_{v^*} \geq \Delta$ then $h_\Delta = h$ and we are done. Otherwise consider the last iteration i in which the degree of v^* is increased in the solution. Let H denote the output of the algorithm at the end of the previous iteration $i - 1$. The degree of v^* in the online solution is increased by at most two at iteration i. Hence $h \leq \ell_H^+(v^*) \leq \tau_i$.

Recall that in our greedy algorithm, τ_i is the minimum uptick load of a path that satisfies the new demand. Let P denote a path that connects s_i and t_i in an optimal solution. Recall that by the definition, $b_v \geq \Delta$ for every vertex v of P. For every vertex v in P we have

$$\begin{aligned}
\tau_i \leq \ell_H^+(v) &\leq h_\Delta + \frac{2}{b_v} && \ell_H(v) \leq \ell_{\mathsf{GA}}(v) \leq h_\Delta \\
&\leq h_\Delta + \frac{2}{\Delta} && b_v \geq \Delta \\
&\leq h_\Delta + 2OPT && OPT \geq 1/\Delta \\
&\leq O(\log n) \cdot OPT && \text{by the above claim}
\end{aligned}$$

Therefore leading to $h \leq O(\log n) \cdot OPT$. □

2.3 An Asymptotically Tight Lower Bound

In the following we show a lower bound for ONLINE DEGREE-BOUNDED STEINER TREE. Consider a graph $G = (X \uplus Z, E)$, with $|Z| = 2^\ell$ and $|X| = \binom{2^\ell}{2}$. For every pair $\{z_1, z_2\}$ of nodes from Z there exists an edge $\{z_1, z_2\} \in E$ and a node $x \in X$ that is connected to z_1 and z_2. An arbitrary node from Z acts as root for the Steiner tree problem.

For a node $z \in Z$, an algorithm A, and a request sequence σ (consisting of nodes from X) we use $\deg_{A,\sigma}(z)$ to denote the number of neighbors of z *among all nodes from* X in the Steiner tree obtained when running algorithm A on request sequence σ. Similarly, we use $\deg'_{A,\sigma}(z)$ to denote the number of those neighbors of z that also appear in σ. Note that $\deg(\cdot)$ and $\deg'(\cdot)$ ignore edges between nodes from Z as an algorithm (online or offline) can simply connect all nodes in Z in a cycle which increases the degree of any node by only 2.

Lemma 2.4. *Fix a possibly randomized online algorithm A. For any subset $S \subseteq Z$, $|S| = 2^s$, $0 \leq s \leq \ell$ there exists a request sequence σ_S consisting of terminals from X s.t.*

- *for a node $x \in \sigma_S$ both its neighbors in G are from set S;*
- *there exists a node $z^* \in S$ with $\mathrm{E}\left[\deg'_{A,\sigma}(z^*)\right] \geq s/2$;*
- *there exists an offline algorithm* OFF *for servicing requests in σ with $\max_{z \in S}\{\deg_{\mathrm{OFF},\sigma}(z)\} \leq 1$, and $\deg_{\mathrm{OFF},\sigma}(z) = 0$ for $z \in (Z \setminus S) \cup \{z^*\}$.*

Proof. We prove the lemma by induction over s. The base case $s = 0$ holds trivially when choosing the empty request sequence. For the induction step consider an arbitrary subset $S \subseteq Z$ with $|S| = 2^{s+1}$. Partition S into two disjoint subsets S_1 and S_2 of cardinality 2^s each.

Let σ_1 be the request sequence that exists due to induction hypothesis for set S_1. Hence, there is a node $z_1^* \in S_1$ with $\mathrm{E}\left[\deg'_{A,\sigma_1}(z_1^*)\right] \geq s/2$. Now, let $\tilde{A}$ behave like algorithm A *after* it already received a request sequence σ_1 (hence, it starts with all edges that are chosen when running A on σ_1; note, however, that $\deg'_{\tilde{A},\sigma_2}(\cdot)$ only takes into account edges incident to nodes from σ_2). Due to induction hypothesis for $\tilde{A}$ and set S_2 there exists a request sequence σ_2 such that $\mathrm{E}\left[\deg'_{\tilde{A},\sigma_2}(z_2^*)\right] \geq s/2$ for a node $z_2^* \in S_2$. Hence, the request sequence $\sigma = \sigma_1 \circ \sigma_2$ fulfills $\mathrm{E}\left[\deg'_{A,\sigma}(z_1^*)\right] \geq s/2$ and $\mathrm{E}\left[\deg'_{A,\sigma}(z_2^*)\right] \geq s/2$.

We extend the request sequence by appending the node x that is connected to z_1^* and z_2^* in G. After serving the request one of the edges $\{x, z_1\}$ or $\{x, z_2\}$ must be chosen with probability at least $1/2$ by A. Hence, afterwards either $\mathrm{E}\left[\deg'_{A,\sigma}(z_1^*)\right]$ or $\mathrm{E}\left[\deg' \delta_{A,\sigma}(z_2^*)\right]$ must be at least $(s+1)/2$.

It remains to argue that there exists a good offline algorithm. Combining the offline algorithms OFF_1 and OFF_2 for σ_1 and σ_2 gives an offline algorithm for $\sigma_1 \circ \sigma_2$ that has $\max_z\{\deg_{\mathrm{OFF},\sigma_1\circ\sigma_2}(z)\} \leq 1$ and $\deg_{\mathrm{OFF},\sigma_1\circ\sigma_2}(z)\} = 0$ for $z \notin S_1 \cup S_2$ and for $z \in \{z_1^*, z_2^*\}$. Now, when the node x connected to z_1^* and z_2^* is appended to the request sequence the offline algorithm can serve the request by either buying edge $\{x, z_1\}$ or $\{x, z_2\}$, and can therefore gurantee that z^* (z_1 or z_2) has degree 0. $\square$

Proof of Theorem 2.2: Choosing $s = \log n$ in the above lemma gives our lower bound. $\square$

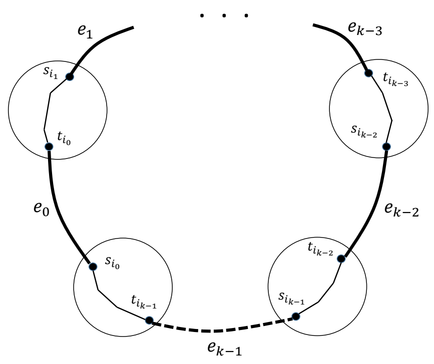


Figure 2.1: Existence of a cycle implies that of a path with low uptick load in its extension part.

2.4 Lower Bounds for Other Degree-Bounded Steiner Connectivity Problems

In this section we consider strong lower bounds for the general form of two variants of degree-bounded network design problems.

2.4.1 Online Degree-Bounded Group Steiner Tree

In this section, presenting an adversary scenario, we show there is no deterministic algorithm for ONLINE DEGREE-BOUNDED GROUP STEINER TREE with competitive ratio $o(n)$ even if G is a star graph.

Proof of Theorem 2.4: For any integer $n > 1$, we provide a graph instance G of size n and an online scenario, in which no deterministic algorithm can obtain a competitive ratio better than $n-1$. Let G be a star with $n-1$ leaves v_2 to v_n, and v_1 be the internal node. For an algorithm $\mathcal{A}$, we describe the adversary scenario as follows.

Let v_1 be the root. Let the first demand group be the set of all leaves. Whenever $\mathcal{A}$ connects a node v_i to v_1 in H, adversary removes the selected nodes v_i from the next demand groups, until all leaves are connected to v_1 in H. In particular let C denote the set of all nodes connected to v_1 in H so far. Let the demand group be the set of all leaves in $\{v_2, v_3, \ldots, v_n\} \setminus C$. We do this until $\{v_2, v_3, \ldots, v_n\} \setminus C = \emptyset$, which means every leaf is connected to v_1 in H. G is a star, thus a leaf v_i is connected to v_1 in H iff H includes the edge between v_i and v_1. Thus after all demands

$\mathcal{A}$ has added all edges in G to H. Hence $deg(v_1) = n - 1$.

Now consider the optimal offline algorithm. Let g_i denote the i-th demand group. Assume we have k demand groups. By construction of the demand groups $g_k \subset g_{k-1} \subset \ldots \subset g_1$. Thus there is a single node that exists in all group demands. Hence the optimal offline algorithms only needs to connect that node to the root in H. Therefore, the degree of each node is at most 1 and the competitive ratio is $n - 1$. $\square$

Chapter 3: Online Weighted Degree-Bounded Steiner Network Design

3.1 Introduction

Degree-bounded network design problems comprise an important family of network design problems since the eighties. Aside from various real-world applications such as vehicle routing and communication networks [55–57], the family of degree-bounded problems has been a testbed for developing new ideas and techniques. The problem of *degree-bounded spanning tree*, introduced in Garey and Johnson's *Black Book* of NP-Completeness [15], was first investigated in the pioneering work of Fürer and Raghavachari [16]. In this problem, we are required to find a spanning tree of a given graph with the goal of minimizing the maximum degree of the vertices in the tree. Let b^* denote the maximum degree in the optimal spanning tree. Fürer and Raghavachari give a parallel approximation algorithm which produces a spanning tree of degree at most $O(\log(n)b^*)$. This result was later generalized by Agrawal, Klein, and Ravi [17] to the case of degree-bounded Steiner tree (DB-ST) and degree bounded Steiner forest (DB-SF) problem. In DB-ST, given a set of terminal vertices, we need to find a subgraph of minimum maximum degree that connects the terminals. In the more generalized DB-SF problem, we are given pairs of terminals and the output subgraph should contain a path connecting each pair. Fürer and Raghavachari [19] significantly improved the result for DB-SF by presenting an algorithm which produces a Steiner forest with maximum degree at most $b^* + 1$.

The study of DB-ST and DB-SF was the starting point of a very popular line of work on various degree-bounded network design problems; e.g. [20, 22–25] and more recently [25, 58]. One particular variant that has been extensively studied was initiated by Marathe *et al.* [20]: In the *edge-weighted degree-bounded spanning tree* problem, given a weight function over the edges and a degree bound b, the goal is to find a minimum-weight spanning tree with maximum degree at most b. The initial results for the problem generated much interest in obtaining approximation algorithms for the edge-weighted degree-bounded spanning tree problem [59–68]. The groundbreaking results obtained by Goemans [21] and Singh and Lau [26] settle the problem by giving an algorithm that computes a minimum-weight spanning tree with degree at most $b + 1$. Singh and Lau [23] generalize their result for the *edge-weighted Steiner tree* (EW-DB-ST) and *edge-weighted Steiner forest* (EW-DB-SF) variants. They design an algorithm that finds a Steiner forest with cost at most twice the cost of the optimal solution while violating the degree constraints by at most three.

Despite these achievements in the offline setting, it was not known whether degree-bounded problems are tractable in the *online setting*. The online counterparts of the aforementioned Steiner problems can be defined as follows. The underlying graph and degree bounds are known in advance. The demands arrive one by one in an online manner. At the arrival of a demand, we need to augment the solution subgraph such that the new demand is satisfied. The goal is to be competitive against an offline optimum that knows the demands in advance.

Recently, Dehghani et al. [69] explore the tractability of the Online DB-SF problem by showing that a natural greedy algorithm produces a solution in which the degree bounds are violated by at most a factor of $O(\log n)$, which is asymptotically *tight*. They analyze their algorithm using a dual fitting approach based on the combinatorial structures of the graph such as the toughness[1] factor. Unfortunately, they can also show that greedy methods are not competitive for the edge-weighted variant of the problem. Hence, it seems unlikely that the approach of [69] can be generalized to EW-DB-SF.

The *online edge-weighted Steiner connectivity* problems (with no bound on the degrees) have been extensively studied in the last decades. Imase and Waxman [34] use a dual-fitting argument to show that the greedy algorithm has a competitive ratio of $O(\log n)$, which is also asymptotically tight. Later the result was generalized to the EW SF variant by Awerbuch *et al.* [35] and Berman and Coulston [36]. In the past few years, various primal-dual techniques have been developed to solve the more general node-weighted variants [37,38,52], prize-collecting variants [39,40], and multicommodity buy-at-bulk [70]. These results are obtained by developing various primal-dual techniques [38, 52] while generalizing the application of combinatorial properties to the online setting [37,40,70]. In this study however, we develop a primal

[1]The toughness of a graph is defined as $\min_{X\subseteq V} \frac{|X|}{|\mathrm{CC}(G\setminus X)|}$; where for a graph H, $\mathrm{CC}(H)$ denotes the collection of connected components of H.

approach for solving *bounded-frequency mixed packing/covering integer programs*. We believe this framework would be proven useful in attacking other online packing and covering problems.

3.1.1 Our Results and Techniques

In this study, we consider the online Steiner tree and Steiner forest problems at the presence of both edge weights and degree bounds. In the Online EW-DB-SF problem, we are given a graph $G = (V, E)$ with n vertices, edge-weight function w, degree bound b_v for every $v \in V$, and an online sequence of connectivity demands (s_i, t_i). Let w_{opt} denote the minimum weight subgraph which satisfies the degree bounds and connects all demands. Let $\rho = \frac{\max_e w(e)}{\min_{e:w(e)>0} w(e)}$.

Theorem 3.1. *There exists an online deterministic algorithm which finds a subgraph with total weight at most $O(\log^3 n)w_{opt}$ while the degree bound of a vertex is violated by at most a factor of $O(\log^3(n)\log(n\rho))$.*

If one favors the degree bounds over total weight, one can find a subgraph with degree-bound violation $O(\log^3(n)\frac{\log(n\rho))}{\log\log(n\rho)})$ and total cost $O(\log^3(n)\frac{\log(n\rho))}{\log\log(n\rho)})w_{opt}$.

We note that the logarithmic dependency on ρ is indeed necessary. It follows from the result of [69] that the competitive ratio of any algorithm is either $\Omega(n)$ or $\Omega(\log\rho)$.

Our technical contribution for solving the EW-DB-SF problem is twofold. First by exploiting a structural result and massaging the optimal solution, we show a formulation of the problem that falls in the restricted family of *bounded-frequency* *mixed packing/cover IPs*, while losing only logarithmic factors in the competitive ratio. We then design a generic online algorithm with a logarithmic competitive ratio that can solve any instance of the bounded-frequency packing/covering IPs. In what follows, we describe our results in detail.

3.1.1.1 Massaging the optimal solution

Initiated by work of Alon *et al.* [52] on online set cover, Buchbinder and Naor developed a strong framework for solving packing/covering LPs *fractionally* online. For the applications of their general framework in solving numerous online problems, we refer the reader to the survey in [53]. Azar *et al.* [54] generalize this method for the fractional *mixed* packing and covering LPs. The natural linear program relaxation for EW-DB-SF, commonly used in the literature, is a special case of mixed packing/covering LPs: one needs to select an edge from every cut that separates the endpoints of a demand (covering constraints), while for a vertex we cannot choose more than a specific number of its adjacent edges (packing constraints). Indeed, one can use the result of Azar *et al.* [54] to find an online *fractional* solution with poly-logarithmic competitive ratio. However, doing the rounding in an online manner seems very hard.

Offline techniques for solving degree-bounded problems often fall in the category of iterative and dependent rounding methods. Unfortunately, these methods are inherently difficult to adapt for an online settings since the underlying fractional solution may change dramatically in between the rounding steps. Indeed, this might be the very reason that despite many advances in the online network design paradigm in the past two decades, the natural family of degree-bounded problems has remained widely open. In this study, we circumvent this by reducing EW-DB-ST to a novel formulation beyond the scope of standard online packing/covering techniques and solving it using a new online integral approach.

The crux of our IP formulation is the following structural property: Let (s_i, t_i) denote the i^{th} demand. We need to augment the solution Q_{i-1} of previous steps by buying a subgraph that makes s_i and t_i connected. Let G_i denote the graph obtained by contracting the pairs of vertices s_j and t_j for every $j < i$. Note that any $(s_i - t_i)$-path in G_i corresponds to a feasible augmentation for Q_{i-1}. Some edges in G_i might be already in Q_{i-1} and therefore by using them again we can save both on the total weight and the vertex degrees. However, in Section 3.2 we prove that there always exists a path in G_i such that even without sharing on any of the edges in G_i and therefore paying completely for the increase in the weight and degrees, we can approximate the optimal solution up to a logarithmic factor. This in fact, enables us to have a formulation in which the covering constraints for different demands are *disentangled*. Indeed, we only have one covering constraint for each demand. Unfortunately, this implies that we have exponentially many variables, one for each possible path in G_i. This may look hopeless since the competitive factors obtained by standard fractional packing/covering methods introduced by Buchbinder and Naor [53] and Azar *et al.* [54], depend on the logarithm of the number of variables. Therefore we come up with a new approach for solving this class of mixed packing/covering integer programs (IP).

3.1.1.2 Bounded-frequency mixed packing/covering IPs

We derive our result for EW-DB-ST by demonstrating a new technique for solving mixed packing/covering *integer* programs. We believe this approach could be applicable to a broader range of online problems. The integer program $\mathbb{IP}1$ describes a general mixed packing/covering IP with the set of integer variables $\mathbf{x} \in \mathbb{Z}^n_{\geq 0}$ and α. The packing constraints are described by a $m \times n$ non-negative matrix P. Similarly, the $q \times n$ matrix C describes the covering constraints. The *covering frequency* of a variable x_i is defined as the number of covering constraints in which x_i has a positive coefficient. The covering frequency of a mixed packing/covering program is defined as the maximum covering frequency of its variables.

$$\begin{aligned} \text{minimize} \quad & \alpha \ , & (\mathbb{IP}1) \\ s.t. \quad & P\mathbf{x} \leq \alpha \ . \\ & C\mathbf{x} \geq 1 \ . \\ & \mathbf{x} \in \mathbb{Z}_{\geq 0}, \alpha \in \mathbb{R}_{>0} \ . \end{aligned}$$

In the online variant of mixed packing and covering IP, we are given the packing constraints in advance. However the covering constraints arrive in an online manner. At the arrival of each covering constraint, we should *increase* the solution $\mathbf{x}$ such that it satisfies the new covering constraint. We provide a deterministic algorithm for solving online mixed packing/covering IPs.

Theorem 3.2. *Given an instance of the online mixed packing/covering IP, there* *exists a deterministic integral algorithm with competitive ratio $O(k \log m)$, where m is the number of packing constraints and k is the covering frequency of the IP.*

We note that the competitive ratio of our algorithm is independent of the number of variables or the number of covering constraints. Indeed, there can be exponentially many variables.

Our result can be thought of as a generalization of the work of Aspnes et al. [71] on virtual circuit routing. Although not explicit, their result can be massaged to solve mixed packing/covering IPs in which all the coefficients are zero or one, and the covering frequency is one. They show that such IPs admit a $O(\log(m))$-competitive algorithms. Theorem 3.2 generalizes their result to the case with arbitrary non-negative coefficients and any bounded covering frequency.

We complement our result by proving a matching lower bound for the competitive ratio of any *randomized* algorithm. This lower bound holds even if the algorithm is allowed to return fractional solutions.

Theorem 3.3. *Any randomized online algorithm A for integral mixed packing and covering is $\Omega(k \log m)$-competitive, where m denotes the number of packing constraints, and k denotes the covering frequency of the IP. This even holds if A is allowed to return a fractional solution.*

As mentioned before, Azar *et al.* [54] provide a fractional algorithm for mixed packing/covering LPs with competitive ratio of $O(\log m \log d)$ where d is the maximum number of variables in a single constraint. They show an almost matching lower bound for deterministic algorithms. We distinguish two advantages of our approach compared to that of Azar *et al*:

- The algorithm in [54] outputs a *fractional* competitive solution which then needs to be rounded online. For various problems such as Steiner connectivity problems, rounding a solution online is very challenging, even if offline rounding techniques are known. Moreover, the situation becomes hopeless if the integrality gap is unbounded. However, for bounded-frequency IPs, our algorithm directly produces an integral competitive solution. Thus it does not depend on rounding methods, and is applicable to problems with large integrality gap, or the problems for which it is shown that rounding methods do not preserve any approximation guarantee, and as such, the traditional approach fails.

- Azar *et al.* find the best competitive ratio with respect to the number of packing constraints and the size of constraints. Although these parameters are shown to be bounded in several problems, in many problems such as connectivity problems and flow problems, formulations with exponentially many variables are very natural. Our techniques provide an alternative solution with a tight competitive ratio, for formulations with bounded covering frequency.

3.1.2 Preliminaries

Let $G = (V, E)$ be an undirected graph of size n ($|V| = n$). Let $w : E \to \mathbb{Z}_{>0}$ be a function denoting the edge weights. For a subgraph $H \subseteq G$, we define $w(H) := \sum_{e \in E(H)} w(e)$. For every vertex $v \in V$, let $b_v \in \mathbb{Z}_{>0}$ denote the degree bound of v. Let $\deg_H(v)$ denote the degree of vertex v in subgraph H. We define the load $l_H(v)$ of vertex v w.r.t. H as $\deg_H(v)/b_v$. In DB-SF we are given graph G, degree bounds, and k connectivity demands. Let σ_i denote the i-th demand. The i-th demand is a pair of vertices $\sigma_i = (s_i, t_i)$, where $s_i, t_i \in V$. In DB-SF the goal is to find a subgraph $H \subseteq G$ such that for each demand σ_i, s_i is connected to t_i in H, for every vertex $v \in V$, $l_H(v) \leq 1$, and $w(H)$ is minimized. In this study without loss of generality we assume the demand endpoints are distinct vertices with degree one in G and degree bound infinity.

In the online variant of the problem, we are given graph G and degree bounds in advance. However the sequence of demands are given one by one. At arrival of demand σ_i, we are asked to provide a subgraph H_i, such that $H_{i-1} \subseteq H_i$ and s_i is connected to t_i in H_i.

The following integer program is a natural mixed packing and covering integer program for EW-DB-SF. Let $\mathcal{S}$ denote the collection of subsets of vertices that separate the endpoints of at least one demand. For a set of vertices S, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. In $\mathbb{SF\_IP}$, for an edge e, $x_e = 1$ indicates that we include e in the solution while $x_e = 0$ indicates otherwise. The variable α indicates an upper bound on the violation of the load of every vertex and an upper bound on the violation of the weight. The first set of constraints ensures that the load of a vertex is upper bounded by α. The second constraint ensures that the violation for the weight is upper bounded by α. The third set of constraints ensures that the endpoints of every demand are connected. Here we assume w_{opt} is known to the algorithm, although this can be waived by standard doubling techniques.

$$\text{minimize} \quad \alpha \ . \tag{$\mathbb{SF\_IP}$}$$

$$\forall v \in V \quad \frac{1}{b_v} \sum_{e \in \delta(\{v\})} x_e \leq \alpha \ . \tag{3.1}$$

$$\frac{1}{w_{\text{opt}}} \sum_{e \in E} w(e) x_e \leq \alpha \ . \tag{3.2}$$

$$\forall S \subseteq \mathcal{S} \quad \sum_{e \in \delta(S)} x_e \geq 1 \ . \tag{3.3}$$

$$x_e \in \{0,1\}, \alpha \in \mathbb{Z}_{>0} \ .$$

3.1.3 Overview

We begin Section 3.2 by providing a bounded frequency IP for EW-DB-SF. The IP is not a proper formulation of the problem, however, we can show that one can map feasible solutions of EW-DB-SF to feasible solutions of the IP without increasing the cost too much. In Section 3.3 we provide a deterministic algorithm for online bounded frequency mixed packing/covering IPs. Finally, in Section 3.4 we merge our techniques to obtain online polylogarithmic-competitive algorithms for EW-DB-SF.

