National & Kapodistrian University of Athens

Graduate Program in Logic, Algorithms and Computation

M.Sc. Thesis

Infrastructure Leasing Problems

by

Paraschos Koutris

Supervised by

Dimitris Fotakis

Athens

December 2010

Abstract

Infrastructure problems are important for a wide variety of applications in networks, business and planning. The standard assumption for modeling these problems is that, when a resource is purchased, it can be used at any time in the future, without inducing further cost. However, this fails to capture how time may influence the cost; in the real-world, resources can be leased for various time periods or the cost may vary according to the amount of time a resource is utilized. Motivated by this, we introduce a leasing framework for infrastructure problems, where a resource can be purchased for various periods of time (day, week, month) with a different cost. Naturally, purchasing a resource for a longer time period costs more, but the average cost per day is smaller. We first examine the structure of leasing problems and show a surprising connection to multistage stochastic optimization. Then, we describe the leasing variants of several problems: Parking Permit, Facility Location, Steiner Forest, Set Cover, and present both offline and online algorithms for most of these. Finally, we introduce the Sum-Radii Clustering problem, which is also related to leasing, and study its online version.

Keywords

Leasing, Facility Location, Steiner Tree, Set Cover, Clustering, Infrastructure Problems, Parking Permit, Stochastic Optimization

Contents

1 Introduction **1**

2 The Framework **5**

2.1 Definition 5

2.2 Examples of Leasing Problems 6

2.2.1 Steiner Tree Leasing 7

2.2.2 Steiner Forest Leasing 7

2.2.3 Facility Leasing 8

2.2.4 Set Leasing 9

2.2.5 Sum-Radii Leasing 9

2.3 The Structure of Leasing Problems 10

3 Multistage Stochastic Optimization and Leasing **13**

3.1 Multistage Stochastic Optimization 14

3.2 A Reduction from Leasing to Multistage Stochastic Optimization 16

4 Offline Leasing Problems **21**

4.1 Algorithms for Leasing Problems through Stochastic Optimization 22

4.2 Set Leasing 23

4.3 Sum-Radii Leasing 25

4.4 Facility Leasing 29

5 The Parking Permit Problem **35**

5.1 Definition 36

5.2 A Deterministic Approach 36

5.3 Randomized Algorithms 39

6 Online Sum-Radii Clustering **45**

6.1 Description of the Model 46

6.2 Equivalent models 47

6.3 The Deterministic Approach 50

6.4 Randomized Sum-Radii Clustering 54

6.4.1 A Lower Bound 55

6.4.2 A Fractional Online Algorithm 56

7 Online Leasing Problems **59**

7.1 Online Steiner Forest Leasing 60

7.2 Online Facility Leasing 61

7.2.1 A Deterministic Algorithm 62

7.2.2 A Randomized Algorithm 66

Chapter 1

Introduction

Nowadays, infrastructure problems are behind a wide variety of applications and systems in networks, business, logistics and planning. In this setting, the goal is to decide which parts of infrastructure should be purchased such that a specific action or plan can be carried out as efficiently as possible. This infrastructure may be servers or links in a network, or even warehouses and trucks for a business. Naturally, an important factor which influences the decisions is how the cost of the infrastructure varies; one has to look for cost-effective solutions. Classic problems which fall into this category are Facility Location, Steiner Tree and Steiner Forest, Set Cover and variants of clustering problems. Such problems comprise the core of the Operations Research field, but they have received a lot of attention from the Computer Science community as well, being not only of theoretical interest, but also of practical one [29, 28, 42].

A standard feature in most models of infrastructure problems is the permanence of the infrastructure purchased. This means that, when a link or a facility is purchased at some point, it can be used at any time in the future, even if we work in the online setting, without incurring any additional cost. This is a simplifying assumption, since in many cases we might want to *lease* parts of the infrastructure for specific periods of time, thus paying less to cover any needs and removing the necessity of investing a large sum of money to buy infrastructure. For example, consider a company which, instead of buying warehouses to store its goods, leases warehouses for varying amounts of time, depending on current and future needs. Moreover, it might be that the cost of infrastructure depends on how much it is used; for example, deploying a server in the web for a long period has a maintenance and update cost, which should be taken into account. Naturally, such a model should capture the nature of economy: a longer lease should cost more, but the average cost per unit of time should be less.

Models which try to describe different economies of scale have been proposed and extensively studied; however, no other model captures the notion of time and how it affects the cost. One such example is the *Buy-at-Bulk* model [5, 8, 37, 55], where the cost varies according to the amount of work we want to serve on a link or a facility. Clearly, buying more "capacity" costs less per unit. We should also note the relevant *Rent-or-Buy* model, where apart from buying infrastructure for various capacities, we can also purchase it forever by paying a larger cost [38]. Another example is the model of *perishable goods* in inventory theory [35, 51]. In this model, we assume that commodities have finite lifetime.

The study of leasing problems has only been recently initiated. The first who introduced the leasing model, where purchases have time durations and expire, was Meyerson [49], with the Parking Permit problem. In the same paper, the author introduced a leasing variant of Steiner Forest. Nevertheless, the leasing model in its general form was described later by Anthony and Gupta [6]. The authors formally defined the model and proved a surprising connection to multistage stochastic optimization. Using powerful tools from this area, they obtained algorithms for several offline leasing problems, such us Facility Leasing and Set Leasing. Subsequently, Nagarajan and Williamson [50] showed that the results from [6] are not optimal; they presented a better approximation algorithm for offline Facility Leasing and proposed the first online algorithm for Facility Leasing.

The purpose of this thesis is not only to present in detail the above results, but also to contribute to the study and better understanding of leasing problems. In particular, our contribution can be summarized as follows:

- We obtain an improved approximation algorithm for the offline Vertex Leasing problem, by presenting an alternative approach to solving offline Set Leasing.
- We give a randomized online algorithm for Facilitiy Leasing, which achieves a better competitive ratio than the previous deterministic algorithm.
- We present the first approximation algorithm for Sum-Radii Leasing, which achieves a constant approximation factor.
- We study the online Sum-Radii Clustering problem, in both the deterministic and randomized setting. We also prove an interesting connection to Parking Permit.

Finally, let us briefly describe the outline of this thesis. Chapter 2 describes in detail the framework of leasing problems and defines the standard and leasing versions of the problems we study. Moreover, in this chapter we prove several reductions and simplifications of the model, which will prove useful throughout the thesis. In chapter 3, we

give a brief overview of multistage stochastic optimization and several important results in this area. Next, we formally show the connection between leasing problems and multistage stochastic optimization. Chapter 4 presents and analyzes approximation algorithms for several leasing problems: Facility Location, Set Leasing, Sum-Radii Leasing, Steiner-Tree Leasing.

Chapter 5 initiates the study of online leasing problems. In this chapter, we present and study the Parking Permit problem, providing deterministic and randomized online algorithms, along with tight lower bounds. The next chapter 6 studies in detail the online Sum-Radii Clustering problem. Finally, in chapter 7 we examine the online versions of two leasing problems: Steiner Forest Leasing and Facility Leasing.

Although the thesis is a complete and thorough presentation of the area of leasing problems, the results we present are still far from optimal. Much remains to be done in order to understand how the notion of time and cost interleaves with the combinatorial structure of the problems. For example, the gap between lower and upper bounds on the competitive ratio of leasing problems is still large. Furthermore, we do not completely understand the connection of leasing problems with stochastic optimization or buy-at-bulk problems. It is possible that further advances on understanding leasing problems will also give insight to these areas of combinatorial optimization.

Chapter 2

The Framework

In this chapter, we will provide a framework which captures the leasing variant for a large class of infrastructure problems. This general framework was first presented by Anthony and Gupta [6]. We will subsequently give detailed examples of specific leasing problems which will be of interest in the next chapters. Finally, we will analyze the structure of the leasing problems, providing several simplifications which will prove useful in designing algorithms and proving lower bounds.

2.1 Definition

Let us first consider a general optimization problem Π. In the context of Π, we are given a set of potential clients (or demands) U. The goal is to serve the set $D \subseteq U$ of demands which will actually appear by constructing a solution from a set of elements X. For every subset $D \subseteq U$, there exists a corresponding set of solutions $\mathtt{Sol}(D) \subseteq 2^X$. Clearly, the structure of the set of solutions is determined by the combinatorial structure of the optimization problem Π. In the standard setting of optimization problems, each element $e \in X$ occurs with a cost $c(e)$ and we wish to minimize the total cost of the solution chosen to serve the set of clients.

In the leasing framework, we introduce a notion of time. The problem Π spans over a period of days $\{1, 2, \ldots, T\}$, where at each day t arrives a set of clients $D_t \subseteq U$. Since our goal is to serve the clients, we would have to purchase a solution $S_t \in \mathtt{Sol}(D_t)$ for the set D_t. However, we have not specified yet the *duration* of the elements we purchase. Can we use an element e we have purchased at day t for the solution of a later day or is it necessary to purchase it again? Clearly, if we assume that an element can be used only for one day, the problem would be trivially reduced to solving an independent optimization

problem for each day. On the other hand, if we assume that an element we buy at day t can be used at any later day for free, we would only have to solve the problem for the client set $D_1 \cup D_2 \cup \ldots \cup D_T$. But we would like to have a model which captures the duration-cost tradeoff: if you buy something for a longer time, you will pay more, but the average cost will be less in the end. Hence, it would be in our interest to purchase an element for a longer period of time if we could use this element enough times in a solution to make up for the amount it costs.

This observation leads naturally to the *leasing model*. Each element $e \in X$ can be purchased (or *leased*) at any day t for several different periods of time (we refer to these as *lease types* $1, 2, \ldots, K$), where the lease of type i for element $e \in X$ costs $c_e(i)$ and e can be used at days $t, t+1, \ldots, t+l(i)-1$. Thus, $l(i)$ refers to the *duration* of lease type i. Moreover, we denote by $S_t(k)$ the set of elements purchased at day t with lease type k. A *feasible leasing solution* consists of a sequence of sets $S_t(k)$ for $t = 1, \ldots, T$, such that for each day t, it holds that $S_t = \bigcup_{k=1,\ldots,K} \bigcup_{t'=t-l(k)+1}^{t} S_{t'}(k) \in \mathtt{Sol}(D_t)$. This means that the set of elements which are purchased with leases that cover day t must belong to the set of solutions $\mathtt{Sol}(D_t)$.

At this point, we should note that the cost of purchasing an element e with a lease type i depends both on e and i in an arbitrary way. However, it will often prove useful to restrict our attention to the case where the lease costs are uniform for each element in X. This leads to the following definition.

Definition 2.1.1. *We say that a leasing problem is* uniform *if the lease costs functions* $c_e(k)$ *depend only on the lease type we choose, namely* $c_e() = c_{e'}()$ *for every* $e, e' \in X$. *Otherwise, we say that the leasing problem is* non-uniform.

2.2 Examples of Leasing Problems

A leasing problem with simple structure is the Parking Permit problem [49]. This problem has essentially no combinatorial structure, as it only captures the notion of time. One could think of Parking Permit as any infrastructure problem where there is only one point in space.

More specifically, we are given K different types of parking permits. Permit k has a duration of $l(k)$ days and costs c_k. We are also given a schedule of T days with marked driving days. The goal is to select a set of permits so as to cover all driving days and minimize the total cost of permits purchased. We will analyze Parking Permit in detail in chapter 5.

Next, we will formally define the leasing variants of several other infrastructure problems. In theory, we could define leasing variants for a large class of combinatorial optimization problems; however, in this study, we will restrict our attention only to this smaller class of infrastructure problems.

2.2.1 Steiner Tree Leasing

In the Steiner Tree Leasing problem, we are given a graph $G = (V, E)$ and a root node $r \in V$. The set of elements X we can lease is the set of edges, i.e. $X = E$. For each day t, a set of terminal nodes $D_t \subseteq V$ arrives. Each edge of the graph can be leased for K possible lease lengths, where the cost of leasing any edge e with type k is $c_e(k)$. A feasible solution for a set of demand nodes $D_t \subseteq V$ consists of a subset of edges S_t such that the edge-induced subgraph on S_t connects each terminal node with the root r.

This problem has important applications in diverse areas, from circuit layout design to network design problems. Assume, for example, that we want to service clients in a network (which correspond to terminal nodes) by connecting them with a central server. These client requests arrive during a long period of time. Naturally, we wish to minimize the cost of the links we purchase such that every client is connected at any time.

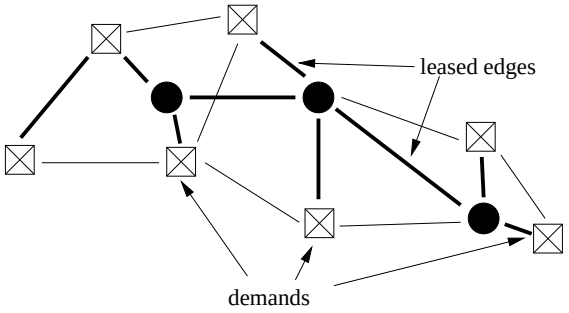


Figure 2.1: A solution for an instance of the Steiner Tree problem.

The Steiner Tree problem has been extensively studied from the perspective of both approximation algorithms [64, 54, 18], where the best approximation ratio achieved is 1.39, and online algorithms [61, 12].

2.2.2 Steiner Forest Leasing

In the Steiner Forest Leasing problem, we are given again a graph $G = (V, E)$. Each day, a family of sets of nodes D_t arrives. The set of elements X is the set of edges E and

each edge can be leased for K different lease durations. We pay $c_e(k)$ in order to lease an edge e with type k.

A feasible solution needs to maintain, at each day t, a set of leased edges such that all nodes of a set of the family D_t are pairwise connected. The goal is naturally to lease a set of edges so that the total leasing cost is minimized. Note that Steiner Forest Leasing collapses to Steiner Tree Leasing when the family D_t consists of only one set.

The classic Steiner Forest problem has been also extensively studied [1, 33, 15].

2.2.3 Facility Leasing

In Facility Location, we are given a set of potential facilities F in a metric space and a set of demands D in the same metric space. Each facility $i \in F$ has an opening cost f_i. The goal is to open a subset $S \subseteq F$ of the facilities so that the total cost of opening the facilities plus the distance of each point to its closest open facility is minimized.

In the leasing version of this problem, Facility Leasing, we are given a set of potential facility locations F (hence $X = F$) and a set of potential demands U in a metric space. At each day t, a set $D_t \subseteq U$ of clients arrives and each client must connect with a facility which is in lease at day t. There are K different possible types for leasing a facility and the cost of leasing a facility $i \in F$ with lease type k is f_i^k. Notice that we can define a uniform version of the problem, where the lease costs are exactly the same for each potential facility.

The goal is again to minimize the leasing cost of the facilities we open plus the total cost of serving each client in each set D_t from some facility open during day t (where the cost of the service is equal to the distance of the demand to the closest open facility).

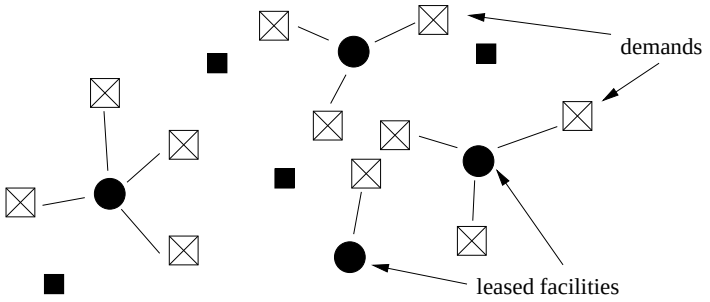


Figure 2.2: A solution for an instance of the Facility Leasing problem.

Facility Location is one of the most important problems in combinatorial optimization, as it has numerous applications to several fields. The bibliography considering

Facility Location and its variants is vast; here, we only refer to some results on approximation algorithms [57, 22, 46, 45, 24, 17], where the currently best approximation ratio is 1.5 [17] and the best lower bound is 1.463 [36]. The problem has also been studied in the online setting [48, 31, 32], where the competitive ratio has been proved to be $\Theta(\frac{\log n}{\log \log n})$ (n is the number of demands).

2.2.4 Set Leasing

In the Set Cover problem, we are given a set of elements U and a family $\mathcal{S}$ of subsets of $\mathcal{U}$. Every set $S \in \mathcal{S}$ has some weight w_S. The goal is to choose a subset $C \subseteq \mathcal{S}$ which covers the elements of a demand set $D \subseteq U$, so that the cost of the sets in C is minimum.

In Set Leasing, a subset $D_t \subseteq U$ of elements arrives at day t and needs to be covered. Each set in $\mathcal{S}$ ($X = \mathcal{S}$) can be leased for K different periods of time with a corresponding cost. The solution is feasible when the set of demands D_t for every day t will be always covered from the sets we have leased for that day.

We also consider a special case of Set Leasing, Vertex Leasing. In the latter problem we are given a graph $G = (V, E)$, demands $D_t \subseteq E$ arrive at day t and we have to find a set of vertices such that every edge has some endpoint in this set. Clearly this reduces to Set Leasing, if we define the universe U to be the set of vertices V and the family $\mathcal{S}$ to consist of $|V|$ sets, where each set contains all edges adjacent to a specific vertex.

Set Cover has an approximation ratio of $O(\ln n)$ [60], which is essentially optimal, unless of course $P = NP$ [4, 53]. We can also design an approximation algorithm with approximation ratio equal to the frequency α of the most frequent element (i.e. the element of U that appears most often in the sets).

2.2.5 Sum-Radii Leasing

In the Sum-Radii Clustering problem [23], we are given a set of potential demands U in a metric space and a set X of potential cluster centers. Opening a cluster with center at point i and radius r costs $f_i + r$. The goal is to specify a set of clusters with designated centers so that each point belongs in a cluster and the total cost of the clusters is minimized.

In the leasing version of this problem, Sum-Radii Leasing, a set $D_t \subseteq U$ of demands arrives at day t. There are K different possible lease types for each potential cluster, and opening a cluster with center $i \in X$, radius r and lease length k costs $f_i^k + r$. The goal is to specify clusters with appropriate centers and lease lengths so that the demand points

arriving each day belong in an open cluster and the cost of the open clusters is minimized.

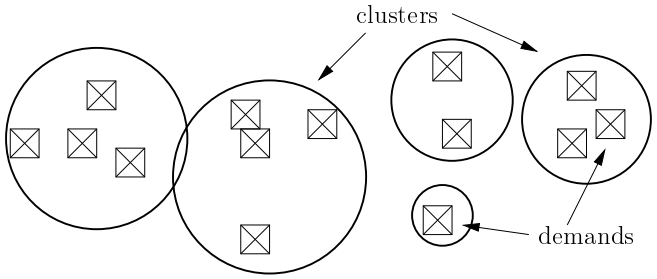


Figure 2.3: A solution for an instance of the Sum-Radii Clustering problem.

2.3 The Structure of Leasing Problems

In this section, we will show how to simplify the structure of any leasing problem without losing more than a constant factor from the cost of the optimum solution. This allows us to present several online and offline algorithms, as well as reductions, in a simpler way.

First of all, it is natural to assume that longer leases are more expensive than shorter leases. Thus, we may discard from any instance of a leasing problem lease types which are shorter and more expensive than some other lease type, since any reasonable algorithm will always prefer a longer and cheaper lease over a shorter and more expensive one.

