University of Paderborn
Germany

Doctoral Thesis

Online Resource Leasing

Christine Markarian

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Department of Computer Science
Supervised by: Prof. Dr. Friedhelm Meyer auf der Heide

June 2015

Reviewers

- Prof. Dr. Friedhelm Meyer auf der Heide
- Prof. Dr. Christian Scheideler

Declaration of Authorship

I, Christine Markarian, declare that this thesis entitled as 'Online Resource Leasing' and the work presented in it are my own. I confirm that:

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

Signed: _____

Date: _____

To my beloved parents.

"A man's heart plans his way but the Lord directs his steps".
Proverbs 16:9

Acknowledgements

I wouldn't be writing these lines nor would have completed this Ph.D. thesis without the support and encouragement of many people who played a significant role throughout this very unique chapter of my life.

Not only did he give me the opportunity to start my Ph.D. journey, but he gave me his trust, his time, his ideas, his advice for all matters, and a very homely atmosphere at all times. He is my supervisor Prof. Dr. Friedhelm Meyer auf der Heide. I even feel happy right now for finally having the chance to tell him: *Thank you.*

A special thanks goes to Prof. Dr. Christian Scheideler who accompanied me in one of the most critical periods of my journey and accepted to review this thesis. *Thank you.*

I would like to thank Prof. Dr. Franz-Josef Rammig and Prof. Dr. Holger Karl for serving on my dissertation committee. Furthermore, I am very grateful for the support of the international graduate school (IGS) which granted me a 3-year fellowship. *Thank you.*

They with whom I spent hours of thinking. They who made research enjoyable. They who accompanied me not only when papers got accepted but when proofs failed as well. They are my colleagues: Michael Schubert, Sebastian Abshoff, Peter Pietrzyk, Peter Kling, Shouwei Li, Sören Riechers, Matthias Feldotto, Alexander Skopalik, and Alexander Mäcker whom I owe much for reviewing this thesis. *Thank you.*

My Ph.D. journey was not only about designing and analyzing algorithms. In fact, it was way more than that. It was leaving my country for the first time to a country I knew nothing about its language, culture, and rules. I can not deny the difficulty I had here, specially during my first months, but *they* were there for me. They cheered me up when I felt homesick. They made me smile when things went gloomy. They were happy for seeing me happy. They are my friends, from my country and those I met here. I am happy you were there for me. *Thank you.*

Among them is Yara Khaluf whom I met here almost since the beginning of my journey. She who was there for me in every stage, as a member of my dissertation committee, as a mentor, as a friend, and I'm proud to say, as a family too. *Thank you.*

They to whom I nag when I need to complain. They who support me in every decision. They who endure all my moods. They are my brothers Sevag and Gregory. *Thank you.*

They who waited for me on Skype *everyday* for three years. Yes, everyday. They who give me energy when I get weak. They who feel with me before even sharing. They are

my parents to whom I dedicate this thesis. If I am here right now, it is because of them. *Thank you.*

In the first year, I met a person I had heard of before but never bothered to know more. Meeting Him again and knowing the real Him was the best thing ever happened to me. He introduced me to myself, to life, to the real meaning of all we do in life, including this thesis. I call him my Father - His name is God. I only am who I am because of You. THANK YOU.

Abstract

Many markets have seen a shift from the idea of *buying* and moved to *leasing* instead. Arguably, the latter has been a major catalyst for their success. In the wake of this shift, we study in this thesis leasing concepts from an algorithmic perspective. In particular, we design theoretic models, study their inherent difficulty, and devise provably good (often optimal), efficient algorithms, with the goal to cope with real-world resource leasing scenarios.

A major difficulty faced by most of these markets is the uncertainty of future demands. Consider a subcontractor who leases expensive resources from other companies to rent them out to clients. The subcontractor may buy long/expensive leases for some resource, just to realize later on that no more requests are issued for this resource in subsequent time steps. Or, the subcontractor may buy short leases, just to notice later on that having bought a longer lease would have cost less.

In attempt to capture this difficulty, our algorithms tend to be *online*, thus providing solutions in the present without knowing the future.

Zusammenfassung

Auf vielen Märkten beobachten wir eine Verschiebung vom Konzept des *Kaufens* zu dem des *Leasings*. Dabei stellt letzteres einen wesentlichen Katalysator für den Erfolg der Märkte dar. Als Folge dieser Verschiebung befassen wir uns in dieser Thesis mit dem Konzept des Leasings aus einer algorithmischen Perspektive. Insbesondere entwerfen wir theoretische Modelle, untersuchen ihre inhärente Schwierigkeit und erarbeiten beweisbar gute (und häufig optimale) und effiziente Algorithmen mit dem Ziel einen Umgang mit echten Leasing-Situationen zu ermöglichen.

Eine wesentliche Problematik, mit der sich viele der betrachteten Märkte konfrontiert sehen, ist die Unsicherheit bezüglich der zukünftigen Nachfrage. Man betrachte beispielsweise einen Subunternehmer, der kostspielige Ressourcen von anderen Unternehmen least, um diese an seine Kunden zu vermieten. Der Subunternehmer könnte lange/teure Leasings für eine Ressource nutzen und anschließend bemerken, dass keine weiteren Anfragen für diese Ressource gestellt werden. Auf der anderen Seite könnte sich der Subunternehmer für kurze Leasings entscheiden und daraufhin feststellen, dass ein längeres Leasing günstiger gewesen wäre.

Um diese Schwierigkeit zu erfassen, sind unsere Algorithmen vornehmlich *online* und ermöglichen somit Lösungen ohne die Zukunft im Voraus zu kennen.