3.2 Finding the Right Integer Program

In this section we design an online mixed packing and covering integer program for EW-DB-SF. We show this formulation is near optimal, i.e. any $f-$approximation for this formulation, implies an $O(f \log^2 n)$-approximation for EW-DB-SF. In Section 3.4 we show there exists an online algorithm that finds an $O(\log n)$-approximation solution for this IP and violates degree bounds by $O(\log^3 n \log w_{\text{opt}})$, where w_{opt} denotes the optimal weight.

First we define some notations. For a sequence of demands $\sigma = \langle (s_1, t_1), \ldots, (s_k, t_k) \rangle$, we define $R_\sigma(i)$ to be a set of i edges, connecting the endpoints of the first i demands. In particular $R_\sigma(i) := \bigcup_{j=1}^{i} e(s_j, t_j)$, where $e(s_j, t_j)$ denotes a direct edge from s_j to t_j. Moreover, we say subgraph H_i satisfies the connectivity of demand $\sigma_i = (s_i, t_i)$, if s_i and t_i are connected in graph $H_i \cup R_\sigma(i-1)$. Let $\mathcal{H}_i$ denote the set of all subgraphs that satisfy the connectivity of demand σ_i. In $\mathbb{PC\_IP}$ variable α denotes the violation in the packing constraints. Furthermore for every subgraph $H \subseteq G$ and demand σ_i, there exists a variable $x_H^i \in \{0, 1\}$. $x_H^i = 1$ indicates we add the edges of H to the existing solution, at arrival of demand σ_i. The first set of constraints ensure the degree-bounds are not violated more than α. The second constraint ensures the weight is not violated by more than α. The third set of constraints ensure the endpoints of every demand are connected.

$$\text{minimize} \quad \alpha \ . \tag{$\mathbb{PC\_IP}$}$$

$$\forall v \in V \qquad \frac{1}{b_v} \sum_{i=1}^{k} \sum_{H \subseteq G} \deg_H(v) x_H^i \leq \alpha \ . \tag{3.4}$$

$$\frac{1}{w_{\text{opt}}} \sum_{i=1}^{k} \sum_{H \subseteq G} w(H) x_H^i \leq \alpha \ . \tag{3.5}$$

$$\forall \sigma_i \qquad \sum_{H \in \mathcal{H}_i} x_H^i \geq 1 \ . \tag{3.6}$$

$$\forall H \subseteq G, 1 \leq i \leq k \qquad \mathbf{x}_H^i \in \{0, 1\} \ .$$

$$\alpha > 0 \ .$$

We are considering the online variant of the mixed packing and covering program. We are given the packing Constraints (3.4) and (3.5) in advance. At arrival of demand σ_i, the corresponding covering Constraint (3.6) is added to the program. We are looking for an online solution which is feasible at every online stage. Moreover the variables x_H should be monotonic, i.e. once an algorithm sets $x_H = 1$ for some H, the value of x_H is 1 during the rest of the algorithm. Figure (3.1) illustrates an example which indicates the difference between the solutions of $\mathbb{PC\_IP}$ and $\mathbb{SF\_IP}$.

Let popt and lopt denote the optimal solutions for $\mathbb{PC\_IP}$ and $\mathbb{SF\_IP}$, respectively. Lemma 3.1 shows that given an online solution for $\mathbb{PC\_IP}$ we can provide a feasible online solution for $\mathbb{SF\_IP}$ of cost popt.

Lemma 3.1. *Given a feasible solution $\{\mathbf{x}, \alpha\}$ for $\mathbb{PC\_IP}$, there exists a feasible solution $\{\mathbf{x}', \alpha\}$ for $\mathbb{SF\_IP}$.*

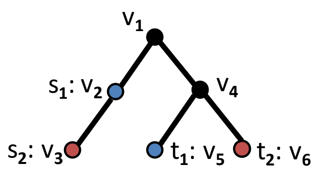


Figure 3.1: An example where every vertex has degree-bound 3 and every edge has weight 1. The first demand is (v_2, v_5) and the second demand is (v_3, v_6). The optimal solution for $\mathbb{SF\_IP}$ is a subgraph, say H, with the set of all edges and vertices, i.e. $H = G$. However an optimal solution for $\mathbb{PC\_IP}$ is: Two subgraphs H_1 for the first request which has edges $\{e(v_1, v_2), e(v_1, v_4), e(v_4, v_5)\}$ and H_2 for the second request which has edges $\{e(v_2, v_3), e(v_4, v_5), e(v_4, v_6)\}$. Note that $w(H) = 5$ and $w(H_1) + w(H_2) = 6$, since we have edge $e(v_4, v_5)$ in both H_1 and H_2. Moreover the number of edges incident with v_4 in the solution of $\mathbb{PC\_IP}$ is 4, i.e. $\deg_{H_1}(v_4) + \deg_{H_2}(v_4) = 4$.

In the rest of this section, we show that we do not lose much by changing $\mathbb{SF\_IP}$ to $\mathbb{PC\_IP}$. In particular we show $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$.

To this end, we first define the *connective* list of subgraphs for a graph G, a forest F, and a list of demands σ. We then prove an existential lemma for such a list of subgraphs with a desirable property for any $\langle G, F, \sigma\rangle$. With that in hand, we prove $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$. Given G, a list of demands $\sigma = \langle (s_1, t_1), \ldots, (s_k, t_k)\rangle$, and a forest $F \subseteq G$:

Definition 3.1. *Let $Q = \langle Q_1, Q_2, Q_3, \ldots, Q_k\rangle$ be a list of k subgraphs of F. We say Q is a connective list of subgraphs for $\langle G, F, \sigma\rangle$ iff for every $1 \leq i \leq k$ there exists no cut disjoint from Q_i that separates s_i from t_i, but does not separate any s_j from t_j for $j < i$.*

The intuition behind the definition of connective subgraphs is the following: If Q is a connective list of subgraphs for an instance $\langle G, F, \sigma\rangle$ then for every i we are guaranteed that the union of all subgraphs $\cup_{j=1}^{i} Q_i$ connects s_i to t_i. In Lemma 3.2 we show for every $\langle G, F, \sigma\rangle$, there exists a connective list of subgraphs for $\langle G, F, \sigma\rangle$, such that each edge of F appears in at most $O(\log^2 n)$ subgraphs of Q.

Lemma 3.2. *Let G be a graph and F be a forest in G. If σ is a collection of k demands $\langle (s_1, t_1), \ldots, (s_k, t_k)\rangle$, then there exists a connective list of subgraphs $Q = \langle Q_1, Q_2, \ldots, Q_k\rangle$ for $\langle G, F, \sigma\rangle$ such that every edge of F appears in at most $3 \log^2 |V(F)|$ number of Q_i's.*

Proof.

Here we give a sketch of the proof of lemma; we refer the reader to the full version for detailed proofs. We first prove a cost-minimization variant of the lemma. Consider an arbitrary weight vector $\hat{w} : F \to \mathbb{R}^{\geq 0}$. We argue that there is a connective list Q, such that $\sum_i \hat{w}(Q_i) \leq O(\log^2 n)\hat{w}(F)$. Let $\hat{H}_i = (V, F \cup R_\sigma(i), \hat{w}_i)$ denote a weighted graph for which $\hat{w}_i(e) = \hat{w}(e)$ for $e \in F$, and $\hat{w}_i(e) = 0$ for $e \in R_\sigma(i)$. Now we note that there is no cost-sharing among Q_i's in the goal $\sum_i \hat{w}(Q_i)$. Therefore the optimal choice for Q_i corresponds to the minimum-weight (s_i, t_i)-path in $\hat{H}_{i-1}$. Hence, we need to analyze the cost of these greedy choices.

Awerbuch et al. [35] showed that the greedy algorithm is indeed $O(\log^2 n)$-competitive for the edge-weighted Steiner forest problem. The standard greedy algorithm is slightly different from the greedy process we discussed above. In the greedy algorithm of Awerbuch et al., at time step i we choose a minimum-cost (s_i, t_i)-

path in a graph in which there is a zero-cost edge between *any* pair of vertices in the same connected component of the current solution; not just the (s_j, t_j) pairs of the previous demands. However, in their analysis they only use the zero-cost edges among the terminals of a previous demand. This is indeed not surprising since we hardly have any control on the greedy choices other than the fact that they satisfy the demands. Therefore the following claim follows from the result of Awebuch et al.[2]:

Claim 3.1 (implicitly proven in Theorem 2.1 of [35]). *For any weight function $\hat{w}$ defined over F, there exists a connective list Q for which*

$$\sum_i \hat{w}(Q_i) \leq O(\log^2 n)\hat{w}(F)$$

.

However, Claim 3.1 is not enough for us. We need a solution in which every edge is used at most $O(\log^2 n)$ times, not just in an amortized sense. Indeed we can show that since there is a solution for *every* weight function, we can have a *fractional* connective list Q in which every edge is used (fractionally) at most $O(\log^2 n)$ times. This implies that we have a fractional connective list. Finally, we provide a rounding argument which obtains an integral connective list by losing only a constant factor; which completes the proof of lemma.

□

[2]There is also a lower bound of $\Omega(\log n)$ for the competitive ratio of the greedy algorithm. Closing the gap between this lower bound and the upper bound of $O(\log^2 n)$ for EW Steiner forest is an important open problem.

Finally, we can leverage Lemma 3.2 to show $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$. This shows we can use $\mathbb{PC\_IP}$ as an online mixed packing/covering IP to obtain an online solution for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST losing a factor of $O(\log^2 n)$.

In Section 3.4 we show this formulation is an online bounded frequency mixed packing/covering IP, thus we leverage our technique for such IPs to obtain a polylogarithmic-competitive algorithm for online EW-DB-SF.

3.3 Online Bounded Frequency Mixed Packing/Covering IPs

In this section we consider bounded frequency online mixed packing and covering integer programs. For every online mixed packing and covering IP with covering frequency k, we provide an online algorithm that violates each packing constraint by at most a factor of $O(k \log m)$, where m is the number of packing constraints. We note that this bound is independent of the number of variables, the number of covering constraints, and the coefficients of the mixed packing and covering program. Moreover the algorithm is for integer programs, which implies obtaining an integer solution does not rely on (online) rounding.

In particular we prove there exists an online $O(k \log m)$-competitive algorithm for any mixed packing and covering IP such that every variable has covering frequency at most k, where the covering frequency of a variable x_r is the number of covering constraints with a non-zero coefficient for x_r.

We assume that all variables are binary. One can see this is without loss of generality as long as we know every variable $x_r \in \{1, 2, 3, \ldots, 2^l\}$. Since we can replace x_r by l variables $y_r^1, \ldots, y_r^l$ denoting the digits of x_r and adjust coefficients accordingly. Furthermore, for now we assume that the optimal solution for the given mixed packing and covering program is 1. In Theorem 3.4 we prove that we can use a doubling technique to provide an $O(k \log m)$-competitive solution for online bounded frequency mixed packing and covering programs with any optimal solution.

The algorithm is as follows. We maintain a family of subsets $\mathcal{S}$. Initially $\mathcal{S} = \emptyset$. Let $\mathcal{S}(j)$ denote $\mathcal{S}$ at arrival of C_{j+1}. For each covering constraint C_{j+1}, we find a subset of variables S_{j+1} and add S_{j+1} to $\mathcal{S}$. We find S_{j+1} in the following way. For each set of variables S, we define a cost function $\tau_S(\mathcal{S}(j))$ according to our current $\mathcal{S}$ at arrival of C_{j+1}. We find a set S_{j+1} that satisfies C_{j+1} and minimizes $\tau_S(\mathcal{S}(j))$. More precisely we say a set of variables S satisfies C_{j+1} if

- $\sum_{x_r \in S} C_{j+1,r} x_r \geq 1$, where $C_{j+1,r}$ denotes the coefficient of C_{j+1} for x_r.
- For each packing constraint P_i, $\sum_{x_r \in S} \frac{1}{k} P_{ir} \leq 1$.

Now we add S_{j+1} to $\mathcal{S}$ and for every $x_r \in S_{j+1}$, we set $x_r = 1$. We note that there always exists a set S that satisfies C_{j+1}, since we assume there exists an optimal solution with value 1. Setting S to be the set of all variables with value one in an optimal solution which have non-zero coefficient in C_{j+1}, satisfies C_{j+1}. It only remains to define $\tau_S(\mathcal{S}(j))$. But before that we need to define $\Delta_i(S)$ and $F_i(\mathcal{S}(j))$. For packing constraint P_i and subset of variables S, we define $\Delta_i(S)$ as

$\Delta_i(S) := \sum_{x_r \in S} \frac{1}{k} P_{ir}$. For packing constraint P_i and $\mathcal{S}(j)$, let

$$F_i(\mathcal{S}(j)) := \sum_{S \in \mathcal{S}(j)} \Delta_i(S) \ . \tag{3.7}$$

Now let $\tau_S(\mathcal{S}(j)) = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}$, where $\rho > 1$ is a constant to be defined later.

Algorithm 2

Input: Packing constraints P, and an online stream of covering constraints $C_1, C_2, \ldots$.

Output: A feasible solution for online bounded frequency mixed packing/covering.

Offline Process:

```
1: Initialize $\mathcal{S} \leftarrow \emptyset$.
```

Online Scheme; assuming a covering constraint C_{j+1} is arrived:

```
1: $S_{j+1} \leftarrow \arg\min_S \{\tau_S(\mathcal{S}(j)) \mid S$ satisfies $C_{j+1}\}$.
2: for all $x_r \in S_{j+1}$ do
3:     $x_r \leftarrow 1$.
```

Let $\mathbf{x}^*$ be an optimal solution, and $\mathbf{x}^*(j)$ denote its values at online stage j. We define $G_i(j)$ as

$$G_i(j) := \sum_{l=1}^{j} \sum_{r:C_{lr}>0} \frac{1}{k} x_r^* P_{ir} \ . \tag{3.8}$$

Now we define a potential function Φ_j for online stage j.

$$\Phi_j = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \ , \tag{3.9}$$

where $\rho, \gamma > 1$ are constants to be defined later.

Lemma 3.3. *There exist constants ρ and γ, such that Φ_j is non-increasing.*

Proof. We find ρ and γ such that $\Phi_{j+1} - \Phi_j \leq 0$. By the definition of Φ_j,

$$\Phi_{j+1} - \Phi_j = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j+1))}(\gamma - G_i(j+1)) - \rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \ . \tag{3.10}$$

By Equation (3.7), $\rho^{F_i(\mathcal{S}(j+1))} - \rho^{F_i(\mathcal{S}(j))} = \rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}$. Moreover by Equation (3.8), $(\gamma - G_i(j+1)) - (\gamma - G_i(j)) = -\sum_{r:C_{j+1,r}>0} \frac{1}{k} x_r^* P_{ir}$. For simplicity of notation we define $B_i(j+1) := \sum_{r:C_{j+1,r}>0} \frac{1}{k} x_r^* P_{ir}$. Thus we can write Equation (3.10) as:

$$\Phi_{j+1} - \Phi_j = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j+1))}(\gamma - G_i(j) - B_i(j+1)) - \rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \tag{3.11}$$

$$= \sum_{i=1}^{m} (\gamma - G_i(j))(\rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j+1))} B_i(j+1) \quad \text{Since } G_i(j) \geq 0$$

$$\leq \sum_{i=1}^{m} \gamma(\rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j+1))} B_i(j+1) \quad F_i(\mathcal{S}(j+1)) \geq F_i(\mathcal{S}(j))$$

$$\leq \sum_{i=1}^{m} \gamma(\rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j))} B_i(j+1) \ .$$

Now according to the algorithm for each subset of variables S' such that $\sum_{x_r \in S'} C_{j+1}(x_r) \geq 1$, either $\tau_S(\mathcal{S}(j)) \leq \tau_{S'}(\mathcal{S}(j))$ or there exists a packing constraint P_i such that $\Delta_i(S') > 1$. In $B_i(j+1)$, we are considering variables x_r such that $x_e^* = 1$, thus for every P_i, $B_i(j+1) \leq 1$. Therefore setting S' to be the set of variables x_r such that $x_r^* = 1$ and $C_{j+1,r} > 0$, we have $\tau_S(\mathcal{S}(j)) \leq \tau_{S'}(\mathcal{S}(j))$. Thus $\sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))} \leq \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j))+B_i(j+1)} - \rho^{F_i(\mathcal{S}(j))}$. Therefore we can rewrite Inequality (3.11) as

$$\Phi_{j+1} - \Phi_j \leq \sum_{i=1}^{m} \gamma(\rho^{F_i(\mathcal{S}(j))+B_i(j+1)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j))} B_i(j+1) \tag{3.12}$$

$$= \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j))}(\gamma \rho^{B_i(j+1)} - \gamma - B_i(j+1)) \ .$$

We would like to find ρ and γ such that Φ_j is non-increasing. We find ρ and γ such that for each packing constraint P_i, $\gamma\rho^{B_i(j+1)} - \gamma - B_i(j+1) \leq 0$. Thus

$$\gamma\rho^{B_i(j+1)} - \gamma \leq B_i(j+1) \qquad \text{Since } 0 \leq B_i(j+1) \leq 1 \tag{3.13}$$

$$\gamma\rho B_i(j+1) - \gamma \leq B_i(j+1) \qquad \text{By simplifying} \tag{3.14}$$

$$\rho \leq 1 + 1/\gamma \ . \tag{3.15}$$

Thus if we set $\rho \leq 1 + 1/\gamma$, Φ_j is non-increasing, as desired. □

Now we prove Algorithm 2 obtains a solution of at most $O(k \log m)$.

Lemma 3.4. *Given an online bounded frequency mixed packing covering IP with optimal value 1, there exists a deterministic integral algorithm with competitive ratio $O(k \log m)$, where m is the number of packing constraints and k is the covering frequency of the IP.*

Proof. By Lemma 3.3 for each stage j, $\Phi_{j+1} \leq \Phi_j$. Therefore $\Phi_j \leq \Phi_0 = \gamma m$. Thus for each packing constraint P_i,

$$\rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \leq \gamma m \ . \tag{3.16}$$

Thus,

$$\rho^{F_i(\mathcal{S}(j))} \leq \frac{\gamma m}{(\gamma - G_i(j))} \leq \frac{\gamma m}{\gamma - 1} \ . \qquad \text{Since } G_i(j) \leq 1 \tag{3.17}$$

Thus we can conclude

$$F_i(\mathcal{S}(j)) \in O(\log m) \ . \tag{3.18}$$

By definition of $F_i(\mathcal{S}(j))$, $F_i(\mathcal{S}(j)) = \sum_{S \in \mathcal{S}(j)} \Delta_i(S) = \sum_{S \in \mathcal{S}(j)} \sum_{x_r \in S} \frac{1}{k} P_{ir}$. Since each variable x_r is present in at most k sets, $\frac{1}{k} P_i \cdot \mathbf{x}(j) \leq F_i(\mathcal{S}(j))$. Thus by Inequality (3.18) $P_i\mathbf{x}(j) \in O(k \log m)$, which completes the proof. □

Finally we prove there exists an online $O(k \log m)$-competitive algorithm for bounded frequency online mixed packing and covering integer programs with any optimal value.

Theorem 3.4. *Given an instance of the online mixed packing/covering IP, there exists a deterministic integral algorithm with competitive ratio $O(k \log m)$, where m is the number of packing constraints and k is the covering frequency of the IP.*

3.4 Putting Everything Together

In this section we consider the online mixed packing/covering formulation discussed in Section 3.2 for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST $\mathbb{PC\_IP}$. In this section we show this formulation is an online bounded frequency mixed packing/covering IP. Therefore we our techniques discussed in Section 3.3 to obtain a polylogarithmic-competitive algorithm for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST.

First we assume we are given the optimal weight w_{opt} as well as degree bounds. We can obtain the following theorem.

Theorem 3.5. *Given the optimal weight w_{opt}, there exists an online deterministic algorithm which finds a subgraph with total weight at most $O(\log^3 n)w_{opt}$ while the degree bound of a vertex is violated by at most a factor of $O(\log^3 n)$.*

Proof. By Lemma 3.1, given a feasible online solution for $\mathbb{PC\_IP}$ with violation α, we can provide an online solution for $\mathbb{SF\_IP}$ with violation α. Moreover, in

Section 3.2 we show that $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$. Thus given an online solution for $\mathbb{PC\_IP}$ with competitive ratio f, there exists an $O(f \log n)$-competitive algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST. We note that in $\mathbb{PC\_IP}$ we know the packing constraints in advance. In addition every variable x_H^i has non-zero coefficient only in the covering constraint corresponding to connectivity of the i-th demand endpoints, i.e. the covering frequency of every variable is 1. Therefore by Theorem 3.4 there exists an online $O(\log m)$-competitive solution for $\mathbb{PC\_IP}$, where m is the number of packing constraints, which is $n + 1$. Thus there exists an online $O(\log^3 n)$-competitive algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST. This means the violation for both degree bounds and weight is of $O(\log^3 n)$. $\square$

Chapter 4: Online Survivable Network Design

4.1 Introduction

In an instance of the *network design* problem, we are given a graph $G = (V, E)$, an edge-cost function $c : E \to \mathbb{R}^{\geq 0}$, and a connectivity criteria. The goal is to find a minimum-cost subgraph H of G that satisfies the connectivity requirements. An important family of this class is the *survivable network design problem* (SNDP): Given non-negative integers r_{uv} for each pair $u, v \in V$, the solution subgraph H should contain r_{uv} edge-disjoint paths for each pair u and v. SNDP arises in fault tolerance management and thus is of much interest in design community: the connectivity of nodes u and v in H is resilient to even $(r_{uv} - 1)$ edge failures. This problem clearly generalizes the *Steiner tree*[1] and *Steiner forest*[2] problems.

For a non-empty cut $S \subset V$, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. SNDP falls in the general class of network design problems that can be characterized by *proper cut functions*. A function $f : 2^V \to \mathbb{Z}^{\geq 0}$ defined over cuts in the graph is *proper*, if it is symmetric ($f(S) = f(V \setminus S)$ for all $S \subset V$) and it

[1]In the Steiner tree problem, given a set of terminal nodes $T \subset V$, the goal is to find a minimum-cost subgraph connecting all terminals.

[2]In the Steiner forest problem, given a set of pairs of vertices $s_i, t_i \in V$, the goal is to find a minimum-cost subgraph in which every pair is connected.

satisfies maximality ($f(S \cup T) \leq \max\{f(S), f(T)\}$ for all $S \cap T = \phi$). For SNDP, one can choose $f(S) = \max_{u \in S, v \notin S} r_{uv}$ for every cut S. Given a proper function f over cuts in the graph, the goal is to find a minimum-cost subgraph H such that

$$|E(H) \cap \delta(S)| \geq f(S) \qquad\qquad \forall \textit{non-empty } S \subset V \ .$$

Over the past decades, the offline SNDP and proper cut functions have been extensively studied especially as an important testbed for primal-dual and iterative rounding methods (see e.g. [72–77]). In this study, we consider SNDP in the *online* setting: we are given a graph $G = (V, E)$ and an edge-cost function c in advance. We receive an online sequence of demands in the form of tuples $(u, v, r_{uv}) \in V \times V \times \mathbb{Z}^{\geq 0}$. We start with the empty subgraph H. Upon the arrival of a demand (u, v, r_{uv}), we need to immediately *augment* H such that there exist at least r_{uv} edge-disjoint paths between u and v in H. The goal is to minimize the cost of H. The *competitive ratio* of an algorithm is defined as the maximum ratio of the cost of its output and that of an optimal offline solution, over all possible input instances.