Nevertheless, no lease type can be arbitrarily cheaper than another lease type. More specifically, a leasing problem makes sense only when buying a lease covering a time period is strictly cheaper than buying many smaller leases which cover the same time period. In a more formal way, we can say that the function $r_e(i) = \frac{c_e(i)}{l(i)}$, which corresponds to the average cost per day, is strictly decreasing. We refer to this assumption as the *subadditivity* property.

Definition 2.3.1 (Subadditivity)**.** *In any instance of a leasing problem, we may assume that the lease types satisfy the following property: for any lease types* $i < j$ *and any element* $e \in X$*, it holds that* $\frac{c_e(i)}{l(i)} > \frac{c_e(j)}{l(j)}$.

It turns out that it is possible to assume a stricter structure for the costs of the different lease types. In fact, we can assume that the lease costs scale at least by a factor of two as the lease duration increases.

Lemma 2.3.2 (Cost Scaling)**.** *[49] For each lease type* $1 \leqslant k \leqslant K$ *and any element*

$e \in X$, *we may assume that* $c_e(k) \geqslant 2 \cdot c_e(k-1)$, *with losing only a 2-factor from the optimum solution.*

Proof. Consider an instance I of a leasing problem, having an optimal solution OPT of cost C_{OPT}. Now, we start with the largest leasing type K and consider the lease types with decreasing order, deleting every lease type k such that $2 \cdot c_e(k) > c_e(k+1)$. Clearly, this procedure leads to an instance I' of the problem with a smaller number of possible lease types. We obtain a solution for I' by modifying the solution OPT as follows.

If an element e is purchased with a lease type k not deleted in I', we also purchase e with the same lease type in OPT'. Otherwise, we purchase e with the cheapest lease type k' of some longer duration. Since k was deleted, it must hold that $c_e(k) > \frac{1}{2}c_e(k')$ and thus we pay at most twice the cost of the lease type k. Thus, $C_{OPT'} \leqslant 2C_{OPT}$.

In general, any solution for I can be mapped to a solution for I' with at most twice the cost. □

The following lemma implies that we can also assume a simpler structure for the durations of the leases.

Lemma 2.3.3 (Length Scaling). *[6] For lease types* i, j *where* $i < j$, *we may assume that* $l(i)$ *divides* $l(j)$, *with losing only a 2-factor from the optimum solution.*

Proof. The easiest way to prove this is to round the length of each lease type down to the closest power of two. In order to obtain a solution for the new instance I', we adopt the following strategy. Assume that OPT buys a lease of type k for a period of time. Since the lease length is rounded down to the closest power of two, the same time period can be covered in I' by at most two leases of type k. By purchasing these two consecutive leases, we obtain a feasible solution and pay at most twice the cost of the optimal solution. □

Assuming that the previous lemma holds, we can impose the restriction that leases of different type are *nested*. This implies a crucial property, which is exploited in several algorithms: at each day, there exists only one possible lease of a specific type we can buy. Intuitively, this means that it suffices to decide only which lease type we need to buy and not when we need to purchase it.

Lemma 2.3.4 (Interval Model). *[49] We may assume that a lease of type* i *is only obtained for intervals of the form* $[t, t + l(i)]$, *where* t *is a multiple of* $l(i)$ *and lose only a 2-factor from the optimum solution.*

Proof. Following lemma 2.3.3, we may assume that $l(i)|l(j)$ for every $i < j$. Furthermore, let us define $R_k = l(k)/l(k-1)$, i.e. the number of leases of type $k-1$ that can be embedded in a lease of type k. Clearly, R_k will be an integer greater than 1.

Next, let us assume that, for each type k, we allow only leases which start at integer multiples of $l(k)$. Consider an instance I of a leasing problem, which has an optimum solution OPT with cost C_{OPT}. We may view the timeline divided in time intervals of length $l(k)$ for any lease type k. Then, any lease of type k purchased crosses at most two of the above intervals. This means that we can obtain a valid solution OPT′ by purchasing at most two leases of type k which fit exactly in the intervals defined. Clearly, we pay at most twice the cost of the optimum solution. □

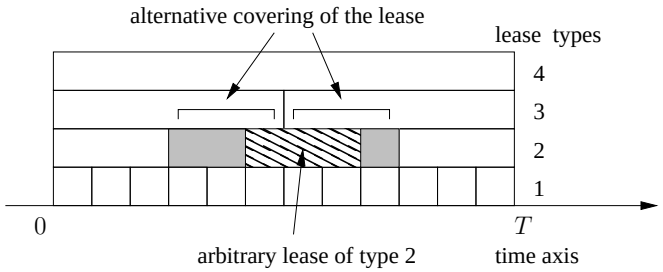


Figure 2.4: The nested structure of the interval model. Any lease of type 2 can be covered by purchasing two consecutive leases of the same type in the nested instance.

To sum up, the nature of the leasing problem allows us to impose several simplifications on the length and structure of the leases for any optimization problem, and this costs only a factor of 4 from the optimal solution. For several problems, we will need to assume this structure and, as long as the competitive or approximation ratio is not constant (but depends on K or n), the multiplicative factor is negligible. Nevertheless, for some problems where we would like to achieve a small constant approximation ratio, dealing with the arbitrary structure of the problem will be necessary.

Chapter 3

Multistage Stochastic Optimization and Leasing

In this section, we show a surprising connection between leasing problems and stochastic optimization, first studied by Anthony and Gupta [6]. In particular, we prove that any instance of a leasing infrastructure problem Π can be reduced to a corresponding instance of the multistage stochastic optimization version for Π. This reduction provides us with a powerful tool to construct algorithms for leasing problems, since we already have efficient approximation algorithms for the multistage stochastic versions of several combinatorial problems.

However, this approach raises the question of whether applying tools from stochastic optimization is actually the best way to solve leasing problems. It may be the case that the use of generic algorithms from stochastic optimization overlooks the structure of leasing problems, which we could exploit to design more efficient and accurate algorithms. Indeed, as we will show in the next section, a more restrictive approach yields better results. Nevertheless, this does not imply that multistage stochastic optimization is strictly more difficult than leasing problems. It seems that multistage stochastic optimization captures a wider variety of problems, but it could also be the case that stochastic optimization and leasing are actually equivalent problems. In this case, techniques and algorithms used for leasing problems could help us improve the algorithms for multistage stochastic optimization problems.

In this chapter, we will first present the notion of multistage stochastic optimization and then summarize the known results and algorithms for problems in this category. Finally, we will show the reduction from leasing to multistage stochastic optimization.

3.1 Multistage Stochastic Optimization

Let us assume that we are given a combinatorial problem Π, with a set U of possible demands and a set X of elements we can purchase to serve the demands. Clearly, for a given U, the goal is to find a set of elements which serves the set of the demands so as to minimize the total cost. The common offline case is that we get the set of demands as an input to our algorithm, hence we can plan a solution by exploiting any information about the location and structure of the demands.

However, this may not be always the case, since there are cases when we need to plan ahead while being uncertain about the future requirements. In fact, multistage stochastic optimization examines the case where the input is revealed through a sequence of stages. A classic example of such a model is the following [13]: we are in possession of a water reservoir and the goal is to minimize the expected cost of operation. The source of income is the amount of irrigation water we sell, but there is a restriction on the level of the water in the reservoir (it must always be larger than some threshold). Clearly, in this problem the source of uncertainty is the rainfall and this uncertainty can be modeled as a multi-stage procedure, since information about the present or previous days allow more accurate predictions of the future. More problems include forest planning models, electricity investment planning and bond investment planning (see [7] for more details).

Let us now formally describe the model. We assume the actual set of demands is not revealed until the last day K, and that it is drawn from a distribution π over the set of all possible demands. At each day, we receive additional information about the distribution of the final set of demands. This information is received in the form of a *signal*, which actually corresponds to updating the distribution of the demands conditioned on the information the signal carries. Hence, as time progresses, we obtain more accurate information about the final distribution.

At each day, after obtaining the signal, we can buy a new set of elements to augment the current solution based on the additional information we have gained. However, information comes with a cost: if one decides to buy an element later on, the cost of the purchase will be bigger. The growth of the cost may be either uniform or non-uniform. In the first case, which we call *uniform inflation*, the cost of an element at day t will be σ_t times bigger than the cost of the element at day $t-1$ (clearly, we must have that $\sigma_t \geqslant 1$), namely $c_e(t) = \sigma_t \cdot c_e(t-1)$. We will call σ_t the *inflation* of the cost at day t. In the non-uniform case, the costs may vary in a different way as the time progresses and more information about the distribution is collected.

When the K-th day arrives, we learn the actual set of demands and augment the solution

to obtain a feasible one for the demand set. The goal is to minimize the expected cost that we paid during the K-day period, where the expectation is taken over each possible scenario that may occur. The model of multistage stochastic optimization is described more thoroughly in [40] and [59].

Stochastic optimization dates back to the 50's and the work of Dantzig [26]. However, the study of classic combinatorial problems in the stochastic optimization framework is relatively recent. Initially, the work was limited to two-stage stochastic optimization. In this case, we choose an initial set of elements at the first day based on the distribution of the demands and at the second day, after the set of demands is revealed, we add the necessary elements to obtain a feasible solution. The first results on two-stage stochastic optimization for several combinatorial problems (Minimum Cost Flow, Bin Packing, Vertex Cover, Steiner Tree) were obtained by Immorlica et al. [44]. However, the authors restricted their attention only to the case when the distribution over the demands has a bounded support or when each demand appears independently in the final set with some probability. Hence, their algorithms, even though they have good approximation factors, are of limited applicability.

In [52], the authors studied a different kind of limitation on the distributions, that is, they demanded that the possible scenarios are polynomially many. Under this assumption, they provided approximation algorithms for several combinatorial problems, including a constant-factor approximation algorithm for stochastic Facility Location.

Nevertheless, it would be desirable to consider a more general model where the distribution over the possible scenarios is not given explicitly, but as a black-box, from which we can sample according to the distribution of the scenarios. We call this model the *black-box model*. Under this model, Shmoys and Swamy [56] provided approximation algorithms for solving a very broad class of stochastic linear problems. Their algorithms are based on a generalization of the ellipsoid method. Subsequently, Charikar, Chekuri and Pal [21] proved that the results of [56] can be obtained using a different approach, the Sample Average Approximation (SSA) method. This method is based on the idea that we can approximate any distribution over a large number of scenarios (or even infinite) by learning about this distribution from polynomially many samples.

Gupta et al. [39] followed a different approach to two-stage stochastic optimization in the black-box model. Their technique, which they call *Boosted Sampling*, applies to a general class of combinatorial problems which admit approximation algorithms with strict cost-sharing functions. The idea is to use the first day to draw a number of samples on the distribution (which depends on the inflation parameter) and then construct a solution for the union of the samples using the approximation algorithm. At the second day, after

the set of demands is revealed, we only need to augment the solution from the first day to cover all the remaining demands.

The Boosted Sampling framework can be extended to solve multistage stochastic optimization problems [40]. The authors obtained approximation factors for several combinatorial problems (Steiner Tree, Facility Location, Vertex Cover). Nevertheless, the approximation ratios depend on the number of stages K, in most cases exponentially, and only for the Steiner Tree problem the dependence is linear to K (in particular they give a $2K$ approximation factor). An $O(K)$ approximation algorithm for K-stage stochastic Steiner Tree was also independently proposed by Hayrapetyan et al. [41].

Furthermore, Swamy and Shmoys [59] independently extended the SSA method for multistage stochastic optimization problems. For fixed K, the authors provide fully polynomial randomized approximation schemes (FPRAS) for a large class of problems. They also provide algorithms with approximation ratios $O(K \log n)$ for K-stage stochastic Set Cover, $2K$ for K-stage stochastic Vertex Cover and $O(K)$ for the K-stage stochastic Facility Location problem. We note here that the approximation guarantees are essentially $O(K)$ times the approximation factor of the non-stochastic versions of these problems, i.e. we have a linear dependence on the number of stages.

Finally, Srinivasan [58] improved the approximation factors for the stochastic Vertex Cover and Set Cover by removing the dependence from the number of stages K. Hence, he provided a constant-time approximation algorithm for K-stage stochastic Vertex Cover and a $O(\log n)$ approximation algorithm for stochastic Set Cover.

3.2 A Reduction from Leasing to Multistage Stochastic Optimization

In this section, we formally present the reduction from any leasing problem to the corresponding multistage optimization problem, which was proved by Anthony and Gupta [6]. This seems surprising at first, since stochastic optimization and leasing are two problems from two different worlds. Stochastic optimization assumes a probability distribution over the set of demands and tries to minimize the expected cost of the solution over this distribution. On the other hand, leasing is a inherently deterministic problem. Hence, it is counter-intuitive how the two versions can be connected.

However, the idea behind the reduction is simple and elegant. In any leasing problem, at each day we are given a set of demands; one can view this as a probability distribution over the sets of demands. The core of the leasing problem is to understand how these sets

correlate in time and identify the clusters they make. Clearly, sets of demands that are close in time are candidates for being included in a larger cluster. How do we interpret this connection in the field of probability distributions? The idea is to force these demand sets to be correlated probabilistically, i.e. the closer the sets are in time, the more correlated will the sets be in the probability distribution. Thus, clustering in time corresponds to clustering in the probability space. It is then easy to see that the different levels on time clustering we have correspond to the stages, since at each stage we get more information and thus look at smaller clusters. By tuning the inflation cost and the stochastic process which reveals information about the distribution at each stage, we can make this correspondence precise.

Theorem 3.2.1. *Any (non-uniform) offline problem* Π *in the leasing framework with* K *lease types can be reduced to the standard* K*-stage stochastic optimization version of* Π*.*

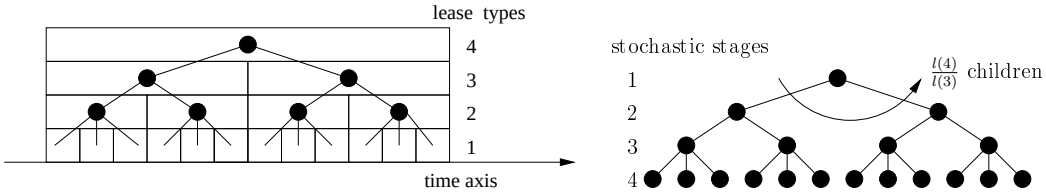
Proof. Consider the leasing version of a combinatorial problem Π, where the number of lease types is K. We will assume that the lease structure is nested; this means that we loose a factor of 4 in approximating the optimal value for the leasing problem. Furthermore, we can restrict our attention only to a time interval of type K, since the decisions we make for this interval do not influence the decisions that we may make for the next interval of type K. Thus, without loss of generality, we assume that the time starts at 0 and ends at $T = l(k) - 1$. The first goal is to express the optimization function we want to minimize for the leasing problem.

We denote by $S_t(i)$ the elements of lease type i we purchase at time t. Notice that $S_t(i) \neq \emptyset$ only when i precisely divides t. For simplicity, let also $C_t(i) = \sum_{e \in S_t(i)} c_e(i)$. Then, the objective function F_{Leas} can be written as

$$F_{Leas} = C_0(K) + \sum_{i=1}^{K} \sum_{t \in [0,T]: i|t} C_t(i)$$

We will next define a corresponding K-stage stochastic optimization instance of Π such that its objective function is exactly the same. Consider a tree T of depth $K-1$, where the nodes at depth i ($i < K-1$) have $\frac{l(K-i)}{l(K-i-1)}$ children (see Fig. 3.2). This means that T has $l(K)$ leaves. We associate each leaf with a corresponding set of demands.

It now remains to define the *signals* the stochastic process sends at each stage. The easiest way to visualize the process is to think of a particle that starts at the root of the tree and at each stage moves to a u.a.r. chosen child. Thus, at each stage, we know that the demands can be drawn only from the distributions that correspond to the leaves of the subtree rooted at the node where the particle stands. When the particle reaches a leaf, we output the set of demands associated with this leaf.



(a) An instance of a leasing problem (b) The tree T of the stochastic model

Figure 3.1: The correspondence between the structure of the leasing and the stochastic optimization on a problem Π.

However, in stochastic optimization the cost of purchasing elements varies according to the stage we buy the element. In our case, purchasing an element at stage $i+1$ costs $p_e(i+1) = c_e(K-i) \cdot \frac{l(K)}{l(K-i)}$. It is easy to see that the inflation of an element e from stage i to stage $i+1$ will be

$$\sigma_e(i+1) = \frac{c_e(K-i) \cdot \frac{l(K)}{l(K-i)}}{c_e(K-i-1) \cdot \frac{l(K)}{l(K-i-1)}} = \frac{c_e(K-i)}{c_e(K-i-1)} \cdot \frac{l(K-i-1)}{l(K-i)}$$

First of all, observe that the subadditivity of the lease costs guarantees that the inflation ratio will be always larger than 1, which is a necessary assumption for the stochastic version to be meaningful (since purchasing elements at later stages should be more expensive). Furthermore, notice that in order to have uniform inflation ratio, i.e. the same inflation for any element e, it is necessary to have a uniform leasing problem.

Now, notice that the structure of the tree T allows us to associate each node of the tree with a corresponding interval of the leasing problem. In particular, the j-th node at depth d (node v) will be associated with the j-th interval of type $K-d$. Notice that we can associate j with the starting time t of this interval. Then, let us denote by $S'(v) = S_t(K-d)$ the set of elements purchased at that node and let $C'(v) = \sum_{e \in S_t(K-d)} p_e(d+1) = \frac{l(K)}{l(K-d)} \cdot C_t(K-d)$. Now, the objective function F_{Stoc} we want to optimize is

$$F_{Stoc} = C'_0(K) + \sum_{v \in V(T)} \mathbb{P}r[\text{particle reaches } v] \cdot C'(v)$$

The probability that the particle reaches a fixed node v at depth d is $\prod_{i=0}^{d-1} \frac{l(K-i-1)}{l(K-i)} = \frac{l(K-d)}{l(K)}$. Hence, for that node, we have that

$$\mathbb{P}r[\text{particle reaches } v] \cdot C'(v) = \frac{l(K-d)}{l(K)} \cdot \frac{l(K)}{l(K-d)} \cdot C_t(K-d) = C_t(K-d)$$

Then, the objective function becomes

$$F_{Stoc} = C_0(K) + \sum_{d=0}^{K-1} \sum_{t:(K-d)|t} C_t(K-d) = C_0(K) + \sum_{i=1}^{K} \sum_{t:i|t} C_t(i)$$

Consequently, the two objective functions coincide and both problems want to minimize the same function. This implies that an approximation algorithm for the stochastic version leads to an approximation algorithm for the leasing version with the same ratio multiplied by 4. □

The reduction can be extended for the case we consider randomized leasing policies, or in the case that the set of demands at each day is not fixed, but drawn at random from a specified probability distribution.

Chapter 4

Offline Leasing Problems

In this chapter, we present algorithms for several leasing problems in the offline setting. Since the standard variants of the problems we study are NP-complete, we do not expect to obtain polynomial time exact algorithms for the harder leasing variant. Instead, we propose approximation algorithms which run in polynomial time.