Contents

List of Figures

Notation

Δ: maximum cardinality of sets

δ: maximum number of sets an element belongs to

K: number of lease types

$l_{\max}$: maximum lease length

l_{min}: minimum lease length

l_k: length of lease type k

d_{max}: longest interval length

d_{min}: shortest interval length

ILP: integer linear program

LP: linear program

Chapter 1

Introduction

Leasing as a model for temporarily acquiring access to goods or services - in contrast to buying physical ownership - is certainly not a new business model. But given the rise of digital services, in which physical ownership is at best a blurry concept, leasing is rapidly becoming not only an alternative but *the* predominant business model in many markets. As an example, consider the cloud computing market: Companies and research institutions needing access to high-tech infrastructure no longer have to acquire or even build their own server farms but can lease computing time from cloud computing services such as Amazon AWS, Microsoft Azure, or Google App Engine. These provide different types of affordable, fine granular, and scalable access to computational power. Arguably, one of the major catalysts for these services' success is their degree of flexibility: There is no need for a huge, upfront investment but instead anyone can leverage access to big data methods, fast and reliable storage, or scalable web services. A recent technical report on the development and challenges of cloud computing can be found in [1].

In the light of this development, we will study in this thesis leasing concepts from an algorithmic perspective. In particular, we will design theoretic models, study their inherent difficulty, and devise provably (optimal) efficient online algorithms, with the goal to cope with real-world leasing scenarios.

Given that at the scale of the above mentioned cloud service providers, even small profit

improvements can have a considerable influence on their revenue, it seems necessary to gain a better understanding of real-world leasing scenarios. This not only is necessary for cloud service providers, but for cloud users needing access to resources in the cloud be it for a private use, research need, or a business requirement, as well. With leasing, small businesses are encouraged, larger ones are given more benefits, and high-tech resources become easily accessible:

- Drop in initial expenses: There is no need for a large outlay of cash upfront so as cost can be spread over a larger period of time. This significantly helps maintain cash flow, which is critical to all businesses: Poor cash flow is a major cause of small business failures.
- Flexible terms: Leased resources need not be worried about in comparison to purchased ones: One can easily end a lease, renew it, or change its duration.
- Up-to-date equipment: Efficient, reliable, and high-tech equipment are often too expensive to buy outright. Leasing not only makes the latter possible, but also allows frequently replacing obsolete equipment with up-to-date ones.

A major difficulty faced by most businesses is the uncertainty aspect of future demands: It is often not clear which demands will be requested in the future. Consider a truck company that leases trucks from a third party based on demands needed. It might happen that the company buys long/expensive leases for some truck, just to realize later on that no more requests for this truck are issued in subsequent time steps. Or, the company buys short leases, just to notice later on that having bought a longer lease would have cost less.

In pursuit of keeping up with current markets, a sound, systematic, and scientific understanding of how to behave in face of such uncertainties is a gap we, scientists, are expected to fill. I hope with this thesis I can take part in filling this gap.

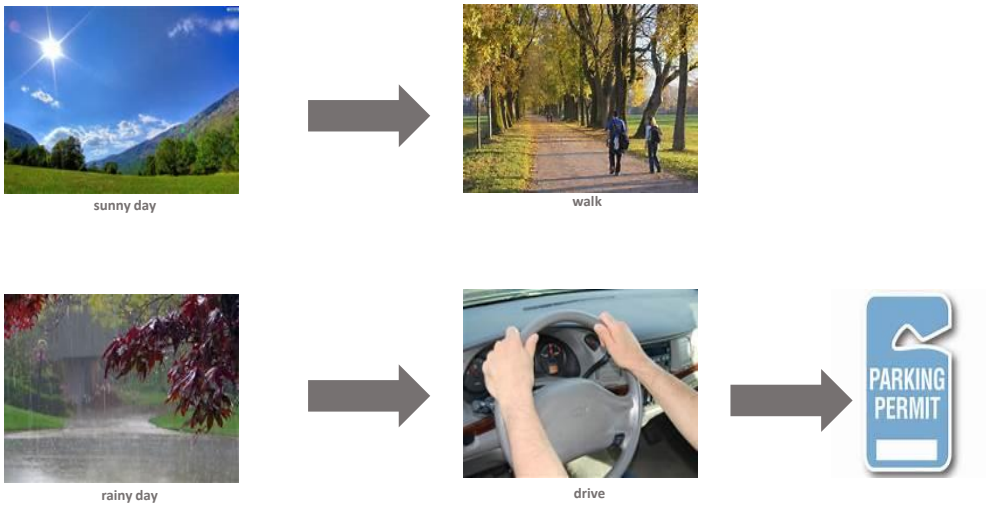


FIGURE 1.1: Parking Permit Problem

1.1 The Price of Leasing Online

The 'price of leasing online' is the price we pay for not knowing the future while making our leasing decisions. In attempt to realize this price on a theoretical level, Meyerson [2] introduced the first leasing model with a simple daily-life problem: the PARKINGPERMITPROBLEM. In this problem, each day (depending on the weather) we have to either use the car (if it is rainy) or walk (if it is sunny). In the former case, we must have a valid parking permit, which we choose among different types of permits, each having a different duration and price. The goal is to buy a set of permits in order to cover all rainy days while minimizing the total cost of purchases (without using weather forecasts). See Figure 1.1.

This simple problem, in which a *single* resource (a permit) is leased, captures the main concept of leasing online. This thesis falls into a series of works that extend this concept to more sophisticated problems - such as involving *multiple* resources - thereby incorporating the leasing concept into classical optimization problems, in which decisions are assumed to be final (resources are *bought*).