The online 1-connectivity problems, in which $r_{uv} \in \{0, 1\}$ for all pairs, have been extensively studied in the last decades. Imase and Waxman [78] (SIAM'91) were first to consider the edge-weighted Steiner tree problem. They used a dual-fitting argument to show that the natural greedy algorithm is $O(\log n)$-competitive where $n = |V|$[3]. Their result is asymptotically tight. Later, Berman and Coulston [79] (STOC'97) and Awerbuch, Azar, and Bartal [80] (TCS'04) demonstrated

[3]In fact, the competitive ratio is $O(\log \min\{n, \mathcal{D}\})$ where $\mathcal{D}$ is the number of demand requests. However, to simplify the comparison with results for SNDP, in this study we ignore the distinction between this factor and $O(\log n)$.

an $O(\log n)$-competitive algorithm for the more general Steiner forest problem by designing an elegant online primal-dual technique. The latter also shows that the greedy algorithm achieves the competitive ratio of $O(\log^2 n)$ for Steiner forest. Indeed, due to the simplicity of greedy approaches, an important open problem is to settle the competitiveness of the greedy algorithm for Steiner forest. In the recent years, several primal-dual techniques are developed for solving node-weighted variants [8, 81, 82], and prize-collecting variants [83, 84] of 1-connectivity problems.

Gupta, Krishnaswamy, and Ravi [7] (SIAM'12) were first to consider the online survivable network design problem. They demonstrate an elegant algorithm with competitive ratio of $\tilde{O}(k \log^3 n)$, where $k = \max_{u,v} r_{uv}$. The crux of their analysis is to use distance-preserving tree-embeddings in an online setting. More precisely, they first pick a random distance-preserving spanning subtree $T \subseteq G$. They satisfy a connectivity demand r_{uv} by iteratively increasing the connectivity of u and v. In each iteration, they show that it is sufficient to use cycles that are formed by an edge $e = (a, b) \notin T$ and the $\{a, b\}$-path in T; hence, reducing the number of *options* for satisfying a connectivity demand. This would enable them to use a set cover approach to solve the problem in an online manner and achieve the first competitive algorithm for online SNDP.

Single-source SNDP is a variant of SNDP where all demands share a same endpoint. Naor, Panigrahi, and Singh [8] (FOCS'11) partially improve the results of Gupta et al. [7] by demonstrating a *bi-criteria* competitive algorithm for single-source SNDP using structural properties of a single-source optimal solution. A bi-criteria competitive ratio of (α, β) for SNDP implies that the solution produced by the online algorithm achieves a connectivity of $\lfloor \frac{r_i}{b} \rfloor$ for every demand σ_i and is at most a factor of α more expensive than the optimal offline solution for connectivity of r_i. The algorithm by Naor et al. achieves the competitive ratio of $O(\frac{k \log n}{\epsilon}, 2+\epsilon)$ for any $\epsilon > 0$. They also study and give bi-criteria algorithms for the vertex-connectivity problem.

The competitive ratio of algorithms by Gupta et al. and Naor et al. grows linearly in k. This seems to be inherent to their methods since they may need to solve a set-cover-like problem in each iteration of incrementing the connectivity of a demand; hence, losing a polylogarithmic factor in each iteration. One would need a new approach to break the linear dependency to k. Indeed, both factors of $O(\log^3 n)$ and $O(k)$ are not plausible in practice, and an important open problem in the online community [7,8] is whether the linear dependency to k can be reduced to a logarithmic dependency. In this study we circumvent this problem by two main approaches, first by considering the stochastic setting of the problem, and also by allowing small congestion on the edges.

Allowing small congestion. Interestingly, we show that if the online algorithm is allowed to buy three copies of an edge, then a simple greedy approach is $O(\log^2 n \log k)$-competitive. More precisely, we demonstrate a deterministic algorithm with a bi-criteria competitive ratio of $(O(\log^2 n \log_{1+\epsilon} k), 2+\epsilon)$ for any constant $\epsilon > 0$. For the single-rooted variant, the competitive ratio is $(O(\log n \log_{1+\epsilon} k), 2+\epsilon)$. In our analysis of the greedy algorithm, our main technical contributions are two folds: (i) establishing the connection between online SNDP and the celebrated *Steiner packing* problem; and (ii) demonstrating the optimal packing ratio of 1/2 for the relaxed *fractional* variant of Steiner packing problem; which is a decades old open problem for the integral variant.

It is worth mentioning that one can think of our bi-criteria algorithm as a solution to the online SNDP with congestion 2. Relaxing a hard-to-approximate problem by allowing small congestion is quite natural and has been very fruitful over the past decade. Perhaps the most important example of such line of research is the *edge-disjoint path* problem with congestion [85–94], for which the work of Chuzhoy and Li [9] (FOCS'12) gives a polylogarithmic approximation algorithm with congestion 2.

Stochastic SNDP. For many online optimization problems it is natural and fruitful to assume that at each online step the online query is drawn independently from a known probability distribution. We call this model the *i.i.d.* model. This model has been considered for many fundamental problems (see e.g. online stochastic matching [95–100], k-server [101, 102], set-cover, Steiner network design, facility location [103], multi-commodity flow [104], among others). There are two important generalizations of the i.i.d. model. The unknown distribution i.i.d. model, where the queries are again drawn independently from the same probability distribution, but the probability distribution is not known, and also the *prophet setting*. In the prophet setting, instead of only one distribution for every online query, the queries are independently drawn from different distributions, inspired by the classical *prophet inequality* problem. The prophet setting is also studied for many online problems (see e.g. [105–108]). In this study we consider all three variants of the stochastic SNDP and show that we can achieve significantly more efficient algorithms having a stochastic information about the input. We first provide an oblivious constant competitive algorithm for the i.i.d. SNDP. Using a similar approach to Garg et al. [103] we provide an $O(\log n)$-competitive algorithm for the unknown distribution i.i.d. SNDP. Then we provide a constant competitive algorithm for the prophet SNDP, through a novel technique which leverages the oblivious algorithm provided for the i.i.d. SNDP.

Interestingly we provide a general framework to obtain competitive algorithms for online optimization problems in the prophet setting. Indeed it is an interesting open problem that what is the relation between the i.i.d. setting and the prophet setting? We show, by a simple but tricky technique that we can obtain a competitive online algorithm in the prophet setting if we can design an oblivious competitive algorithm for the problem in the i.i.d. setting. Using this technique we provide asymptotically tight competitive algorithms for fundamental online problems in prophet setting, such as set cover and facility location.

4.1.1 Our Results and Techniques

Let $\sigma_i = (u_i, v_i, r_i)$ denote the i-th connectivity demand. Consider the following intuitive greedy approach. Upon the arrival of σ_i, we augment the solution subgraph H, by finding the minimum-cost set of edges whose addition to H creates r_i edge-disjoint paths between u_i and v_i. Awerbuch et al. [80] (TCS'04) show that if all the demands require 1-connectivity (i.e., $r_i = 1$ for every i), this algorithm achieves a competitive ratio of $O(\log^2 n)$. This leads to a natural question that whether greedy works for higher connectivity problems as well. However, we show an instance of online SND in Section 4.3, for which the greedy algorithm has a competitive ratio of $\Omega(n)$. Indeed, the connectivity demands in the instance are either zero or two, hence greedy is not competitive even for low connectivity demands.

In this study we interestingly show that *greedy-like* algorithms do surprisingly well, if we consider the stochastic version of the problem, or if we allow a small congestion on the edges. In the following we present our algorithms techniques in a high-level perspective.

Allowing small congestion. We show that a greedy algorithm does surprisingly well, if we relax the connectivity requirements by a constant factor. Let α denote an arbitrary scale factor. We define an *α-scaled* variant of the greedy algorithm in which the goal is to find only $\lfloor \frac{r_i}{\alpha} \rfloor$ disjoint paths between the endpoints of σ_i. Our main result states that the scaled greedy algorithm is polylogarithmic competitive.

Theorem 4.1. *For any constant $\epsilon > 0$, the $(2+\epsilon)$-scaled greedy algorithm is $(O(\log^2 n \log_{1+\epsilon} k), 2+\epsilon)$-competitive. For the single-source variant, the competitive ratio is $(O(\log n \log_{1+\epsilon} k), 2+\epsilon)$.*

Furthermore, for uniform SNDP, 2-scaled greedy is $(O(\log^2 n), 2)$-competitive.

We start by demonstrating a deep connection between the greedy method for SNDP and the *Steiner packing problems.* The Steiner packing problems are motivated by vast applications in VLSI-layout and has been used as an algorithmic toolkit in computer science. In the *Steiner tree packing* problem, we are given a graph $G = (V, E)$ and a set S of vertices and the goal is to find the *Steiner decomposition number* (SDN), the maximum number of edge-disjoint subgraphs that each connects the vertices of S. We note that a minimal connecting subgraph is a *Steiner tree* with respect to S. In the *Steiner forest packing* problem, we are given a set of demand pairs $u_i, v_i \in V$ and the goal is to find SDN, the maximum number of edge-disjoint subgraphs that in each, the demand pairs are connected.

For simplicity, let us assume we have an *uniform* instance. Let opt denote the optimal SNDP solution, with the Steiner decomposition number q. In Section 4.3, we show that the $(\frac{k}{q})$-scaled greedy algorithm approximates opt up to logarithmic factors. Intuitively, every forest in the Steiner forest decomposition, gives us a path to satisfy a demand. Hence, we need to bound the overall cost of satisfying demands in all the q forests. The crux of our analysis is then to charge the cost of the scaled greedy to that of a parallel set of greedy algorithms that solve 1-connectivity instances on every forest. Finally, to get a polylogarithmic competitive algorithm, we need to find a universal lower bound on the SDN number q with respect to k.

It is shown that finding SDN is NP-hard and cannot be computed in polynomial time unless P=NP [109] (Algorithmica'06). Given that there exist q disjoint Steiner forests connecting a set of demands, it is straight forward to show the graph is q-connected on the demands. Therefore, a natural upper bound on SDN is the minimum connectivity of the endpoints of demands. For the case of spanning trees (Steiner tree with $S = V(G)$), it is proven that the above upper bound also provides a good approximation guarantee for the problem. In other words, any k-connected graph can be decomposed into $k/2$ edge-disjoint spanning trees [110] (JCT'03). This is also followed by a matching upper bound. The problem is much subtler when S does not encompass all vertices of the graph. The first lower bound for the Steiner tree packing problem was achieved by Petingi and Rodriguez [111] (CON'03) who proved every S-k-connected[4] has $\lfloor (2/3)^{(|V(G)-|S|)} k/2 \rfloor$ dijsoint Steiner trees. This was later improved by Kriesell [110] (JCT'03), Jain, Mahdian, and Salavatipour [112] (SODA'03), Lua [113] (FOCS'04), and DeVos, McDonald, and Pivotto [114] (Man'13), the most recent of which shows for every S-$(5k+4)$-connected graph, we can find k edge-disjoint Steiner trees. However, the main conjecture is that, similar to the case of spanning trees, every S-k-connected graph admits a $k/2$-dijsoint Steiner tree decomposition [110]. In this study, we prove this conjecture for the *fractional* variant of the problem.

For a set of demand pairs (u_i, v_i)'s, let $\mathcal{T}$ denote the set of Steiner forests that satisfy all the demands. In the *fractional Steiner forest packing* problem, the output is a fractional assignment x over $\mathcal{T}$ such that for every edge e, $\sum_{T \in \mathcal{T}: e \in T} x_T$ is not more than one. The goal is to find a fractional Steiner forest decomposition with maximum $||x||$. In Section 4.2, we prove the fractional variant of the conjecture of Kriesell. We believe this result can be of independent interest in improving upon other problems that rely on Steiner packing problems. We would like point out that the fractional Steiner forest decomposition is also presented in [114].

Theorem 4.2. *Given a set of demand pairs (u_i, v_i), if G is k-connected for every demand pair, then the fractional Steiner decomposition number is at least $k/2$.*

[4]A graph which is k-connected on a set of vertices S

Indeed, in Section 4.3, we use a dependent rounding method to show that the connection between SDN and the competitiveness of the greedy approach holds even for the stronger fractional variant of SDN. Hence, Theorem 4.2 implies that the 2-scaled greedy algorithm, achieves a polylogarithmic competitive ratio for the uniform SNDP. Finally, in Section 4.4, we prove Theorem 4.9 by showing that the scaled greedy is also competitive for the non-uniform variant, if one is willing to lose an extra $\log k$ factor in the competitive ratio.

Stochastic SNDP. A single-source *uniform* instance of online SNDP is an instance in which for every demand σ_i, $u_i = u$, $r_i = k$ for some vertex $u \in V$ and integer k. For a non-uniform variant, let $k = \max_i r_i$. Let D be a given probability distribution over V. In i.i.d. SNDP at each online step i, a random connectivity demand $\sigma_i = (u, v_i, k)$ arrives, where v_i is drawn independently at random from distribution D. We call the problem unknown distribution SNDP if the probability distribution D is not given in advance. Another interesting generalization of the i.i.d. model, which we call the prophet SNDP is defined as follows. In prophet SNDP, instead of only a single probability distribution D, we are given T probability distributions $D_1, \ldots, D_T$, such that the i-th demand is $\sigma_i = (u, v_i, k)$, where v_i is drawn independently at random from distribution D_i. In all three variants of the stochastic SNDP, the competitive-ratio is defined as the the expected cost of an algorithm $\mathcal{A}$ over the expected cost of an optimal offline algorithm while the distribution(s) is chosen by an adversary. More precisely let $E[A(\omega)]$ and $E[opt(\omega)]$ denote the expected cost of an algorithm $\mathcal{A}$ and the expected cost of an optimal offline algorithm for an online scenario ω, respectively. Thus the competitive-ratio of algorithm $\mathcal{A}$ is defined as follows.

$$cr(\mathcal{A}) := \max_{D} \frac{E_{\omega \sim D}[\mathcal{A}(\omega)]}{E_{\omega \sim D}[opt(\omega)]}.$$

We first provide an oblivious algorithm for the i.i.d. SNDP. The algorithm has two steps. First we realize a random scenario from distribution D. Then we compute a 2-approximate solution for this random offline instance using Jain [75]. We buy all the edges in the offline solution. Now considering the edges already selected, for each vertex v we compute a minimum-cost k-flow to the root u. In the second step, upon arrival of each demand, we buy the computed minimum-cost k-flow to satisfy the k-connectivity to the root. While the algorithm is similar to the algorithm in Garg et al. [103] for the 1-connectivity case, the analysis needs careful consideration of the k-connected graphs. In fact, despite previous works on the online survivable network design which do not take the structures of k-connected graphs into account, we obtain a structural result about k-connected graphs. Then we leverage the structural result to analyze our algorithm, and to prove that our algorithm is 4-competitive. This is quite surprising, since it is dramatically improving the known $O(k \log^3 n)$-competitive-ratio.

Theorem 4.3. *There exist a 4-competitive algorithm for i.i.d. SNDP.*

Afterwards, we show that a greedy algorithm is $O(\log n)$-competitive if the input is drawn from an even unknown distribution. This is very interesting since, in the adversary setting where there is no stochastic information about the input, we show that a greedy algorithm may be $\Omega(n)$-competitive. Note that due to [103] even the 1-connectivity version of this problem is $\Omega(\frac{\log n}{\log \log n})$-hard.

Theorem 4.4. *There exist an $O(\log n)$-competitive algorithm for unknown distribution SNDP.*

Then we consider the prophet SNDP. In fact we provide a general framework to obtain competitive algorithms for online optimization problems in prophet setting.

From oblivious i.i.d. to prophet. We show if there exists a competitive oblivious algorithm for an online problem in i.i.d. setting, we can obtain a competitive algorithm for the same problem in prophet setting.

Theorem 4.5. *Given an oblivious α-competitive online algorithm for problem $\mathcal{P}$ in the i.i.d. setting, there exists a $\alpha\frac{2e}{e-1}(1+o(1))$-competitive online algorithm for $\mathcal{P}$ in prophet setting.*

Roughly speaking, we show that we can combine different distributions in the prophet setting to obtain a single distribution. Then using the i.i.d. oblivious algorithm, we may not lose more than a constant factor in the competitive ratio. Thus the following is a direct corollary of this technique.

Corollary 4.1. *There exists a $O(1)$-competitive algorithm for prophet SNDP.*

Using this framework, we can obtain competitive algorithms for many fundamental and classical problems in prophet setting. For example define $D_1, \dots, D_T$ be T probability distributions over the elements of a set cover instance. Now let i-th demand of a set cover problem be an element randomly and independently drawn from distribution D_i. We call this problem the prophet set cover problem. Similarly one may define prophet facility location the same as the classical facility location problem, with the difference that the i-th demand is randomly drawn from a known distribution D_i. Garg et al. [103] provide oblivious online algorithms for i.i.d. facility location and i.i.d. vertex cover. Grandoni et al. [115] also provide an oblivious online algorithm for i.i.d. set cover. Thus we can directly obtain the following corollaries.

Corollary 4.2. *There exists an $O(1)$-competitive algorithm for prophet vertex cover.*

Corollary 4.3. *There exists an $O(1)$-competitive algorithm for prophet facility location.*

Corollary 4.4. *There exists an $O(\log n)$-competitive algorithm for prophet set cover.*

4.1.2 Further Related Work

Over the past decades, SNDP and proper cut functions have been an important testbed for primal-dual and iterative rounding methods. Goemans and Williamson [73] (SIAM'95) were first to consider the case of $\{0,1\}$-proper functions. They used a primal-dual method to obtain a 2-approximation algorithm for the problem; which later on got generalized to the celebrated moat-growing framework for solving connectivity problems. Klein and Ravi [76] (IPCO'93) considered the two-connectivity problem and the case of $\{0,2\}$-proper functions. They gave a primal-dual 3-approximation algorithm for the problem. Williamson, Goemans, Mihail, and Vazirani [77] (Combinatorica'95) were first to consider general proper functions. They too developed a primal-dual algorithm with approximation ratio $2k$, where $k = \max_S f(S)$. Subsequently, Goemans, Goldberg, Plotkin, Shmoys, Tardos, and Williamson [72] (SODA'94) presented a primal-dual $2H(k)$-approximation algorithm, where $H(k)$ is the k^{th} harmonic number. Finally, in his seminal work [75] (Combinatorica'01), Jain introduced the *iterative rounding* method by developing a 2-approximation algorithm for network design problems characterized by proper cut functions[5]. We refer the reader to [74] for a survey of results for (offline) network design problems.

Prophet inequality has been first studied in 70s by Krengel and Sucheston [116, 117]. Hajiaghayi, Kleinberg and Sandholm [118] study the relation between online auctions and prophet inequality. In particular, they show that algorithms used in derivation of prophet inequality can be reinterpreted as truthful mechanisms for online auctions. In the prophet inequality setting, an onlooker is given an online sequence of independent random variables $X_1, X_2, \ldots X_n$, such that X_i is drawn from known probability distribution D_i. The onlooker has to choose at most one variable from the succession of the values. The onlooker can choose a value only at the time of arrival. The onlookers goal is to maximize her revenue. The onlooker's revenue is compared with the expected revenue of an optimal offline algorithm which known all the realized random variables in advance, like a prophet. Many online optimization problem have been studied under a prophet type of stochastic setting (see e.g. [105–108]), i.e. at each online step the demand, or the input is drawn from a specific known distribution.

[5]Indeed, the results in [72] and [75] applies to the more general class of weakly or skew supermodular cut functions.

In particular, the first generalization of the prophet inequality is the *multiple-choice prophet inequality* [119–121]. In the multiple-choice prophet inequality we are allowed to pick k values, and the goal is to maximize the total sum of picked values. Alaei [122] gives an almost tight $(1-\frac{1}{\sqrt{k+3}})$-approximation algorithm for the k-choice prophet inequality (the lower bound is proved in Hajiaghayi, Kleinberg, and Sandholm [118]). Prophet inequalities have been studied under complicated combinatorial structures such as matroid, polymatroid, and matching.

Kleinberg and Weinberg [123] consider matroid prophet inequalities, in which the set of selected values should be an independent set of a predefined matroid. They give a tight 0.5-approximation worst order algorithm for this problem. Later, Dütting and Kleinberg extended this result to polymatroids [124].

Alaei, Hajiaghayi and Liaghat study matching prophet inequalities [105–107]. They extend the multiple-choice prophet inequality and give an almost tight $(1-\frac{1}{\sqrt{k+3}})$-approximation worst order algorithm for any matching prophet inequality instance, where k is the minimum capacity of a vertex.

Rubinstein considers the prophet inequalities restricted to an arbitrary downward-closed set system [125]. He provides a $O(\log n \log r)$-approximation algorithm for this problem, where n is the number of distributions and r is the size of the larges feasible set. Babaioff, Immorlica and Kleinberg show a lower bound of $\Omega(\frac{\log n}{log^2(logn)})$ for this problem [126]. Prophet inequalities has also been studied restricted to independent set in graphs [127].

4.2 Steiner Tree Packing

In this section we study a variant of the Steiner tree packing problem which we call "*the fractional Steiner tree packing problem*" and show the conjecture of Kriesell holds for this variant. We use this tool in Section 4.3 to analyze the behavior of the greedy algorithm in survivable network design. An immediate corollary of this theorem is a simple logarithmic approximation algorithm for the Steiner tree packing problem.

4.2.1 Fractional Steiner Tree Packing

One way to formulate the Steiner tree packing problem is via an integer program. Let $\mathcal{T}_S(G)$ be the set of all Steiner trees of G on the vertices of S. By definition, the Steiner tree packing problem is the solution of the following integer program.

$$\begin{aligned}
\text{maximize:} \quad & \sum_{T \in \mathcal{T}_S(G)} x_T \\
\text{subject to:} \quad & \sum_{T \ni e} x_T \le 1 \qquad && \forall e \in E(G) \\
& x_T \in \{0,1\} \qquad && \forall T \in \mathcal{T}_S(G)
\end{aligned} \tag{4.1}$$

In Program 4.1, for every Steiner tree $T \in \mathcal{T}_S(G)$ we have a variable x_T that is either 0 or 1. The goal is to maximize the number of trees T with $x_T = 1$ while every edge appears in no more than one of such trees. One way to relax this problem is by lifting the constraint of $x_T \in \{0,1\}$ and instead assume $0 \le x_T \le 1$. This results in the following linear program:

$$\begin{aligned} \text{maximize:} \quad & \sum_{T \in \mathcal{T}_S(G)} x_T & & \qquad (4.2) \\ \text{subject to:} \quad & \sum_{T \ni e} x_T \le 1 & & \forall e \in E(G) \\ & 0 \le x_T \le 1 & & \forall T \in \mathcal{T}_S(G) \end{aligned}$$

We call the optimal solution of LP 4.2 the fractional Steiner tree packing problem. We show in Section 4.2.2 that for every graph G that is k-connected on a set S of vertices, the answer of the fractional Steiner tree packing problem on set S is greater than or equal to $k/2$.

4.2.2 Fractional Steiner Tree Packing of k-connected Graphs

In this section we show the conjecture of Kriesell for Steiner tree packing holds when we relax the problem to the fractional case. More precisely, we show any graph which is k-connected on a set S of vertices, has a fractional Steiner tree packing of at least $k/2$. We begin the proof by an auxiliary lemma.

Lemma 4.1. *Let $\mathcal{R}$ be a subspace of $\mathbb{R}^n$ that contains all points of $\mathbb{R}^n$ with non-negative coordinates and P be a convex set of points in $\mathcal{R}$. If for every point $\hat{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{R}$ there exists a point $\hat{p} \in P$ such that*

$$\hat{p}.\hat{x} \le k \sum_{i=1}^{n} x_i$$

then P contains a point $\hat{r}$ such that $\max_{i=1}^{n} r_i \le k$.