In the first section of the chapter, we will give approximation results to several leasing problems by reducing them to the corresponding multistage stochastic optimization problem and then applying known approximation algorithms to the stochastic version. This is a general approach to such problems [6]; however, the approximation factors we obtain are not always optimal.

In particular, the subsequent sections present an alternative approach which uses the powerful primal-dual method. Following this approach, Williamson and Nagarajan [50] designed an algorithm which achieves constant approximation ratio for Facility Leasing, in comparison with the $O(K)$ approximation ratio achieved through the reduction to the K-stage stochastic problem.

We show that their technique is applicable to other combinatorial problems as well. More specifically, we derive a constant approximation algorithm for Sum-Radii Leasing and improve the approximation ratio of the Vertex Leasing problem, by showing how to solve the more general Set Leasing problem. For ease of exposition, we will first present the technique applied to Set Leasing, since its application is simple, and then proceed to the other two problems, which are technically more demanding.

Based on the above results, it seems that we have one the following two situations. First, stochastic optimization may be a tool which is more powerful than we need in order to solve leasing problems. In this case, the question is whether we can find a general framework which captures the structure of leasing problems in a tighter way and thus allows us to

understand the problem better. The second case is that stochastic optimization perfectly captures leasing problems (so the two variants are essentially equivalent). This would imply that the algorithms we have for multistage stochastic optimization are far from optimal.

Moreover, it would be interesting to examine whether studying a problem in the leasing frameowrk implies that its approximability factor will grow. In Facility Leasing, the algorithm of [50] achieves exactly the same approximation ratio as the algorithm for the standard variant. However, much better approximation algorithms are known for Facility Location. Is it possible to adapt these algorithms so as to obtain an even better approximation ratio? The cases of Set Leasing and Sum-Radii Leasing are more interesting, since we lose a constant factor when we move to the leasing variant of the problem. Is this inherent to the leasing structure or can we match the approximation ratio?

4.1 Algorithms for Leasing Problems through Stochastic Optimization

We present approximation algorithms for Set Cover, Vertex Cover, Facility Leasing and Steiner Tree by reducing them to multistage stochastic optimization instances. More specifically, we apply the reduction theorem from Chapter 3 and thus reduce a leasing instance with K lease types to a corresponding instance of a K-stage stochastic optimization problem. Before applying the reduction, it is necessary to force a nested structure on the leases; this means that we lose a factor of 4 from the approximation ratio. Hence, an α-approximation algorithm for the corresponding K-stage stochastic optimization problem implies a 4α-approximation algorithm for the leasing variant.

For K-stage stochastic Set Cover, an $O(\ln n)$ approximation algorithm was presented in [58]. This directly implies a $O(\ln n)$ approximation algorithm for Set Leasing. For the special case of the Vertex Cover, [58] gives an approximation ratio of 2. Thus, for the Vertex Leasing we can achieve a constant approximation ratio of 8.

For the Facility Location problem, Swamy and Shmoys [59] obtained an $O(K)$ approximation algorithm for the K-stage stochastic version. Consequently, we can easily obtain an $O(K)$ approximation algorithm for the leasing version as well. Similarly, the $O(K)$ approximation algorithm from [40] for K-stage stochastic Steiner Tree implies an $O(K)$ approximation algorithm for the leasing version.

The above results can be summarized in the following table, which also presents the best approximation factors we have so far.

Problem	Approximation Ratio		
	Stochastic	Leasing (from Stochastic)	Best
Facility Leasing	$O(K)$	$O(K)$	3
Steiner Tree Leasing	$O(K)$	$O(K)$	$O(K)$
Set Leasing	$O(\ln n)$	$O(\ln n)$	$\min\{f, \ln n\}$
Vertex Leasing	2	8	6

4.2 Set Leasing

We will show how to approach the Set Leasing problem by applying the primal-dual schema. For the Set Cover problem, there exists a standard primal-dual algorithm which achieves an approximation ratio of α, where α is the frequency of the most frequent element [60, 10]. We apply here the same technique to solve the leasing variant.

Let us first present the primal linear program which describes this problem. We first introduce some useful notation. An element $e \in D_t$ is denoted by e_t. Let us also denote by c_S^k the cost for buying a lease of type k for set S (lease S_t^k). We use $x_{S,k}^t$ for the indicator variable which is set to 1 iff the set S is purchased at time t with a lease of type k. Moreover, $\mathcal{S}$ denotes the set of all possible leases. We can now write the primal program.

$$\begin{aligned} \min \quad & \sum_{S_t^k \in \mathcal{S}} x_{S,k}^t \cdot c_S^k \\ \text{subject to} \quad & \\ & \forall e_t \in U: \quad \sum_{S_t^{k'}: e_t \in S_t^{k'}, t \in I_{t'}^k} x_{S,k}^t \geqslant 1 \\ & \forall S_t^k \in \mathcal{S}: \quad x_{S,t}^k \in \{0,1\} \end{aligned} \tag{4.1}$$

The first inequality denotes that each element must be covered by a set which is in lease at the time when the element appears. The relaxation of this linear program is achieved when we allow the variables to assume any real value greater than or equal to zero. In order to express the dual program, we need to introduce the dual variables y_e^t. Then, the dual can be written as

$$\begin{aligned} \max \quad & \sum_{e_t \in U} y_e^t \\ \text{subject to} \quad & \\ & \forall S_t^k \in \mathcal{S} : \quad \sum_{e_{t'} : e_{t'} \in S_t^k, t' \in I_t^k} y_e^{t'} \leqslant c_S^k \\ & \forall e_t \in U : \quad y_e^t \geqslant 0 \end{aligned} \tag{4.2}$$

Algorithm. The algorithm works in a primal dual fashion. We initially set every dual variable y_e^t to zero and declare that every element is *uncovered*. Then, as long as uncovered elements exist, we perform the following steps. Let L be the set which contains the sets S_t^k we choose to lease, which is initially empty.

- We pick an arbitrary uncovered element e_t.
- We increase its dual variable until the dual constraint becomes tight for some set $S_{t'}^k$.We then add $S_{t'}^k$ and any other tight set to L and declare e_t covered. We also declare any other elements covered by these sets *covered*.

After the procedure has ended, all elements have been covered by some set in L. Thus, L is a valid set cover. The algorithm then performs one final step. The step starts with an empty set L'. For each set S, the algorithm examines the leases $S_t^k \in L$ in decreasing order of lease type. On considering a lease S_t^k, the algorithm deletes all intersecting leases in time (i.e. all leases $S_{t'}^{k'}$ such that $I_t^k \cap I_{t'}^{k'} \neq \emptyset$) and adds the sets $S_t^k, S_{t-l(k)}^k, S_{t+l(k)}^k$ to L'. Finally, the algorithm outputs L'.

Theorem 4.2.1. *The primal-dual algorithm for* Set Leasing *admits a* 3α *approximation ratio.*

Proof. First, notice that L is a feasible solution, since every element will be covered by the end of the algorithm. Moreover, L' maintains this feasibility, since every element in L covered by a deleted lease will be covered by the new leases we open in L'.

Denote by L_d the set L after having deleted the time-intersecting leases. Notice that the cost of the solution L' is 3 times the cost of L_d, since for each lease in L_d, solution L purchases 3 leases. We will thus focus on bounding the total cost of L_d. For any set $S_t^k \in L_d$, it holds that it is tight and hence $\sum_{e_{t'} : e_{t'} \in S_t^k, t' \in I_t^k} y_e^{t'} = c_S^k$. Thus, we have that

$$C_{L_d} = \sum_{S_t^k \in L_d} c_S^k = \sum_{S_t^k \in L_d} \sum_{e_{t'} : e_{t'} \in S_t^k, t' \in I_t^k} y_e^{t'} = \sum_{e_{t'}} y_e^{t'} \cdot \sum_{S_k^t : e_{t'} \in S_t^k, t' \in I_t^k} 1$$

The goal is to bound the number of sets which may be covering a single element. However, notice that there at most α different sets which cover a single element, since the element belongs in at most α sets and for each set there exists only one lease which covers that time (since L_d has no intersecting leases). Consequently, we have that

$$C_{L_d} \leqslant \alpha \cdot \sum_{e_t} y_e^t \leqslant \alpha \cdot C_{OPT}$$

Using the relation between primal, dual and optimal solutions for linear programs, we conclude that $C_{L'} \leqslant (3\alpha) \cdot C_{OPT}$ and the proof is complete. □

As we have previously shown, Vertex Leasing is a special case of Set Leasing, where it holds that $\alpha = 2$, since every element (edge) can be covered by at most two sets (vertices). We thus obtain the following corollary.

Corollary 4.2.2. Vertex Leasing *admits a* 6*-approximation algorithm.*

This is an improvement over the 8 approximation factor obtained by reducing Vertex Leasing to the multistage stochastic problem. It still remains an open question whether we can further improve this factor.

4.3 Sum-Radii Leasing

We will apply the primal-dual schema for Sum-Radii Leasing, following the approach from [23]. For simplicity, we will assume that the leases follow the simpler nested structure. This implies that we lose a 4-factor from the approximation ratio, but the goal of this section is to show that this problem admits a constant approximation factor, independent of the number of lease types K.

Let us first introduce some notation. We denote by C^r_{ikt} the lease of type k we purchase at time t for a cluster opening at point i with radius r. This cluster costs $c^k_i + r$. Let $\mathcal{C}$ be the set of all possible leases. Furthermore, I^k_t denotes the time interval $[t, t + l(k) - 1]$ and d^t_i denotes a demand point at i arriving at time t.

Let us next formulate the primal linear program for this problem. The variable x^r_{ikt} is the indicator variable of whether the lease C^r_{ikt} has been purchased.

$$\begin{aligned} \min \quad & \sum_{C^r_{ikt} \in \mathcal{C}} x^r_{ikt}(r + c^k_i) \\ \text{subject to} \quad & \\ & \forall d^t_j \in \mathcal{D}: \quad \sum_{C^r_{ikt'}: t \in I^k_{t'}, d(i,j) \leqslant r} x^r_{ikt} \geqslant 1 \\ & \forall C^r_{ikt} \in \mathcal{C}: \quad x^r_{ikt} \in \{0,1\} \end{aligned} \tag{4.3}$$

In order to obtain the linear relaxation of the primal program, we let y^r_{ikt} admit any value greater or equal to 0. As for the dual program, we need to introduce the dual variables y_{jt} for each demand d^t_j. We can now express the dual program.

$$\begin{aligned} \max \quad & \sum_{d^t_j \in \mathcal{D}} y_{jt} \\ \text{subject to} \quad & \\ & \forall C^r_{ikt} \in \mathcal{C}: \quad \sum_{d^{t'}_j: d(i,j) \leqslant r, t' \in L^k_t} y_{jt'} \leqslant r + c^k_i \\ & \forall d^t_j \in \mathcal{D}: \quad y_{jt} \geqslant 0 \end{aligned} \tag{4.4}$$

The algorithm we present here works in two phases. The first phase works in a primal-dual fashion, whereas the second phase manipulates the solution from the first phase to lower its cost, while maintaining feasibility.

First Phase. Initially, we set all the dual variables to zero, $y_{jt} = 0$. Next, we start increasing the dual variables in any arbitrary way, as long as they do not violate any dual constraint. Each time the first inequality of the dual program becomes tight, we declare the corresponding lease *tight* and add it to a set L, which is initially empty. The phase terminates when every point participates in a tight constraint, i.e. we can not increase the value of any dual variable without violating some constraint of the dual program.

Second Phase. The first phase outputs a set L of tight leases. The second step essentially performs a pruning procedure, which ensures that every point will contribute to a unique lease of a cluster. The new leases will be added to the initially empty set L'. We examine the leases in L in decreasing order of their lease type. Consider a tight lease $C^{r_0}_{i_0kt} \in L$. For this cluster, we find all leases in L which intersect it both in time and space. Consider the lease $C^{r_1}_{i_1k't'}$ with the largest radius among the intersecting leases. If

$r_0 < r_1$, we search again for the lease with the largest radius among the leases intersecting $C^{r_1}_{i_1 k' t'}$ (not including $C^{r_0}_{i_0 k t}$). This process constructs a sequence of clusters with increasing radius and ends when we find a lease $C^{r_m}_{i_m k'' t''}$ such that $r_{m-1} \geqslant r_m$ or until we cannot find any more intersecting leases.

Thus, we have constructed a sequence of leases $C^{r_0}_{i_0 k t}, \ldots, C^{r_m}_{i_m k_m t_m}$ such that $r_m > r_{m-1} > \ldots > r_0$. Based on whether m is odd or even, we distinguish two cases:

- m is even: We purchase the lease $C^{R_e}_{i_0 k t}$, where $R_e = r_0 + 4r_2 + \ldots + 4r_{m-2} + 6r_m$.
- m is odd: We purchase the lease $C^{R_o}_{i_0 k t}$, where $R_o = 3r_1 + 4r_3 + \ldots + 4r_{m-2} + 6r_m$.

We add to L' this lease and delete from L all the leases of the sequence, along with the intersecting leases. This process continues until L becomes empty.

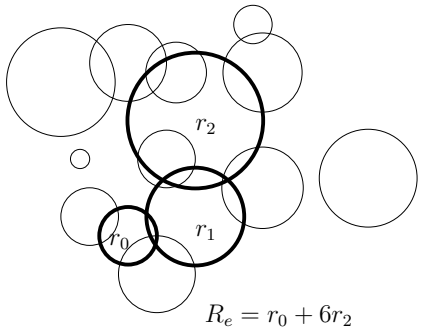


Figure 4.1: Constructing the sequence of clusters in the second phase. The radii are in increasing sequence: $r_2 > r_1 > r_0$. All the intersecting clusters will be removed and a cluster with radius R_e will open.

Lemma 4.3.1. *The solution L' is feasible. In particular, every demand belongs to at least one cluster which opens from a lease in L'.*

Proof. Every demand is covered by a cluster in L by the end of the first phase. Thus, it suffices to prove that every cluster is contained, both in time and space, in a cluster chosen in the final solution L'. Let us consider such a cluster $C^r_{ikt} \in L$ and consider the step of the second phase when it was examined. Assume that the sequence of clusters produced by this step is $C^{r_0}_{i_0 k t}, \ldots, C^{r_m}_{i_m k_m t_m}$ and that m is even (the odd case is similar).

First, since the structure of leases is nested, it is easy to see that every lease in the sequence or intersecting the sequence will belong in the interval I^k_t, hence the lease $C^{R_e}_{i_0 k t}$ will cover in time C^r_{ikt}. It remains to prove that C^r_{ikt} will be covered in space as well by

$C^{R_e}_{i_0kt}$. We can inductively prove that a cluster $C^{R_j}_{i_0kt}$ (j even), where $R_j = r_0 + 4r_2 + \ldots + 4r_j$ will cover all leases of the sequence $C^{r_0}_{i_0kt}, \ldots, C^{r_j}_{i_jk_jt_j}$, along with every lease which intersects any lease of the sequence apart from the last one.

Indeed, cluster $C^{r_0}_{i_0kt}$ trivially covers cluster $C^{r_0}_{i_0kt}$, hence the induction base holds. Now, assume that we know the proposition holds for the cluster $C^{R_j}_{i_0kt}$. Let us consider a cluster $C^{r'}_q$ intersecting $C^{r_j}_{i_jk_jt_j}$. This implies that there exists a point p which belongs to both clusters. By the induction assumption, p also belongs to $C^{R_j}_{i_0kt}$. Hence $d(p, i_0) \leqslant R_j$. Note also that, by the construction at the second phase, $r' \leqslant r_{j+1}$ and $r_{j+2} > r_{j+1} > r_j$. Thus, for any point p' in the cluster $C^{r'}_q$, it holds that $d(p', i_0) \leqslant d(p', q) + d(q, p) + d(p, i_0) \leqslant 2r_{j+2} + R_j$ using the triangle inequality. We can slightly extend this argument to show that every point p'' of $C^{r_{j+2}}_{i_{j+2}k_{j+2}t_{j+2}}$ has distance at most $4r_{j+2} + R_j$ from i_0.

This proves the proposition we need; the last $2r_m$ added to R_e corresponds to the covering of any cluster intersecting the last cluster of the sequence $C^{r_m}_{i_mk_mt_m}$, which is not covered by R_m. □

We partition the cost of L' into two parts: the cost C_R, which corresponds to the sum of the radii of the clusters we open, and C_F, which corresponds to the cost of buying the leases for the clusters. The following lemma charges these costs to the dual variables.

Lemma 4.3.2. $C_R + C_F \leqslant 7 \cdot \sum_{d^t_j} y_{jt}$

Proof. Let us first study the cost C_F. The cost we pay for the leasing of the clusters in L' is equal to the cost of a subset L_d of disjoint clusters in L, since for each cluster in L', all intersecting clusters are not considered. Thus, each dual variable pays exactly once for a cluster in L_d, and consequently at most once for the leasing of a cluster in L'. This implies that $C_R \leqslant \sum_{d^t_j} y_{jt}$.

The second step is to bound the cost C_R. Consider a cluster C^R_{ikt} picked in the final solution, where $R \leqslant 6(r_m + r_{m-2} + \ldots)$. Notice that the clusters which contribute to this sum are always disjoint by construction. Moreover, the clusters contributing to each lease in L' are again disjoint. Hence, C_R is at most the cost of a disjoint subset of clusters of L. For this subset of clusters, each demand pays at most once. Thus, $C_F \leqslant 6 \cdot \sum_{d^t_j} y_{jt}$.

Summing for the two cases, we conclude that $C_R + C_F \leqslant 7 \cdot \sum_{d^t_j} y_{jt}$. □

Applying standard facts from the theory of linear programming, we obtain the following theorem.

Theorem 4.3.3. *The primal-dual algorithm for* Sum-Radii Leasing *achieves a constant approximation ratio.*

The approximation ratio we obtain, although constant, is too large (28). It seems that the approximation ratio can be improved using a more detailed analysis; here, however, we focused on making the analysis as compact as possible.

4.4 Facility Leasing

In the previous section, we showed how to obtain an algorithm for the Facility Leasing problem with approximation ratio $O(K)$ using the reduction of leasing problems to multi-stage stochastic optimization. However, this is not the best we can do; we can construct a more specialized approximation algorithm which achieves a constant ratio, independent of the number of lease types K. The algorithm was presented by Nagarajan and Williamson [50] and is based on the classic primal-dual algorithm for Facility Location introduced by Jain and Vazirani [46].

We first describe the problem in the context of linear programming, that is, we present the primal program which solves Facility Leasing, we then give the LP-relaxation and finally the dual linear program of the relaxed one. For a detailed presentation of linear programming and the primal-dual schema, we refer the reader to [27, 60, 34].

Let us first introduce some useful notation. We denote by I_t^k the time interval of length $l(k)$ starting at time t. We also remind that the set of demands arriving at time t is denoted by $D_t \subseteq \mathcal{D}$. A demand at point j arriving at time t will be denoted by $d_j^t \in D_t$. Finally, let $f_i^k(t) \in \mathcal{F}$ be a facility at point i of type k which opens at time t. This facility costs c_i^k.