Consider, as an example of multiple resources, a company that provides services in a

network it does not own and must therefore lease multiple nodes in order to use them as servers. A customer may get the service from a server until the lease at the server node expires, after which the node must be leased again to be used as a server. The main problem the company is faced with is the unpredictable behavior of customer nodes: it is not known which nodes at which point of time will request the service. Thus, the company might buy long and expensive leases for some nodes, just to realize later on that no more requests are issued to those nodes.

At the core of this example lies a complex *infrastructure problem*. The term infrastructure problems is used to refer to classical optimization problems that consider scenarios where one acquires certain goods or resources (e.g., facilities, network nodes, or network connections) in order to generate or improve a given infrastructure (e.g., a supply network).

1.2 Thesis Overview

My thesis aligns with a prominent body of literature built upon the seminal work [2] by Meyerson, who initiated the study of leasing in theoretical research. It has three parts, the first two of which study two infrastructure problems, respectively. Part one (Chapter 3) introduces and gives results for SetMulticoverLeasing. Part two (Chapter 4) defines and gives results for FacilityLeasing. Part three (Chapter 5) extends the leasing model by Meyerson [2] to a natural model with the aim to capture more real-world leasing scenarios.

A short overview of results obtained in this thesis and their connection to the cloud computing market are given in the following two publications, respectively.

> Christine Markarian and Friedhelm Meyer auf der Heide. "Online Resource Leasing". To appear in: *Proceedings of the 34th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, **2015** [3].

Sebastian Kniesburges, Christine Markarian, Friedhelm Meyer auf der Heide, and Christian Scheideler. "Algorithmic Aspects of Resource Management in the Cloud". In: *Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, **2014** [4].

The remainder of this section provides a short description of each part and the corresponding results.

Set Multicover Leasing. In this part, SetMulticoverLeasing is introduced. Elements U ($|U| = n$), each with some value p specifying the number of sets to be covered by, arrive over time and must be covered by sets from a family F ($|F| = m$) of subsets of U. Each set can be leased for K different periods of time. Leasing a set S for a period k incurs a cost c_{Sk} and allows S to cover its elements for the next l_k time steps. SetMulticoverLeasing asks to minimize the total cost of sets leased, such that each element arriving at time t with value p is covered by p different sets containing it and leased during time t. SetMulticoverLeasing generalizes SetCoverLeasing by setting $p = 1$ for all elements. The latter was introduced by Anthony et al. [5] who only studied the problem in the offline setting. In this part, the following is achieved:

(i) SetMulticoverLeasing is introduced and an $\mathcal{O}(\log(\delta \cdot K) \log n)$-competitive online algorithm for the latter, where δ is the maximum number of sets an element belongs to, is given.

(ii) This algorithm is modified to yield the first competitive online algorithm for SetCoverLeasing, with a competitive factor $\mathcal{O}(\log(\delta \cdot K) \log n) = \mathcal{O}(\log(m \cdot K) \log n)$.

(iii) This also implies:

- An optimal $\mathcal{O}(\log \delta \log n)$-competitive algorithm for OnlineSetMulticover introduced by Berman and DasGupta [6] which is a special case of SetMulticoverLeasing if we just set $K = 1$ and $l_1 = \infty$.

- An improvement for OnlineSetCoverWithRepetitions introduced by Alon et al. [7] in which elements arrive over time and may appear multiple times such that an element must be covered by a different set at each arrival, from $\mathcal{O}(\log^2(m \cdot n))$ [7] to $\mathcal{O}(\log \delta \log(\delta \cdot n)) = \mathcal{O}(\log m \log(m \cdot n))$.

The results in this part are based on the following publication.

> Sebastian Abshoff, Christine Markarian, and Friedhelm Meyer auf der Heide. "Randomized Online Algorithms for Set Cover Leasing Problems". In: *Proceedings of the 8th Annual International Conference on Combinatorial Optimization & Applications (COCOA)*, **2014** [8].

Facility Leasing. FacilityLeasing was introduced along with SetCoverLeasing by Anthony et al. [5] who gave offline algorithms for the two problems. Clients D ($|D| = n$) arrive over time and must be connected to open facilities F ($|F| = m$). Each facility can be leased for K different periods of time. Leasing a facility i for a period k incurs a cost c_{ik} and ensures that i is open for the next l_k time steps. Connecting a client j to facility i incurs a connecting cost d_{ij}. FacilityLeasing asks to connect each client to an open facility while minimizing the total leasing and connecting costs. Nagarajan et al. [9] gave the first online algorithm, with an $\mathcal{O}(K \log n)$-competitive factor for the problem. In this part, the following is presented:

(i) The first online algorithm, with a time-independent competitive factor $\mathcal{O}(l_{\max} \log(l_{\max}))$, where $l_{\max}$ denotes the maximum lease length, is given.

(ii) The algorithm is shown to have an $\mathcal{O}(\log^2(l_{\max}))$-competitive factor for many 'natural' cases, such as, situations in which the number of clients arriving in each time step does not vary too much, or is non-increasing, or is polynomially bounded in $l_{\max}$.

The results in this part were achieved by Kling et al. in [10] and later incorporated into the following publication. They are presented here in this thesis for completeness.

Sebastian Abshoff, Peter Kling, Christine Markarian, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. "Towards the Price of Leasing Online". To appear in: *Journal of Combinatorial Optimization (JOCO)*, **2015** [11].

Flexible Demands. In this part, a new model for online leasing problems is introduced. In this model, demands with *deadlines* arrive over time and need to be served by leased resources. A resource can be leased for K different periods of time each incurring a different cost, such that longer leases cost less per unit time. Each demand j can be served anytime between its arrival a_j and its deadline $a_j + d_j$. The objective is to meet all deadlines while minimizing the total leasing costs. This model is a natural generalization of Meyerson's ParkingPermitProblem [2] in which $d_j = 0$ for all j. In this part, the following is achieved:

(i) A new leasing model that considers demands with deadlines is introduced.