Proof. We define P' as the set of all points in $\mathbb{R}^n$ whose all indices are greater than or equal to the corresponding indices of a point in P. In other words

$$P' = \{\hat{p} \in \mathbb{R}^n | \exists \hat{q} \in P \text{ such that } p_i \geq q_i \text{ for all } 1 \leq i \leq n\}.$$

We show in the rest that $(k, k, \ldots, k) \in P'$ which immediately implies the lemma. To this end, suppose for the sake of contradiction that $(k, k, \ldots, k) \notin P'$. Note that, since P is a convex set, so is P'. Therefore, there exists a hyperplane that separates all points of P' from point $(k, k, \ldots, k)$. More precisely, there exist coefficients $h_0, h_1, \ldots, h_n$ such that

$$\sum_{i=1}^{n} h_i p_i > h_0 \tag{4.3}$$

for all points $\hat{p} \in P'$ and

$$\sum_{i=1}^{n} h_i k < h_0. \tag{4.4}$$

Due to the construction of P' we are guaranteed that all coefficients $h_1, h_2, \ldots, h_n$ are non-negative numbers since otherwise for any index i such that $h_i < 0$ there exists a point in P' whose i'th index is large enough to violate Inequality (4.3). Now let $\hat{x} = (h_1, h_2, \ldots, h_n)$. By Inequalities (4.3) and (4.4) we have

$$\hat{p}.\hat{x} = \sum_{i=1}^{n} x_i p_i = \sum_{i=1}^{n} h_i p_i > h_0 \geq k \sum_{i=1}^{n} h_i = k \sum_{i=1}^{n} x_i$$

for every $\hat{p} \in P'$ which means there is no $\hat{p} \in P$ such that $\hat{p}.\hat{x} \leq k \sum_{i=1}^{n} x_i$. This contradicts the assumption of the lemma. $\square$

Now based on Lemma 5.2, we give a lower bound on the fractional Steiner tree packing of any S-k-connected graph. Let G be a graph which is k connected on a set S, and R be the set of all randomized algorithms that randomly select a Steiner tree of G. In other words, every element of R is an algorithm that can be specified with a distribution of probabilities over the Steiner trees of G. We associate every element of R to a point in a $|E(G)|$ dimensional space in the following way:

$$f(A) = \langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{|E(G)|} \rangle$$

where A is an algorithm and for every edge $e \in E(G)$, $\hat{x_e}$ is the probability that e appears in a random tree of algorithm A. Now, let $\mathcal{R} = \bigcup_{A \in \mathsf{R}} f(A)$ be the set of all points associated to the elements of R. Convexity of R is a direct consequence of its definition; For every two algorithms $A, B \in \mathsf{R}$, one can construct a randomized procedure C that selects a random Steiner tree based on each of the procedures with probability $1/2$ and hence $f(C) = (f(A) + f(B))/2$. Thus, for every two points $\hat{x}, \hat{y} \in \mathcal{R}$, $(\hat{x} + \hat{y})/2$ is also in $\mathcal{R}$. Moreover, all indices of the points in $\mathcal{R}$ are non-negative. Next, we show the following important property of $\mathcal{R}$.

Lemma 4.2. *For any point $\hat{y} \in \mathbb{R}^{|E(G)|}$ with non-negative indices, there exists a point $\hat{x}$ in $\mathcal{R}$ such that*

$$\hat{x} \cdot \hat{y} \leq \sum 2\hat{y}_i/k.$$

Proof. To prove this lemma, we assume every edge e of G has a weight equal to $\hat{y}_e$. Moreover, let $W = \sum_{e \in E(G)} w_e$ be the total sum of the weights of the edges. We show the minimum Steiner tree of G on set S, has a weight of at most $2W/k$. This implies the lemma since the point associated to an algorithm that deterministically selects the minimum Steiner tree of G, trivially satisfies the condition of the lemma. Therefore, it only suffices to show that the minimum Steiner tree of G has a weight of at most $2w/k$. To this end we write the LP relaxation of the Steiner tree problem as follows:

$$\begin{aligned}
\text{minimize:} \quad & x_e w_e \\
\text{subject to:} \quad & \sum_{e \in \sigma} x_e \geq 1 \qquad \forall \text{ cut } \sigma \text{that separates the vertices of } S \\
& 0 \leq x_e \leq 1 \qquad \forall e \in E(G)
\end{aligned} \tag{4.5}$$

One feasible solution to LP 4.5 can be achieved by setting $x_e = 1/k$ for all edges of the graph. The reason such a solution is feasible is that every cut that separates two vertices of S has at least k edges and therefore the summation of x_e's for every separating cut is at least 1. Thus, the optimal solution of LP 4.5 is bounded by W/k. It has been shown that the integrality gap of the Steiner tree problem is less than 2 [1]. Therefore, the weight of the minimum Steiner tree of graph G is at most $2W/k$ which concludes the lemma.

□

According to Lemmas 5.2 and 4.2, and the fact that $\mathcal{R}$ is convex, there exists a point $\hat{x} \in \mathcal{R}$ such that $\hat{x}_e \leq 2/k$ for every edge e of the graph. Now, we construct a solution to LP 4.2 in the following way: Let $A^{\hat{x}}$ be a randomized algorithm which is associated to $\hat{x}$ ($f(A^{\hat{x}}) = \hat{x}$). For every $T \in \mathcal{T}_S(G)$, we set $x_T = k/2A^{\hat{x}}(T)$, where $A^{\hat{x}}(T)$ is the probability that $A^{\hat{x}}$ selects tree T. Since the probability that every edge appears in a tree of $A^{\hat{x}}$ is bounded by $2/k$, then all constrains of LP 4.2 are satisfied. Note that the objective function of LP 4.2 for solution x is equal to $k/2$ since we multiplied the probabilities by $k/2$.

Theorem 4.6. *The fractional Steiner tree packing problem for any graph which is k-connected on a set S of vertices is at least $k/2$.*

4.2.3 Steiner Forest Packing

In this section we generalize the Steiner tree packing problem and again, give a lower bound for the fractional variant of this problem. Let G be a graph and S be a sequence of vertex pairs $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$. We call a subgraph of G a Steiner forest for S, if it connects all of the pairs in S and is minimal (it contains no cycles). Now, the Steiner forest packing problem is defined as follows: Given a graph G and a sequence of vertex pairs S, what is the maximum number of edge-disjoint Steiner forests in G with respect to set S? It is trivial to show that the Steiner forest packing problem is a generalization of the Steiner tree packing problem and hence this problem is also NP-hard. Similar to Section 4.2.1, we formulate the Steiner forest problem as follows:

$$\begin{aligned} \text{maximize:} \quad & \sum_{F \in \mathcal{F}_S(G)} x_F & & \\ \text{subject to:} \quad & \sum_{F \ni e} x_F \leq 1 & & \forall e \in E(G) \\ & x_F \in \{0, 1\} & & \forall F \in \mathcal{F}_S(G) \end{aligned} \tag{4.6}$$

where $\mathcal{F}_S(G)$ stands for the set of all Steiner forests of G with respect to S. Again, we relax the integer constraints to obtain the following linear program:

$$\begin{aligned}
\text{maximize:} \quad & \sum_{F \in \mathcal{F}_S(G)} x_F \\
\text{subject to:} \quad & \sum_{F \ni e} x_F \le 1 & \forall e \in E(G) \\
& 0 \le x_F \le 1 & \forall F \in \mathcal{F}_S(G)
\end{aligned} \tag{4.7}$$

We call the optimal solution of LP 4.7 *"the fractional Steiner forest packing"* problem. A similar analysis to what we present in Section 4.2.2 yields to Theorem 4.2.

4.3 Uniform SNDP

In this section we consider the uniform-connectivity version of the online survivable Steiner network design problem. The assumption of this version is that all connectivity requirements are equal to a given number. For this problem we first give a very simple algorithm and then analyze it using the tools introduced in the previous section. The next section explains how we generalize our algorithm to make it work for inputs with non-uniform connectivity requirements.

In the online uniform-connectivity survivable Steiner forest problem we are given an offline graph $G = \langle V(G), E(G) \rangle$, an integer k, and an online stream of demands $S = (s_1, t_1), (s_2, t_2), \ldots$. Every time a demand (s_i, t_i) arrives we have to add some of the edges of G to our current solution H in order to make k edge-disjoint paths between s_i and t_i in H. The online uniform-connectivity survivable Steiner tree problem is a special case of the forest problem in which the second endpoints of all demands are fixed at some vertex *root*. The objective of the problems is to minimize the cost of the selected subgraph H according to a given cost function.

A simple approach to solve these problems is to choose edges based on the following greedy method: for every demand add a minimum-cost subset of edges that satisfies the k-connectivity between its endpoints. In this section we show that this algorithm is not competitive to the optimum offline solution. This is shown by Lemma 4.6 in which we give an instance graph and a series of demands for which the greedy algorithm gives a solution of cost $\Omega(n)$ times the cost of the optimum offline solution.

However, we show a modified version of the greedy algorithm can be a viable approach for these problems if we lose some factor on the connectivity requirement. This can be done by satisfying half of the required connectivity. In particular, for every demand we add a minimum-cost subset of the edges that makes the current solution $(k/2)$-connected between the endpoints of that demand. Let us call this algorithm *GA*. In this section we show the cost of the edges GA selects is poly-logarithmically competitive to that optimum offline solution which satisfies k-connectivity for every demand.

Theorem 4.7. *For the online survivable Steiner forest problem, the output of GA satisfies $(k/2)$-connectivity for every demand and its cost is $O(\log^2 n)$-competitive.*

Theorem 4.8. *For the online survivable Steiner tree problem, the output of GA satisfies $(k/2)$-connectivity for every demand and its cost is $O(\log n)$-competitive.*

As a direct consequence of adding edges according to GA, the $(k/2)$-

Algorithm 3 2-scaled Greedy

Input: A graph G, an integer k, and an online stream of demands $(s_1, t_1), (s_2, t_2), \ldots$.

Output: A set H of edges such that every given demand (s_i, t_i) is connected through k edge-disjoint paths in H.

Offline Process:

```
1: Initialize $H = \varnothing$.
```

Online Scheme; assuming a demand (s_i, t_i) is arrived:

```
1: $P_i$ = A minimum-cost subset of edges, such that $s_i$ is $k/2$-connected to $t_i$ in $H \cup P_i$.
2: Update $H = H \cup P_i$.
```

connectivity is guaranteed for every demand. To complete the proof of the theorems, we need to show that the cost of the solution produced by GA is upper bounded by a factor of $O(\log^2 n)$ for forests, and $O(\log n)$ for trees.

Let $c : E(G) \to \mathbb{R}^{\geq 0}$ be the cost function on the edges. With some abuse of notation, we also use $c(Y)$ for a subset of edges $Y \subseteq E(G)$ as the sum of the cost of the edges in Y. With this notation we can say at every step i GA chooses a subset of edges P_i that satisfies $(k/2)$-connectivity and minimizes $c(P_i)$.

The overall idea of the proofs is as follows. We take an optimum solution and charge every $c(P_i)$ to $c(L_i)$, where L_i is a set of edges chosen from the optimum solution. The way we define L_i's allows them to have overlapping edges, but we show that their total cost is limited by the desired poly-logarithmic factor of the cost of the optimum solution. More specifically, we charge $c(L_i)$ to the cost of a fractional routing Q_i between s_i and t_i. Every Q_i is itself a linear combination of routes on different Steiner forests of the optimum solution. The coefficients of this linear combination are achieved from an Steiner forest packing of the optimum solution. In this fashion, the problem boils down to finding an upper bound for the total cost of routings on each Steiner forest. In the following we formally prove every step in detail.

Let OPT be an optimum offline solution of the survivable Steiner forest problem on graph G, a stream of demands S, and the connectivity requirement k. Now we define L_i for every demand i as a minimum-cost set of edges in OPT that is $(k/2)$-connected between s_i and t_i assuming the endpoints of every previous demand are contracted. In particular, we call a set of edges a *pseudo-path* between s_i and t_i if there is a path between these vertices using those edges and the edges in $\{(s_j, t_j)|\forall j < i\}$. A *pseudo-routing* between s_i and t_i is hence a set of pseudo-paths between s_i and t_i. With these definitions, L_i is a minimum-cost pseudo-routing between s_i and t_i in OPT that consists of $k/2$ pseudo-paths. The following lemma shows the relation between the costs of L_i and P_i.

Lemma 4.3. *For every demand i, $c(P_i) \leq c(L_i)$.*

Proof. Every time a demand i arrives, GA finds a set P_i with the minimum cost and adds it to H in order to satisfy $(k/2)$-connectivity between s_i and t_i. Note that the endpoints of every demand $j < i$ are already connected with $k/2$ disjoint paths in H. Besides, L_i is a pseudo-routing between s_i and t_i which is $(k/2)$-connected between s_i

and t_i if we contract the two endpoints of every previous demand. Therefore adding L_i to H makes H $(k/2)$-connected between s_i and t_i. Since GA finds a minimum-cost set of edges that satisfies $(k/2)$-connectivity in H, $c(P_i)$ never exceeds $c(L_i)$.

$\square$

In the remaining we show how to charge the total cost of L_i's to $c(OPT)$. As a property of an optimum solution, OPT contains k edge-disjoint paths between the endpoints of every demand $(s_i, t_i) \in S$. Therefore, according to Theorem 4.2 there exists a solution for the fractional Steiner forest packing of OPT and demand set S with value at least $k/2$. Let $\mathbf{z}$ be a Steiner forest packing of OPT with value $k/2$. In the following we use $\mathcal{F}_S(OPT)$ to denote the collection of all Steiner forests of OPT with respect to demand set S. The theorem states there exists a vector $\mathbf{z}$ such that

$$\sum_{F \in \mathcal{F}_S(OPT)} z_F = k/2 \tag{4.8}$$

$$\sum_{F \in \mathcal{F}_S(OPT): e \in F} z_F \leq 1 \qquad \forall e \in OPT\ . \tag{4.9}$$

Moreover, the following inequality holds for the summation of the costs of these forests.

Lemma 4.4. $\sum_{F \in \mathcal{F}_S(OPT)} z_F . c(F) \leq c(OPT)\ .$

Proof. For each forest we replace its cost with the sum of the cost of its edges.

$$\begin{aligned}\sum_{F\in\mathcal{F}_S(OPT)} z_F.c(F) &= \sum_{F\in\mathcal{F}_S(OPT)} z_F \sum_{e\in F} c(e)\\ &= \sum_{e\in OPT}\ \sum_{F\in\mathcal{F}_S(OPT):e\in F} z_F c(e)\\ &= \sum_{e\in OPT} c(e)\left(\sum_{F\in\mathcal{F}_S(OPT):e\in F} z_F\right)\ .\end{aligned}$$

Now we use the fact that the load on every edge in the fractional Steiner forest packing is no more than 1.

$$\begin{aligned}\sum_{F\in\mathcal{F}_S(OPT)} z_F.c(F) &\leq \sum_{e\in OPT} c(e) \qquad\qquad \textit{Inequality}\quad (4.9)\\ &= c(OPT)\ .\end{aligned}$$

□

Now for every forest $F \in \mathcal{F}_S(OPT)$ and every demand i we define $Q_i(F)$ as a minimum-cost pseudo-path between s_i to t_i in F. This definition allows using an edge $e \in F$ multiple times in $Q_i(F)$ of different demands. Note that $Q_i(F)$ can be considered as a fractional pseudo-routing between s_i and t_i with value z_F. Considering this for all forests in $\mathcal{F}_S(OPT)$, we achieve a fractional pseudo-routing between s_i and t_i that has a value of $k/2$. We use Q_i to refer to this fractional pseudo-routing and $c(Q_i) = \sum_{F\in\mathcal{F}_S(OPT)} z_F.c(Q_i(F))$ to refer to its cost.

For every demand i we have mentioned two different pseudo-routings between s_i and t_i in OPT with value $k/2$: an integral pseudo-routing L_i, and a fractional pseudo-routing Q_i. The following lemma shows the relation between the costs of these two.

Lemma 4.5. *For every L_i and Q_i pseudo-paths defined as above, we have:*

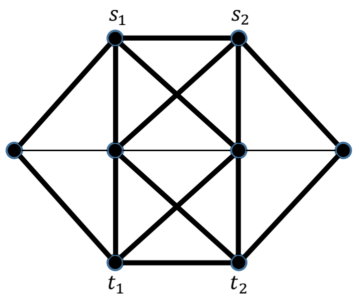
$$c(L_i) \leq c(Q_i)$$

Proof. Let $\mathcal{P}$ be the family of all pseudo-paths that connects s_i to t_i in OPT. Now Consider the following LP:

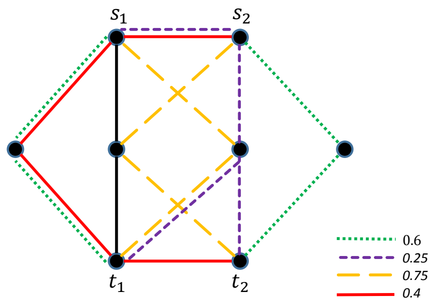
$$\begin{aligned}
\text{minimize:} \quad & \sum_{p \in \mathcal{P}} x_p c(p) & & \\
\text{subject to:} \quad & \sum_{p \in \mathcal{P} : e \in p} x_p \leq 1 & \quad & \forall e \in OPT \\
& \sum_{p \in \mathcal{P}} x_p = k/2 & & \\
& 0 \leq x_p \leq 1 & & \forall p \in \mathcal{P}
\end{aligned} \tag{4.10}$$

A feasible solution to this LP is in fact a pseudo-routing between s_i and t_i in OPT with value $k/2$. Since every $F \in \mathcal{F}_S(OPT)$ is a subset of OPT, the set of pseudo-paths between s_i and t_i in F is a subset of $\mathcal{P}$. As a result, every $Q_i(F)$ is also a member of $\mathcal{P}$ and thus Q_i corresponds to a feasible fractional solution to LP 4.10 with an objective function equal to $c(Q_i)$. Similarly, L_i is corresponding to a feasible integral solution to this LP. Due to the definition, L_i is an optimum integral solution of this LP, meaning that $c(L_i)$ is the minimum among the objective functions of all integral solutions. Note that this LP is essentially a minimum-cost flow which has an integrality gap of 1. Therefore, $c(L_i)$ equals an optimum solution of the LP, and thus does not exceed $c(Q_i)$. □

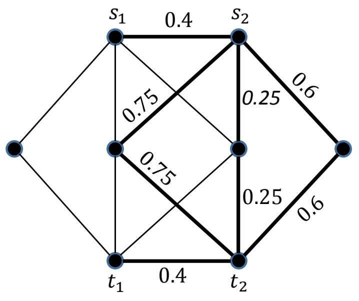
Finally for a particular $F \in \mathcal{F}_S(OPT)$ we show an upper bound for the sum of $c(Q_i(F))$ over all demands. First let us take a closer look at every $Q_i(F)$ on a



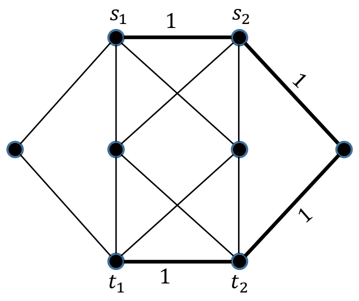
(a) Graph G with unit cost edges. Bold edges show an optimum offline solution for demand set $\{(s_1, t_1), (s_2, t_2)\}$ and $k = 4$.



(b) A Steiner forest packing of OPT in which four forests have non-zero values in vector $\mathbf{z}$.



(c) Bold edges show Q_2, the fractional pseudo-routing from s_2 to t_2 achieved from pseudo-paths on Steiner forests of OPT with respect to vector $\mathbf{z}$.



(d) Bold edges represent L_2, a minimum-cost integral pseudo-routing from s_2 to t_2 in OPT which consists of $k/2 = 2$ pseudo-paths.

Figure 4.1: An example to show the pseudo-routings Q_i and L_i in OPT which we use to bound $c(P_i)$.

particular F. Every time a new demand (s_i, t_i) arrives $Q_i(F)$ connects its endpoints through a pseudo-path in F. This can be generalized to an algorithm for the online single-connectivity Steiner forest problem that greedily connects the endpoints of every demand by fully buying a minimum-cost pseudo-path between s_i and t_i. This is very similar to the greedy algorithm proposed in [80]. Theorem 2.1 of that paper states that their greedy algorithm is $O(\log^2 n)$-competitive. The statement of that theorem is slightly different than Claim 4.1, but the same proof verifies the correctness of the claim.

Claim 4.1. *For the online Steiner forest problem, the algorithm that connects every demand with a minimum-cost pseudo-path is $O(\log^2 n)$-competitive.*

Now we are ready to wrap up the proof of Theorem 4.7.

Proof of Thorem 4.7: Let ALG denote the output of GA. The cost of ALG is the sum of the cost of P_i's over all demands. Therefore, by applying lemmas 4.3 and 4.5 we have

$$\begin{aligned} c(ALG) &= \sum_{(s_i,t_i)\in S} c(P_i) \\ &\leq \sum_{(s_i,t_i)\in S} c(L_i) && \textit{Lemma} \quad 4.3 \\ &\leq \sum_{(s_i,t_i)\in S} c(Q_i) && \textit{Lemma} \quad 4.5 \end{aligned}$$

Now we replace $c(Q_i)$ with the weighted sum of $c(Q_i(F))$'s with respect to $\mathbf{z}$.

$$\begin{aligned} c(ALG) &\leq \sum_{(s_i,t_i)\in S} \sum_{F\in\mathcal{F}_S(OPT)} z_F.c(Q_i(F)) \\ &= \sum_{F\in\mathcal{F}_S(OPT)} z_F \sum_{(s_i,t_i)\in S} c(Q_i(F)) \qquad (4.11) \end{aligned}$$

By applying Claim 4.1 to Inequality (4.11) we achieve an $O(\log^2 n)$-competitive ratio for GA.

$$\begin{aligned}
c(ALG) &\leq \sum_{F \in \mathcal{F}_S(OPT)} z_F \Big(O(\log^2 n) c(F) \Big) && \textit{Claim} \quad 4.1 \\
&\leq O(\log^2 n) \sum_{F \in \mathcal{F}_S(OPT)} z_F . c(F) \\
&\leq O(\log^2 n) c(OPT) \ . && \textit{Lemma} \quad 4.4
\end{aligned}$$

□

Finally, for the survivable Steiner tree problem we show that GA is $O(\log n)$-competitive. In other words, if one endpoint of every demand is fixed at the root, then the output of GA is at most $O(\log n)$ times the optimum offline solution. To complete the proof of Theorem 4.8 we use a result from [8]. In that paper the authors prove a competitive ratio of $O(\log n)$ for the algorithm which satisfies every demand using a minimum-cost pseudo-path. The following claim is a restatement of their result.

Claim 4.2. *For the online Steiner tree problme, the algorithm that satisfies each demand with a minimum-cost pseudo-path is $O(\log n)$-competitive.*

Proof of Theorem 4.8: Note that the tree problem is a special case of the forest problem, hence Inequality (4.11) also holds for it. By applying Claim 4.2 to that inequality the proof is complete.

$$\begin{aligned} c(ALG) &\leq \sum_{F \in \mathcal{F}_S(OPT)} z_F \Big(O(\log n) c(F) \Big) && \textit{Claim} \quad 4.2 \\ &\leq O(\log n) \sum_{F \in \mathcal{F}_S(OPT)} z_F . c(F) \\ &\leq O(\log n) c(OPT) \ . && \textit{Lemma} \quad 4.4 \end{aligned}$$

□

The following Lemma shows that there exists a graph G and a sequence of demands σ such that Greedy algorithm performs $\Omega(n)$ times worse than the optimal solution.

Lemma 4.6. *The competitive ratio of the greedy algorithm for survivable Steiner network design is $\Omega(n)$, even if every connectivity requirement is exactly 2.*

Proof. First we provide an online instance of the survivable network design problem where every connectivity requirement is exactly 2 and show the greedy algorithm performs poorly in comparison with the optimal solution. We construct a graph G of size n as follows. For each $1 \leq i \leq n-1$, there exist two undirected edges from node i to node $i+1$ of weights 1 and $n-i-\epsilon$ for some small $\epsilon > 0$. There exist two undirected edges from node n to node 1 with weights 1 and $n-\epsilon$. Thus G is the union of two cycles of size n. Figure (4.2) illustrates graph G. We construct a set of demands S as follows. For each $1 \leq i \leq n-1$, let $(i, i+1)$ be the i'th demand in S.