In order to express the linear program, we need to introduce some more notation. Let y_{ikt} be the indicator variable of the facility $f_i^k(t)$. This means that $y_{ikt} = 1$ if we open $f_i^k(t)$, otherwise it is 0. Furthermore, we denote by $x_{jt,ikt'}$ the variable which indicates whether the demand d_j^t is assigned to facility $f_i^k(t)$. We can now write down the primal linear program.

$$\min \sum_{f_i^k(t)\in\mathcal{F}} y_{ikt} \cdot c_i^k + \sum_{d_j^t\in\mathcal{D}} \sum_{f_i^k(t')\in\mathcal{F}:t\in I_{t'}^k} x_{jt,ikt'} \cdot d(i,j)$$

subject to

$$\begin{aligned}
&\forall d_j^t \in \mathcal{D}: && \sum_{f_i^k(t')\in\mathcal{F}:t\in I_{t'}^k} x_{jt,ikt'} \geqslant 1 && (4.5)\\
&\forall d_j^t \in \mathcal{D}, f_i^k(t') \in \mathcal{F}: && x_{jt,ikt'} \leqslant y_{ikt'} &\\
&\forall d_j^t \in \mathcal{D}, f_i^k(t') \in \mathcal{F}: && x_{jt,ikt'}, y_{ikt'} \in \{0,1\} &
\end{aligned}$$

The first inequality denotes that each demand must be assigned to at least one facility with duration that covers the time of arrival. Note that since the objective function must be minimized, each demand will be assigned to exactly one facility. The second inequality implies that, in order to assign a demand to some facility, this facility must be open. To obtain the LP-relaxation of the above linear program, it suffices to let the variables y_{ikt} and $x_{jt,ikt'}$ take any real value greater than or equal to 0.

For the dual program, we need to introduce the dual variables ν_{jt} and $w_{jt,ikt'}$, which correspond to the first and second inequality of the primal program respectively. Intuitively, ν_{jt} expresses how much the demand d_j^t *pays* towards the opening of facilities. Now we can write the dual linear program.

$$\max \quad \sum_{d_j^t \in \mathcal{D}} \nu_{jt}$$

subject to

$$\begin{aligned} &\forall d_j^t \in \mathcal{D}, f_i^k(t') \in \mathcal{F}: \quad \nu_{jt} - w_{jt,ikt'} \leqslant d(i,j) \\ &\forall f_i^k(t') \in \mathcal{F}: \quad \sum_{d_j^t \in \mathcal{D}} w_{jt,ikt'} \leqslant c_i^k \\ &\forall d_j^t \in \mathcal{D}, f_i^k(t') \in \mathcal{F}: \quad w_{jt,ikt'}, d_{jt} \geqslant 0 \end{aligned} \tag{4.6}$$

The algorithm for offline Facility Leasing proceeds in two phases. In the first phase, the algorithm works in a primal-dual fashion. By the end of the first phase, the algorithm has computed a feasible integer dual solution and a corresponding primal solution. The solution includes a set of facilities $\mathcal{F}_T$ which are *temporarily open* and the assigned facilities to each demand. However, this procedure alone could potentially output a solution with unacceptable cost. The second phase deals exactly with this case: the algorithm refines the set of facilities which will finally open and also determines the mapping from demands to the new set of facilities.

Let us now describe the algorithm in full detail.

First Phase. The algorithm starts with a feasible dual solution, where all ν_{jt} and $w_{jt,ikt'}$ are set to zero and the set $\mathcal{F}_T$ of temporarily open facilities is empty. Moreover, it initially declares all demands *unconnected*. Then, it starts increasing in a uniform way all dual variables ν_{jt} corresponding to unconnected demands. The dual variables $w_{jt,ikt'}$ behave so that it holds that $w_{jt,ikt'} = \max\{0, \nu_{jt} - d(i,j)\}$, i.e. they start increasing their value as soon as ν_{jt} equals $d(i,j)$ for some facility placed at point i. This

is necessary so as to ensure the feasibility of the dual program. The procedure continues until we come up with one of the following cases:

- It holds that $\nu_{jt} = d(i,j)$ for the demand d_j^t and some facility $f_i^k(t')$ such that $t \in I_{t'}^k$. Then, we say that the "edge" $(d_j^t, f_i^k(t'))$ becomes *tight*.
- The second inequality of the dual program becomes *tight* for some facility $f_i^k(t)$. Then, $f_i^k(t)$ becomes *temporarily open* and is added to the set $\mathcal{F}_T$.
- As soon as a temporarily open facility $f_i^k(t')$ is connected with a demand d_j^t through a tight edge, we declare the demand *connected* and stop increasing the dual variables ν_{jt}, $w_{jt,ikt'}$. We also say that demand d_j^t *contributes* to facility $f_i^k(t')$ iff $\nu_{jt} > d(i,j)$ and $t \in I_{t'}^k$.

The first phase continues until all demands are declared connected. Then, the algorithm proceeds with the second phase.

Second Phase. Let us consider the graph $G_T = (V, E)$, where V is the set $\mathcal{F}_T$ and two facilities are connected with an edge iff there is a demand contributing to both facilities. We next construct a maximal independent set of G_T, following the algorithm of Jain and Vazirani. However, for our case it does not suffice to consider an arbitrary maximal independent set. Instead, we have to construct the independent set by examining the facilities in order of decreasing lease length (or type equivalently) and by greedily choosing the next vertex of the independent set.

After constructing the independent set $\mathcal{F}_I$, the algorithm executes one more step to finalize the set of open facilities. Consider a facility $f_i^k(t) \in \mathcal{F}_I$. Instead of opening only the facility which starts at time t, the algorithm also opens the facilities $f_i^k(t - l(k))$ and $f_i^k(t + l(k))$. Hence, we end up with a set of open facilities which we denote by $\mathcal{F}'$.

Finally, the algorithm determines the mapping from each demand to a facility in $\mathcal{F}'$. Based on the assignment constructed by the end of phase one, we distinguish two cases for a demand d_j^t:

- The demand is connected to a facility in $\mathcal{F}_I$. The algorithm then keeps this assignment and we say that d_j^t is *directly* connected to this facility.
- The demand is connected to a facility $f_i^k(t') \in \mathcal{F}_T \setminus \mathcal{F}_I$. Then $f_i^k(t')$ does not belong to the set of facilities which the algorithm finally opens. However, since $\mathcal{F}_I$ is a maximal independent set, there must be some facility $f_{i'}^{k'}(t'')$ adjacent to $f_i^k(t')$ in

the graph T with the property that $k' \geqslant k$. It is always possible to find such a facility, following the fact that facilities where added to the independent set $\mathcal{F}_I$ in decreasing order of lease type.

This means that there exists a demand $d_j^{\bar{t}}$ such that it contributes to both facilities and also that $\bar{t} \in I_k^{t'}$ and $\bar{t} \in I_{k'}^{t''}$. Hence, the intervals $I_k^{t'}$ and $I_{k'}^{t''}$ intersect and since $I_{k'}^{t''}$ has at least the same length as $I_k^{t'}$ ($k' \geqslant k$), the interval $I_{k'}^{t''}$ is fully covered by the interval $[t'' - l(k'), t'' + l(k')]$. Consequently, one of the three facilities we open due to the facility $f_i^k(t') \in \mathcal{F}_I$ must cover d_j^t.

Thus, the second phase yields a feasible solution for the instance of the problem. The next step is to analyze the algorithm and bound the approximation ratio.

Analysis. The first step of the proof is to show that the dual variables ν_{jt} fully pay for the facility cost and the assignment cost. For this, we need to partition the contribution of each variable ν_{jt} to a contribution towards the opening of a facility ν_{jt}^f and a contribution towards the connection cost ν_{jt}^e, such that

$$\nu_{jt} = \nu_{jt}^f + \nu_{jt}^e$$

Note that if a demand is directly connected to a facility in $\mathcal{F}_I$, it will have contributed $\nu_{jt} - d(i,j)$ towards opening the facility; thus, we have that $\nu_{jt}^f = \nu_{jt} - d(i,j)$ and $\nu_{jt}^e = d(i,j)$. Let us denote by $D(f)$ the set of demands directly connected to facility f.

As for any demand d_j^t which is indirectly connected to a facility $f_i^k(t)$, it holds that the dual variable contributes only to the connection cost of the demand; hence, $\nu_{jt} = \nu_{jt}^e$ and $\nu_{jt}^f = 0$.

Now, observe that for each facility $f_i^k(t) \in \mathcal{F}_I$, the second inequality of the dual program is tight and hence

$$\sum_{d_j^t \in \mathcal{D}} w_{jt,ikt'} = c_i^k$$

However, $w_{jt,ikt'} > 0$ only for the demands directly connected to this facility. In this case, we also have that $w_{jt,ikt'} = \nu_{jt} - d(i,j) = \nu_{jt}^f$. We thus conclude that

Lemma 4.4.1. *For any facility $f_i^k(t) \in \mathcal{F}_I$, we have that*

$$\sum_{d_j^t \in D(f_i^k(t))} \nu_{jt}^f = c_i^k$$

It is now straightforward to charge the dual variables with the total facility cost of the final solution $\mathcal{F}'$.

Lemma 4.4.2. *It holds that*

$$\sum_{f_i^k(t)\in\mathcal{F}'} c_i^k \leqslant 3\cdot\sum_{d_j^t} v_{jt}^f$$

Proof. Since each facility in $\mathcal{F}_I$ leads to the opening of 3 equal cost facilities for the final solution, we have that

$$\sum_{f_i^k(t)\in\mathcal{F}'} c_i^k = 3\cdot\sum_{f_i^k(t)\in\mathcal{F}_I} c_i^k = 3\cdot\sum_{f_i^k(t)\in\mathcal{F}_I}\sum_{d_j^t\in D(f_i^k(t))} v_{jt}^f \leqslant 3\cdot\sum_{d_j^t\in\mathcal{D}} v_{jt}^f$$

where the last equality is derived from lemma 4.4.1. □

The next lemma charges the dual variables with the assignment cost of the final solution.

Lemma 4.4.3. *For any demand* d_j^t *connected to facility* $f_i^k(t')$*, it holds that* $d(i,j)\leqslant 3\cdot v_{jt}^e$.

Proof. For a directly connected demand d_j^t, we have that $d(i,j)=v_{jt}^e\leqslant 3\cdot v_{jt}^e$. The difficult case is when the demand d_j^t is indirectly connected to a facility $f_i^k(t')$. Let $f_{i'}^{k'}(t'')$ be the temporarily open facility to which the demand was assigned at the end of the first phase. Since $f_i^k(t')$ and $f_{i'}^{k'}(t'')$ are adjacent facilities in the graph T, there must be a demand $d_{\bar{j}}^{\bar{t}}$ which contributes to both of them. Then, using the triangle inequality, we get that $d(j,i)\leqslant d(j,i')+d(i',\bar{j})+d(\bar{j},i)$.

First, note that since d_j^t was connected to $f_{i'}^{k'}(t'')$, we have that $d(j,i')\leqslant v_{jt}=v_{jt}^e$. In the same fashion, it holds that $d(i',\bar{j})\leqslant v_{\bar{j}\bar{t}}^e$ and $d(\bar{j},i)\leqslant v_{\bar{j}\bar{t}}^e$. It thus remains to show that $v_{\bar{j}\bar{t}}\leqslant v_{jt}$.

Consider the moment when $f_{i'}^{k'}(t'')$ becomes temporarily open. Since d_j^t is connected to this facility, its dual variable will be increasing at least until that moment. However, $d_{\bar{j}}^{\bar{t}}$ contributes to both facilities, hence it is the case that as soon as one of these is declared open, its dual variable will stop increasing. We can now easily infer that $v_{\bar{j}\bar{t}}\leqslant v_{jt}$. □

We can now sum up the cost of the dual variables. We denote by $\phi(d_j^t)$ the facility to which d_j^t is assigned.

Theorem 4.4.4.

$$\sum_{d_j^t\in\mathcal{D}} v_{jt}\geqslant 3\cdot\sum_{f_i^k(t)\in\mathcal{F}'} c_i^k + 3\cdot\sum_{\phi(d_j^t)=f_i^k(t)} d(i,j)$$

By setting $y_{ikt} = 1$ when facility $f_i^k(t)$ is open (otherwise 0) and $x_{jt,ikt'} = 1$ when demand d_j^t is assigned to facility $f_i^k(t')$ (else 0), we get that

$$C_{OPT} \geqslant C_{DUAL} = \sum_{d_j^t \in \mathcal{D}} \nu_{jt} \geqslant 3 \cdot C_{ALG}$$

Theorem 4.4.5. *The primal-dual algorithm achieves a 3 approximation ratio for the offline* Facility Leasing *problem.*

It is interesting that this algorithm achieves the same approximation ratio with the algorithm of Jain and Vazirani, even if leasing actually generalizes the problem. It remains an open question whether it is possible to deploy better approximation algorithms designed for Facility Location to obtain an even better approximation algorithm for Facility Leasing.

Chapter 5

The Parking Permit Problem

In this chapter, we will present the Parking Permit problem, which will function as a transition from offline to online leasing problems. As we will later see, Parking Permit can be considered as the simplest form of a leasing problem. Although it is straightforward to solve its offline version, the online version is much more interesting and will provide a cornerstone for analyzing leasing problems with a more complicated structure.

The problem is motivated by the following real-life scenario. Let us assume that the office we work is nearby and we prefer to walk to work instead of driving at sunny days. However, when it rains, we always use the car. In this case, we must always have a valid parking permit in order to park our car in the nearby parking. There are various types of parking permits with different durations: day, week, month or even year. Naturally, we have to pay more for a parking permit with long duration. However, the cost per day is less if we buy a longer parking permit.

Our goal is to choose which types of permits we need to purchase and when so that every driving day is covered by a parking permit. Furthermore, we want to find the solution which minimizes the total cost is. It is easy to see that this problem falls into the category of leasing problems. If we know the rainy days in advance, there exists a simple dynamic programming solution for the problem. Hence, contrary to the other leasing problems we have studied, the offline version of Parking Permit is easy to solve. However, the problem becomes much harder if we assume that we do not know in advance the rainy days and we have to make the best decision based only on the information we are given so far. Thus, we focus on the *online* version of the problem.

The Parking Permit problem was presented by Meyerson in [49] and it is the first paper which introduces the leasing model. The setting is as simple as possible, since the notion of space is not involved; we are interested only in time. Clearly, any infrastructure

leasing problem where the metric space is a single point is equivalent to Parking Permit. Thus, the study of the Parking Permit, although trivial for the offline case, provides important intuition for the online case.

First, we describe the formal definition of the problem. Then, we examine deterministic and randomized online algorithms for the Parking Permit. We present a deterministic algorithm with competitive ratio $O(K)$ and prove that it is the best possible ratio we can hope for in the deterministic setting. Finally, we show how randomization helps us improve the competitive ratio to $O(\log K)$ and prove its optimality.

5.1 Definition

The online Parking Permit problem can be described in detail as follows.

Definition 5.1.1 (Parking Permit)**.** *We are given* K *different types of parking permits. Permit* k *has a duration of* $l(k)$ *days and costs* c_k*. We are given a schedule of* T *days with marked driving days; this schedule is revealed one day at a time. Our goal is to select a set of permits so as to cover all driving days and minimize the total cost of permits purchased.*

Since Parking Permit follows the general structure of the leasing framework, we may consider the simple nested structure of the problem, with losing only a constant factor in the solution cost. In particular, we may assume that the problem has the following structure.

- For each permit type $1 < k \leqslant K$, we have that $c_k \geqslant 2 \cdot c_{k-1}$ and $l(k) \geqslant l(k-1)$
- There is exactly one permit of type k which can possibly cover a specific day
- Each permit of type k has exactly $\frac{l(k)}{l(k-1)}$ permits of type $k-1$ embedded within

5.2 A Deterministic Approach

We first turn our attention to deterministic online algorithms. The algorithm we present here (Simple-Online) [49] is a generalization of the deterministic online algorithm for the Ski-Rental problem [47, 14]. The algorithm proceeds essentially in a primal-dual fashion, by purchasing a longer lease type as soon as it discovers that buying the shorter permits was at least as expensive as buying the longer one.

Notice that since we deal with the interval model, the algorithm buys a permit of type k as soon as the total cost of the permits of type $k-1$ which cover the driving days in the

Algorithm 1: SIMPLE-ONLINE

- We start by purchasing a permit of type 1 for a single driving day.
- For any interval of type k, as soon as the optimum solution would purchase this permit, we purchase it as well .

interval becomes greater than c_k. Hence, the algorithm can be described as conservative, since it purchases a permit only when it is completely sure that the optimum solution would do the same.

While the corresponding online deterministic algorithm for the SKI-RENTAL problem admits a constant competitive ratio, the competitive ratio for SIMPLE-ONLINE depends on the number of types of permits K. The following theorem proves this:

Theorem 5.2.1. *The algorithm* SIMPLE-ONLINE *is* $O(K)$*-competitive.*

Proof. We will use induction to prove that the online algorithm pays at most k times the cost of the optimal solution (OPT) during an interval of type k. For $k = 1$, clearly the online algorithm makes the optimal choice. For any interval of type $k > 1$, we distinguish two cases:

- OPT **does not buy a permit of type** k**:** Then, OPT must cover each sub-interval of type $k-1$ separately. For each sub-interval, we pay at most $(k-1)$ times the optimum (from the induction hypothesis). Thus, for the whole interval we pay at most $(k-1)C_{OPT}$.
- OPT **buys a permit of type** k**:** Then, OPT pays c_k. If we consider the days of the interval in reverse order, there exists some day where OPT would not buy the whole permit, but pay separately for each sub-interval. Until then, the online algorithm pays at most $(k-1) \cdot c_k$ (this follows from the same argument as in the first case). Then, since OPT buys permit k, the algorithm buys it too. Thus, the algorithm pays at most $(k-1) \cdot c_k + c_k = k \cdot c_k$.

This completes the induction and the proof. □

A natural question is whether there exists a deterministic algorithm which can do better than the above naive algorithm. The following theorem answers this question negatively. The theorem was proved in [49], but we will prove the lower bound in a slightly different way, which will allow us to generalize it easily in the next chapter.

Theorem 5.2.2. *No deterministic online algorithm for the* Parking Permit *problem admits a competitive ratio better than* $\Omega(K)$.

Proof. The idea behind the lower bound is to force the online algorithm ALG to buy a large number of smaller permits before realizing that a longer permit should be bought from the start. The adversary has a very simple adaptive strategy: if ALG has no valid permit covering a day, mark this day as a driving day. For the lower bound, we consider the interval model and we assume that we have K different permits with costs $c_i = 2^i$ and durations $l(i) = 3^i$.

We say that an interval is *active* if it has contains at least one driving day. Let us also denote by n_i the number of intervals of type i covered by ALG by a permit i (*good* intervals), and by b_i the number of intervals i which ALG covers by using smaller permits (let us call them *bad* intervals). It is clear that $C_{ALG} = \sum_{i=1}^{K} n_i \cdot c_i$. In order to get a bound on the cost OPT pays, we note that it would be a feasible solution to buy a permit of type i for every active interval of type i. However, we have b_i bad active intervals and at most $\sum_{j \geqslant i} n_j$ good intervals . Thus, the number of active intervals of type i is bounded by $b_i + \sum_{j \geqslant i} n_j$. Thus, we have that $C_{OPT} \leqslant 2^i \cdot (b_i + \sum_{j \geqslant i} n_j)$ for any i.