(ii) An online algorithm for the proposed model, with a $\Theta(K + \frac{d_{max}}{l_{min}})$-competitive factor where d_{max} and l_{min} denote the largest d_j and the shortest available lease length, respectively, is given.

(iii) The SetCoverLeasingWithDeadlines problem, an extension of SetCoverLeasing which includes deadlines, is introduced.

(iv) An online competitive algorithm for SetCoverLeasingWithDeadlines, which also improves results for SetCoverLeasing, is proposed.

The results in this part are based on the following publication.

Shouwei Li, Alexander Mäcker, Christine Markarian, Friedhelm Meyer auf der Heide, and Sören Riechers. "Towards Flexible Demands in Online Leasing Problems". In: *Proceedings of the 21st International Computing and Combinatorics Conference (COCOON)*, **2015** [12].

The next section discusses an interesting connection between the results obtained in this thesis and the cloud computing market.

1.3 Leasing in the Cloud

"Security is still the elephant in the cloud" said an information security architect who builds security cloud services for international software vendors and whose team currently has the largest adopted cloud encryption solutions worldwide [13].

Although organizations are becoming more comfortable with the idea of putting at least some of their resources into the cloud, this is still a comparatively new paradigm and thus introduces a high level of uncertainty and questions regarding how *secure* the cloud is. Ensuring a level of security to costumers, however, is clearly not an easy task. At best, costumers are advised to pay particular attention to contractual language addressing security-related issues. Nevertheless, cloud contracts often have vague terms regarding the maintenance of data confidentiality, reliability, and recovery, making it difficult to rely on cloud providers and manage risk. Consequently, costumers either decide to stay away from the cloud - despite the benefits they would get, or are tempted to pass the task off to a subcontracting company under the "the more eyes on it, the better" theory, thus handing over full responsibility to a third party. This party is not only expected to assure reliability to clients but to promptly respond to their expectations and requirements.

It is well argued that involving subcontracting companies leads to a wider diversity of providers in the cloud computing market thus avoiding the latter to be dominated by few large providers. As a result of competition, a variety of prices is expected, thus giving strong cost benefits for both providers and clients. These benefits are not only restricted to cost, but benefits resulting from sharing resources as well - which already constitute a key strength of the cloud computing market and become more prominent when subcontractors come into play.

In this thesis, we will be the subcontractors - who despite making the life of clients easier and the cloud a more desirable place, are faced with difficult challenges - challenges we try to soften by identifying critical scenarios, proposing solutions, and calling research for help.

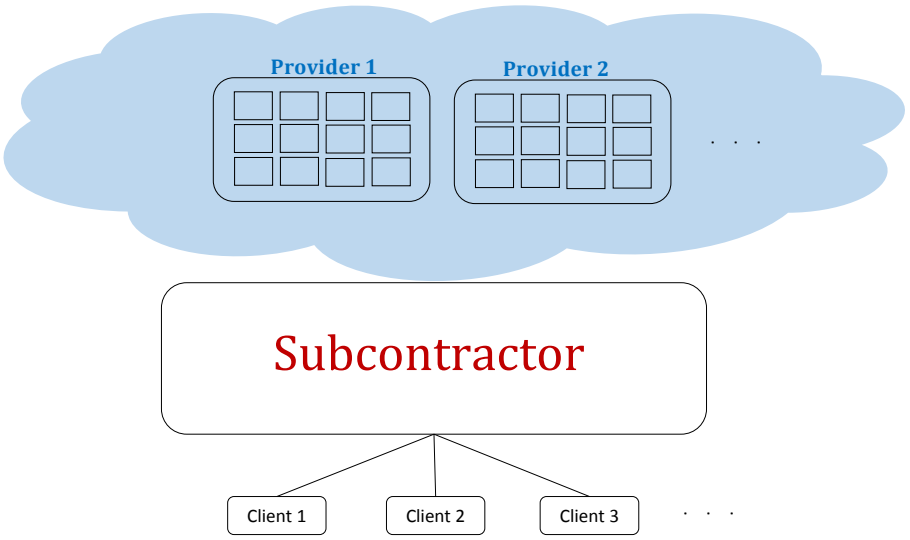


FIGURE 1.2: Leasing in the Cloud

Imagine you are a subcontractor and you have promised to satisfy all your costumers not only by fulfilling their demands on time but by also giving them best possible prices. You have agreed with a number of providers - all giving similar services but at different costs. See Figure 1.2 for an illustration. Each day, you may receive a phone call from a costumer asking to use a machine in the cloud - which although is offered by all providers, the distance between the client and the provider affects the price and so you need to make smart decisions. The closer the provider to the client is, the cheaper is the *connection cost*. The overall price the client pays for using the machine depends on the connection cost and the provider you pick - clearly you need to pay the provider and here comes your challenge. As is typical for accessing machines in the cloud, you need to decide how long to *lease* the machine - obviously, the longer the lease, the less you pay per unit time. If you could know who will call on each day, things would have been easy - there are plenty of efficient *offline* algorithms in the literature which one can use. But the problem here is that you do not know *which* clients will demand *which* machines in the future. Yet, you still need to decide *when* to lease a machine, from *which* provider, and for *how long*. Chapter 4 of this thesis handles typically such scenarios.

Now what if, each provider offers only some of the services and all connection costs are waved. You are asked to answer all phone calls and satisfy each costumer by selecting

an appropriate provider which is offering the requested service. Again, for the same reason of not knowing future phone calls, you are faced with a similar challenge - one Chapter 3 of this thesis tries to solve.

Imagine you receive a phone call from a client saying: "I would like to use machine 'X' but I do not mind to wait as long as you can give me a better price - I just need to use it any time before three weeks". For a moment, you are happy because you have more freedom - but does this make your decision any easier? We answer this question in Chapter 5 and propose online algorithms for such situations.