Now we analyze the output of the greedy algorithm for the input instance. We claim that after satisfying demand i the greedy algorithm has selected both edges

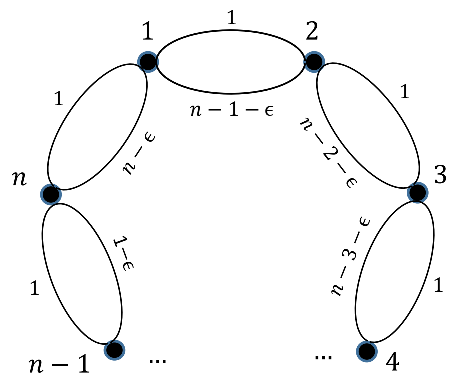


Figure 4.2: An example graph illustrating that the greedy algorithm has $\Omega(n)$-competitive ratio.

between j and $j+1$ for every $j \leq i$. We prove this claim by induction. For the base case, when the first demand arrives the greedy algorithm chooses both edges between nodes 1 and 2 which costs $n-\epsilon$. Now assume the greedy algorithm has selected every edge between j and $j+1$ for every $j < i$ before the arrival of the i'th demand. When the i'th demand arrives, the set of edges with minimum cost that provides two edge-disjoint paths from i to $i+1$ is the two edges between i and $i+1$ which costs $n-i-\epsilon$. Thus the total cost of the greedy algorithm at the end is $\frac{n(n-1)}{2} - \epsilon n$. However, the optimum offline solution chooses the cycle containing all edges of weight 1. Thus the competitive ratio of the greedy algorithm is $\Omega(n)$.

$\square$

4.4 Non-Uniform SNDP

In Section 4.3 we considered uniform online survivable network design, where all connectivity requirements are the same, or in other words, for every $\sigma_i = \langle s_i, t_i, r_i \rangle$, r_i equals some fixed integer k. Theorems 4.8 and 4.7 show a greedy algorithm which satisfies $k/2$ edge connectivity of the demands, is $O(\log n)$-competitive for online survivable Steiner tree, and $O(\log^2 n)$ competitive for online survivable Steiner forest, respectively. Now we consider the non-uniform case where connectivity requirements are arbitrary numbers between 1 and some value k. In particular, each demand $\sigma_i = \langle s_i, t_i, r_i \rangle$ indicates an r_i edge-connectivity requirement between s_i and t_i. Theorem 4.9 shows one can use algorithms provided in Section 4.4 to obtain an online competitive algorithm that satisfies $\frac{r_i}{2+\epsilon}$ connectivity of the demands.

Theorem 4.9. *Given an α-competitive online algorithm $\mathcal{A}$ for online survivable Steiner network design with equal connectivity demands, which partially satisfies every demand of k connectivity with $\frac{k}{2}$ edge-disjoint paths, there exists an $O(\frac{\alpha \log k}{\log(1+\epsilon)})$-competitive algorithm for online survivable Steiner network design that provides a solution that partially satisfies every demand of r_i connectivity with $\frac{r_i}{2+\epsilon}$ edge-disjoint paths.*

Proof. Let l be $\lceil \frac{\log k}{\log(1+\epsilon/2)} \rceil + 1$. Roughly speaking we define l subgraphs of G, $H_1, \ldots, H_l$, such that H_j is supposed to maintain a $\lceil (1+\epsilon/2)^{j-1} \rceil$-connected graph on the subset of demand pairs with connectivity requirement $(1+\epsilon/2)^{j-1} \leq r_i < (1+\epsilon/2)^j$.

We use l parallel and independent greedy algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_l$, such that H_j is the solution of $\mathcal{A}_j$ at every time. Let $\sigma_i = \langle s_i, t_i, r_i \rangle$ be the i-th demand. Assume $(1+\epsilon/2)^{j-1} \leq r_i < (1+\epsilon/2)^j$, then we use $\mathcal{A}_j$ for demand i. At arrival of σ_i, we define another request σ'_i as follows. Set $\sigma'_i = \langle s_i, t_i, r'_i \rangle$, where $r'_i = \lceil (1+\epsilon/2)^{j-1} \rceil$. Now we use $\mathcal{A}_j$ to satisfy demand σ'_i. Let the solution of the algorithm S be the union of the selected edges in $H_1, \ldots, H_l$.

First we prove S partially satisfies every demand of r_i connectivity with $\frac{r_i}{2+\epsilon}$ edge-disjoint paths. Since $\mathcal{A}_j$ provides $k/2$ edge-disjoint paths for a connectivity demand of k , if σ'_i is assigned to $\mathcal{A}_j$, $\mathcal{A}_j$ provides $\frac{\lceil (1+\epsilon/2)^{j-1} \rceil}{2}$ edge-disjoint paths between s_i and t_i. Moreover, since we assign σ'_i to $\mathcal{A}_j$ only if $(1+\epsilon/2)^{j-1} \leq r_i < (1+\epsilon/2)^j$, $\frac{r_i}{\lceil (1+\epsilon/2)^{j-1} \rceil} \leq (1+\epsilon/2)$. Thus $\frac{\lceil (1+\epsilon/2)^{j-1} \rceil}{2} \geq \frac{r_i}{2+\epsilon}$, hence there are at least $\frac{r_i}{2+\epsilon}$ edge-disjoint paths between s_i and t_i in H_j. Since $H_j \subseteq S$, there are at least $\frac{r_i}{2+\epsilon}$ edge-disjoint paths between s_i and t_i in S.

Let opt denote the cost of an optimal offline solution which maintains edge-connectivity of r_i for each demand i. Let $\mathrm{cost}(H)$ denote the total cost of edges in graph H, where H is a subset of G. We prove $\mathrm{cost}(S)$ is no more than $O(\frac{\alpha \log k}{\log(1+\epsilon/2)}) \cdot \mathsf{opt}$. Let opt_j denote the cost of an optimal solution for maintaining edge-connectivity of r_i for each demand i such that $(1+\epsilon/2)^{j-1} \leq r_i < (1+\epsilon/2)^j$. Since the set of such demands is a subset of all demands $\mathsf{opt}_j \leq \mathsf{opt}$. $\mathcal{A}_j$ is α-competitive to opt_j, thus

$$\mathrm{cost}(H_j) \leq \alpha \cdot \mathsf{opt}_j \leq \alpha \cdot \mathsf{opt} \ . \tag{4.12}$$

Since $S = \bigcup_{j=1}^{l} H_j$, $\text{cost}(S) = \sum_{j=1}^{l} \text{cost}(H_j)$. Thus by Equation (4.12),

$$\text{cost}(S) \leq \sum_{j=1}^{l} \alpha \cdot \mathsf{opt}_j \leq \sum_{j=1}^{l} \alpha \cdot \mathsf{opt} \ .$$

Since $l \leq O(\frac{\log k}{\log(1+\epsilon/2)})$,

$$\text{cost}(S) \leq O(\frac{\alpha \log k}{\log(1+\epsilon)}) \cdot \mathsf{opt} \ .$$

□

Using Theorems 4.8 and 4.7 for online survivable Steiner tree and forest in Section 4.3 and Theorem 4.9, we can immediately imply the following corollaries.

Corollary 4.5. *There exists an algorithm for the Survivable Steiner tree network design problem that: (i) provides a solution that partially satisfies every demand of r_i connectivity with $\frac{r_i}{2+\epsilon}$ edge-disjoint paths, and (ii) is $O(\frac{\log n \log k}{\log(1+\epsilon)})$-competitive to an optimal solution that maintains r_i-connectivity for every demand.*

Corollary 4.6. *There exists an algorithm for the Survivable Steiner forest network design problem that: (i) provides a solution that partially satisfies every demand of r_i connectivity with $\frac{r_i}{2+\epsilon}$ edge-disjoint paths, and (ii) is $O(\frac{\log^2 n \log k}{\log(1+\epsilon)})$-competitive to an optimal solution that maintains r_i-connectivity for every demand.*

4.5 From Oblivious I.I.D. to Prophet and Applications to Online Problems

In this section we show how one can use an oblivious competitive algorithm for an online optimization problem in the i.i.d. setting to obtain a competitive algorithm for that problem in the prophet setting. We first define and formulate a general set of online problems. Note that for simplicity, we only consider cost minimization problems, but one can similarly obtain the same statements for welfare maximization problems as well.

Let $\mathcal{P}$ be an online problem. Let Q be a set of queries or demands. Let $\mathcal{E}$ be a set of elements, such that each response of an algorithm to a demand $\sigma \in Q$ is a subset of $\mathcal{E}$. At each online step i, an online algorithm is given a demand $\sigma_i \in Q$. Then the algorithm needs to provide a response $R_i \subseteq \mathcal{E}$ that satisfies the given demand. Finally let $C : 2^{\mathcal{E}} \to \mathbb{R}$ denote a monotone and sub-additive cost function that maps each subset of $\mathcal{E}$ to a real number denoting the cost. The overall cost of an online algorithm that responds $R_1, \ldots, R_T$ to demands $\sigma_1, \ldots, \sigma_T$ is computed as $C(\bigcup_{i=1}^{T} R_i)$. To clarify the notations, consider the online SNDP. Given a graph $G = (V, E)$, Q is the set of triples (u, v, r) such that $u, v \in V$ and r is an integer such that there exist at least r edge-disjoint paths from u to v in G. Let $\mathcal{E}$ be the set of edges E. Now $R_i \subseteq E$ is a feasible response to a given demand $\sigma_i = (u_i, v_i, r_i)$, if adding R_i to the existing graph guarantees r_i edge-disjoint paths from u_i to v_i. The overall solution of an algorithm up to time T is $R = \bigcup_{i=1}^{T} R_i$. The cost function is defined as the total cost of edges in the solution, i.e. $C(R) := \sum_{e \in R} C(e)$, where $C(e)$ denotes the cost of a single edge e. Similarly one can formulate other fundamental online optimization problems such as online set-cover, online facility location, etc. in this way.

For an online problem in the i.i.d. setting we are given a probability distribution D over Q. At each time i, a random demand σ_i is drawn randomly and independently from distribution D. Let $\mathcal{P}_{iid}^{D}$ denote problem $\mathcal{P}$ in the i.i.d. setting given probability distribution D. For an online problem in the prophet setting we are given T probability distributions over Q, $D = \langle D_1, \ldots, D_T \rangle$. At each time i, a random demand σ_i is drawn randomly and independently from distribution D_i. Let $\mathcal{P}_{pht}^{D}$ denote problem $\mathcal{P}$ in the prophet setting given the sequence of probability distributions D. Now given $\mathcal{P}_{pht}^{D}$, we define a corresponding i.i.d. instance of the problem as follows. Define D^* to be the average of all distributions $D_1, \ldots, D_T$, i.e. $D^* = \sum_{i=1}^{T} \frac{D_i}{n}$. Let $\mathcal{A}$ be an oblivious α-competitive algorithm for $\mathcal{P}_{iid}^{D^*}$. In the following we show that $\mathcal{A}$ is $\frac{2e}{e-1}(1 + o(1))\alpha$-competitive for $\mathcal{P}_{iid}^{D^*}$.

First we need to define another problem W_{pht}^{D}, which is the same as $\mathcal{P}_{pht}^{D}$, but in the beginning, with probability $e^{-\frac{T(1-\frac{1}{e})}{8}}$ we do not provide any demand at all, and with the remaining probability we remove $\frac{1}{2}(1 - \frac{1}{e})$ fraction of the T distributions uniformly at random, i.e. we do not provide any demands at those times.

Consider a subset of demands $\sigma = \sigma_1, \ldots \sigma_{\frac{1}{2}(1-\frac{1}{e})}$ in W_{pht}^{D}. We prove that the probability that a super set of σ is an online scenario for $\mathcal{P}_{iid}^{D^*}$ is no less than the probability that σ is an online scenario for W_{pht}^{D}, or roughly speaking if one ignores the order of the online demands, W_{pht}^{D} is an easier problem than $\mathcal{P}_{iid}^{D^*}$.

Lemma 4.7. *For every subset of demands $\sigma = \sigma_1, \ldots \sigma_{\frac{1}{2}(1-\frac{1}{e})}$, the probability the online scenario for $\mathcal{P}_{iid}^{D^*}$ is a super set of σ is no less than the probability that the online scenario for W_{pht}^{D} is σ.*

Proof. Without loss of generality we can assume the demands in $D_1, \ldots, D_T$ are different. In other words each demand either arrive at a specific time, or never arrives. We can easily add dummy demands, if a demand can possibly arrive at two or more different time steps. For drawing a random online scenario in $\mathcal{P}_{iid}^{D^*}$, we define an equivalent random process as follows. At each time, first we draw a distribution from all T distributions uniformly at random, and then we draw a random demand from that distribution. Now for every subset S of distributions of size $\frac{1}{2}(1-\frac{1}{e})T$ we show that the probability that S is drawn in W_{pht}^{D} is less than or equal to the probability that a super set of S is drawn in $\mathcal{P}_{iid}^{D^*}$. Note that this shows that for every subset of demands $\sigma = \sigma_1, \dots \sigma_{\frac{1}{2}(1-\frac{1}{e})}$, the probability that a super set of σ is an online scenario for W_{pht}^{D} is less than or equal to the probability that σ is an online scenario for $\mathcal{P}_{iid}^{D^*}$. Since if the set of distributions is fixed, we can use the same random coin for drawing random demands from the same distributions in W_{pht}^{D} and $\mathcal{P}_{iid}^{D^*}$.

Using a Chernoff bound we show that with probability $e^{-\frac{T(1-\frac{1}{e})}{8}}$ there are at least $\frac{1}{2}(1-\frac{1}{e})T$ distinct distributions drawn by $\mathcal{P}_{iid}^{D^*}$. Let Y be the number of distinct distributions that are drawn. Let X_i be 1 if distribution i is drawn and 0 if not. We have

$$E[Y] = \sum_{i=1}^{T} E[X_i] = \sum_{i=1}^{T} P[X_i = 1] = T - \frac{(T-1)^T}{T^{T-1}} \geq T(1-\frac{1}{e}). \tag{4.13}$$

Since X_i's are negatively correlated we can use a Chernoff bound to bound the probability that we have less than $\frac{1}{2}T(1-\frac{1}{e})$ distinct distributions in $\mathcal{P}_{iid}^{D^*}$.

$$Pr[Y \leq \frac{1}{2}T(1-\frac{1}{e})] \leq e^{-\frac{T(1-\frac{1}{e})}{8}}. \tag{4.14}$$

Since there is a symmetry across distributions in the random process, the probability for every subset of distributions of size k to be drawn is the same. Thus for every subset of distributions S of size $\frac{1}{2}(1-\frac{1}{e})T$ the probability that S is drawn in W^D_{pht} is less than or equal to the probability that a super set of S is drawn in $\mathcal{P}^{D^*}_{iid}$. $\square$

Let $C^D_W(\mathcal{A})$ and $C^{D^*}_{iid}(\mathcal{A})$ denote the expected cost of algorithm $\mathcal{A}$ for online random scenarios drawn by W^D_{pht} and $\mathcal{P}^{D^*}_{iid}$, respectively. By Lemma 4.7, since $\mathcal{A}$ is oblivious and indifferent to the order of the input,

$$C^D_W(\mathcal{A}) \leq C^{D^*}_{iid}(\mathcal{A}). \tag{4.15}$$

Recall that in W^D_{pht} with probability $e^{-\frac{T(1-\frac{1}{e})}{8}}$ we do not provide any demand at all, and with probability $1-e^{-\frac{T(1-\frac{1}{e})}{8}}$ we remove $\frac{1}{2}(1-\frac{1}{e})$ fraction of the T distributions uniformly at random. Moreover $\mathcal{A}$ is oblivious and the cost function is monotone and subadditive. Thus

$$\frac{1-e^{-\frac{T(1-\frac{1}{e})}{8}}}{\frac{1}{2}(1-\frac{1}{e})} C^D_{pht}(\mathcal{A}) \leq C^D_W(\mathcal{A}), \tag{4.16}$$

where $C^D_{pht}(\mathcal{A})$ is the expected cost of algorithm $\mathcal{A}$ for an online random scenario drawn by $\mathcal{P}^D_{pht}$.

Now we are ready to prove that one can use $\mathcal{A}$ to obtain a competitive algorithm for $\mathcal{P}^D_{pht}$. Let opt^D_{pht} and $opt^{D^*}_{iid}$ denote the expected cost of an optimal offline solution for a random online scenario drawn in $\mathcal{P}^D_{pht}$ and $\mathcal{P}^{D^*}_{iid}$, respectively.

$$
\begin{aligned}
\frac{\frac{1-e^{-\frac{T(1-\frac{1}{e})}{8}}}{\frac{1}{2}(1-\frac{1}{e})} C^D_{pht}(\mathcal{A})}{opt^D_{pht}} &\le \frac{\frac{1-e^{-\frac{T(1-\frac{1}{e})}{8}}}{\frac{1}{2}(1-\frac{1}{e})} C^D_{pht}(\mathcal{A})}{opt^{D^*}_{iid}} && \text{Since } opt^{D^*}_{iid} \le opt^D_{pht} \\
&\le \frac{C^D_W(\mathcal{A})}{opt^{D^*}_{iid}} && \text{By Inequality (4.16)} \\
&\le \frac{C^{D^*}_{iid}(\mathcal{A})}{opt^{D^*}_{iid}} && \text{By Inequality (4.15)} \\
&\le \alpha. && \text{Since } \mathcal{A} \text{ is } \alpha\text{-competitive}
\end{aligned}
$$

Thus $\frac{C^D_{pht}(\mathcal{A})}{opt^D_{pht}} \le \frac{2e}{e-1}(1+o(1))\alpha$.

Theorem 4.10. *Given an oblivious α-competitive online algorithm for an online problem in the i.i.d. setting, there exists a $\frac{2e}{e-1}(1+o(1))\alpha$-competitive online algorithm for the problem in the prophet setting, where the competitive ratio approaches $\frac{2e}{e-1}\alpha$ exponentially fast as the number of online steps T grows.*

Interestingly, we can use Theorem 4.10 to obtain online competitive algorithms for other fundamental problems in the prophet setting. Using the oblivious i.i.d. algorithms in [103] for i.i.d. vertex cover and i.i.d. facility location, we may obtain $O(1)$-competitive online algorithms for prophet vertex cover and prophet facility location. Moreover using the oblivious i.i.d. algorithm for i.i.d. set cover in [115], we can obtain $O(\log n)$-competitive algorithm for the prophet set cover problem.

4.6 Stochastic Survivable Network Design

In this section we study the stochastic variant of survivable network design. Recall that in this model, the input consists of both offline and online data. The offline input is given in advance to the algorithm and specifies a graph $G = \langle V(G), E(G)\rangle$, a source node s, an integer k denoting the connectivity requirement, a distribution D of probabilities over the vertices of the graph, and an integer l denoting the number of demands. Next, an online stream of demands $t_1, t_2, \ldots, t_l$ arrive one by one, upon every arrival of which we are required to update our solution to make sure the newly arrived vertex is k-connected to the source node s. No prior information about the demands is given in advance, however, we're guaranteed that the demands are randomly and independently drawn from the given distribution D.

In this section we show that a slight variation of the greedy algorithm performs almost optimally in this setting. This improves upon the result of Garg *et al.* [103] which is a constant bound for the case where the connectivity is equal to 1. This is surprising since the proven bounds of the online algorithms for survivable network design are much worse than that of single the connectivity [7]. From a high-level perspective our method is similar to the greedy algorithm in Garg *et al.* [103], however, such a generalization requires a deep and innovative study of the k-connected graphs. In Section 4.6.2 we present a structural lemma which basically simplifies the analysis of our greedy algorithm.

Theorem 4.11. *There exists an oblivious* 4*-approximation algorithm for the stochastic survivable network design problem.*

Moreover, we generalize the algorithm to the setting in which the demands are drawn from independently from an unknown distribution D.

Theorem 4.12. *There exists an oblivious* $O(\log n)$*-approximation algorithm for the* *stochastic survivable network design problem with an unknown distribution.*

Note that achieving a constant factor approximation algorithm is not possible for an unknown distribution due to the work of Garg *et al.* [103]. In particular, they show there is an $\Omega(\frac{\log n}{\log \log n})$ lower bound for Steiner tree, therefore our greedy algorithm is almost tight.

The rest of this section is summarized in the following. In Section 4.6.1 we describe our algorithm and show a sufficient condition for obtaining the constant approximation factor. We also prove the approximation factor of the algorithm for unknown distribution. Finally, in Section 4.6.2 we provide a study of k-connected graphs and prove the structural lemma.

4.6.1 Algorithm

In this section we explain our algorithm and outline the analysis. The algorithm is as follows. Before any demand arrives, we simulate a stream $t_1^*, t_2^*, \ldots, t_l^*$ of demands by drawing l random vertices from probability distribution D. Next, we find a 2-approximation Steiner network of the graph that k-connects all the simulated demands to the source node via the algorithm of Jain [75]. Let H denote this network. Based on this solution, for every node v of the graph, we find a minimum cost k-flow from v to s that uses the edges of H for free and call that the *partial solution* of v. Now we're ready to run the algorithm on the actual demands. We start with an empty graph for our initial solution. Every time a demand t_i arrives, we update our solution by adding all of the edges of the partial solution of t_i to our current solution. Notice that our solution is *oblivious* in the sense that the k-flow of each demand is regardless of the queries that have arrived prior to that demand.

In the rest we show that the approximation factor of our algorithm is bounded by 4. Let for every list L of vertices, $\mathsf{sol}(L)$ be the set of all subsets of $E(G)$ that k-connect all vertices of L to s. Moreover, let for a subset of edges Q, $\text{cost}(Q)$ denote the total cost of the edges in Q. We define a pseudo-cost function for a list of vertices $L = \langle v_1, v_2, \ldots, v_{|L|} \rangle$ and another vertex u as follows:

$$\beta(L, u) = \max_{F_1 \in \mathsf{sol}(L)} \min_{F_2 \in \mathsf{sol}(L \cup \{u\})} \text{cost}(F_2 \setminus F_1)$$

In words, $\beta(L, u)$ is the smallest cost that we need to pay in order to k-connect u to the source in **any** solution that already k-connects all vertices of L to the source. Monotonicity of β follows from its definition; the more vertices we add to L, the less costly it will be to satisfy another node in any solution that satisfies L. In other words, by adding more vertices to L, the max in the formulation of β will be more constrained.

The main observation that enables us to prove a constant approximation factor for our algorithm is a bound on the psudocost of a pair (L, u). More precisely, in Section 4.6.2 we prove the following theorem.

Theorem 4.13 [to be proven in Section 4.6.2]. *For any set S of vertices we have*

$$\sum_{v \in S} \beta(S \setminus \{v\}, v) \leq 2c(\mathsf{opt}_S)$$

where opt_S is a minimum cost Steiner network that k-connects all vertices of S to the source node s.

Roughly speaking, the idea is to consider a minimum weight subgraph that k-connects all vertices of S to the source node. Next, we find a k-flow for each node in S that k-connects this node to other vertices of the set. We do this in a way that k-flows use only the edges of the Steiner network, and that every edge appears in at most two k-flows. This in turn implies that summation of the β functions for all nodes is bounded by two times $c(\mathsf{opt}_S)$. This is discussed in details in Section 4.6.2.

An immediate corollary of Theorem 4.13 is that if we randomly draw l demands $d_1, d_2, \ldots, d_l$ from D, $\beta(\{d_1, d_2, \ldots, d_{l-1}\}, d_l)$ is no more than $2/l$ times the minimum cost of k-connecting all vertices of $d_1, d_2, \ldots, d_l$ to the source.