Now, by summing for $i = 1, \ldots, K$, we have that

$$K \cdot C_{OPT} \leqslant \sum_{i=1}^{K} \left(2^i \cdot (b_i + \sum_{j \geqslant i} n_j) \right) \leqslant \sum_{i=1}^{K} b_i \cdot 2^i + \sum_{i=1}^{K} 2^i \cdot \sum_{j \geqslant i} n_j$$

Next, we can manipulate part of the right hand side of the equation as follows.

$$\begin{aligned} \sum_{i=1}^{K} 2^i \cdot \sum_{j \geqslant i} n_j = \sum_{i=1}^{K} n_i \cdot \sum_{j=1}^{i} 2^j &\leqslant \sum_{i=1}^{K} n_i \cdot 2^{i+1} \\ &= 2 \cdot \sum_{i=1}^{K} n_i \cdot c_i = 2 \cdot C_{ALG} \end{aligned}$$

In order to conclude the proof of the lower bound, it remains to bound the quantity $R = \sum_{i=1}^{K} b_i \cdot 2^i$. Let us fix some permit type $i > 1$ and consider any bad interval of type i. Then, all 3 sub-intervals of type $i-1$ are active and covered by permits of smaller type. Hence, we have at most $b_{i-1} + n_{i-1}$ such sub-intervals. It follows immediately that $3 \cdot b_i \leqslant b_{i-1} + n_{i-1}$ for any $i > 1$. Thus, we have that $(3/2) \cdot b_i \cdot 2^i \leqslant (b_{i-1} + n_{i-1}) \cdot 2^{i-1}$. Now, we can sum for $i = 2, \ldots, K$

$$\frac{3}{2} \cdot \sum_{i=2}^{K} b_i \cdot 2^i \leqslant \sum_{i=2}^{K} b_{i-1} \cdot 2^{i-1} + \sum_{i=2}^{K} n_{i-1} \cdot 2^{i-1}$$

Since $b_1 = 0$, we have that

$$\frac{3}{2} \cdot R \leqslant \sum_{i=1}^{K-1} b_i \cdot 2^i + \sum_{i=1}^{K-1} n_i \cdot 2^i \leqslant R + C_{ALG}$$

Thus, we have that $R \leqslant 2 \cdot C_{ALG}$ and consequently

$$K \cdot C_{OPT} \leqslant 4 \cdot C_A$$

This concludes the proof of the lower bound. The original proof in [49] achieves a $K/3$ lower bound, but it assumes that each interval contains a much bigger number of sub-intervals ($2K$ instead of 3 for our case). □

5.3 Randomized Algorithms

Since deterministic algorithms can not achieve competitive ratio better than $\Omega(K)$, it is natural to ask whether randomization leads to more competitive online algorithms. Indeed, using randomization, we can achieve a much lower competitive ratio of $O(\log K)$. We follow a common approach to designing online randomized algorithms. Instead of dealing directly with randomization, we will use an equivalence between randomized and fractional algorithms and describe a deterministic online algorithm for the fractional case. The latter algorithm is inspired by a recent general-purpose technique for designing algorithms for online problems [2, 3, 16].

Let us first introduce the fractional version of the Parking Permit problem. A fractional algorithm may choose to buy a permit fractionally, for example we may buy 1/3 or 1/2 of a permit. We must ensure though that for each driving day the sum of the permits is at least 1. A fractional algorithm can be viewed as the linear relaxation of the integer linear program behind the Parking Permit. The key point is that randomized and fractional algorithms for the parking permit problem are equivalent with respect to the competitive ratio. The following theorem proves this equivalence by presenting a randomized rounding procedure (see [60] for more details on randomized rounding).

Theorem 5.3.1. *[49] There exists a randomized online algorithm for the* Parking Permit *problem with competitive ratio $\Theta(a(K))$ iff there exists a deterministic fractional algorithm with competitive ratio $\Theta(a(K))$.*

Proof. The intuition behind this equivalence is that we can view the fractional values of the permits as an indication of the probability with which a permit should be purchased. The one direction is quite straightforward; if we have an online randomized algorithm,

we may set design an online fractional one by setting the fractional value of each permit equal to the probability that the permit is purchased. Clearly, the cost of the fractional algorithm is equal to the expected cost of the randomized algorithm.

The other direction describes how to round the fractional solution. The randomization we use is restricted only to choosing a threshold τ uniformly at random between 0 and 1. As the fractional algorithm runs, the fractional values of each permit may increase. Consider a day t, where we will have K possible permits to choose from. Denote by $F_k(t)$ the fraction of the permit of type k which covers day t. By the feasibility of the fractional algorithm, we are guaranteed that $\sum_k F_k(t) \geqslant 1$. For day t, the randomized algorithm buys the permit k such that

$$\sum_{i=k+1}^{K} F_i(t) < \tau \leqslant \sum_{i=k}^{K} F_i(t)$$

Notice that this ensures the feasibility of the randomized solution: there will always be such a permit of some type which satisfies the above inequality.

In order to compute the expected cost of the algorithm, consider an interval $[t, t + l(k) - 1]$ of type k. The randomized algorithm may attempt to buy the permit at any day in this interval. However, it buys the permit i only when for some day t', it holds that $\tau > \sum_{i=k+1}^{K} F_i(t') \geqslant \sum_{i=k+1}^{K} F_i(t)$. Similarly, we must have that $\tau \leqslant \sum_{i=k}^{K} F_i(t') \leqslant \sum_{i=k}^{K} F_i(t + l(k) - 1)$. Note that the probability that the permit is purchased is bounded by the probability that τ falls between these two values, which is exactly $\sum_{i=k}^{K} F_i(t + l(k) - 1) - \sum_{i=k+1}^{K} F_i(t)$. Hence, the expected cost we pay for this permit is

$$c_k \cdot \sum_{i=k}^{K} F_i(t + l(k) - 1) - c_k \cdot \sum_{i=k+1}^{K} F_i(t)$$

We can now sum over all intervals of type k and over all types of permits. We now need to exploit the following two facts: the first and last days of neighboring intervals coincide, and $c_{i+1} \geqslant 2c_i$. Using these facts, we can manipulate the equation and obtain that the total expected cost of the algorithm will be at most twice the cost of the fractional solution. □

We now present a fractional online algorithm for the Parking Permit problem (Online-Random). Let $F_i(t)$ denote the fractional value of the unique permit of type i which covers day t.

A first important observation is that we only need a finite sequence of operations in order to reach a feasible solution. The analysis of the algorithm implies the following theorem.

Algorithm 2: ONLINE-RANDOM

Initially, all fractional permits are set to zero ;

When we need to drive and the sum of the permits covering the day is less than one, we perform an *operation*, which consists of the following two steps:

1. $\forall\, 1 \leqslant i \leqslant K$, multiply $F_i(t)$ with $1 + \frac{1}{c_i}$
2. $\forall\, 1 \leqslant i \leqslant K$, add $\frac{1}{K \cdot c_i}$ to $F_i(t)$

until we achieve feasibility for the specific day.

Theorem 5.3.2. *[49] The fractional algorithm* ONLINE-RANDOM *has* $O(\log K)$ *competitive ratio.*

Proof. In order to prove the theorem, we will first show that each operation has constant cost. Then, we will show that we can bound the total number of operations involved in a specific time interval of the optimal solution.

Indeed, if we perform an operation at day t, then the sum of the permits at that day must be strictly less than 1, so it holds that $\sum_{i=1}^{K} F_i(t) < 1$. The first step of each operation increases the fraction of permit i by $F_i(t)/c_i$, whereas the second step increases the fraction by $1/K \cdot c_i$. By summing for all permit types, we conclude that the total increase of the cost is at most

$$\sum_{i=1}^{K} c_i \cdot \left(\frac{F_i(t)}{c_i} + \frac{1}{K \cdot c_i} \right) = \sum_{i=1}^{K} F_i(t) + 1 < 2$$

We next bound the number of operations performed. For this, consider an interval of type i where the optimal solution OPT buys a permit i and pays c_i. We will calculate the cost of the fractional algorithm for the same interval. After c_i operations, the second step guarantees that the fraction of the permit i is at least $1/K$. From then on, the first step of each operation multiplies the fraction by the factor $1 + \frac{1}{c_i}$. Thus, after $O(c_i \log K)$ operations, the fractional value of the permit is larger than 1, which implies that the algorithm buys the whole permit. Consequently, any other driving day during this interval will be covered and no more operation will be performed until the end of the interval. Since each operation costs at most 2, we pay $O(\log K)$ times more than OPT for the specific interval.

By summing for each interval where OPT buys a permit, we conclude that the competitive ratio of the fractional algorithm is at most $O(\log K)$. □

Why does SIMPLE-ONLINE perform so well? One important reason is that the algorithm is not *memoryless*, i.e. decisions for the present day influence decisions on future days. This is achieved by incrementing the fractional values of all permits for any day considered. By investing a small fraction to larger permit types, we make sure that the algorithm will realize that it needs to purchase a longer permit before it pays too much. In fact, a memoryless approach is bound to fail, even when we use randomization [49].

It can be proved that ONLINE-RANDOM achieves the best possible competitive ratio within randomized algorithms.

Theorem 5.3.3. *[49] Any randomized online algorithm for* PARKING PERMIT *has expected competitive ratio at least* $\Omega(\log K)$.

Proof. In order to prove the lower bound, we will construct a randomized instance of the PARKING PERMIT problem and show a bound on the expected competitive ratio of the best deterministic algorithm. We can then apply Yao's minimax principle [62] to derive the lower bound for randomized algorithms.

Let us assume that we have K permit types and permit i costs $c_i = 2^i$. The instance will be nested and we will assume that the duration of a lease of type i will be arbitrarily larger than the duration of lease type $i-1$. We construct randomized instances of our problem using *active* intervals. An active interval has the following property: the i-th sub-interval of the interval is active with probability 2^{i-1}. Notice that this means that the first sub-interval will always be active. Moreover, the top-level interval is always active. The base of this recursive definition is that an active interval corresponding to type 1 has a driving day.

Let us first compute the expected cost of any deterministic algorithm on this randomized sequence of driving days. Suppose that the algorithm has to make a choice about whether or not to buy a permit of type k for an active interval. Buying a permit of type k costs $c_k = 2^k$. However, the algorithm may choose to pay separately for each sub-interval of type $k-1$ which will be active. In this case, the algorithm pays an expected cost of $\sum_{i=0}^{l(k)-1} \frac{2^{k-1}}{2^i} = 2^{k-1} \sum_{i=0}^{l(k)-1} 2^{-i} < 2^k$. This means that the algorithm pays in expectation less by preferring to buy smaller permits. We may repeat this argument for any type of permit and conclude that the best strategy for the algorithm is to buy only type 1 permits whenever a driving day appears. Hence, the expected cost of the algorithm will be the expected number of driving days of the instance.

Let us denote by r_k the expected number of active intervals at level k. Then, we have that $r_k = r_{k-1} \cdot \sum_{i=0}^{l(k)-1} 2^{-i}$. By assuming that $l(k)$ is arbitrarily large for any k, we can prove that $r_k \approx 2r_{k-1}$. Moreover, $r_1 = 1$. This implies that $r_K \approx 2^K$.

Can we find an upper bound for what the optimum offline solution will pay in expectation? Consider the following strategy: we buy a permit of type k for an interval only if this interval includes at least $\log \mathsf{k} + 1$ active sub-intervals. Using again recursion to describe the number of active intervals at each level, we can inductively prove that the expected cost of the algorithm will be upper bounded by $\frac{2^{\mathsf{K}+1}}{\log \mathsf{K}}$. Hence, the optimum offline algorithm pays at least $\mathsf{O}(\log \mathsf{K})$ times less in expectation than any deterministic algorithm. This concludes the proof. □

Chapter 6

Online Sum-Radii Clustering

In a clustering problem, the goal is to partition points into sets, which we call *clusters*, so as to minimize an objective function. The typical setting is to assign a cost to each cluster and let the objective function be the sum of the cluster costs.

In our case, we will assume that each cluster has a fixed opening cost plus a cost related to the size of the cluster, which might be the radius or the diameter of the cluster. Other models proposed by Charikar et al. [20], Chan and Zarrabi-Zadeh [19, 63] assume that we are allowed only clusters of a fixed size and cost, with which the entire point set must be covered. In this case, the objective function is just the number of clusters opened.

In the online variant of this problem, the request points appear one by one and have to be assigned to clusters as soon as they arrive. In order to serve a new point, we may either open a new cluster, assign the point to an existing cluster, or even expand an existing cluster so as to include the new point. However, it is not possible to delete an existing cluster, merge two clusters into one, or split a cluster into several smaller ones.

We will study the problem in the case where the requests are points of a metric space. Csirik et al. [25] initiated the study of this variant; however, they studied only on the restriction of the problem to the real line. They assumed that clusters may be opened anywhere on the line and that each cluster pays a fixed cost for set-up plus its diameter. Their goal was to minimize the sum of the cost of the clusters. On this framework, they considered several variants, which differ in the way the clusters open: (a) a cluster must be fixed when it is initialized (b) the algorithm can shift the cluster or expand it, as long as it contains all points already assigned to it and (c) the diameter is fixed in advance while the exact location can be modified.

In the following sections, we will first define the problem formally. Then, we will prove the equivalence of several online variations. Next, we will present a primal-dual

deterministic algorithm and show its optimality. Finally, we prove a randomized lower bound and give a fractional algorithm which improves the competitive ratio obtained in the deterministic case.

6.1 Description of the Model

We are given a metric space with a distance function $d(i, j)$. The requests are points of this metric space, and arrive one after the other. Each request point must be covered by a cluster; this means that the distance of the point to the center of the cluster is at most the cluster radius. A cluster is centered at some point p and is allowed to have any radius r; we denote such a cluster by $C(p, r)$.

A cluster with radius r costs a fixed set-up cost f plus its radius r. Note here that we measure the cost of the cluster based on its radius instead of its diameter; both measures are essentially equivalent, since a c competitive algorithm for the radius variant yields a $(2c)$-competitive algorithm for the diameter variant.

We will also assume that a cluster may be centered at any point of the metric space. This is not a necessary assumption; we could as well request that the clusters are centered only at locations of request points and this would mean that we would lose a multiplicative factor of 2 on the competitive ratio. The argument is quite simple; assume that an algorithm opens a cluster $C(p, r)$ centered at an arbitrary point p. Clearly, the cluster contains some request point q and $d(q, p) \leqslant r$. If we open the cluster $C(q, 2r)$, then any request point in the cluster $C(p, r)$ will be covered and we pay at most twice the cost of the original cluster.

However, it yet remains to be specified how the clusters open in the online setting. Here we consider several scenarios, based on the models proposed in [25].

- Fixed-Cluster: when a new cluster opens, we specify its center and radius, which will remain fixed throughout the algorithm. Only points which arrive inside the cluster may be assigned to it.

- Fixed-Radius: when we open a new cluster, we only fix its radius. This means that we may move its center when new points arrive; however, the cluster must still cover all the points originally assigned to it.

- Flexible-Cluster: when we open a new cluster, we only specify the points that are covered by it. This means that its radius may increase and its center may change when new requests arrive. The cluster costs only its final radius plus the set-up cost.

It is clear that a c-competitive online algorithm for the Fixed-Cluster model will also give a c-competitive algorithm for Fixed-Radius and Flexible-Cluster. However, as the following theorem suggests, all three models are equivalent regarding the competitive ratio within a constant factor.

Proposition 6.1.1. *A c-competitive algorithm for the* Flexible-Cluster *or the* Fixed-Radius *model gives a $O(c)$-competitive algorithm for the* Fixed-Cluster *model.*

Proof. Let us assume a c-competitive algorithm A_R for the Fixed-Radius model. Now, consider the following algorithm A_S for the Fixed-Cluster model. When A_R decides that some point p will be covered by a cluster with radius r, A_S opens the cluster $C(p, 2r)$, which costs at most twice the cluster $C(q, r)$ which A_R will finally open. However, note that each point i of the cluster $C(q, r)$ will be also served by the cluster $C(p, 2r)$, since $C(q, r)$ covers p and thus $d(i, p) \leqslant d(i, q) + d(q, p) \leqslant r + r = 2r$. Thus, we obtain a $2c$-competitive algorithm for the Fixed-Cluster model.

Now, let us assume a c-competitive algorithm A_C for the Flexible-Cluster model. We will construct an algorithm A_S which works for the Fixed-Cluster model. Let us fix a cluster C which A_C finally opens with center at some point p and radius r_m (the radius may increase when new points are added to the cluster). The algorithm A_C pays $f + r_m$. As for A_S, we will construct a sequence of clusters $\mathcal{C}$ which follow the progress of C and cover the same points.

Initially, when the first request point q covered by C arrives, we open the cluster $C(q, f)$ (if A_C opens a cluster with radius $\leqslant f$). Then, whenever A_C covers a point not covered by any cluster of $\mathcal{C}$, we keep opening clusters by doubling their radius until the new point is covered. Moreover, let us assume that A_S will perform k such doublings. Thus, A_S pays $\sum_{i=0}^{k} f \cdot (2^i + 1) \leqslant f \cdot \sum_{i=0}^{k} 2^{i+1} = 2 \cdot f \cdot (2^{k+1} - 1) \leqslant 4 \cdot f \cdot 2^k$.

However, notice the distance between q and any request point p' covered by C is at most $d(q, p') \leqslant d(q, p) + d(p, p') \leqslant r_m + r_m = 2r_m$. Furthermore, due to the doubling procedure, A_S will open at most a cluster with radius $2 \cdot (2r_m) = 4r_m$. Thus, it holds that $4r_m \geqslant 2^k f$. Consequently, A_S pays at most $4 \cdot 4 \cdot r_m \leqslant 16(r_m + f)$. □

Since the models are equivalent within a constant, we will assume throughout the chapter that we follow the stricter model, namely the Fixed-Cluster model.

6.2 Equivalent models

We first prove a proposition which allows us to simplify the structure of the online problem.

Proposition 6.2.1 (Doubling Radii)**.** *We may assume that the radius of a cluster is restricted only to values of the form* $2^i \cdot f$*, where* $i \geqslant 0$ *is an integer, without losing more than a multiplicative factor of two in the competitive ratio.*

Proof. Let us consider an optimal solution OPT which opens a set of clusters $\mathcal{C}$ and costs $C_{OPT} = \sum_{i \in \mathcal{C}} (f + r_i)$, where we denote by r_i the radius of cluster i. Now, consider an alternative solution OPT′ where the centers of the clusters in $\mathcal{C}$ are maintained, but the corresponding radii are set to $r'_i = 2^k \cdot f$, where $2^{k-1} \cdot f < r_i \leqslant 2^k \cdot f$ and $k = 1, 2, \ldots$. For $r_i \leqslant f$, we take $r'_i = f$. In practice, this means that we round the radii up to the closest power of two.