I can go on with more of such scenarios but as long as I will have to go beyond the scope of this thesis, I'd rather stop here and head to the technical contribution of this thesis, in the coming chapters.

Chapter 2

Preliminaries

The aim of this chapter is to familiarize the reader with a background necessary to grasp the research done in this thesis. Those familiar with approximation, online, and primal-dual algorithms may skip the first section (Section 2.1). Section 2.2 provides the first theoretic results in leasing: deterministic and randomized provably optimal algorithms for the ParkingPermitProblem. The last section (Section 2.3) describes a framework that transforms a given online problem with certain properties to its leasing variant. Both problems SetMulticoverLeasing and FacilityLeasing studied in Chapters 3 and 4, respectively, are derived from this framework.

2.1 Approximation, Online, & Primal-dual Algorithms

The main purpose of this section is to give the reader an intuitive idea and understanding of how we measure the quality of our algorithms. To do so, we define approximation, online, and primal-dual algorithms.

Approximation Algorithms. Many real-world optimization problems (e.g., nurse scheduling, logistics, flights management) are often hard to solve optimally within a reasonable period of time. In scientific terms, these problems are referred to as NP-hard

problems, which cannot be solved in time polynomial in the input size if the complexity classes NP and P are not equal. From here comes the idea of *approximation algorithms*, which are algorithms running in time polynomial in the input while computing solutions that are not necessarily optimal but provably close to optimal. More detailed introduction to approximation algorithms can be found in [14].

To measure the quality of an approximation algorithm, *worst-case approximation ratio* is commonly used.

Definition 2.1. (Worst-case Approximation Ratio) Given a minimization problem and a set Π of all feasible instances of this problem. Let $Opt(I)$ denote the cost of an optimal solution for an instance $I \in \Pi$. We say an approximation algorithm Alg has a worst-case approximation ratio γ if:

$$\gamma := \sup_{I \in \Pi} \frac{Alg(I)}{Opt(I)}$$

where $Alg(I)$ is the cost of Alg for an instance I and $\gamma = 1$ if Alg is optimal.

Online Algorithms. Approximation algorithms are essential should time be crucial and so we trade quality for computation time. There are other real-world problems, however, in which time is not necessarily prominent. Instead, we lack critical information such as the *future*. This is specifically true in the business world, in which demands are not known in advance, or at least nothing is guaranteed about the future. Despite that, we are expected to make smart decisions to keep the businesses alive. Computer science models such scenarios as *online problems* where online means decisions are to be made on-the-fly without having the entire input in advance. The difficulty in an online problem lies in the irrevocable decisions: We cannot go back in time and change our mind. More detailed introduction to online algorithms can be found in [15].

To measure the quality of an online algorithm, *worst-case competitive ratio* is commonly used. Here, we basically compare the cost of an online algorithm to that of an optimal offline algorithm, which knows all the future.

Definition 2.2. (Worst-case Competitive Ratio) Given a minimization problem and a set Π of all feasible instances of this problem such that only part of an instance is revealed in each time step. Let $Opt(I)$ denote the cost of an optimal offline solution for an instance $I \in \Pi$. We say an online algorithm Alg has a worst-case competitive ratio γ (or, γ-competitive) if:

$$\gamma := \sup_{I \in \Pi} \frac{Alg(I)}{Opt(I)}$$

where $Alg(I)$ is the cost of Alg for an instance I and $\gamma = 1$ if Alg is optimal.

Throughout this thesis, we consider that instances are revealed by an *adaptive* adversary which knows all the actions taken by an online algorithm thus far and can accordingly choose the next instances. Moreover, we will talk about *deterministic* and *randomized* online algorithms. Unlike a deterministic algorithm, the worst-case competitive ratio of a randomized algorithm is measured using the *expected* cost resulting from the random choices made by the randomized online algorithm.

Primal-dual Algorithms. The core of a primal-dual algorithm is a linear program. A linear program (LP) is an optimization (minimization or maximization) of a linear objective function over a feasible set defined by a system of linear inequalities. A key aspect of optimization problems is that they come in pairs: Every minimization problem has a maximization counterpart and vice versa. In Figure 2.1, we find a linear program of a minimization problem, such that $A \in R^{n \times m}$ is a matrix and $c, x \in R^m$ are column vectors. The upper part is called the primal program (the original problem) and its counterpart in the lower part is the dual program. x and y are called *primal variables* and *dual variables*, respectively. Similarly, the inequalities $Ax \geq b$ and $A^T y \leq c$ are referred to as *primal constraints* and *dual constraints*, respectively. An integer linear program (ILP) is a special case of LP where $x \in \{0, 1\}$. An introduction to linear programming can be found in [16].

At a high level, a primal-dual algorithm works as follows:

$$\begin{array}{c} \min c^T \cdot x \\ \text{Subject to: } Ax \geq b \\ x \geq 0 \end{array}$$

$$\begin{array}{c} \max b^T \cdot y \\ \text{Subject to: } A^T y \leq c \\ y \geq 0 \end{array}$$

FIGURE 2.1: Linear Program

- Formulate a given optimization problem as a linear program.
- Construct a dual solution.
- Derive a primal solution from the dual solution.

Primal-dual algorithms have served as efficient approximation algorithms for many NP-hard problems, not only in the offline but in the online setting as well. For more detailed introduction to primal-dual algorithms, see [17].

The reason for their success mainly goes to the properties of linear programs and in particular, the relationship between primal and dual programs. Among these properties is the weak duality theorem, defined as follows.