Now, recall that before the stream of demands arrives, our algorithm randomly draws l demands $t_1^*, t_2^*, \ldots, t_l^*$ and finds a 2-approximation solution for these demands. We refer to this subgraph as H. Since we use a 2-approximation algorithm and all demands are drawn from D, the expcted cost of H is bounded by $2c(\mathsf{opt})$. Moreover, for every actual demand t_i, the total cost of the edges in the partial solution of t_i that are not in H is bounded by $\beta(\{t_1^*, t_2^*, \ldots, t_l^*\}, t_i)$ which is by monotonicity bounded by $\beta(\{t_1^*, t_2^*, \ldots, t_{i-1}^*, t_{i+1}^*, \ldots, t_l^*\}, t_i)$. Notice that all t_i^*'s and t_i's are independently drawn from D and thus the expected cost of such edges in the partial solution of a demand t_i is no more than $2c(\mathsf{opt})/l$ in expectation. Therefore, if we add the cost of all such edges for all of the demands to the cost of H, it yields an upperbound on the total cost of our algorithm as follows

$$c(T) + l\frac{2c(\mathsf{opt})}{l} \leq 4c(\mathsf{opt}) \ .$$

This proves a 4-approximation bound on the cost of the greedy algorithm.

Now suppose the distribution of demands is unknown. In this case, the greedy algorithm is simply k-connecting new demand t_i to the source by buying a minimum cost set of edges. Let T_i denote the set of these edges. We show that as we serve more demands the expected cost of T_i decreases for larger i. For every demand i let $L_i = \{t_1, \dots, t_i\}$ and $p(i) = 2^{\lfloor \log i \rfloor}$. We have

$$c(T_i) \leq \beta(L_{i-1}, t_i) \leq \beta(L_{p(i)-1}, t_i) \ .$$

Using Theorem 4.13 and the fact that all items are drawn from independently from the same distribution, in expectation we have

$$\beta(L_{p(i)-1}, t_i) \leq 2c(\mathsf{opt}_{L_{p(i)-1}})/p_i \ .$$

Now we use $c(\mathsf{opt})$ as an upper bound for $c(\mathsf{opt}_{L_{p(i)-1}})$. Therefore,

$$\sum_{i=1}^{l} c(T_i) \leq \sum_{i=1}^{l} \frac{2c(\mathsf{opt})}{2^{\lfloor \log i \rfloor}} \leq 2\log(l)c(\mathsf{opt}) \ .$$

The number of demands is at most n, and thus the greedy algorithm is $O(\log n)$-approximation for unknown distribution.

4.6.2 Structural Lemma for k-connected Graphs

In this section we provide a study of k-connected Steiner graphs. Roughly speaking, we state a lemma that shows the k-connectivity property suffices for the existence of concurrent k-flows for all the vertices such that the congestions on the edges are bounded by a factor of 2. We formally define k-flows in the remainder. As a result of this lemma, we can prove the following theorem, which has been used in Section 4.6.1.

Theorem 4.13. *For any set S of vertices we have*

$$\sum_{v \in S} \beta(S \setminus \{v\}, v) \leq 2c(\mathsf{opt}_S)$$

where opt_S is a minimum cost Steiner network that k-connects all vertices of S to the source s.

The theorem states that the overall cost of k-connecting every vertex $v \in S$ to the optimum solution that k-connects $S \setminus \{v\}$ is no more two times the cost of the optimum solution that k-connects all vertices in S. We prove this theorem via a structural lemma on unweighted k-connected Steiner graphs. To this end let us first define k-flows.

Definition 4.1. *Consider an S-k-connected graph G which is undirected and unweighted. A k-flow for a vertex $v \in S$ in G is the union of k edge-disjoint directed paths that k-connects v to $S \setminus \{v\}$.*

Let opt_S be a minimum cost solution that k-connects every vertex to the source node s. Since every vertex is k-connected to s it follows that every other pairs of vertices are k-connected too. Therefore opt_S is an S-k-connected graph. Now for every $v \in S$ let $F(v)$ be a k-flow in opt_S from v to $S \setminus \{v\}$. We note that $\beta(S \setminus \{v\}, v) \leq c(F(v))$ because one can k-connect v to $S \setminus \{v\}$ through $F(v)$. As a result we have

$$\sum_{v \in S} \beta(S \setminus \{v\}, v) \leq \sum_{v \in S} c(F(v)) \ .$$

The following structural lemma states that one can find such k-flows in an S-k-connected graph for every vertex in S in a way that every edge appears in at most two k-flows. Therefore, for such set of k-flows in opt_S we have

$$\sum_{v \in S} c(F(v)) \leq 2c(\mathsf{opt}_S)$$

which completes the proof of Theorem 4.13.

Lemma 4.8. *In every S-k-connected graph G there exists a set of k-flows $\{F(v)|v \in S\}$ such that every edge is used at most once in each direction.*

Proof. Without loss of generality we can assume that the graph is minimal, i.e. removing any edge decreases the connectivity of S to $k-1$. This minimality assumption implies that every edge participates in a separating cut of S with size k. As a result, every minimum size separating cut of S has exactly k edges. We prove the lemma by induction on the number of vertices in G. In particular, we find a minimum size separating cut on S and considering the following two cases:

- *Basis*: In every min-cut $C = [A, B]$ either A or B has one vertex. Recall that every edge e belongs to a min-cut of S. For such cut, we now that one of the sides has size one, therefore at least one endpoints of e belongs to S.

  Now we explain how to find the k-flows for every $v \in S$. Take a neighbor u of v. If $u \in S$ then we draw a direct flow from v to u. If $u \notin S$, then every neighbor of u belongs to S, because every edge (u, w) has to have at least one endpoint in S. Without loss of generality assume that the neighbors of u are ordered such that each of them has a unique *next*. Let w be the next vertex after v. We draw a flow from v to u and from u to w. Since the degree of v is at least k the total number of outflows from v to $S \setminus \{v\}$ is at least k. Moreover, in this manner every edge has at most one flow in each direction.

- *Inductive step*: There exists a min-cut $C = [A, B]$ of S such that both A and B have more than one vertex. In this case we proceed with the following two actions. We once contract all the vertices in B into a vertex v_B. Let $S_A = \{S \cap A\} \cup \{v_B\}$ be the set of Steiner vertices in the new graph. As a consequence, this contracted graph is S_A-k-connected and its size is smaller than G. Therefore, due to the induction we can find k-flows for every $v \in S_A$ to $S_A \setminus \{v\}$ such that every edge in A has a flow of at most one in each direction. Similarly, we can find flows in B by contracting A and then using the induction.

  In this way, for every $v \in S$ we get k flows that leave v, but may not end up to a vertex in $S \setminus \{v\}$. This is because there are some flows that go to v_B or v_A in the contracted graphs, and become incomplete after mapping them to the original graph. However, we show that the flows of the other side can be used in order to continued the incomplete flows to reach S.

  Consider the graph achieved from contracting B. Let $in_A(v_B)$ be the set of flows from $S_A \setminus \{v_B\}$ to v_B and $out_A(v_B)$ be the set of flows from v_B to $S_A \setminus \{v_B\}$. Note that $out_a(v_B)$ is of size k and $in_A(v_B)$ is of size at most k, because there are k edges in the cut. Likewise, we we define $in_B(v_A)$ and $out_B(v_A)$ for the graph achieved from contracting A

  Now consider a flow of $in_A(v_B)$. This flow ends up to and edge (x, y) of the cut, where $x \in A$ and $y \in B$. Since $out_B(v_A)$ is of size k, for every such y there exists a flow from y to S_B. Therefore, we can continue that flow in A

such that it reaches a vertex of S. By doing this for all such flows in $in_A(v_B)$ and $in_B(v_A)$ every $v \in S$ has a k-flow such that every edge of G is used at most once in each direction.

$\square$

Chapter 5: Online Stochastic k-Server

5.1 Introduction

The k-server problem is one of the most fundamental problems in online computation that has been extensively studied in the past decades. In the k-server problem we have k mobile servers on a metric space $\mathcal{M}$. We receive an online sequence of t requests where the i^{th} request is a point $r_i \in \mathcal{M}$. Upon the arrival of r_i, we need to move a server to r_i, at a cost equal to the distance from the current position of the server to r_i. The goal is to minimize the total cost of serving all requests.

Manasse, McGeoch, and Sleator [128] introduced the k-server problem as a natural generalization of several online problems, and a building block for other problems such as the metrical task systems. They considered the adversarial model, in which the online algorithm has no knowledge of the future requests. Following the proposition of Sleator and Tarjan [129], they evaluate the performance of an online algorithm using competitive analysis. In this model, an online algorithm ALG is compared to an *offline* optimum algorithm OPT which is aware of the entire input in advance. For a sequence of requests ρ, let $|\text{ALG}(\rho)|$ and $|\text{OPT}(\rho)|$ denote the total cost of ALG and OPT for serving ρ. An algorithm is *c-competitive* if for every

ρ, $|\mathrm{ALG}(\rho)| \leq c\,|\mathrm{OPT}(\rho)| + c_0$ where c_0 is independent of ρ.

Manasse *et al.* [128] showed a lower bound of k for the competitive ratio of any deterministic algorithm in any metric space with at least $k+1$ points. The celebrated *k-server conjecture* states that this bound is tight for general metrics. For several years the known upper bounds were all exponential in k, until a major breakthrough was achieved by Koutsoupias and Papadimitriou [130], who showed that the so-called *work function algorithm* is $(2k-1)$-competitive. Proving the tight competitive ratio has been the "holy grail" of the field in the past two decades. This challenge has led to the study of the problem in special spaces such as the uniform metric (also known as the paging problem), line, circle, and trees metrics (see [131, 132] and references therein). We also refer the reader to Section 5.1.3 for a short survey of randomized algorithms, particularly the recent result of Bansal, Buchbinder, Madry, and Naor [133] which achieves the competitive ratio of $O(\log^3 n \log^2 k)$ for discrete metrics that comprise n points.

The line metric (or Euclidean 1-dimensional metric space) is of particular interest for developing new ideas. Chrobak, Karloof, Payne, and Vishwnathan [131] were the first to settle the conjecture in the line by designing an elegant k-competitive algorithm. Chrobak and Larmore [132] generalized this approach to tree metrics. Later, Bartal and Koutsoupias [134] proved that the work function algorithm is also k-competitive in line. Focusing on the special case of $k = 2$ in line, Bartal *et al.* [135] show that, using randomized algorithms, one can break the barrier of lower bound k by giving a 1.98-competitive algorithm for the case where we only have two servers.

Despite the strong lower bounds for the k-server problem, there are heuristics algorithms that are *constant* competitive in practice. For example, for the paging problem- the special case of uniform metric- the least recently used (LRU) strategy is shown to be experimentally constant competitive (see Section 5.1.3). We also run our algorithm on real world data to measure its performance. In particular we use the distribution of car accidents obtained from road safety data. Our experiments illustrate our algorithm is performing even better in practice. The idea of comparing the performance of an online algorithm (with zero-knowledge of the future) to the request-aware offline optimum has led to crisp and clean solutions. However, that is not without its downsides. The results in the online model are often very pessimistic leading to theoretical guarantees that are hardly comparable to experimental results. Indeed, one way to tighten this gap is to use stochastic information about the input data as we describe in this study.

We should also point out that the competitive analysis is not the only possible or necessarily the most suitable approach for this problem. Since the distributions from which the input is generated are known, one can use dynamic programming (or enumeration of future events) to derive the optimal movement of servers. Unfortunately, finding such an optimal online solution using the distributions is an NP-hard problem [1], thus the dynamic programming or any other approach takes exponential time. This raises the question that how well one can perform in comparison to the best online solution. In the rest of this chapter we formally define the model and

[1]Reduction from k-median to Stochastic k-server: to find the k median of set S of vertices, one can construct an instance of stochastic k-server with $t = 1$ and $P_1(v) = 1/|S|$ for every $v \in S$. The best initialization of the servers gives the optimum solution to k-median of S.

address this question.

5.1.1 The Stochastic Model

In this study, we consider the *stochastic k-server problem* where the input is not chosen adversarially, but consists of draws from given probability distributions. This problem has applications such as equipment replacement in data centers. The current mega data centers contain hundreds of thousands of servers and switches with limited life-span. For example servers usually retire after at most three years. The only efficient way to scale up the maintenance in data centers is by automation, and robots are designed to handle maintenance tasks such as repairs or manual operations on servers. The replacement process can be modeled as requests that should be satisfied by robots, and robots can be modeled as servers. This problem also has applications in physical networks. As an example, suppose we model a shopping service (e.g. Google Express) as a k-server problem in which we receive an online sequence of shopping requests for different stores. We have k shopping cars (i.e., servers) that can serve the requests by traveling to the stores. It is quiet natural to assume that on a certain time of the week/day, the requests arrive from a distribution that can be discovered by analyzing the history. We formalize this stochastic information as follows.

For every $i \in [1 \cdots t]$, a discrete probability distribution P_i is given in advance from which request r_i will be drawn at time step i. The distributions are chosen by the adversary and are assumed to be independent but not necessarily identical.

This model is inspired by the well-studied model of *prophet inequalities* [2] [118, 136]. As mentioned before, the case of line metric has proven to be a very interesting restricted case for studying the k-server problem. In this study, we focus mainly on the class of line metric though our results carry over to circle metric and general metrics as well.

In the adversarial model, the competitive ratio seems to be the only well-defined notion for analyzing the performance of online algorithms. However, in the presence of stochastic information, one can derive a much better benchmark that allows us to make fine-grained distinctions between the online algorithms. We recall that in the offline setting, for a class of algorithms $\mathcal{C}$, the natural notion to measure the performance of an algorithm ALG $\in \mathcal{C}$ is the *approximation ratio* defined as the worse case ratio of $|\text{ALG}|$ to $|\text{OPT}(\mathcal{C})|$ where $\text{OPT}(\mathcal{C})$ is the optimal algorithm in the class. In this study, we also measure the performance of an online algorithm by its approximation ratio– compared to the *optimal online solution*. We note that given distributions $P_1, \ldots, P_t$, one can iteratively compute the optimal online solution by solving the following exponential-size dynamic program: for every $i \in [0 \cdots t]$ and every possible placement A of k servers (called a *configuration*) on the metric, let $\tau(i, A)$ denote the minimum expected cost of an online algorithm for serving the first i requests and then moving the servers to configuration A. Note that $\tau(i, A)$

[2] In the prophet inequality setting, given (not necessarily identical) distributions $P_1, \ldots, P_t$, an online sequence of values $x_1, \ldots, x_n$ where x_i is drawn from P_i, an onlooker has to choose one item from the succession of the values, where x_i is revealed at step i. The onlooker can choose a value only at the time of arrival. The goal is to maximize the chosen value.

can inductively be computed via the following recursive formula

$$\tau(i, A) = \min_{B} \tau(i-1, B) + \mathrm{E}_{r_i \sim P_i} \left[\text{min. distance from } B \text{ to } A \text{ subject to serving } r_i\right] ,$$

where $\tau(0, A)$ is initially zero for every A.

5.1.2 Our Results

Our first main result is designing a constant approximation algorithm in the line metric when the distributions for different time steps are not necessarily identical.

Theorem 5.1. *There exists a* 3*-approximation online algorithm for the stochastic* k*-server problem in the line metric. The running time is polynomial in* k *and the size of the support of the input distributions. The same guarantee holds for the circle metric.*

For the general metric, we present an algorithm with a logarithmic approximation guarantee.

Theorem 5.2. *There exists a* $O(\log n)$*-approximation online algorithm for the stochastic* k*-server problem in the general metric.*

We prove the theorems using two important structural results. The first key ingredient is a general reduction from class of online algorithms to a restricted class of *non-adaptive algorithms* while losing only a constant factor in the approximation ratio. Recall that a configuration is a placement of k-servers on the metric. We say an algorithm ALG is *non-adaptive* if it follows the following procedure: ALG pre-computes a sequence of configurations $A_0, A_1, \ldots, A_t$. We start by placing the k-servers on A_0. Upon the arrival of r_i, (i) we move the servers to configuration A_i; next (ii) we move the closest server s to r_i; and finally (iii) we return s to its original position in A_i. We first prove the following structural result.

Theorem 5.3. *For the stochastic k-server problem in the general metric, the optimal non-adaptive online algorithm is within* 3*-approximation of the optimal online algorithm.*

Using the aforementioned reduction, we focus on designing the optimal non-adaptive algorithm. We begin by formulating the problem as an integer program. The second ingredient is to use the relaxation of this program to formalize a natural *fractional variant* of the problem. In this variant, a configuration is a fractional assignment of *server mass* to the points of the metric such that the total mass is k. To serve a request at point r_i, we need to move some of the mass to have at least one amount of server mass on r_i. The cost of moving the server mass is naturally defined as the integral of the movement of infinitesimal pieces of the server mass. By solving the linear relaxation of the integer program, we achieve the optimal fractional non-adaptive algorithm. We finally prove Theorems 5.1 and 5.2 by leveraging the following rounding techniques. The rounding method in line has been also observed by Türkoglu [137]. We provide the proof for the case of line in Section 5.5 for the sake of completeness. The rounding method for general metrics is via the well-known embedding of a metric into a distribution of well-separated trees while losing a logarithmic factor in the distortion. Bansal *et al.* [133] use a natural rounding method similar to that of Blum, Burch, and Kalai [138] to show that any fractional k-server movement on well-separated trees can be rounded to an integral counterpart by losing only a constant factor.

Theorem 5.4 (first proven in [137]). *Let* ALG_f *denote a fractional* k*-server algorithm in the line, or circle. One can use* ALG_f *to derive a randomized integral algorithm* ALG *such that for every request sequence* σ, $\mathrm{E}\left[|\mathrm{ALG}(\sigma)|\right] = |\mathrm{ALG}_f(\sigma)|$*. The expectation is over the internal randomness of* ALG*. Furthermore, in the stochastic model* ALG *can be derandomized.*

Theorem 5.5 (proven in [133]). *Let* ALG_f *denote a fractional* k*-server algorithm in any metric. One can use* ALG_f *to derive a randomized integral algorithm* ALG *such that for every request sequence* σ, $\mathrm{E}\left[|\mathrm{ALG}(\sigma)|\right] \leq O(\log n) \left|\mathrm{ALG}_f(\sigma)\right|$.

We further show that in the stochastic setting, if the number of possible input scenarios is m, even if the distributions are correlated, one can compute the best *fractional* online competitive algorithm in time polynomial in m and n. Note that since the number of placements of k servers on n points is exponential, it is not possible to enumerate all the possible choices of an online algorithm. We solve this problem by presenting a non-trivial LP relaxation of the problem with size polynomial in n and m; therefore obtaining the following result. We present the formal model and analysis in Appendix 5.7.

Theorem 5.6. *The optimal online algorithm of the stochastic* k*-server problem with correlated setting in line and circle can be computed in polynomial time w.r.t.*

the number of possible scenarios. In general metrics, an $O(\log n)$-approximation algorithm can be obtained.

5.1.3 Further Related Work

The randomized algorithms often perform much better in the online paradigm. For the k-server problem, a lower bound of $\Omega(\log k)$ is known for the competitive ratio of randomized algorithms in most common metrics. Despite the exponential gap, compared to the lower bound of deterministic algorithms, very little is known about the competitiveness of randomized algorithms. In fact, the only known algorithms with competitive ratios below k, work either in the uniform metric (also known as the paging problem [139–142]), a metric comprising $k + 1$ points [143], and two servers on the line [135]. Two decades after the introduction of the k-server problem, a major breakthrough was achieved by Bansal *et al.* [133] in discrete metrics with sub-exponential size. If $\mathcal{M}$ comprise n points, their randomized algorithm achieves a competitive ratio of $O(\log^3 n \log^2 k)$.

The case of uniform metric has been extensively studied under various stochastic models motivated by the applications in computer caching. Koutsoupias and Papadimitriou [130] consider two refinements of the competitive analysis for server problems. First, they consider the *diffuse adversary* model. In this model, at every step i the adversary chooses a distribution D_i over the uniform metric of the paging problem. Then the i^{th} request is drawn from D_i which needs to be served. The distribution D_i is not known to the online algorithm and it may depend on the previous requests. However, in their paper, they consider the case wherein it is guaranteed that for every point p, $D_i(p) \leq \epsilon$ for a small enough ϵ; i.e., the next request is not predictable with absolute certainty for the adversary. The results of Koutsoupias and Papadimitriou and later Young [144] shows that the optimum competitive ratio in this setting is close to $1 + \Theta(k\epsilon)$.

The second refinement introduced in [130] restricts the optimal solution to having lookahead at most ℓ. Hence, one can define a *comparative ratio* which indicates the worst-case ratio of the cost of the best online solution to the best solution with lookahead ℓ. They show that for the k-server problem, and more generally the metrical task system problem, there are online algorithms that admit a comparative ratio of $2\ell + 1$; for some instances this ratio is tight.

Later, Panagiotou and Souza [145] considered the paging problem when the adversary is restricted to certain local constraints on the request. These constraints are motivated by the locality of reference in a memory cache and typical memory-access patterns. They show that under these constraints the LRU algorithm is constant competitive, which indeed gives a theoretical explanation to why LRU works pretty well in practice.

Various other models of restricting the adversary (access graph model [146–148], fault rate model [149–151], etc) have also been considered for the paging problem (see [145, 152] and references therein for a further survey of these results). Unfortunately, many of the stochastic settings considered for the paging problem do not seem to have a natural generalization beyond the uniform metric setting. For example, in the diffuse adversary model, most of the studied distributions do not weaken the adversary in the general metric. In this study, we look for polynomial-time *approximation* algorithms in the class of online algorithms that have access to the distributions.

We would like to mention that various online problems have been previously considered under prophet inequality model or i.i.d. model (where all distributions are identical). Motivated by ad auctions, the maximum matching problem has been extensively studied in these models, achieving near one competitive ratios [105–107]. In the graph connectivity problems, Garg, Gupta, Leonardi,and Sankowski [153] consider the online variants of Steiner tree and several related problems under the i.i.d. stochastic model. In the adversarial model, there exists an $\Omega(\log n)$ lower bound on the competitive ratio of any online algorithm, where n is the number of demands. However, Garg *et al.* show that under the i.i.d. assumption, these problems admit online algorithms with constant or $O(\log \log n)$ competitive ratios. We refer the reader to the excellent book by Borodin and El-Yaniv [154] for further study of online problems.

5.2 Preliminaries

In this section we formally define the stochastic k-server problem. The classical k-server problem is defined on a metric $\mathcal{M}$ which is consisted of points that could be infinitely many. For every two points x and y in metric $\mathcal{M}$, let $d(x, y)$ denote the distance of x from y which is a symmetric function and satisfies the triangle inequality. More precisely for every three points x, y, and z we have

$$d(x, x) = 0 \tag{5.1}$$

$$d(x, y) = d(y, x) \tag{5.2}$$

$$d(x, y) + d(y, z) \geq d(x, z). \tag{5.3}$$

In the k-server problem the goal is to place k servers on k points of the metric, and move these servers to satisfy the requests. We refer to every placement of the servers on the metric points by a *configuration*. Let $\rho = \langle r_1, r_2, \ldots, r_t \rangle$ be a sequence of requests, the goal of the k-server problem is to find configurations $\langle A_0, A_2, \ldots, A_t \rangle$ such that for every i there exists a server on point r_i in configuration A_i. We say such a list of configurations is *valid* for the given list of requests. A valid sequence of configurations is optimal if $\sum d(A_{i-1}, A_i)$ is minimized where $d(X, Y)$ stands for the minimum cost of moving servers from configuration X to configuration Y. An optimal sequence $\langle A_0, A_2, \ldots, A_t \rangle$ of configurations is called an optimal offline solution of $\text{OFKS}(\mathcal{M}, \rho)$ when ρ is known in advance. We refer to the optimal cost of such movements with $|\,\text{OFKS}(\mathcal{M}, \rho)| = \sum d(A_{i-1}, A_i)$.