Clearly, for any $r_i > f$, it holds that $r'_i < 2r_i$. In the case that $r_i \leqslant f$, the optimal solution pays at least a cost of f for the cluster i, while OPT′ pays at most $2f$. In any case, OPT′ pays at most double what OPT pays for any cluster i. Summing for all clusters, we conclude that OPT′ pays at most twice as much as OPT. □

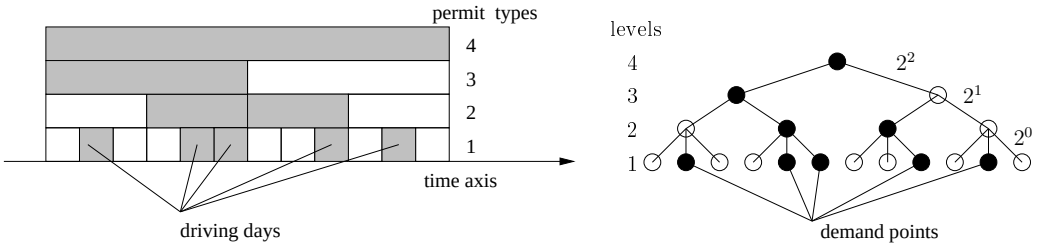
We next show a correspondence between the Parking Permit problem and the Sum-Radii Clustering problem. We consider the *interval* version of the Parking Permit problem, together with the assumption that the costs for each type are rounded up to powers of two. This loses only a constant factor from the competitive ratio of the problem. The following theorem establishes the exact connection between these two problems.

Theorem 6.2.2. *Consider an instance* $\mathcal{I}$ *of the* Parking Permit *problem. Then, there exists an instance* $\mathcal{I}'$ *of the* Sum-Radii Clustering *problem such that any solution of* $\mathcal{I}$ *can be mapped to a solution of* $\mathcal{I}'$ *of equal cost and vice versa.*

Proof. Let us assume that $\mathcal{I}$ has K types of permit with corresponding costs $c_1, c_2, \ldots, c_K$ such that w.l.o.g. $c_i = 2^{i-1}$ $(i = 1, \ldots, K)$ and durations $D_1, D_2, \ldots, D_K$. We will construct a tree metric $\mathcal{T}$ which comprises the metric space of the Sum-Radii Clustering instance $\mathcal{I}'$. The tree T corresponding to $\mathcal{T}$ has K levels and level i $(i = 1, \ldots, K)$ includes all nodes at depth $K - i$. Clearly, the root is at level K, whereas all the nodes at level 1 are the leaves of T. A node at level i has exactly D_i / D_{i-1} children. The opening cost of any cluster will be 1. Finally, the distance between a node at level i and its child at level $i - 1$ is exactly $c_{i-1} = 2^{i-2}$.

First, we have to prove that the tree metric satisfies the properties of a metric space. Indeed, it is easy to see that $\mathcal{T}$ corresponds to a 2-HST (Hierarchically Separated Tree), since the distances at every path from the root to the leaves decrease by a factor of 2 at every level and a node has equal distance to all its children.

Next, let us describe the one-to-one mapping of a solution $S_{\mathcal{I}}$ of $\mathcal{I}$ to a solution $S_{\mathcal{I}'}$ of $\mathcal{I}'$. Note that the construction allows us to map each time interval of the instance $\mathcal{I}$ to a subtree of T. More specifically, the i-th interval of type k can be mapped to the subtree with root the i-th node of level k and vice versa. Then, every day of the instance $\mathcal{I}$ corresponds to a leaf of the tree T. Thus, every marked day of instance $\mathcal{I}$ corresponds to a request point of $\mathcal{I}'$.



(a) A solution for an instance of Parking Permit (b) A solution for an instance of Sum-Radii Clustering

Figure 6.1: The correspondence of two solutions for the equivalent instances of the Parking Permit and the Sum-Radii Clustering problems.

In a similar way, the i-th permit of type k is mapped to the cluster with center the i-th node of level k and radius $c_{i-1} + c_{i-2} + \ldots + c_1 = 2^{i-2} + \ldots + 1 = 2^{i-1} - 1$. Thus, the total cost of the cluster is $(2^{i-1} - 1) + 1 = 2^{i-1} = c_i$. This means that the cost of the permit equals the cost of the corresponding cluster, which directly implies that the cost of $S_{\mathcal{I}}$ equals the cost of $S_{\mathcal{I}'}$.

It remains to show that a valid solution of $\mathcal{I}$ leads to a valid solution of $\mathcal{I}'$ and vice versa. For the one direction, we have to show that $S_{\mathcal{I}'}$ covers all request points. Indeed, consider a marked day of $\mathcal{I}$. This day is covered by a permit of type k and corresponds to a request point p. However, the solution $S_{\mathcal{I}'}$ opens a cluster which covers all the points of the subtree which is rooted at level k and contains p. Consequently, p is covered by a cluster. For the opposite direction, notice that we may assume w.l.o.g. that any solution of $\mathcal{I}'$ covers a point p by opening a cluster centered along the path from p to the root of T. Such a cluster centered at level k must have a radius of $c_{k-1} + c_{k-2} + \ldots + c_1$ and thus can be mapped back to a permit of type k covering the marked day which corresponds to p. □

We can use this theorem to show the following lemma.

Lemma 6.2.3. *Any c-competitive online algorithm for the* Sum-Radii Clustering *problem gives a c-competitive algorithm for the* Parking Permit *problem.*

Proof. Consider a c-competitive online algorithm A for Sum-Radii Clustering. Fix an instance $\mathcal{I}$ for Parking Permit and consider the corresponding instance $\mathcal{I}'$ from theorem 6.2.2. We have that $C_A(\mathcal{I}') \leqslant c \cdot C_{OPT}(\mathcal{I}')$. Moreover, consider the mapping of the solutions $OPT(\mathcal{I}')$ and $A(\mathcal{I}')$ to the corresponding solutions of $\mathcal{I}$, which we denote by $B(\mathcal{I})$ and $OPT(\mathcal{I})$. Since it holds that $C_B(\mathcal{I}) = C_A(\mathcal{I}')$ and $C_{OPT}(\mathcal{I}) = C_{OPT}(\mathcal{I}')$, it follows that $C_B(\mathcal{I}) \leqslant c \cdot C_{OPT}(\mathcal{I})$. □

6.3 The Deterministic Approach

In this section, we study deterministic online algorithms for the Sum-Radii Clustering problem. We first show a lower bound of $\Omega(\log n)$ on the competitive ratio of any deterministic algorithm. Then, we give a deterministic primal-dual algorithm which matches this lower bound.

Theorem 6.3.1. *The competitive ratio of any online deterministic algorithm for the* Sum-Radii Clustering *problem is* $\Omega(\log n)$*, where* n *is the number of points.*

Proof. We prove the lower bound for metric spaces which are described by an a-HST T with branching factor 3 and height $K = \log_3 n$ (a is a constant we will fix later). We also require that the set-up cost of any cluster is $f = 1$.

Let us fix any deterministic algorithm A. The adversary brings points only at the leaves of the tree, starting from the leftmost leaf and advancing towards the rightmost leaf. Specifically, the adversary brings a request at the next leaf of the tree not covered by an already open cluster. Since T has exactly $3^K = n$ leaves, it is possible that the adversary will bring strictly less than n points before all of them are covered. In this case, the adversary brings the remaining requests at any leaf which already has a demand.

We denote by C_{OPT} the cost paid by the optimum solution OPT, and by C_A the cost paid by A. Furthermore, we define that $G_k^a = 1 + \sum_{j=1}^{k-1} a^{j-1}$. The quantity G_k^a corresponds to the cost of a cluster centered at level k with radius equal to the distance of the center to the nearest leaf. Moreover, the following property holds for any $a \geqslant 2$: $G_j^a \leqslant a \cdot G_{j-1}^a$.

The next step is to classify the clusters opened by A to sets according to their cost; set L_k ($k = 1, \ldots, K$) contains all clusters with cost c, $G_k^a \leqslant c \leqslant 2 \cdot G_k^a$. The key property is that a cluster in L_k may cover a subtree rooted at level at most k and not higher. An equivalent way to define the sets L_k would be to assume w.l.o.g. that clusters are centered only along the path from the request point to the root. In this case, L_k includes

exactly all clusters centered at level k. We also define $n_k = |L_k|$. It is clear that A pays $C_A \geqslant \sum_{k=1}^{K} n_k \cdot G_k^a$.

Let us also call a subtree of T *active* when there exists a demand at some leaf of it. The points of an active subtree T_k at level k may be covered in two ways, depending on the largest cluster C which A opens after a point of T_k arrives.

1. $C \in \bigcup_{i \geqslant k} L_i$: we then call T_k a *good* subtree.

2. $C \in \bigcup_{i<k} L_i$: we then call T_k a *bad* subtree. Note that for a bad subtree, its three children subtrees are also active. We denote by b_k the number of bad subtrees rooted at level k.

In order to get a bound on the cost OPT pays, we note that it would be a feasible solution to open a cluster at level k for every active subtree rooted at level k. However, we have b_k bad active subtrees and at most $\sum_{j \geqslant k} n_j$ good subtrees. Thus, the number of active subtrees rooted at level k is bounded by $b_k + \sum_{j \geqslant k} n_j$. Thus, we have that $C_{OPT} \leqslant G_k^a \cdot (b_k + \sum_{j \geqslant k} n_j)$ for any k.

Now, by summing for $k = 1, \ldots, K$, we have that

$$K \cdot C_{OPT} \leqslant \sum_{k=1}^{K} \left(G_k^a \cdot (b_k + \sum_{j \geqslant k} n_j) \right) \leqslant \sum_{k=1}^{K} b_k \cdot G_k^a + \sum_{k=1}^{K} G_k^a \cdot \sum_{j \geqslant k} n_j$$

Using the fact that $G_j^a \leqslant a \cdot G_{j-1}^a$ and consequently that $G_j^a \leqslant a^{j-1}$, it holds that

$$\sum_{k=1}^{K} G_k^a \cdot \sum_{j \geqslant k} n_j = \sum_{k=1}^{K} n_k \cdot \sum_{j=1}^{k} G_j^a \leqslant \sum_{k=1}^{K} n_k \cdot \sum_{j=1}^{k} a^{j-1} \leqslant a \cdot \sum_{k=1}^{K} n_k \cdot G_k^a \leqslant a \cdot C_A$$

In order to conclude the proof of the lower bound, it remains to bound the quantity $R = \sum_{k=1}^{K} b_k \cdot G_k^a$. Let us fix some level $k > 1$ and consider any bad subtree T_k rooted at level k. As we have mentioned, all 3 subtrees at level $k-1$ of T_k are active and covered by clusters in $\bigcup_{i<k} L_i$. Hence, we have at most $b_{k-1} + n_{k-1}$ such subtrees. It follows immediately that $3 \cdot b_k \leqslant b_{k-1} + n_{k-1}$ for any $k > 1$. Thus, we have that $(3/a) \cdot b_k \cdot G_k^a \leqslant (b_{k-1} + n_{k-1}) \cdot G_{k-1}^a$. Now, we can sum for $k = 2, \ldots, K$

$$\frac{3}{a} \cdot \sum_{k=2}^{K} b_k \cdot G_k^a \leqslant \sum_{k=2}^{K} b_{k-1} \cdot G_{k-1}^a + \sum_{k=2}^{K} n_{k-1} \cdot G_{k-1}^a$$

Since $b_1 = 0$, we have that

$$\frac{3}{a} \cdot R \leqslant \sum_{k=1}^{K-1} b_k \cdot G_k^a + \sum_{k=1}^{K-1} n_k \cdot G_k^a \leqslant R + C_A$$

Thus, we have that $R \leqslant \frac{a}{3-a} \cdot C_A$ (for $a < 3$) and consequently

$$K \cdot C_{OPT} \leqslant \left(a + \frac{a}{3-a}\right) \cdot C_A$$

which holds for any constant $2 \leqslant a < 3$. This concludes the proof of the lower bound. □

It is known that Sum-Radii Clustering admits a constant competitive ratio when restricted to the 1-dimensional line [25]. Hence, it would be interesting to examine whether metric spaces of lower dimensionality admit a better competitive ratio than $\Omega(\log n)$. It seems though that the lower bound construction can be described on a 2-dimensional euclidean space. Thus, online Sum-Radii Clustering may be hard even for simple metric spaces.

Now, we turn our attention to an online algorithm for the problem. We consider the case where we have only radii equal to powers of two (by lemma 6.2.1, we lose only a constant factor). The approach will be based on the primal-dual schema. Let us first write the relaxed primal linear program for Sum-Radii Clustering, where we assume uniform opening costs. For ease of exposition, we will also assume that we can open clusters of at least radius f. The indicator variable x_{ir} denotes whether we open a cluster at point i with radius $2^r \cdot f$.

$$\begin{aligned} \min \quad & \sum_{i,r=0,1,\ldots} x_{ir} \cdot (2^r + 1) \cdot f \\ \text{subject to} \quad & \\ \forall j: \quad & \sum_{C(i,2^r \cdot f): d(i,j) \leqslant 2^r \cdot f} x_{ir} \geqslant 1 \\ \forall i, r = 0, 1, \ldots: \quad & x_{ir} \geqslant 0 \end{aligned} \tag{6.1}$$

Next, we present the dual linear program. For this, we introduce a dual variable a_j for every demand j.

$$\begin{aligned} \max \quad & \sum_{j} a_j \\ \text{subject to} & \\ \forall i, r = 0, 1, \ldots : \quad & \sum_{j : d(i,j) \leqslant 2^r \cdot f} a_j \leqslant (2^r + 1) \cdot f \\ \forall j : \quad & a_j \geqslant 0 \end{aligned} \tag{6.2}$$

Algorithm. At each step of the algorithm, we hold a set of open clusters S, which is initially empty. When a new point p arrives, we check whether p belongs in some cluster in S. In this case, we do nothing. Otherwise, we set $a_p = f$. When a_p increases from 0 to f, it is possible that some constraints (clusters) become tight. Then, we find the tight cluster $C(i, r)$ containing p with the largest radius and we add the cluster $C(i, 3r)$ to S.

We claim that this algorithm gives a $O(\log n)$ competitive ratio. In order to prove this, we first observe that no cluster with radius $\geqslant n$ ever becomes tight, since we have n points and thus $\sum_j a_j \leqslant n \cdot f$. Thus, for each point as the cluster center, we have at most $\log n$ constraints. We then need the following lemmas.

Lemma 6.3.2. *The dual solution we obtain satisfies all the dual constraints.*

Proof. In order to prove the lemma, it suffices to show that when a constraint becomes tight, then it will never be violated. Equivalently, we have to show that when a cluster becomes tight, every new point which arrives and belongs in this cluster will belong to some open cluster and thus its dual value will not be increased (and it remains equal to zero).

Let us assume that cluster $C(i, r)$ becomes tight after point p arrives. We distinguish the following cases:

1. We open the cluster $C(i, 3r)$. Clearly, $C(i, 3r)$ covers all the points of the cluster $C(i, r)$ and thus every point arriving in the cluster belongs in an already open cluster.

2. We open a cluster $C(j, 3r')$. First, notice that the algorithm guarantees that $r' \geqslant r$. Now, consider any point p' of $C(i, r)$. Using the triangle inequality, we have that $d(p', j) \leqslant d(p', r) + d(r, p) + d(p, r') \leqslant r + r + r' \leqslant 3r'$. Thus, every point arriving inside cluster $C(i, r)$ will also belong to the open cluster $C(j, 3r')$ and thus its dual value will not increase.

□

Let OPT be the optimum solution of the problem. For the dual values the algorithm has computed, we have that $\sum_j a_j \leqslant \mathsf{OPT}$, since the dual solution we obtain is a valid solution and thus a lower bound for the optimum cost of the primal program. Now, we have to compute the cost of the online algorithm C_{ALG} in terms of the cost of the dual solution.

Lemma 6.3.3. *It holds that* $C_{ALG} \leqslant 3 \cdot \log_2 n \cdot \sum a_j$

Proof. We will first show that each point p contributes to the opening of at most one cluster with radius $3r$, i.e. the algorithm opens at most one cluster $C(i, 3r)$ where p belongs in $C(i, r)$. Indeed, assume that the opposite holds and p belongs in two open clusters $C(i, 3r), C(j, 3r)$, where both $C(i, r), C(j, r)$ contain p. One of these clusters must have opened before the other, w.l.o.g. assume that i opens before j. However, $C(i, 3r)$ covers all points of the cluster $C(j, r)$ and thus no arrival of a point will force the algorithm into opening the cluster $C(j, 3r)$.

Thus, a point p contributes to the opening of at most $\log_2 n$ clusters. Furthermore, for each cluster $C(i, 3r)$ we open, it holds that $C_{ir} = 3 \cdot \sum_{j \in C(i,r)} a_j$. By summing, we have that

$$\begin{aligned} C_{ALG} &= \sum_{C(i,3r)} C_{ir} = 3 \cdot \sum_{C(i,3r)} \sum_{j \in C(i,r)} a_j \\ &= 3 \cdot \sum_j a_j \sum_{j \in C(i,r)} 1 \leqslant (3 \cdot \log_2 n) \cdot \sum_j a_j \end{aligned}$$

□

Combining the above lemmas, we can prove the main theorem of this section.

Theorem 6.3.4. *The primal-dual algorithm achieves a* $O(\log n)$ *competitive ratio for* Sum-Radii Clustering.

6.4 Randomized Sum-Radii Clustering

In this section, we deploy randomization in order to overcome the deterministic lower bound of the previous section and achieve a better competitive ratio. In particular, we first prove a lower bound of $\Omega(\log \log n)$ on the competitive ratio of any randomized online algorithm. Then, we present a fractional algorithm which actually achieves this competitive ratio, with the hope of providing an optimal randomized algorithm by rounding the fractional solution.

6.4.1 A Lower Bound

We will first show a lower bound for randomized algorithms. The lower bound is based on the randomized lower bound for the Parking Permit problem [49].

Theorem 6.4.1. *Any randomized online algorithm for the* Sum-Radii Clustering *problem has* $\Omega(\log\log n)$ *competitive ratio.*

Proof. As we have already shown, any Parking Permit instance can be mapped to an equivalent instance of the Sum-Radii Clustering problem. Thus, we can deploy the $\Omega(\log K)$ lower bound of the Parking Permit problem for our case. Nevertheless, we are presented with a difficulty: the lower bound of the Parking Permit depends on the number of permit types K whereas we would like to express the lower bound for the Sum-Radii Clustering problem in terms of the number of points n.

Let us consider an adversary which brings a fixed number of points n. We then consider the distribution of points for the Parking Permit bound with $K = \log(n/c)$ permit types, where $c > 1$ is a constant. Moreover, let us denote by N the random variable which denotes the number of points which the adversary brings. Following [49], we have that $\mathbb{E}[N] \approx n/c$.

We now distinguish two cases according to the value N finally assumes. In the case that $N \leqslant n$, the adversary brings any remaining points at a location where a point has already arrived before. Furthermore, any deterministic algorithm pays at least an expected cost of $\mathbb{E}[N] \approx n/c$.