Theorem 2.3. *(Weak Duality) Let x be a feasible solution for the primal program and y a feasible solution for the dual program, then*

$$c^T \cdot x \leq b^T \cdot y$$

Proof. Since y is a feasible solution for the dual program and $x \geq 0$, we have

$$c^T \cdot x = x^T \cdot c \leq x^T \cdot (A^T \cdot y)$$

Since x is a feasible solution for the dual program and $y \geq 0$, we deduce

$$x^T \cdot (A^T \cdot y) = (Ax)^T \cdot y \leq b^T \cdot y$$

□

This means that any dual solution is a tight lower bound to a primal solution. Another useful theorem regarding the optimal solution is described in the following.

Theorem 2.4. *(Strong Duality) Let x^* be an optimal solution for the primal program and y^* an optimal solution for the dual program, then*

$$c \cdot x^* = b \cdot y^*.$$

2.2 The Parking Permit Problem

Before we head to complex leasing problems, it is important to have an understanding of the first theoretic model, algorithms, and results in leasing. As mentioned earlier, Meyerson [2] introduced the first leasing model with the ParkingPermitProblem. In this section, a formal definition of the problem along with two online approaches (deterministic and randomized) proposed by Meyerson, are given.

2.2.1 The Leasing Model

In this section, a formal definition of the ParkingPermitProblem followed by a simplified version of the latter are presented.

Problem Definition. On each day t, we are either given a client (sunny day) or not (rainy day). There are K different types of leases (permits), each with its own duration and cost (longer leases tend to cost less per day). A client arriving on day t is *served* if there is a lease in the solution which covers t. The goal is to buy a set of leases such that all arriving clients are served while minimizing the total cost of purchases.

$$\min \sum_{(k,t)\in L} x_{(k,t)} \cdot c_k$$

$$\text{Subject to: } \forall t' \in D : \sum_{(k,t)\in L, t'\in[t,t+l_k]} x_{(k,t)} \geq 1$$

$$\forall (k,t) \in L : x_{(k,t)} \in \{0,1\}$$

$$\max \sum_{t'\in D} y_{t'}$$

$$\text{Subject to: } \forall (k,t) \in L : \sum_{t'\in D, t'\in[t,t+l_k]} y_{t'} \leq c_k$$

$$\forall t' \in D : y_{t'} \geq 0$$

FIGURE 2.2: ILP Formulation of the PARKINGPERMITPROBLEM

To allow more coherence throughout the thesis, particularly in regard to Chapter 5 which extends the PARKINGPERMITPROBLEM model within the primal-dual scheme, we will formulate the latter as an integer linear program (ILP).

Figure 2.2 shows this formulation. A lease of type k has cost c_k and length l_k. l_{min} and l_{max} denote the shortest and the longest lease length, respectively. We refer to a type k lease starting at time t as (k,t), a client arriving on day t as t, and an interval $[a, a+b]$ as I_a^b. The collection of all leases is L and the collection of all clients is D. We say a lease $(k,t') \in L$ is a *candidate* to client $t \in D$ if $t \in I_{t'}^{l_k}$. The sum in the objective function represents the costs of buying the leases. The indicator primal variable $X_{(k,t)}$ tells us whether lease (k,t) is bought or not. The primal constraints guarantee that each client $t \in D$ is served. A dual variable Y_t is assigned to each client t.

A Simplified Model. As a convenience for his analysis, Meyerson introduced a model he referred to as the *interval model*. The interval model simplifies the original leasing model at the cost of a small constant factor in the competitive ratio.

Definition 2.5. (Interval Model) Leases in the interval model satisfy the following two properties:

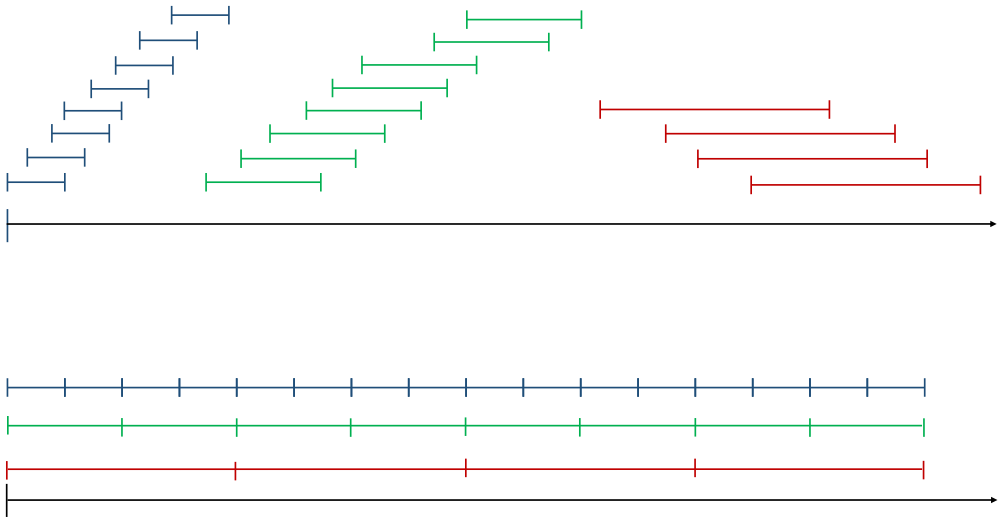


FIGURE 2.3: Interval Model

In the upper image, leases can have arbitrary length and start at arbitrary times. In the lower image (interval model), we allow only lengths that are a power of two and are non-overlapping for each lease type. The proof of Lemma 2.6 merely uses that we can map between the two models by opening two consecutive intervals of the same time without sacrificing feasibility.

- Lease lengths l_k are powers of two.
- Leases of the same type do not overlap.

The following lemma states the effect of the interval model on the competitive ratio. An illustration can be found in Figure 2.3.