We also define the notion of *fractional configuration* as an assignment of the metric points to non-negative real numbers. More precisely, each number specifies a mass of fractional server on a point. Every fractional solution adheres to the following condition: The total sum of the values assigned to all points is exactly equal to k. Analogously, a fractional configuration serves a request r_i if there is a mass of size at least 1 of server assigned to point r_i. An offline fractional solution of the k-server problem for a given sequence of requests ρ is defined as a sequence of fractional configurations $\langle A_0, A_1, \ldots, A_t \rangle$ such that A_i serves r_i.

In the online k-server problem, however, we're not given the whole sequence of requests in the beginning, but we will be informed of every request once it is realized. An algorithm $\mathcal{A}$ is an online algorithm for the k-server problem if it reports a configuration A_0 as an initial configuration and upon realization of every request r_i it returns a configuration A_i such that $\langle A_0, A_1, A_2, \ldots, A_i \rangle$ is valid for $\langle r_1, r_2, \ldots, r_i \rangle$. If $\mathcal{A}$ is deterministic, it generates a unique sequence of configurations for every sequence of requests. Let $\mathcal{A}(\mathcal{M}, \rho)$ be the sequence that $\mathcal{A}$ generates for requests in ρ and $|\mathcal{A}(\mathcal{M}, \rho)|$ denote its cost.

In the online stochastic k-server problem, in addition to metric $\mathcal{M}$, we are also given t independent probability distributions $\langle P_1, P_2, \ldots, P_t \rangle$ which show the probability that every request r_i is realized on a point of the metric at each time. An algorithm $\mathcal{A}$ is an online algorithm for such a setting, if it generates a configuration for every request r_i not solely based on $\langle r_1, r_2, \ldots, r_i \rangle$ and $\langle A_0, A_1, \ldots, A_{i-1} \rangle$ but also with respect to the probability distributions. Similarly, we define the cost of an online algorithm $\mathcal{A}$ for a given sequence of requests ρ with $|\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t \rangle)|$. We define the expected cost of an algorithm $\mathcal{A}$ on metric $\mathcal{M}$ and with probability distributions $\langle P_1, P_2, \ldots, P_t \rangle$ by

$$|\mathcal{A}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t \rangle)| = \mathbb{E}_{\forall i, r_i \sim P_i} |\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t \rangle)|.$$

For every metric $\mathcal{M}$ and probability distributions $\langle P_1, P_2, \ldots, P_t \rangle$ we refer to the online algorithm with the minimum expected cost by $\mathrm{OPT}_{\mathcal{M}, \langle P_1, P_2, \ldots, P_t \rangle}$.

An alternative way to represent a solution of the k-server problem is as a vector of configurations $\langle B_0, B_1, \ldots, B_t \rangle$ such that B_i does not necessarily serve request r_i. The cost of such solution is equal to $\sum d(B_{i-1}, B_i) + \sum 2d(B_i, r_i)$ where $d(B_i, r_i)$ is the minimum distance of a server in configuration B_i to request r_i. The additional cost of $2d(B_i, r_i)$ can be thought of as moving a server from B_i to serve r_i and returning it back to its original position. Thus, every such representation of a solution can be transformed to the other representation. Similarly, $d(B_i, r_i)$ for a fractional configuration B_i is the minimum cost which is incurred by placing a mass 1 of server at point r_i. We use letter B for the configurations of such solutions throughout this chapter.

In this study the emphasis is on the stochastic k-server problem on the line metric. We define the line metric $\mathcal{L}$ as a metric of points from $-\infty$ to $+\infty$ such that the distance of two points x and y is always equal to $|x - y|$. Moreover, we show that deterministic algorithms are as powerful as randomized algorithms in this setting, therefore we only focus on deterministic algorithms. Thus, from here on, we omit the term deterministic and every time we use the word algorithm we mean a deterministic algorithm unless otherwise is explicitly mentioned.

5.3 Structural Characterization

In this section we define a class of online algorithms for the stochastic k-server problem and show an important structural property for this class. Later, we leverage this property to provide a polynomial time algorithm for the problem. Although we give the algorithm for the line metric, the structural characterization holds for general graphs and is of independent interest.

Recall that an online algorithm $\mathcal{A}$ has to fulfill the task of reporting a configuration A_i upon arrival of request r_i based on $\langle A_0, A_1, \ldots, A_{i-1}\rangle$, $\langle r_1, r_2, \ldots, r_i\rangle$, and $\langle P_1, P_2, \ldots, P_t\rangle$. We say an algorithm $\mathcal{B}$ is *request oblivious*, if it reports configuration B_i regardless of request r_i. As such, $\mathcal{B}$ generates configurations $\langle B_0, B_1, \ldots, B_t\rangle$ for a sequence of requests $\langle r_1, r_2, \ldots, r_t\rangle$ and the cost of such configuration is $\sum d(B_{i-1}, B_i) + \sum 2d(B_i, r_i)$. More precisely, no matter what request r_i is, $\mathcal{B}$ will generate the same configuration for a given list of past configurations $\langle B_0, B_1, \ldots, B_{i-1}\rangle$, a given sequence of past requests $\langle r_1, r_2, \ldots, r_{i-1}\rangle$, and the sequence of probability distributions $\langle P_1, P_2, \ldots, P_t\rangle$. In the following we show that every online algorithm $\mathcal{A}$ can turn into a request oblivious algorithm $\mathcal{B}_{\mathcal{A}}$ that has a cost of at most $|3\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t\rangle)|$ for a given sequence of requests ρ.

Lemma 5.1. *Let $\mathcal{A}$ be an online algorithm for the stochastic k-server problem. For any metric $\mathcal{M}$, there exists a request oblivious algorithm $\mathcal{B}_{\mathcal{A}}$ such that*

$$|\mathcal{B}_{\mathcal{A}}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)| \leq 3|\mathcal{A}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)|.$$

Proof. Let ρ be a sequence of requests. We define online algorithm $\mathcal{B}_{\mathcal{A}}$ as follows: The configuration that $\mathcal{B}_{\mathcal{A}}$ reports for a given list of input arguments $\langle B_0, B_1, \ldots, B_i\rangle$, $\langle r_1, r_2, \ldots, r_i\rangle$, and $\langle P_1, P_2, \ldots, P_t\rangle$ is the output of algorithm $\mathcal{A}$ on inputs $\langle B_0, B_1, \ldots, B_i\rangle$, $\langle r_1, r_2, \ldots, r_{i-1}\rangle$, and $\langle P_1, P_2, \ldots, P_t\rangle$ (The same input except that r_i is dropped from the sequence of requests). We show the cost of such algorithm for input ρ is at most 3 times the cost of $\mathcal{A}$ for the same input.

Let $\langle A_0, A_1, \ldots, A_t\rangle$ be the sequence of configurations that $\mathcal{A}$ generates for requests ρ and $\langle B_0, B_1, \ldots, B_t\rangle$ be the output of algorithm $\mathcal{B}_A$. According to the construction of $\mathcal{B}_\mathcal{A}$, $B_0 = A_0$ and $B_i = A_{i-1}$ for all $1 \leq i \leq t$. Note that for algorithm $\mathcal{A}$, we assume every A_i serves request r_i. By definition, the cost of solution $\langle B_0, B_1, B_2, \ldots, B_t\rangle$ is equal to $\sum d(B_{i-1}, B_i) + 2\sum d(B_i, r_i)$. Since $B_0 = B_1 = A_0$ and $B_i = A_{i-1}$,

$$\sum_{i=1}^{t} d(B_{i-1}, B_i) = \sum_{i=1}^{t-1} d(A_{i-1}, A_i) \leq \sum_{i=1}^{t} d(A_{i-1}, A_i) = |\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t\rangle)|. \tag{5.4}$$

Moreover, since every A_i servers request r_i, $d(B_i, r_i) \leq d(B_i, A_i) = d(A_{i-1}, A_i)$. Hence,

$$2\sum_{i=1}^{t} d(B_i, r_i) \leq 2\sum_{i=1}^{t} d(B_i, A_i) = 2\sum_{i=1}^{t} d(A_{i-1}, A_i) = 2|\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t\rangle)|. \tag{5.5}$$

Inequality (5.4) along with Equation (5.5) implies

$$|\mathcal{B}_\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t\rangle)| \leq 3|\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t\rangle)|.$$

Since this holds for all requests $\rho \sim \langle P_1, P_2, \ldots, P_t\rangle$, we have

$$|\mathcal{B}_\mathcal{A}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)| \leq 3|\mathcal{A}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)|$$

and the proof is complete. $\square$

An immediate corollary of Lemma 5.1 is that the optimal request oblivious algorithm has a cost of at most $|3\,\mathrm{OPT}_{\mathcal{M},\langle P_1,P_2,\ldots,P_t\rangle}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)|$. Therefore, if we only focus on the request oblivious algorithms, we only lose a factor of 3 in comparison to the optimal online algorithm. The following lemma states a key structural lemma for an optimal request oblivious algorithm.

Lemma 5.2. *For every request oblivious algorithm $\mathcal{B}$, there exists a randomized request oblivious algorithm $\mathcal{B}'$ with the same expected cost which is not only oblivious to the last request, but also oblivious to all requests that have come prior to this.*

Proof. For any given request oblivious online algorithm $\mathcal{B}$, we construct an online algorithm $\mathcal{B}'$ which is oblivious to all of the requests as follows: For an input $\langle B_1, B_2, \ldots, B_{i-1}\rangle$ of configurations and probability distributions $\langle P_1, P_2, \ldots, P_t\rangle$, draw a sequence of requests $\langle r_1, r_2, \ldots, r_i\rangle$ from $\langle P_1, P_2, \ldots, P_t\rangle$ conditioned on the constraint that $\mathcal{B}$ would generate configurations $\langle B_1, B_2, \ldots, B_{i-1}\rangle$ for requests $\langle r_1, r_2, \ldots, r_{i-1}\rangle$. Now, report the output of $\mathcal{B}$ for inputs $\langle B_1, B_2, \ldots, B_{i-1}\rangle$, $\langle r_1, r_2, \ldots, r_i\rangle$, and $\langle P_1, P_2, \ldots, P_t\rangle$.

We define the cost of step i of Algorithm B' as $d(B_{i-1}, B_i) + 2d(B_i, r_i)$. Due to the construction of algorithm $\mathcal{B}'$, the expected cost of this algorithm at every step i for a random sequence of requests is equal to the expected cost of algorithm $\mathcal{B}$ for a random sequence of requests drawn from $\langle P_1, P_2, \ldots, P_t\rangle$. Therefore, the expected cost of both algorithms for a random sequence of requests are equal and thus $|\mathcal{B}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)| = |\mathcal{B}'(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)|$. □

Lemma 5.2 states that there always exists an optimal randomized request oblivious online algorithm that returns the configurations regardless of the requests. We call such an algorithm *non-adaptive*. Since a non-adaptive algorithm is indifferent to the sequence of the requests, we can assume it always generates a sequence of configurations just based on the distributions. For an optimal of such algorithms, all such sequence of configurations should be optimal as well. Therefore, there always exists an optimal non-adaptive online algorithm which is deterministic. By Lemma 5.1 not only do we know the optimal request oblivious algorithm is at most 3-approximation, but also the same holds for the optimal non-adaptive algorithm.

Theorem 5.7. *There exists a sequence of configurations $\langle B_0, B_1, \ldots, B_t \rangle$ such that an online algorithm which starts with B_0 and always returns configuration B_i upon arrival of request r_i has an approximation factor of at most 3.*

5.4 Fractional Solutions

In this section we provide a fractional online algorithm for the k-server problem that can be implemented in polynomial time. Note that by Theorem 5.7 we know that there exist configurations $\langle \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_t \rangle$ such that the expected cost of a non-adaptive algorithm that always returns these configurations is at most 3. Therefore, we write an integer program to find such configurations with the least expected cost. Next, we provide a relaxed LP of the integer program and show that every feasible solution of such LP corresponds to a fractional online algorithm for the stochastic k-server problem. Hence, solving such a linear program, that can be done in polynomial time, gives us a fractional online algorithm for the problem.

5.4.1 Linear Program

Recall that given t independent distributions $\langle P_1, \ldots, P_t \rangle$ for online stochastic k-server, an adaptive algorithm can be represented by $t + 1$ configurations $\langle B_0, \ldots, B_t \rangle$. Upon the arrival of each request r_i, we move the servers from configuration B_{i-1} to B_i and then one server serves r_i and goes back to its position in B_i. The objective is to find the configurations such that the cost of moving to new configurations in addition to the expected cost of serving the requests is minimized. Therefore the problem can formulated in an offline manner. First we provide an integer program in order to find a vector of configurations with the least cost.

The decision variables of the program represent the configurations, the movement of servers from one configuration to another, and the way that each possible request is served. In particular, at each time step τ:

- For each node v there is a variable $b_{\tau,v} \in N$ denoting the number of servers on node v.
- For each pair of nodes u and v, there is a movement variable $f_{\tau,u,v} \in N$ denoting the number of servers going from u to v for the next round.
- For each node v and possible request node r, there is a variable $x_{\tau,v,r} \in \{0,1\}$ denoting whether r is served by v or not.

In the following integer program, the first set of constraints ensures the number of servers on nodes at each time is updated correctly according to the movement variables. The second set of constraints ensures that each possible request is served by at least one server. The third set of constraints ensures that no possible request is served by an empty node. By the definition, the cost of a sequence of configurations $\langle B_0, \ldots, B_t \rangle$ is $\sum_{i=1}^{t} d(B_{i-1}, B_i) + 2\sum_{i=1}^{t} d(B_i, r_i)$. Thus the objective is to minimize the expression

$$\sum_{\tau}\sum_{u,v} f_{\tau,u,v} d(u,v) + 2\sum_{\tau}\sum_{v}\sum_{r} x_{\tau,v,r}\Pr(z \sim P_\tau = r) d(v,r)$$

, where $\Pr(z \sim P_\tau = r)$ denotes the probability that r is requested at time τ.

$$\begin{array}{ll}
\text{min.} & \sum_{\tau}\sum_{u,v} f_{\tau,u,v} d(u,v) + 2\sum_{\tau}\sum_{v}\sum_{r} x_{\tau,v,r}\Pr(z \sim P_\tau = r) d(v,r) \\
\forall \tau, v & b_{\tau+1,v} = b_{\tau,v} + \sum_{u} f_{\tau,u,v} - \sum_{u} f_{\tau,v,u}. \\
\forall \tau, u, v & \sum_{v} x_{\tau,v,r} \geq 1. \\
\forall \tau, v, r & x_{\tau,v,r} \leq b_{\tau,v}. \\
\forall \tau & \sum_{v} b_{\tau,v} \leq k. \\
\forall \tau, v, r & x_{\tau,v,r} \in \{0,1\}. \\
\forall \tau, u, v & f_{\tau,u,v} \in N. \\
\forall \tau, v & b_{\tau,v} \in N.
\end{array}$$

Now we consider the following relaxation of the above integer program.

$$\begin{array}{ll}
\text{min.} & \sum_{\tau}\sum_{u,v} f_{\tau,u,v} d(u,v) + 2\sum_{\tau}\sum_{v}\sum_{r} x_{\tau,v,r}\Pr(z \sim P_\tau = r) d(v,r) \\
\forall \tau, v & b_{\tau+1,v} = b_{\tau,v} + \sum_{u} f_{\tau,u,v} - \sum_{u} f_{\tau,v,u}. \\
\forall \tau, u, v & \sum_{v} x_{\tau,v,r} \geq 1. \\
\forall \tau, v, r & x_{\tau,v,r} \leq b_{\tau,v}. \\
\forall \tau & \sum_{v} b_{\tau,v} \leq k.
\end{array}$$

Next, in Section 5.5 we show how a fractional solution can be rounded to an integral solution.

5.5 Reduction from Integral k-server to Fractional k-server

In this section we show how we can obtain an integral algorithm for the stochastic k-server problem from a fractional algorithm. We first show that every fractional algorithm for the line metric can be modified to an integral algorithm with the same cost. Next, we study the problem on HST metrics; we give a rounding method that produces an integral algorithm from a fractional algorithm while losing a constant factor. Finally, we leverage the previously known embedding techniques to show every metric can be embedded into HST's with a distortion of at most $O(\log n)$. This will lead to a rounding method for obtaining an integral algorithm from every fractional algorithm on general metrics while losing a factor of at most $O(\log n)$. Combining this with the 3 approximation fractional algorithm that we provide in Section 5.4, we achieve an $O(\log n)$ approximation algorithm for the stochastic k-server problem on general graphs.

5.5.1 Integrals Are as Strong as Fractionals On the Line

In this section we show every fractional algorithm on the line metric can be derandomized to an integral solution with the same expected cost. The rounding method is as follows: For every fractional configuration A, we provide an integral configuration $\mathrm{I}(A)$ such that (i) the distance of two configurations A_1 and A_2 is equal to the expected distance of two configurations $\mathrm{I}(A_1)$ and $\mathrm{I}(A_2)$. (ii) for every point x in the metric that A has a server mass of size at least 1 on x, there exists a server on point x in $\mathrm{I}(A)$.

Let for every point x in the metric, $A(v)$ denote the amount of server mass on node v of the line. For every fractional configuration B, we define a mass function $f_A : (0, k] \to V$ as follows. $f_A(x) = v_j$ if and only if j is the minimum integer such that $\sum_{i=1}^{j-1} A(i) < x$ and $\sum_{i=1}^{j} A(i) \geq x$. Intuitively, if one gathers the server mass by sweeping the line from left to right, $f_A(x)$ is the first position on which we have gathered x amount of server mass. The rounding algorithm is as follows:

- Pick a random real number r in the interval $[0, 1)$.
- $\mathrm{I}(A)$ contains k servers on positions $f_A(r)$, $f_A(r+1)$, $f_A(r+2)$, $\ldots$, $f_A(r+k-1)$.

Note that the rounding method uses the same r for all of the configurations. More precisely, we draw r from $[0, 1)$ at first and use this number to construct the integral configurations from fractional configurations. The following two lemmas show that both of the properties hold for the rounding algorithm we proposed.

Lemma 5.3. *Let A be a fractional configuration and x be a point such that $A(x) \geq 1$. Then $\mathrm{I}(A)$ has a server on x.*

Proof. Due to the construction of our rounding method, for every two consecutive servers a and b in $\mathrm{I}(\mathcal{A})$, the total mass of servers after a and before b in the fractional solution is less than 1. Therefore, $\mathrm{I}(A)$ should put a server on point x, otherwise the total mass of servers in the fractional solution between the first server before x and the first server after x would be at least 1. $\square$

The next lemma shows that the rounding preserves the distances between the configurations in expectation.

Lemma 5.4. *Let A_1 and A_2 be two fractional configurations and $|A_1 - A_2|$ be their distance. The following holds for the distances of the configurations*

$$\mathbb{E}|\,\mathrm{I}(A_1) - \mathrm{I}(A_2)| = |A_1 - A_2|.$$

Proof. The key point behind the proof of this lemma is that the distance of two fractional configurations A_1 and A_2 can be formulated as follows

$$|A_1 - A_2| = \int_0^1 |\,\mathrm{I}_\omega(A_1) - \mathrm{I}_\omega(A_2)| d_\omega$$

where $\mathrm{I}_\omega(A)$ stands for an integral configurations which places the servers on points $f_A(\omega)$, $f_A(\omega+1)$, $f_A(\omega+2)$, $\ldots$, $f_A(\omega+k-1)$. Since at the beginning of the rounding method we draw r uniformly at random, the expected distance of the two rounded configurations is exactly equal to

$$\int_0^1 |\,\mathrm{I}_\omega(A_1) - \mathrm{I}_\omega(A_2)| d_\omega$$

which is equal to the distance of A_1 from A_2. □

Theorem 5.8. *For any given fractional online algorithm $\mathcal{A}$ for the k-server problem on the line metric, there exists an online integral solution for the same problem with the same expected cost.*

5.5.2 Reduction for General Graphs

An HST is a undirected rooted tree in which every leaf represents a point in the metric and the distance of a pair of points in the metric is equal to the distance of the corresponding leaves in the tree. In an HST, weights of the edges are uniquely determined by the depth of the vertices they connect. More precisely, in a σ-HST the weight of an edges between a vertex v and its children is equal to σ^{h-d_v} where h stands for the height of the tree and d_v denotes the depth of vertex v.

Since HSTs are very well structured, designing algorithms on HSTs is relatively easier in comparison to a more complex metric. Therefore, a classic method for alleviating the complexity of the problems is to first embed the metrics into HSTs with a low distortion and then solve the problems on these trees.

Perhaps the most important property of the HSTs is the following:

Observation 5.9. *For every pair of leaves $u, v \in T$ of an HST, the distance of u and v is uniquely determined by the depth of their deepest common ancestor.*

Note that, the higher the depth of the common ancestor is, the lower the distance of the leaves will be. Therefore, the closest leaves to a leaf v are the ones that share the most common ancestors with v. Bansal *et al.* propose a method for rounding every fractional solution of the k-server problem to an integral solution losing at most a constant factor [133].

Theorem 5.10. *[133] Let T be a σ-HST with n leaves, $\sigma > 5$, and let $A = \langle A_0, A_1, A_2, \ldots, A_t \rangle$ be a sequence of fractional configurations. There is an on-line procedure that maintains a sequence of randomized k-server configurations $S = \langle S_0, S_1, S_2, \ldots, S_t \rangle$ satisfying the following two properties:*

- *At any time i, the state S_i is consistent with the fractional state A_i.*

- *If the fractional state changes from x_{i-1} to x_i at time i, incurring a movement cost of c_i, then the state S_{i-1} can be modified to a state S_i while incurring a cost of $O(c_i)$ in expectation.*

Embedding general metrics into trees and in particular HSTs has been the subject of many studies. The seminal work of Fakcharoenphol *et al.* [155] has shown that any metric can be randomly embedded to σ-HSTs with distortion $O(\frac{\sigma \log n}{\log \sigma})$.

Theorem 5.11. *[155] There exists a probabilistic method to embed an arbitrary metric $\mathcal{M}$ into σ-HSTs with distortion $\frac{\sigma \log n}{\log \sigma}$.*

Therefore, to round a fractional solution on a general metric, we first embed it into 6-HSTs with a distortion of at most $O(\log n)$ and then round the solution while losing only a constant factor. This will give us an integral algorithm that has an expected cost of at most $O(\log n)$ times the optimal.

Theorem 5.12. *For any given fractional online algorithm $\mathcal{A}$ for the k-server problem on an arbitrary metric, there exists an online integral solution for the same problem having a cost of no worse that $O(\log n)$ times the cost of $\mathcal{A}$ in expectation.*

5.6 Acknowledgment

We would like to thank Shi Li for having helpful discussions, and anonymous reviewers for their comments [3].

[3]We exploit the structural properties of non-adaptive algorithms and provide a simple approximation algorithm for the stochastic k-server problem. Nonetheless, one might think our structural result has a simpler proof via sampling and simulation methods. However, to the best of our

5.7 Correlated Setting

In this section, we study the k-server problem when the probability distributions are not independent. Recall that in the independent setting the sequence of requests is referred to by $\rho = \langle r_1, \ldots, r_t \rangle$. In the correlated model we assume all different possibilities for ρ have been given in the form of a set $\mathcal{R} = \{\rho_1, \ldots, \rho_m\}$ of m sequences $\rho_i = \langle r_{i,1}, \ldots, r_{i,t} \rangle$. Moreover, we assume the probability of each scenario ρ_i is denoted by p_i and given in advance. Given the list of different scenarios and probabilities, the goal is to design an online algorithm to serve each request $r_{i,j}$ prior to arrival of the next request such that the overall movement of the servers is minimized.