In the other case, the adversary has to bring more points than n, which is not allowed since it has a budget of only n points. However, the probability that this happens can be bounded using the Markov's inequality.

$$\Pr[N \geqslant n] \leqslant \frac{\mathbb{E}[N]}{n} \approx \frac{n/c}{n} = 1/c$$

Thus, we conclude that the expected cost of any deterministic algorithm A is at least

$$\mathbb{E}[C_A] \geqslant \Pr[N \leqslant n] \cdot \mathbb{E}[N] \geqslant (1 - 1/c) \cdot n/c$$

Now, let us compute the expected cost of an optimal offline algorithm OPT. Following [49], it is easy to see that the expected cost of OPT is upper bounded by $\frac{2^{K+1}}{\log K} = \frac{2n/c}{\log\log(n/c)}$. Thus, it follows that

$$\frac{\mathbb{E}[C_A]}{\mathbb{E}[OPT]} \geqslant \frac{(1 - 1/c) \cdot n/c}{\frac{2n/c}{\log\log(n/c)}} = \frac{(1 - 1/c)}{2} \cdot \log\log(n/c)$$

Now, applying Yao's lemma, we conclude that the competitive ratio of any randomized online algorithm for the SUM-RADII CLUSTERING problem is at least $\Omega(\log\log n)$. □

6.4.2 A Fractional Online Algorithm

We present a fractional online algorithm for the SUM-RADII CLUSTERING problem, which admits a $O(\log\log n)$ competitive ratio; hence it matches the randomized lower bound.

In the *fractional* problem of SUM-RADII CLUSTERING, we are allowed to have fractions of clusters instead of a whole cluster. However, the fractions of the clusters which cover any demand point must sum to a value greater than one. We will denote by f_i^k the fraction by which the cluster $C(i, 2^k \cdot f)$ opens. Hence, for any point p, the following condition must hold: $F_p = \sum_{i,k:p\in C(i,2^k f)} f_i^k \geqslant 1$. For the algorithm, we will assume that the clusters open with an actual radius double their radius (which increases the total cost only by a factor of 2). Hence, the required condition becomes $F_p = \sum_{i,k:p\in C(i,2^{k+1} f)} f_i^k \geqslant 1$. Furthermore, the total cost of a solution will now be $2 \cdot \sum_{i,k}(f_i^k \cdot c_k)$, where c_k is the cost of a cluster of type k.

We will first assume that the number of demand points n which will arrive is known to the online algorithm. We will later show how this assumption can be dropped. Since the number of points is n, the algorithm needs to consider only $K = \log n$ distinct types of clusters. Denote by F_p^k the fraction of F_p associated only with cluster of types k, i.e. $F_p^k = \sum_{i\in C(p,2^{k+1} f)} f_i^k$. We now describe the online algorithm in full detail.

Algorithm. Notice that during the course of the algorithm, new clusters will be available for opening. As soon as a cluster becomes available (this happens when a new point arrives), its fractional value is set to zero. Now, consider the case when a new demand point p arrives. We distinguish two cases based on the value of F_p.

- If $F_p \geqslant 1$, then p is already covered.
- Else, $F_p < 1$. While $F_p < 1$, perform the following *operation*:
  1. For every k : $f_p^k \leftarrow f_p^k + \frac{1}{K \cdot c_k}$
  2. For every k and $i \in C(p, 2^{k+1} \cdot f)$: $f_i^k \leftarrow f_i^k \cdot (1 + \frac{1}{c_k})$

Theorem 6.4.2. *The competitive ratio of the online fractional algorithm is* $O(\log\log n)$.

Proof. We will first prove that the cost of each operation is constant. Then, we will only need to bound the number of operations performed by the online algorithm.

For the first part, let us consider a single operation. Since the algorithm performs an operation, it means that $F_p < 1$. For the first step of the operation, the fractional cost increases by $1/K$ for each cluster type. Hence, the total increase will be exactly 1. For the second part of the operation, notice that the total increase of the fractional cost will be

$$\sum_{i,k:i\in C(p,2^{k+1}f)} \frac{f_i^k}{c_k} \cdot c_k = \sum_{i,k:i\in C(p,2^{k+1}f)} f_i^k = F_p < 1$$

Summing, we conclude that each operation increases the cost by at most 2.

Thus, it remains to bound the number of operations needed. Let us assume that the optimal solution OPT opens a cluster $C_0 \equiv C(p, 2^k \cdot f)$ with cost c_k. We will compute the corresponding cost of the online algorithm for the points which are covered by the cluster C_0. We denote by F_p^c the total fraction for type k clusters centered within the optimal cluster. Note that as soon as $F_p^c \geqslant 1$, each point q which arrives at an later time inside C_0 will be covered, since $C(q, 2^{k+1} \cdot f)$ includes all points inside C_0 and thus $F_p^k \geqslant F_p^c \geqslant 1$. Hence, we have to count the number of operations required so that F_p^c reaches 1.

Notice that after the first c_k operations for points inside the cluster C_0, it holds that $F_p^c \geqslant \frac{1}{K \cdot c_k} \cdot c_k = 1/K$. After that, when an operation is performed for a point inside C_0, the fractional value of every point within C_0 is multiplied by $(1 + 1/c_k)$ and thus the total fraction F_p^c is multiplied by the same factor as well. It is easy to see that after $O(c_k \log K)$ more operations, we will have that $F_p^c \geqslant 1$ and thus no more operations are needed for any point arriving inside the cluster.

Thus, the online algorithm pays $O(c_k \log K)$, whereas OPT pays c_k. Summing for all clusters of the optimal solution, we conclude that we pay $O(\log K)$ times more than OPT, which implies a competitive ratio of $O(\log \log n)$ for the fractional online algorithm. □

Estimating the number of types K. We will now describe a method to drop the assumption that n is a priori known to the algorithm. The algorithm keeps a current estimation of K according to the number of points already arrived and behaves according to the specific value of K. It starts by assuming that $K = 1$ and increases the value of K when more points arrive. Specifically, the algorithm sets $K \leftarrow k$ as soon as 2^k demands have arrived. When K increases, the algorithm also redistributes the fractions so that they match the fractions the algorithm would output in the case it started by knowing the new value of K.

It is easy to see that the above modification does not influence the competitive ratio of the algorithm. The only difference is that, since we may have an underestimation of K, for an optimal cluster of type k we may pay $O(\log K')$ times more than OPT, where

$k \leqslant K' \leqslant \log n$. Nevertheless, this implies again a $O(\log \log n)$ competitive ratio since K will never increase beyond $\log n$.

An open problem is to describe a randomized rounding procedure for the fractional solution, so as to obtain a randomized online algorithm for Sum-Radii Clustering with optimal competitive ratio.

Chapter 7

Online Leasing Problems

In the context of offline leasing problems, we studied the case where we were given the whole sequence of demands at the beginning of the algorithm. However, we have to take into account real-life situations where we want to act based on present demands, without knowing about the future.

We model this situation by demands arriving one after the other as the time flows. The *online* algorithm must decide on how the demands will be served without having knowledge of the future, based only on current and past information. In chapter 5, we studied the online Parking Permit problem, presenting optimal deterministic and randomized algorithms. In this chapter, we will study the online versions of leasing problems with a more complex structure.

Since all leasing problems generalize Parking Permit, it is clear that the competitive ratio of any deterministic algorithm will be lower-bounded by $\Omega(K)$. We also have to take into account the inherent difficulty of finding a good competitive ratio for the online non-leasing version of the combinatorial problem as well. Thus, if a problem has a lower bound of $\Omega(c(n))$ on the competitive ratio, we get an immediate lower bound of $\Omega(K + c(n))$ for the competitive ratio of any deterministic online algorithm for the leasing problem. The same holds for randomized algorithms, were we cannot hope to achieve a competitive ratio better than $\Omega(\log K + c'(n))$, where $\Omega(c'(n))$ is the lower bound for randomized online algorithms.

Can we design online algorithms for leasing problems that achieve this competitive ratio or can we prove a stronger lower bound? As we will see in this chapter, the algorithms that we present achieve a competitive ratio which combines *orthogonally* the ideas behind algorithms for Parking Permit and algorithms for the non-leasing variants of the problems. This way, the competitive ratio obtained is the product (and not the sum) of the

competitive ratios of the two directions of the problem, that is, it is of the form $O(K \cdot c(n))$ or $O(\log K \cdot c'(n))$. It remains an open question whether there exists an inherent difficulty in finding better solutions for these problems or whether we can approach online leasing problems by combining techniques in an interleaved way.

In this chapter, we will first give a fairly straightforward generalization of the randomized online algorithm for Parking Permit to Steiner Forest Leasing [49]. Next, we will study online Facility Leasing. For this problem, we first present a deterministic algorithm by Nagarajan and Williamson [50], which is based on an algorithm of Fotakis [31]. Finally, we present a randomized algorithm which achieves a slightly better competitive ratio and extends ideas from [48].

7.1 Online Steiner Forest Leasing

We briefly remind the setting for online Steiner Forest Leasing. We are given a graph $G = (V, E)$ and at each day, sets of nodes, which are subsets of V arrive. The elements we can lease are the edges of the graph. Leasing an edge e with a type k lease costs $c_k \cdot w_e$, where w_e is the weight of edge e (we have thus assumed *uniform* lease costs). In order to serve a set of nodes, we must ensure that the edges we lease connect the nodes of the demand set.

The algorithm we present here combines ideas from randomized Parking Permit (see chapter 5) and the Buy-at-Bulk problem [8]. The non-leasing online version of Steiner Forest has been extensively studied. In particular, Imaze and Waxman [43] proved a lower bound of $\Omega(\log n)$, where n is the number of demands (pairs of nodes). As for the upper bound, Awerbuch et al. [9] first designed an $O(\log^2 n)$-competitive algorithm, which was then improved to an optimal $O(\log n)$-competitive algorithm [12].

In order to be able to extend the randomized algorithm for the Parking Permit problem, we first have to simplify the metric space and impose a tree structure to the graph G. In order to achieve this, we apply standard results for probabilistically approximating metric spaces by tree metrics [11, 30]. This means that we lose a multiplicative $O(\log n)$ factor from the competitive ratio; however, the simplicity of the algorithm used for the tree compensates for the factor we lose.

Let us first describe an $O(\log K)$ competitive algorithm for Steiner Forest Leasing when the graph G is a tree. We may assume that the leases follow the interval model and the cost scaling model, since we will lose only a constant factor.

Consider a day t and a request node set $D_t \subseteq V$. Since the graph is a tree, there is a unique spanning subgraph G_{D_t} which connects all the demand nodes. Thus, the optimal

algorithm must ensure that every edge of G_{D_t} is leased at day t. Notice that in the case that the graph was not a tree, there might be that more than one subgraphs connecting the demand nodes. Now, it is easy to see that solving the Steiner Forest Leasing is equivalent to solving a Parking Permit instance I_e for each edge e of the tree. In this instance, a day t is a driving day only if edge e belongs in the spanning subgraph of some demand set arriving at day t.

Thus, we may use the randomized algorithm for the Parking Permit to independently solve the instance I_e for each edge e and pay at most $O(\log K)$ times what the optimum algorithm pays for this particular edge. Summing over all edges, we obtain a $O(\log K)$ competitive ratio for the case that G is a tree.

Theorem 7.1.1. *If graph G is a tree, there exists an $O(\log K)$ competitive algorithm for* Steiner Forest Leasing.

Now, suppose that we are given an arbitrary graph G. Using the result from [30], we embed G in a randomly chosen tree T. Then, we use the above online algorithm for trees to obtain a $O(\log K)$-competitive solution on T. The last step is to map the solution for T back to a solution for the original graph G. Since the embedding from graphs to trees gives an expected $O(\log n)$ stretch to the cost of each edge, we obtain that the final solution achieves a competitive ratio of $O(\log K \cdot \log n)$.

Theorem 7.1.2. *Online* Steiner Forest Leasing *admits an $O(\log K \log n)$ competitive ratio.*

Note that the competitive ratio we achieve is the product of the competitive ratios for Parking Permit ($O(\log K)$) and Steiner Forest ($O(\log n)$). It seems that in order to improve the competitive ratio we need to introduce new ideas and follow a different approach.

7.2 Online Facility Leasing

We can view the online Facility Leasing problem as the generalization of two problems, the Facility Location and the Parking Permit problem. Given that the first problem admits an $\Omega(\frac{\log n}{\log \log n})$ lower bound [32] and the latter an $\Omega(K)$ lower bound on the competitive ratio, we easily obtain a $\Omega(K + \frac{\log n}{\log \log n})$ lower bound for deterministic algorithms. We also obtain a $\Omega(\log K + \frac{\log n}{\log \log n})$ lower bound for randomized algorithms in a similar way.

In the algorithms we present in the next two sections, the two directions of the problem are considered in an independent way, hence the competitive ratio we achieve will be much

larger than the lower bound. For the deterministic case, we present an algorithm from [50] with competitive ratio $O(K \cdot \log n)$. We manage to slightly improve the competitive ratio by using randomization, achieving an upper bound of $O(K \cdot \frac{\log n}{\log \log n})$ for the uniform case.

It remains an open question whether we can lower the upper bound on the competitive ratio or find a better lower bound. It is still not clear whether it would be sufficient to combine already known techniques from both directions or whether one has to proceed with a different method.

7.2.1 A Deterministic Algorithm

In this section, we describe and analyze a deterministic algorithm with competitive ratio of $O(K \log n)$. The algorithm, presented by [50], generalizes an online algorithm for Facility Location introduced by Fotakis in [31].

We consider the nested version of the problem, which is equivalent up to a constant factor in the competitive ratio. We have already described the primal and dual programs for the offline Facility Leasing problem. We keep the same notation here.

In the online setting, each demand arrives one after the other. The algorithm maintains K sets of facilities opened so far, where F_k is the set of type-k facilities we have opened. Moreover, let us denote by $d(F_k, d_j^t) = \min_{f_i^k(t') \in F_k, t \in I_{t'}^k} d(i,j)$, i.e. the distance of a demand from the closest open facility which can serve it.

During the algorithm, each demand d_j^t keeps a *bid* towards every facility $f_i^k(t') \notin F = \bigcup_k F_k$ such that $t \in I_{t'}^k$. Specifically, the bid is

$$\mathtt{bid}(d_j^t, f_i^k(t')) = \max\{0, \min\{v_{jt}, d(F_k, d_j^t)\} - d(i,j)\}$$

The bidding intuitively corresponds to the value the demand would be willing to pay so as to open this particular facility. Clearly, if the demand does not profit from opening the facility (i.e. an open facility is closer), then the demand bids nothing. The algorithm maintains the invariant that the sum of bids for each facility which is not open is no more than the cost of opening this facility.

Analysis. The analysis of the competitive ratio is based on *dual fitting* ([60]) and involves several steps. First, let us express the connection cost of the demands in terms of the dual variables of the algorithm. The following lemma holds.

Lemma 7.2.1. *The sum of the connection costs* C_d *is at most* $\sum_{d_j^t \in \mathcal{D}} v_{jt}$.

Algorithm 3: ONLINE FACILITY LEASING

Consider a new demand d_j^t arriving at time t and let D be the set of the current demands. The algorithm then increases the dual variable ν_{jt}, which is initially set to zero. This is equivalent to the demand bidding $\max\{0, \nu_{jt} - d(i,j)\}$ towards any facility $f_i^k(t)$. The increase stops as soon as one of the following happens:

- $\nu_{jt} = d(i,j)$ for some facility $f_i^k(t') \in F_k$ with $t \in I_{t'}^k$. Then, notice that we cannot increase ν_{jt} any further, since then the sum of the bids would be greater than the opening cost of $f_i^k(t)$, thus violating the dual constraint.
- For a facility $f_i^k(t') \notin F$ such that $t \in I_{t'}^k$, it holds that

$$\sum_{\hat{d_j^t} \in D \cup \{d_j^t\}} bid(\hat{d_j^t}, f_i^k(t')) = c_i^k$$

  In this case, we add $f_i^k(t')$ to F_k (and hence to F).

Finally, the demand d_j^t is connected to the closest open facility in F.

Proof. When a demand d_j^t is assigned to an already open facility $f_i^k(t')$, it holds that $\nu_{jt} = d(j,i)$; hence its dual variable equals the connection cost. Otherwise, the demand must have contributed to the opening of the facility $f_i^k(t')$ the demand was assigned to. Then, it holds that $\nu_{jt} > d(i,j)$. In either case, the dual variable is always greater than or equal to its connection cost. Summing for each demand, we easily obtain the lemma. □

Lemma 7.2.2. *The sum of the facility costs* C_f *is at most* $K \cdot \sum_{d_j^t \in \mathcal{D}} \nu_{jt}$.

Proof. Clearly, when a facility opens, the opening cost is equal to the sum of the bids made at that time. Hence, it suffices to show that for any demand d_j^t, the contribution of the bids towards opening facilities of type k is at most the value of the dual variable ν_{jt}. Since we have exactly K different lease types, the above fact implies the bound the lemma states.

Fix a demand d_j^t and a lease type k. We first observe that at any time and for any facility $f_i^k(t')$, it holds that $bid(d_j^t, f_i^k(t')) \leqslant \nu_{jt}$, i.e. the demand never bids more than its dual variable. Now, let us consider the sequence of facilities in F_k where d_j^t contributes, ordered by time of opening: $f_0, f_1, \ldots, f_m$. Clearly, the facilities in this sequence are in order of decreasing distance to the demand d_j^t (this happens because a demand never profits by bidding for a facility which is further than the closest one). For the first facility

f_0, the demand contributes at most $\nu_{jt} - d(f_0, d_j^t)$. After that, it holds that $d(F_k, d_j^t) \leqslant d(f_0, d_j^t)$, so the demand may not contribute more than $d(f_0, d_j^t) - d(f_1, d_j^t)$ for the second facility. Generally, the demand contributes at most $d(f_s, d_j^t) - d(f_{s+1}, d_j^t)$ for any facility $f_s, 0 \leqslant s < m$. Summing, the total contribution of d_j^t will be at most

$$\nu_{jt} - d(f_0, d_j^t) + \sum_{0 \geqslant s < m} \{d(f_s, d_j^t) - d(f_{s+1}, d_j^t)\} = \nu_{jt} - d(f_m, d_j^t) \leqslant \nu_{jt}$$

and this concludes the proof of the lemma. □

By combining the two lemmata, we easily obtain the following corollary.

Corollary 7.2.3. *The total cost of the solution is at most* $(K+1) \cdot \sum_{d_j^t \in \mathcal{D}} \nu_{jt}$.

If we would have that the dual solution is feasible, then the above corollary would imply a $(K+1)$ competitive ratio. However, the dual solution produced by the algorithm is infeasible. Nevertheless, we can prove using a *dual fitting* argument that, dividing the dual variables by the harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$, we get a feasible dual solution. This implies a competitive ratio of $O(K \cdot \log n)$. We next present the lemma along with its formal proof. Let $\alpha = \frac{1}{2(H_n+1)}$.

Lemma 7.2.4. *For any subset of demands* $S \subseteq \mathcal{D}$*, and any facility* $f_i^k(t')$*, it holds that*

$$\sum_{d_j^t : t \in I_{t'}^k} (\alpha \cdot \nu_{jt} - d(i,j)) \leqslant c_i^k$$

Proof. We will say that a demand l is *related* to an open facility $f_i^k(t')$ when $\nu_l \geqslant d(l,i)$. Consider the set of demands $S = \{1, 2, \ldots, m\}$ ordered according to the time they get related to some k-type facility. If there exist demands which become related at the same iteration, we order them in increasing order of $\nu_j - d(i,j)$, breaking ties arbitrarily. Clearly, we may assume w.l.o.g. that every such demand arrives within the interval $I_{t'}^k$, otherwise it would not contribute to the left hand sum of the inequality.