Lemma 2.6. *Any c-competitive leasing algorithm for the interval model yields a $4c$-competitive leasing algorithm for the original model.*

Proof. Consider a leasing problem instance I for general leases and construct a new instance I' by rounding each lease length $l_k \in N$ to the next power of two. That is, the lease lengths of I' are $l'_k := 2^{\lceil \log l_k \rceil}$. Let S' denote the solution constructed by the c-approximation algorithm for the interval model when given I'. From S', we construct a solution S for I as follows: for each lease of type k bought at time t in solution S', buy two consecutive leases of type k at times t and $t + l_k$. Since $l_k + l_k \geq l'_k$, any lease pair in S covers at least all the demands covered by the original lease in S'. Moreover, we have

$\text{cost}(S) = 2\,\text{cost}(S') \leq 2c\,\text{cost}(OPT')$, where OPT' denotes an optimal solution for I' *in the interval model*. Now, note that an optimal solution OPT for I in the general model yields a solution $\tilde{S}$ for I' in the interval model as follows: for each lease of type k bought at time t in solution OPT, buy two leases of type k at times $\lfloor \frac{t}{l'_k} \rfloor \cdot l'_k$ and $\lfloor \frac{t}{l'_k} \rfloor \cdot l'_k$. These leases cover at least all the demands of the original lease and obey the interval model. Thus, we get $\text{cost}(OPT') \leq \text{cost}(\tilde{S}) = 2\,\text{cost}(OPT)$. The lemma's statement follows by combining both inequalities. □

2.2.2 Deterministic Approach

Meyerson gave a deterministic competitive algorithm for the ParkingPermitProblem and showed that it is optimal. In what follows, a description of the algorithm, its analysis, and the deterministic lower bound achieved - i.e., no deterministic algorithm for the problem can obtain better than this bound - are presented.

Deterministic Algorithm. Although Meyerson does not explicitly present his deterministic algorithm in a primal-dual fashion, his algorithm is, in essence, a primal-dual algorithm and can be described as follows.

Algorithm 1 Deterministic Primal-dual Algorithm for the ParkingPermitProblem

When a client t' arrives,

(i) increase the corresponding dual variable $y_{t'}$ until the dual constraint corresponding to a candidate (k, t) becomes tight.

(ii) set the primal variable $x_{(k,t)}$ of every tight candidate to one.

Analysis. Meyerson proved the competitive ratio of his algorithm using induction. Nevertheless, it is also possible to use primal-dual techniques to do so, as we shall see in what follows.

The theorem below states the competitive ratio of the primal-dual algorithm described above.

Theorem 2.7. *Algorithm 1 yields a competitive ratio $\mathcal{O}(K)$ for the* ParkingPermit-Problem.

Proof. Since Algorithm 1 never violates the dual constraints (it stops increasing the dual variables until one constraint is tight), the dual solution constructed is clearly feasible. Furthermore, since the algorithm makes sure the primal variable of at least one candidate is set to 1 for each arriving client, the primal solution is also feasible. It remains to show a relationship between the cost of the primal solution and that of the dual solution, in order to prove the competitive ratio as we shall see next.

Let $P \subseteq L$ denote the primal solution constructed by the algorithm. Since the dual constraint is tight for each $(k,t) \in P$, we have

$$c_k = \sum_{t' \in D : t' \in I_t^{t+l_k}} Y_{t'}. \tag{2.1}$$

Then, we have

$$\sum_{(k,t)\in P} c_k = \sum_{(k,t)\in P} \sum_{t' \in D : t' \in I_t^{t+l_k}} Y_{t'} = \sum_{t' \in D} Y_{t'} \sum_{(k,t)\in P : t' \in I_t^{t+l_k}} 1. \tag{2.2}$$

Assuming the interval model (Definition 2.6) for the leases, we have that on each day t, there are exactly K leases covering t and hence are candidates for a client arriving on day t. This means that the algorithm can only set the primal variables corresponding to at most K leases to 1. Thus,

$$\sum_{(k,t)\in P : t' \in I_t^{t+l_k}} 1 \leq K. \tag{2.3}$$

This implies that the cost of the primal solution ($\sum_{(k,t)\in P} c_k$) is upper bounded by K times the cost of the dual solution ($\sum_{t' \in D} Y_{t'}$). By weak duality (Theorem 2.3), we have that any any feasible dual solution is a lower bound to any feasible primal solution, which includes an optimal primal solution. This means the cost of the primal solution

constructed by the algorithm is upper bounded by K times the cost of an optimal primal solution, which concludes the proof. □

Lower Bound. The theorem below shows a lower bound to the best deterministic competitive ratio for the ParkingPermitProblem.

Theorem 2.8. *No deterministic algorithm for the* ParkingPermitProblem *whose competitive ratio depends solely on K can obtain better than $\Omega(K)$-competitive ratio.*

Proof. Consider an adversary with a very simple adaptive strategy: a client is given as long as the algorithm can not serve it with the leases bought so far. Assume that leases satisfy the interval model and there are K lease types with costs $c_k = 2^k$ and length $l_k = 2Kl_{k-1} = (2k)^k$. We now look at intervals of length l_k which have at least one client and divide them into three classes.

(i) The algorithm buys a lease of length l_k for this interval.

(ii) The algorithm buys a lease of length larger than l_k for this interval.

(iii) The algorithm does not buy a lease of length l_k or larger for this interval.

Let n_k, $n_j (j > k)$, and q_k denote the number of intervals of class (i), (ii), and (iii), respectively. We denote by *Alg* and *Opt* the cost of an online algorithm and an optimal offline algorithm, respectively. Now, we observe how two such algorithms behave.