We model this problem by an integer program. We first write an integer program and show that every solution of this program is uniquely mapped to a deterministic online algorithms for the problem. Moreover, every online algorithm can be mapped to a feasible solution of the program. More precisely, each solution of the program is equivalent to an online algorithm of the problem. Furthermore, we show how to derive an online algorithm from the solution of the integer program. These two imply that the optimal deterministic online algorithm can be obtained from the optimal solution of the program.

knowledge such approaches do not deliver the right bounds.

5.7.1 Program

To better convey the idea behind the integer program, we first introduce the tree T which is a trie containing all sequences ρ_1 to ρ_m. Let us use $w(v)$ to denote the path from the root to a node v. With these notations, a node $v \in T$ represents a request which may occur conditioning all requests in $w(v)$ occur beforehand. Besides, every leaf of T uniquely represents one of the ρ_i's. Let us use $l(v)$ to denote the set of those indices i for which ρ_i is a leaf of the subtree of v. At each step t, only those ρ_i's can be a final option for R that $\langle \rho_{i,1}, \ldots, \rho_{i,t} \rangle = \langle r_1, \ldots, r_t \rangle$. Hence, a new request r_t can be informative since we know that none of the ρ_i's in $l(r_{t-1}) \backslash l(r_t au)$ will occur anymore. For a node v we define $Pr(v)$ as the probability of all requests in $w(v)$ happening i.e. $Pr(v) = \sum_{i \in l(v)} Pr(R = \rho_i)$.

We extend the tree T by adding $k-1$ additional nodes. As shown in Figure 1, these nodes form a path leading to the root of T. These nodes plus the root represent the initial configuration of the k servers. Let us call these nodes the initial set I. Now we can show the movement of the servers in our metric space by means of k tokens in T. To do so, we begin with putting one token on each of the k nodes of I. Each token corresponds to one of the servers. After a server moves to serve a request r_t, we move its corresponding token to a node of T which represents the request r_t. Note that at this step, there is no discrimination between any of the sequences in $l(r_t)$ in terms of occurrence. This causes a deterministic online algorithm A to serve the first $|w(r_t)|$ requests of R in the same way if R is going to be one of ρ_i's ($i \in l(r_t)$). A result of this uniquely serving is that we can use some downward links on T in order to show how each request v gets served. In the next paragraphs we explain about these links and how we construct the integer program.

Let us use $x_{u,v}$ to denote a link from a node $u \in T$ to its descendant v. $x_{u,v}$ is one if and only if A uses the same server to serve u and then v without using that server to serve any other request between u and v. This consecutive serving may occur with probability $Pr(v) = Pr(u)Pr(v|u)$. In this case, the algorithm moves a server from u to v and pays $|u - v|$ as the distance cost between the two points of the metric space corresponding to u and v.

There are two conditions for these links that we must care about. First, since each request v should be served with a server, at least one of the $x_{u,v}$'s should be one for all u in $w(v)$. Without loss of generality, we assume this is exactly one of them, i.e. there is no need to serve a request with more than one server. Second, after serving a request u, a server can go for serving at most one other request. That is, for each $i \in l(u)$, there should be at most one $v \in \rho_i$ such that $x_{u,v} = 1$. This condition guarantees that in serving the sequence of requests R, a server which serves $r_{t_1} \in R$ has always at most one other request $r_{t_2} \in R$ as the next serving request.

The following integer program maintains both conditions for $x_{u,v}$'s and has the expected overall movement of all servers as the objective function:

$$\begin{array}{ll}
\text{min.} & \sum_{u,v \in T; u \in w(v)} Pr(v)|u-v| x_{u,v} \\
\forall v \in T \backslash I & \sum_{u \in w(v)} x_{u,v} = 1. \\
\forall u \in T, i \in l(u) & \sum_{v \in \rho_i} x_{u,v} \leq 1. \\
\forall u, v \in T, u \in w(v) & x_{u,v} \in \{0,1\}
\end{array}$$

Next, we can relax the constraints of the program to make it linear. Therefore, instead of assigning either $\{0\}$ or $\{1\}$, to each $x_{u,v}$ we let it be a real number between 0 and 1. Thus, the integer program turns to the following linear program with the same objective function but more relaxed constraints.

$$\begin{array}{ll}
\text{min.} & \sum_{u,v \in T; u \in w(v)} Pr(v)|u-v| x_{u,v} \\
\forall v \in T \backslash I & \sum_{u \in w(v)} x_{u,v} = 1. \\
\forall u \in T, i \in l(u) & \sum_{v \in \rho_i} x_{u,v} \leq 1. \\
\forall u, v \in T, u \in w(v) & x_{u,v} \leq 1 \\
\forall u, v \in T, u \in w(v) & x_{u,v} \geq 0
\end{array}$$

Note that every feasible solution of the linear program is corresponding to a fractional solution of the problem. Since the optimal solution of the linear program can be found in polynomial time, using the rounding methods presented in Section 5.5 we obtain an optimal online algorithm for the line metric and a $O(\log n)$ approximation algorithm for general metrics as stated in Theorem 5.6.

5.8 Experimental Results

The goal of this section is to make an evaluation of our method for the line on a real world data set. The line can be an appropriate model for a plenty of applications. For example, it could be sending road maintenance trucks to different points of a road or sending emergency vehicles to accident scenes along a highway. For this experiment, we take the case of car accidents.

Data sets. We use Road Safety Data[4] to find the distribution of the accidents along the A1[5] road in Great Britain. In 2015, over 1600 accidents occurred on this highway, with an average of 140 accidents per month. We assume a point every 10 miles along the highway. That is 40 points in total. Then we build the distributions with respect to how the accidents are spread over the days of month. In this way, we achieve 30 distributions for 40 points along the line.

Algorithms. We compare the performance of our method to that of the optimum algorithm. To find the optimum solution we use backtracking. The running time of the algorithm is exponential to k. However, we use techniques such as *branch and bound* and *exponential dynamic programming* to get a fast implementation.

Results. We run different experiments with k from 2 to 11 on the line and distributions explained above. In previous sections we showed an upper bound of 3 for the approximation factor of our algorithm. Interestingly, in these experiments we can observe a better performance as shown by Figure 5.1. We compare the running

[4]https://data.gov.uk/dataset/road-accidents-safety-data/

[5]https://en.wikipedia.org/wiki/A1_road_(Great_Britain)

Number of Servers	2	3	4	5	6	7	8	9	10	11
Algorithm	6.5	7.6	6.7	7.1	7.5	8.3	8.5	8.4	9.3	8.2
Optimum	0.2	0.8	3.1	8.4	29.4	57.9	126.3	406.7	1477.1	6173.6

Table 5.1: The running time of our algorithm and the optimum algorithm in seconds. For higher number of servers, the optimum solution was not calculable within 5 hours.

time of the algorithms in Table 5.1. Note that the size of our LP our method solvers does not vary by k. This is in fact the reason behind why its running time remains almost the same. In contrast, the running time of the optimum algorithm grows exponentially.

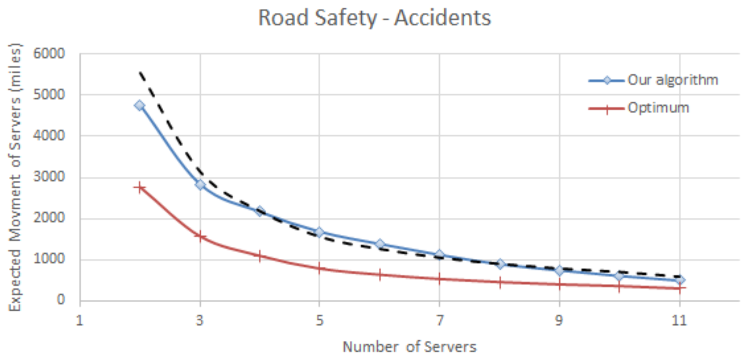


Figure 5.1: Performance of our algorithm compared to the optimum. The dashed curve indicates two times the optimum.

Bibliography

[1] Frank K Hwang, Dana S Richards, and Pawel Winter. *The Steiner tree problem*, volume 53. Elsevier, 1992.

[2] Dingzhu Du and Xiaodong Hu. *Steiner tree problems in computer communication networks*. World Scientific, 2008.

[3] Hans Jürgen Prömel and Angelika Steger. *The Steiner tree problem: a tour through graphs, algorithms, and complexity*. Springer Science & Business Media, 2012.

[4] Xiuzhen Cheng and Ding-Zhu Du. *Steiner trees in industry*, volume 11. Springer Science & Business Media, 2013.

[5] EN Gilbert and HO Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[6] Ding-Zhu Du, JM Smith, and J Hyam Rubinstein. *Advances in Steiner trees*, volume 6. Springer Science & Business Media, 2013.

[7] Anupam Gupta, Ravishankar Krishnaswamy, and R Ravi. Online and stochastic survivable network design. *SIAM Journal on Computing*, 41(6):1649–1672, 2012.

[8] Joseph Naor, Debmalya Panigrahi, and Monika Singh. Online node-weighted steiner tree and related problems. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 210–219. IEEE, 2011.

[9] Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. In *FOCS 2012*.

[10] Sein Win. On a connection between the existence ofk-trees and the toughness of a graph. *Graphs and Combinatorics*, 5(1):201–205, 1989.

[11] Vašek Chvátal. Tough graphs and hamiltonian circuits. *Discrete Mathematics*, 5(3):215–228, 1973.

[12] Paolo M Camerini, Giulia Galbiati, and Francesco Maffioli. Complexity of spanning tree problems: Part i. *European Journal of Operational Research*, 5(5):346–352, 1980.

[13] Paolo M Camerini and Giulia Galbiati. The bounded path tree problem. *SIAM Journal on Algebraic Discrete Methods*, 3(4):474–484, 1982.

[14] Christos H Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM*, 29(2):285–309, 1982.

[15] R Garey Michael and S Johnson David. Computers and intractability: a guide to the theory of np-completeness. *WH Freeman & Co., San Francisco*, 1979.

[16] Martin Fürer and Balaji Raghavachari. An NC approximation algorithm for the minimum degree spanning tree problem. In *Allerton Conf. on Communication, Control and Computing*, pages 274–281, 1990.

[17] Ajit Agrawal, Philip Nathan Klein, and R Ravi. How tough is the minimum-degree steiner tree?: A new approximate min-max equality. *Technical Report CS-91-49, Brown University*, 1991.

[18] Prabhakar Raghavan and Clark D Thompson. Provably good routing in graphs: regular arrays. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 79–87, 1985.

[19] Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, 1994.

[20] Madhav V Marathe, R Ravi, Ravi Sundaram, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Bicriteria network design problems. *Journal of Algorithms*, 28(1):142–171, 1998.

[21] Michel X Goemans. Minimum bounded degree spanning trees. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 273–282, 2006.

[22] Zeev Nutov. Degree-constrained node-connectivity. In *LATIN 2012: Theoretical Informatics*, pages 582–593. 2012.

[23] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. *SIAM Journal on Computing*, 42(6):2217–2242, 2013.

[24] Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. On some network design problems with degree constraints. *Journal of Computer and System Sciences*, 79(5):725–736, 2013.

[25] Alina Ene and Ali Vakilian. Improved approximation algorithms for degree-bounded network design problems with node connectivity requirements. *STOC*, 2014.

[26] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 661–670, 2007.

[27] Douglas Bauer, Hajo Broersma, and Edward Schmeichel. Toughness in graphs–a survey. *Graphs and Combinatorics*, 22(1):1–35, 2006.

[28] Hector Garcia-Molina and Boris Kogan. An implementation of reliable broadcast using an unreliable multicast facility. In *Reliable Distributed Systems, 1988. Proceedings., Seventh Symposium on*, pages 101–111, 1988.

[29] Yongqiang Huang and Hector Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. In *Mobile Data Management*, pages 122–140, 2003.

[30] Jochen Könemann and R Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 537–546, 2000.

[31] Jochen Könemann and R Ravi. Primal-dual meets local search: approximating msts with nonuniform degree bounds. *SIAM Journal on Computing*, 34(3):763–773, 2005.

[32] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

[33] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree-bounded directed network design. *SIAM J. Comput.*, 39(4):1413–1431, 2009.

[34] Makoto Imase and Bernard M Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

[35] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 68–74, 1996.

[36] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 344–353, 1997.

[37] Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 210–219, 2011.

[38] Mohammad Taghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 558–567, 2013.

[39] Jiawei Qian and David P Williamson. An o (logn)-competitive algorithm for online constrained forest problems. In *Automata, Languages and Programming*, pages 37–48. 2011.

[40] MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-optimal online algorithms for prize-collecting steiner problems. In *Automata, Languages, and Programming*, pages 576–587. 2014.

[41] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

[42] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 117–126, 2009.

[43] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596, 2011.

[44] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606, 2011.

[45] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007.

[46] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.

[47] Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1690–1701, 2012.

[48] Saeed Alaei, Mohammad T Hajiaghayi, Vahid Liaghat, Dan Pei, and Barna Saha. Adcell: Ad allocation in cellular networks. In *Algorithms–ESA 2011*, pages 311–322. 2011.

[49] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 18–35, 2012.

[50] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 11–25. 2013.

[51] Aranyak Mehta. Online matching and ad allocation. *Theoretical Computer Science*, 8(4):265–368, 2012.

[52] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.

[53] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal: dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.

[54] Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 85–100. SIAM, 2013.

[55] Fred Bauer and Anujan Varma. Degree-constrained multicasting in point-to-point networks. In *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE*, pages 369–376. IEEE, 1995.

[56] Carlos AS Oliveira and Panos M Pardalos. A survey of combinatorial optimization problems in multicast routing. *Computers & Operations Research*, 32(8):1953–1981, 2005.

[57] Stefan Voß. Problems with generalized steiner problems. *Algorithmica*, 7(1):333–335, 1992.

[58] Takuro Fukunaga and R Ravi. Iterative rounding approximation algorithms for degree-bounded node-connectivity network design. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 263–272. IEEE, 2012.

[59] Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. What would edmonds do? augmenting paths and witnesses for degree-bounded msts. *Algorithmica*, 55(1):157–189, 2009.

[60] Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. A push-relabel algorithm for approximating degree bounded msts. In *Proceedings of the 33rd international conference on Automata, Languages and Programming-Volume Part I*, pages 191–201. Springer-Verlag, 2006.

[61] Michel X Goemans. Minimum bounded degree spanning trees. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 273–282. IEEE, 2006.

[62] Philip N Klein, Radha Krishnan, Balaji Raghavachari, and R Ravi. Approximation algorithms for finding low-degree subgraphs. *Networks*, 44(3):203–215, 2004.

[63] Jochen Könemann and R Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 537–546. ACM, 2000.

[64] Jochen Könemann and R Ravi. Primal-dual meets local search: approximating mst's with nonuniform degree bounds. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 389–395. ACM, 2003.

[65] Lap Chi Lau, Joseph Naor, Mohammad R Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM Journal on Computing*, 39(3):1062–1087, 2009.

[66] Balaji Raghavachari. Algorithms for finding low degree structures. In *Approximation algorithms for NP-hard problems*, pages 266–295. PWS Publishing Co., 1996.

[67] R Ravi, Madhav V Marathe, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001.

[68] R Ravi and Mohit Singh. Delegate and conquer: An lp-based approximation algorithm for minimum degree msts. In *Automata, Languages and Programming*, pages 169–180. Springer, 2006.

[69] Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online degree-bounded steiner network design. In *SODA*, 2016. http://web.stanford.edu/~vliaghat/uploads/notes.pdf.

[70] Deeparnab Chakrabarty, Alina Ene, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. In *FOCS*, 2015.

[71] James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. Online routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM (JACM)*, 44(3):486–504, 1997.

[72] Michel X Goemans, Andrew V Goldberg, Serge A Plotkin, David B Shmoys, Eva Tardos, and David P Williamson. Improved approximation algorithms for network design problems. In *SODA*, volume 94, pages 223–232, 1994.

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

[74] Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011.

[75] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[76] R Ravi and Philip Klein. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *3rd Conference on Integer Programming and Combinatorial Optimization*, pages 39–56, 1993.

[77] David P Williamson, Michel X Goemans, Milena Mihail, and Vijay V Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

[78] Makoto Imase and Bernard M Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

[79] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 344–353. ACM, 1997.

[80] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. *Theoretical Computer Science*, 324(2):313–324, 2004.

[81] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.

[82] Mohammad T Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 558–567. IEEE, 2013.

[83] Jiawei Qian and David P Williamson. An o (logn)-competitive algorithm for online constrained forest problems. In *Automata, Languages and Programming*, pages 37–48. Springer, 2011.

[84] MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-optimal online algorithms for prize-collecting steiner problems. In *Automata, Languages, and Programming*, pages 576–587. Springer, 2014.

[85] Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[86] Yossi Azar and Oded Regev. Strongly polynomial algorithms for the unsplittable flow problem. In *Integer Programming and Combinatorial Optimization*, pages 15–29. Springer, 2001.

[87] Alok Baveja and Aravind Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *Mathematics of Operations Research*, 25(2):255–280, 2000.

[88] Stavros G Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004.

[89] Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via raecke decompositions. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 277–286. IEEE, 2010.

[90] Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 855–874. ACM, 2012.

[91] Ken-ichi Kawarabayashi and Yusuke Kobayashi. Breaking o (n 1/2)-approximation algorithms for the edge-disjoint paths problem with congestion two. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 81–88. ACM, 2011.

[92] András Frank. Edge-disjoint paths in planar graphs. *Journal of Combinatorial Theory, Series B*, 39(2):164–178, 1985.

[93] Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. Edge-disjoint paths in planar graphs with constant congestion. *SIAM Journal on Computing*, 39(1):281–301, 2009.

[94] Löc Séguin-Charbonneau and F Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 200–209. IEEE, 2011.

[95] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *European Symposium on Algorithms*, pages 170–181. Springer, 2010.

[96] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 29–38. ACM, 2011.

[97] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 117–126. IEEE, 2009.

[98] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In

International Workshop on Internet and Network Economics, pages 170–181. Springer, 2011.

[99] Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646, 2013.

[100] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.

[101] Aris Anagnostopoulos, Clément Dombry, Nadine Guillotin-Plantard, Ioannis Kontoyiannis, and Eli Upfal. Stochastic analysis of the k-server problem on the circle. *DMTCS Proceedings*, (01):21–34, 2010.

[102] Randy Cogill and Sanjay Lall. On decentralized policies for the stochastic k-server problem. *arXiv preprint math/0605188*, 2006.

[103] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *SODA*, pages 942–951. Society for Industrial and Applied Mathematics, 2008.

[104] MohammadTaghi Hajiaghayi, Jeong Han Kim, Tom Leighton, and Harald Räcke. Oblivious routing in directed graphs with random demands. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 193–201. ACM, 2005.

[105] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 18–35. ACM, 2012.

[106] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 11–25. Springer, 2013.

[107] Saeed Alaei, Mohammad T Hajiaghayi, Vahid Liaghat, Dan Pei, and Barna Saha. Adcell: Ad allocation in cellular networks. In *European Symposium on Algorithms*, pages 311–322. Springer, 2011.

[108] Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 123–135. SIAM, 2015.

[109] Joseph Cheriyan and Mohammad R Salavatipour. Hardness and approximation results for packing steiner trees. *Algorithmica*, 45(1):21–43, 2006.

[110] Matthias Kriesell. Edge-disjoint trees containing some given vertices in a graph. *Journal of Combinatorial Theory, Series B*, 88(1):53–65, 2003.

[111] Louis Petingi and J Rodriguez. Bounds on the maximum number of edge-disjoint steiner trees of a graph. *Congressus Numerantium*, pages 43–52, 2000.

[112] Kamal Jain, Mohammad Mahdian, and Mohammad R Salavatipour. Packing steiner trees. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 266–274. Society for Industrial and Applied Mathematics, 2003.

[113] Lap Chi Lau. An approximate max-steiner-tree-packing min-steiner-cut theorem. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 61–70. IEEE, 2004.

[114] Matt DeVos, Jessica McDonald, and Irene Pivotto. Packing steiner trees. *arXiv preprint arXiv:1307.7621*, 2013.

[115] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 347–356. IEEE, 2008.

[116] U Krengel and L Sucheston. Semiamarts and finite values. In *Bull. Am. Math. Soc.*, 1977.

[117] U Krengel and L Sucheston. On semiamarts, amarts, and processes with finite value. In *Kuelbs, J., ed., Probability on Banach Spaces*, 1978.

[118] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, volume 7, pages 58–65, 2007.

[119] DP Kennedy. Prophet-type inequalities for multi-choice optimal stopping. *Stochastic Processes and their Applications*, 24(1):77–88, 1987.

[120] DP Kennedy et al. Optimal stopping of independent random variables and maximizing prophets. *The Annals of Probability*, 13(2):566–571, 1985.

[121] Robert P Kertz. Comparison of optimal value and constrained maxima expectations for independent random variables. *Advances in applied probability*, pages 311–340, 1986.

[122] S. Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *FOCS*. 2011.

[123] Robert Kleinberg and Seth Matthew Weinberg. Matroid prophet inequalities. In *STOC*. ACM, 2012.

[124] Paul Dütting and Robert Kleinberg. Polymatroid prophet inequalities. In *Algorithms-ESA 2015*, pages 437–449. Springer, 2015.

[125] Aviad Rubinstein. Beyond matroids: Secretary problem and prophet inequality with general constraints. *arXiv preprint arXiv:1604.00357*, 2016.

[126] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2007.

[127] Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *International Colloquium on Automata, Languages, and Programming*, pages 508–519. Springer, 2014.

[128] Mark S Manasse, Lyle A McGeoch, and Daniel D Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

[129] Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[130] Elias Koutsoupias and Christos H Papadimitriou. On the k-server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983, 1995.

[131] Marek Chrobak, H Karloof, T Payne, and S Vishwnathan. New ressults on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.

[132] Marek Chrobak and Lawrence L Larmore. An optimal on-line algorithm for k servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.

[133] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. 2011.

[134] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theoretical computer science*, 324(2):337–345, 2004.

[135] Yair Bartal, Marek Chrobak, and Lawrence L Larmore. A randomized algorithm for two servers on the line. *Information and Computation*, 158(1):53–69, 2000.

[136] Ulrich Krengel, Louis Sucheston, et al. Semiamarts and finite values. *Bull. Amer. Math. Soc*, 83(4), 1977.

[137] Duru Türkoglu. *The k-Server Problem and Fractional Analysis.* PhD thesis, Masters Thesis, The University of Chicago, 2005. http://people. cs. uchicago. edu/ duru/papers/masters. pdf, 2005.

[138] Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, 1999.

[139] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[140] Lyle A McGeoch and Daniel D Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(1-6):816–825, 1991.

[141] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1):203–218, 2000.

[142] Nikhil Bansal, Niv Buchbinder, and Joseph Seffi Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):19, 2012.

[143] Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.

[144] Neal E Young. Bounding the diffuse adversary. In *SODA*, volume 98, pages 420–425, 1998.

[145] Konstantinos Panagiotou and Alexander Souza. On adequate performance measures for paging. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 487–496. ACM, 2006.

[146] Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995.

[147] Sandy Irani, Anna R Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, 1996.

[148] Amos Fiat and Manor Mendel. Truly online paging with locality of reference. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 326–335. IEEE, 1997.

[149] Anna R Karlin, Steven J Phillips, and Prabhakar Raghavan. Markov paging. *SIAM Journal on Computing*, 30(3):906–922, 2000.

[150] Susanne Albers, Lene M Favrholdt, and Oliver Giel. On paging with locality of reference. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 258–267. ACM, 2002.

[151] Peter J Denning. The working set model for program behavior. *Communications of the ACM*, 26(1):43–48, 1983.

[152] Luca Becchetti. Modeling locality: A probabilistic analysis of lru and fwf. In *Algorithms–ESA 2004*, pages 98–109. Springer, 2004.

[153] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 942–951. Society for Industrial and Applied Mathematics, 2008.

[154] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* cambridge university press, 2005.

[155] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455. ACM, 2003.