Now, let us fix a demand $l \in S$ such that l is related to a facility $f_i^k(t')$. Let F_k be the set of open facilities at the beginning of the iteration when l first becomes related to the facility. Moreover, denote by $h < l$ the demand that first becomes related to $f_i^k(t')$ during this iteration according to our ordering. Consider the invariant the algorithm maintains for the demand l and this facility at the iteration l became related to it.

$$c_i^k \geqslant \sum_{j\in\{1,\ldots,l\}} bid(j, f_i^k(t')) \geqslant \sum_{j\in\{1,\ldots,l\}} \min\{\nu_j, d(F_k, j)\} - d(i,j)$$
$$\geqslant \sum_{j\in\{1,\ldots,h-1\}} \{d(F_k, j) - d(i,j)\} + \sum_{j\in\{h,\ldots,l\}} \{\nu_j - d(i,j)\}$$
$$\geqslant \sum_{j\in\{1,\ldots,h-1\}} d(F_k, j) - d(i,j) + (l-h+1)\cdot(\nu_l - d(i,l))$$

The last inequality holds because we have ordered demands that become related during the same iteration in increasing order of $\nu_j - d(i,j)$. In order to further manipulate this equation, consider a previous demand j related to $f_i^k(t')$. Clearly, the dual variable ν_l will never increase more than the distance of l to the closest open facility. Hence, we have that $\nu_l \leqslant d(l, F_k) \leqslant d(l,j) + d(j, F_k) \leqslant d(i,l) + d(i,j) + d(j, F_k)$, using repeatedly the triangle inequality. From this, we obtain that

$$d(F_k, j) - d(i,j) \geqslant \nu_l - d(i,l) - 2d(i,j)$$

Hence, the equation becomes

$$c_i^k \geqslant \sum_{j<h} \{\nu_l - d(l,i) - 2d(i,j)\} + (l-h+1)\cdot(\nu_l - d(i,l))$$
$$\geqslant l\cdot(\nu_l - d(i,l)) - 2\sum_{j<l} d(i,j)$$

Dividing the equation by l, we get that

$$\frac{c_i^k}{l} \geqslant \nu_l - d(i,l) - \frac{2}{l}\sum_{j<l} d(i,j)$$

However, set S may contain demands which are not related to any facility of type k. Consider such a demand with the smallest possible index $q+1$. Clearly, the sum of the dual variables of these demands will never be larger than c_i^k, i.e.

$$\sum_{j\in S: j>q} (\nu_j - d(i,j)) \leqslant c_i^k$$

Adding this inequality and all the other inequalities for $l = 1, \ldots, q$, we get that:

$$(H_q + 1)\cdot c_i^k \geqslant \sum_{l=1,\ldots,m} (\nu_l - d(i,l)) - \sum_{l=1,\ldots,q} \frac{2}{l}\sum_{j<l} d(i,j)$$

Simple algebraic manipulations of the above inequality obtain that

$$\sum_{l\in S} (\alpha\cdot\nu_l - d(i,l)) \leqslant c_i^k$$

which concludes the proof of the lemma. □

Theorem 7.2.5. *[50] There is an online algorithm for the* Facility Leasing *problem with competitive ratio* $O(K \cdot \log n)$.

7.2.2 A Randomized Algorithm

In this section, we present a simple randomized algorithm for the online uniform Facility Leasing problem (Simple-Random). Simple-Random achieves an expected competitive ratio of $O(K \cdot \frac{\log n}{\log \log n})$, which is smaller in expectation than the competitive ratio of the deterministic algorithm of the previous section. The algorithm is based on the randomized online algorithm for Facility Location presented by Meyerson in [48].

Let us now describe the algorithm in full detail.

Algorithm 4: Simple-Random

Each time a new demand at point p arrives:

- For any type $k = 1, \ldots, K$: open a facility of type k at point p with probability $\min\{\frac{\delta}{c_k}, 1\}$, where δ is the distance to the closest open facility at that time.
- Assign the demand to the closest open facility at that time.

Analysis. Let us provide an analysis for the expected competitive ratio of the above algorithm. Consider an optimal cluster C_i^* around an optimum facility c_i^* of type k. Denote by A_i^* the sum of the assignment costs of all demands attributed to c_i^*. Furthermore, let $a_i^* = A_i^*/|C_i^*|$, where $|C_i^*|$ denotes the number of demands assigned to c_i^*. Clearly, the total cost of the cluster in the optimal solution is $A_i^* + c_k$.

We next partition the demands in C_i^* into sets S_j, $1 \leqslant j \leqslant u$. The set S_j includes all demands with distance from the optimal center between $d^{j-1}a_i^*$ and $d^j a_i^*$ (d and u are parameters which will be fixed later). The demands with distance at most a_i^* are called *inner* demands (set S_0). For the purposes of our analysis, we have to ensure that S_{u+1} will be empty. Indeed, if we guarantee that $d^u > n$, then S_{u+1} must be empty, as otherwise the total assignment cost would be $A_i^* \geqslant d^u \cdot a_i^* > n \cdot a_i^* \geqslant |C_i^*| \cdot a_i^* = A_i^*$, which is a contradiction. Hence, u and d must be chosen such that $d^u > n$.

Now, let us consider some set S_j. We will bound the total amount that any demands in S_j cost to the online algorithm. We will first compute the expected cost we pay for assignment costs before we open a facility of type k. It is easy to see that the expected assignment cost of the demands will be at most c_k before a facility of type k first opens in S_j. Any subsequent demand w in S_j will be at distance from the optimum center

$d^*_w \geqslant d^{j-1}a^*_i$ and, since we have a facility of type k open inside S_j, the optimal center c^*_i will be at distance at most $d^j a^*_i$ from a facility of type k. Using the triangle inequality, we obtain for w that the minimum distance δ from any open facility is bounded by $\delta \leqslant d^j \cdot a^*_i + d^*_w \leqslant (d+1)d^*_w$. Thus, the expected cost for any such demand w is bounded by

$$\delta + \sum_{i=1}^{i=K} \frac{\delta}{c_i} \cdot c_i = (K+1) \cdot \delta \leqslant (K+1) \cdot (d+1) \cdot d^*_w$$

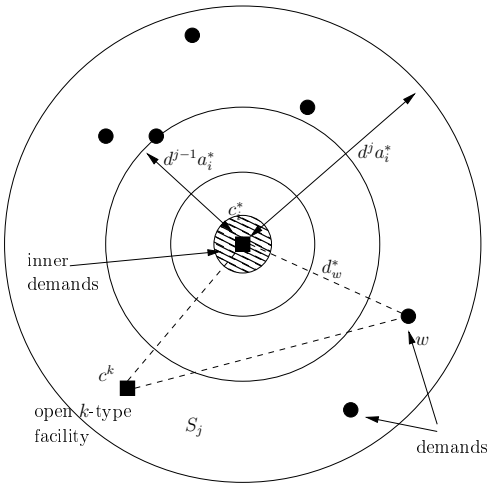


Figure 7.1: The partition of an optimal cluster C^*_i to the sets S_j and the inner demands. One can observe how we can bound the minimum distance from any demand w to the closest facility when a facility has already opened in the same set.

However, before we open the first facility of type k, we have to pay for the possible opening of facilities of smaller type. How much do we expect to pay? Notice that each demand contributes on expectation the same for the opening of facilities of any type. Hence, we pay an expected c_k for the facilities of any type $l < k$. Summing for each type smaller than k, we obtain that the expected cost for the opening of facilities of smaller type will be $(k-1) \cdot c_k$.

Now, it remains to bound the cost for the inner demands. After the opening of any facility of type k in a distance less than a^*_i from the optimum center, for any inner demand w we will have that $\delta \leqslant d^*_w + a^*_i$ (applying again the triangle inequality). By the same argument as before, we pay at most $(k-1) \cdot c_k$ before opening such a facility of type k, plus c_k for the opening of the facility, plus an expected $(K+1) \cdot (d^*_w + a^*_i)$ for any subsequent

demand.

Summing up, we have a total expected cost C which is bounded by

$$\begin{aligned} C &= \sum_{i=1\ldots u} \{k \cdot c_k + (K+1) \cdot (d+1) \cdot \sum_{w \in S_i} d^*_w\} + k \cdot c_k + (K+1) \cdot \sum_{w \text{ inner}} (d^*_w + a^*_i) \\ &\leqslant (u+1) \cdot k \cdot c_k + (K+1) \cdot (d+1)\{ \sum_{i=1\ldots u} \sum_{w \in S_i} d^*_w + \sum_{w \text{ inner}} d^*_w + \sum_{w \text{ inner}} a^*_i\} \\ &\leqslant (u+1) \cdot K \cdot c_k + (K+1) \cdot (d+1) \cdot 2A_i* \end{aligned}$$

It is easy to see that by setting the parameters $u = d = \frac{\log n}{\log \log n}$, we ensure that $d^u > n$ and we obtain an $O(K \cdot \frac{\log n}{\log \log n})$ competitive ratio for Simple-Random. Thus, we have proved the following theorem.

Theorem 7.2.6. Simple-Random *achieves an expected competitive ratio of* $O(K \cdot \frac{\log n}{\log \log n})$ *for the online* Facility Leasing *problem.*

Bibliography

[1] Agrawal, A., Klein, P. N., and Ravi, R. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput. 24*, 3 (1995), 440–456.

[2] Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. A general approach to online network optimization problems. *ACM Transactions on Algorithms 2*, 4 (2006), 640–660.

[3] Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. The online set cover problem. *SIAM J. Comput. 39*, 2 (2009), 361–370.

[4] Alon, N., Moshkovitz, D., and Safra, S. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms 2* (April 2006), 153–177.

[5] Andrews, M., and Zhang, L. Wavelength assignment in optical networks with fixed fiber capacity. In *ICALP* (2004), J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, Eds., vol. 3142 of *Lecture Notes in Computer Science*, Springer, pp. 134–145.

[6] Anthony, B. M., and Gupta, A. Infrastructure leasing problems. In *IPCO* (2007), M. Fischetti and D. P. Williamson, Eds., vol. 4513 of *Lecture Notes in Computer Science*, Springer, pp. 424–438.

[7] Ariyawansa, K. A., and Felt, A. J. On a new collection of stochastic linear programming test problems. *INFORMS Journal on Computing 16*, 3 (2004), 291–299.

[8] Awerbuch, B., and Azar, Y. Buy-at-bulk network design. In *FOCS* (1997), pp. 542–547.

[9] Awerbuch, B., Azar, Y., and Bartal, Y. On-line generalized steiner problem. In *SODA* (1996), pp. 68–74.

[10] Bar-Yehuda, R., and Even, S. A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms 2*, 2 (1981), 198–203.

[11] Bartal, Y. Probabilistic approximations of metric spaces and its algorithmic applications. In *FOCS* (1996), pp. 184–193.

[12] Berman, P., and Coulston, C. On-line algorithms for steiner tree problems (extended abstract). In *STOC* (1997), pp. 344–353.

[13] Birge, J. R., and Louveaux, F. *Introduction to Stochastic Programming*, corrected ed. Springer Series in Operations Research and Financial Engineering. Springer, July 1997.

[14] Borodin, A., and El-Yaniv, R. *Online computation and competitive analysis*. Cambridge University Press, New York, NY, USA, 1998.

[15] Borradaile, G., Klein, P. N., and Mathieu, C. A polynomial-time approximation scheme for euclidean steiner forest. In *FOCS* (2008), IEEE Computer Society, pp. 115–124.

[16] Buchbinder, N., and Naor, J. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science 3*, 2-3 (2009), 93–263.

[17] Byrka, J., and Aardal, K. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM J. Comput. 39*, 6 (2010), 2212–2231.

[18] Byrka, J., Grandoni, F., Rothvoss, T., and Sanità, L. An improved lp-based approximation for steiner tree. In *Proceedings of the 42nd ACM symposium on Theory of computing* (New York, NY, USA, 2010), STOC '10, ACM, pp. 583–592.

[19] Chan, T. M., and Zarrabi-Zadeh, H. A randomized algorithm for online unit clustering. *Theory Comput. Syst. 45*, 3 (2009), 486–496.

[20] Charikar, M., Chekuri, C., Feder, T., and Motwani, R. Incremental clustering and dynamic information retrieval. *SIAM J. Comput. 33*, 6 (2004), 1417–1440.

[21] Charikar, M., Chekuri, C., and Pál, M. Sampling bounds for stochastic optimization. In *APPROX-RANDOM* (2005), pp. 257–269.

[22] Charikar, M., and Guha, S. Improved combinatorial algorithms for the facility location and k-median problems. In *FOCS* (1999), pp. 378–388.

[23] Charikar, M., and Panigrahy, R. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci. 68*, 2 (2004), 417–441.

[24] Chudak, F. A., and Shmoys, D. B. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput. 33*, 1 (2003), 1–25.

[25] Csirik, J., Epstein, L., Imreh, C., and Levin, A. Online clustering with variable sized clusters. In *MFCS* (2010), P. Hlinený and A. Kucera, Eds., vol. 6281 of *Lecture Notes in Computer Science*, Springer, pp. 282–293.

[26] Dantzig, G. Linear programming under uncertainty. *Management Science 1* (1955), 197–206.

[27] Dantzig, G. *Linear Programming and Extensions.* Princeton University Press, Princeton,NJ, 1998.

[28] Daskin, M. S. *Network and Discrete Location: Models, Algorithms, and Applications.* Wiley-Interscience, 1995.

[29] Drezner, Z., and Hamacher, H. *Facility Location: Applications and Theory.* Springer, 2004.

[30] Fakcharoenphol, J., Rao, S., and Talwar, K. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci. 69*, 3 (2004), 485–497.

[31] Fotakis, D. A primal-dual algorithm for online non-uniform facility location. *J. Discrete Algorithms 5*, 1 (2007), 141–148.

[32] Fotakis, D. On the competitive ratio for online facility location. *Algorithmica 50*, 1 (2008), 1–57.

[33] Goemans, M. X., and Williamson, D. P. A general approximation technique for constrained forest problems. *SIAM J. Comput. 24*, 2 (1995), 296–317.

[34] Goemans, M. X., and Williamson, D. P. *The primal-dual method for approximation algorithms and its application to network design problems.* PWS Publishing Co., Boston, MA, USA, 1997, pp. 144–191.

[35] Goyal, S. K., and Giri, B. C. Recent trends in modeling of deteriorating inventory. *European Journal of Operational Research 134*, 1 (2001), 1–16.

[36] Guha, S., and Khuller, S. Greedy strikes back: Improved facility location algorithms. *J. Algorithms 31*, 1 (1999), 228–248.

[37] Guha, S., Meyerson, A., and Munagala, K. Hierarchical placement and network design problems. In *FOCS* (2000), pp. 603–612.

[38] Gupta, A., Kumar, A., Pál, M., and Roughgarden, T. Approximation via cost sharing: Simpler and better approximation algorithms for network design. *J. ACM 54*, 3 (2007), 11.

[39] Gupta, A., Pál, M., Ravi, R., and Sinha, A. Boosted sampling: approximation algorithms for stochastic optimization. In *STOC* (2004), L. Babai, Ed., ACM, pp. 417–426.

[40] Gupta, A., Pál, M., Ravi, R., and Sinha, A. What about wednesday? approximation algorithms for multistage stochastic optimization. In *APPROX-RANDOM* (2005), pp. 86–98.

[41] Hayrapetyan, A., Swamy, C., and Tardos, É. Network design for information networks. In *SODA* (2005), SIAM, pp. 933–942.

[42] Hwang, F. K., Richards, D. S., and Winter, P. *The Steiner Tree Problem.* North-Holland, 1992.

[43] Imase, M., and Waxman, B. M. Dynamic steiner tree problem. *SIAM J. Discrete Math. 4*, 3 (1991), 369–384.

[44] Immorlica, N., Karger, D. R., Minkoff, M., and Mirrokni, V. S. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *SODA* (2004), J. I. Munro, Ed., SIAM, pp. 691–700.

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

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

[47] Karlin, A. R., Manasse, M. S., Rudolph, L., and Sleator, D. D. Competitive snoopy caching. *Algorithmica 3* (1988), 77–119.

[48] Meyerson, A. Online facility location. In *FOCS* (2001), pp. 426–431.

[49] Meyerson, A. The parking permit problem. In *In FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (2005), pp. 274–284.

[50] Nagarajan, C., and Williamson, D. P. Offline and online facility leasing. In *IPCO* (2008), A. Lodi, A. Panconesi, and G. Rinaldi, Eds., vol. 5035 of *Lecture Notes in Computer Science*, Springer, pp. 303–315.

[51] Nahmias, S. Perishable inventory theory: a review. *Oper Res 30*, 4 (1982), 680–708.

[52] Ravi, R., and Sinha, A. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *IPCO* (2004), G. L. Nemhauser and D. Bienstock, Eds., vol. 3064 of *Lecture Notes in Computer Science*, Springer, pp. 101–115.

[53] Raz, R., and Safra, S. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *STOC* (1997), pp. 475–484.

[54] Robins, G., and Zelikovsky, A. Tighter bounds for graph steiner tree approximation. *SIAM J. Discrete Math. 19*, 1 (2005), 122–134.

[55] Salman, F. S., Cheriyan, J., Ravi, R., and Subramanian, S. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization 11*, 3 (2001), 595–610.

[56] Shmoys, D. B., and Swamy, C. Stochastic optimization is (almost) as easy as deterministic optimization. In *FOCS* (2004), IEEE Computer Society, pp. 228–237.

[57] Shmoys, D. B., Tardos, É., and Aardal, K. Approximation algorithms for facility location problems (extended abstract). In *STOC* (1997), pp. 265–274.

[58] Srinivasan, A. Approximation algorithms for stochastic and risk-averse optimization. In *SODA* (2007), N. Bansal, K. Pruhs, and C. Stein, Eds., SIAM, pp. 1305–1313.

[59] Swamy, C., and Shmoys, D. B. Sampling-based approximation algorithms for multistage stochastic optimization. In *In Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (2005), pp. 357–366.

[60] Vazirani, V. V. *Approximation Algorithms*. Springer, July 2001.

[61] Westbrook, J., and Yan, D. C. K. Greedy algorithms for the on-line steiner tree and generalized steiner problems. In *WADS* (1993), F. K. H. A. Dehne, J.-R. Sack, N. Santoro, and S. Whitesides, Eds., vol. 709 of *Lecture Notes in Computer Science*, Springer, pp. 622–633.

[62] Yao, A. C.-C. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *FOCS* (1977), IEEE, pp. 222–227.

[63] Zarrabi-Zadeh, H., and Chan, T. M. An improved algorithm for online unit clustering. *Algorithmica 54*, 4 (2009), 490–500.

[64] Zelikovsky, A. An 11/6-approximation algorithm for the network steiner problem. *Algorithmica 9*, 5 (1993), 463–470.