Clearly, we have that $Alg = \sum_{k=1}^{K} n_k c_k$. Moreover, we can show that $Alg \geq Kq_k c_k$. Consider any interval of length l_k containing at least one client. We show by induction that any online algorithm must pay at least c_k on each such interval.

- (base case) k=0, obviously true.
- (hypothesis) Assume the algorithm pays at least c_{k-1} for an interval of length l_{k-1}.

- (induction step) If the algorithm buys a lease of type k, then it spends at least c_k. Otherwise, we can divide the interval into $2K$ intervals of length l_{k-1}, each starting with a day containing a client. The algorithm spends for each of these intervals, recursively, at least c_{k-1}, thus summing up to a total cost of at least $2Kc_{k-1} = Kc_k$ for an interval of length l_k.

An optimal offline algorithm, on the other hand, cannot do worse than purchasing a lease of type k for each interval of length l_k. Thus, we have that $Opt \leq c_k(q_k + \sum_{j \geq k} n_j)$. Summing up over all K lease types, we get

$$\begin{aligned} KOpt &\leq \sum_{k=1}^{K}(c_k(q_k + \sum_{j \geq k} n_j)) \\ &= \sum_{k=1}^{K}(n_k(\sum_{j=1}^{k} c_k + c_k q_k) \\ &\leq \sum_{k=1}^{K}(2n_k c_k + q_k c_k) \\ &\leq 3Alg \end{aligned}$$

□

2.2.3 Randomized Approach

Besides the optimal deterministic algorithm, Meyerson used randomization to improve the $\mathcal{O}(K)$-competitive ratio to $\mathcal{O}(\log K)$ and showed that it is optimal. In what follows, a description of the randomized algorithm, its analysis, and the lower bound achieved - i.e., no randomized algorithm for the problem can obtain better than this bound - are presented.

Randomized Algorithm. The core of Meyerson's randomized algorithm is a fractional solution - fractions of leases are bought (e.g., 1/4, 1/3, 1/2) such that for each arriving client, fractions of its candidates sum up to 1. The fractional version of the ParkingPermitProblem can be viewed as a relaxation of the linear program - $x_{(k,t)} \in [0,1]$ rather than $x_{(k,t)} \in \{0,1\}$. Meyerson's algorithm first constructs a fractional solution and then converts it online to an integer solution using randomization.

The algorithm maintains a fraction $f_{(k,t)}$ (which, for ease of description, will be referred to as f_k) for each lease $(k,t) \in L$ (similarly here: $k \in L$), initially set to zero and non-decreasing throughout the algorithm. Let Q_t be the collection of candidates of client t. The randomization used is restricted only to choosing a threshold τ uniformly at random between 0 and 1.

Algorithm 2 Randomized Algorithm for the ParkingPermitProblem

When a client t arrives,
(i) (fractional solution) While $\sum_{k \in Q_t} f_k < 1$,
$f_k = f_k \cdot (1 + 1/c_k) + \frac{1}{|Q_t| \cdot c_k}$
(ii) (integer solution) buy $k \in Q_t$ with $\sum_{i=k+1}^{K} f_i < \tau \leq \sum_{i=k}^{K} f_i$

Analysis. Meyerson proved that the randomized algorithm above has an $\mathcal{O}(\log K)$-competitive ratio.

To achieve his competitive ratio, he proved that:

(i) The cost of the fractional solution constructed is at most $\mathcal{O}(\log K)$ times that of an optimal offline solution.

(ii) The cost of the integer solution constructed is at most that of the fractional solution.

Proof of (i) When a client t arrives, in each while loop, the fraction of each candidate $(k,t') \in Q_t$ is increased by $\left(\frac{f_k}{c_k} + \frac{1}{|Q_t| \cdot c_k}\right)$. If we sum up over all candidates in Q_t, the total fractional cost increases by $\sum_{(k,t') \in Q} c_k \cdot \left(\frac{f_k}{c_k} + \frac{1}{|Q_t| \cdot c_k}\right) = \sum_{k \in Q_t} f_k + 1$. Since

$\sum_{k \in Q_t} f_k < 1$, the total fractional cost thus increases by 2 in each while loop. Moreover, in each while loop, at least one candidate $k \in Q_t$ is in the optimal solution. After c_k loops, the fraction of this candidate reaches to at least $\frac{1}{|Q_t|}$ due to the second part of the equation $\frac{1}{|Q_t| \cdot c_k}$. After this, the first part of the equation keeps multiplying the fraction by $(1 + 1/c_k)$ and stops when it becomes larger than 1. Hence, after $\mathcal{O}(c_k \cdot \log |Q_t|)$ loops, $\sum_{k \in Q_t} f_k$ will be greater than 1 and since $|Q_t| = K$, (i) follows. □

Proof of (ii) Note that $\sum_{i=k+1}^{K} f_i < \tau \leq \sum_{i=k}^{K} f_i$ ensures feasibility of the integer solution, because there will always be some lease k satisfying this inequality. To compute the expected cost, consider an interval $[t, t + l_k]$ and its corresponding lease. The algorithm may buy this lease on any day in this interval and, in particular, on a day in which $\tau > \sum_{i=k+1}^{K} f_i$ and $\tau \leq \sum_{i=k}^{K} f_i$ hold. The probability of buying a lease is thus bounded by the probability that τ falls between these two values. To complete the total expected cost, we sum up over all intervals of all l_k's. The next trick here is to exploit the following two facts:

- The first and last days of neighboring intervals coincide.
- $c_{i+1} \geq 2c_i$.

Manipulating the obtained equation using these two facts concludes the proof of (ii). □

Lower Bound. The theorem below shows a lower bound to the best randomized competitive ratio for the ParkingPermitProblem.

Theorem 2.9. *No randomized algorithm for the* ParkingPermitProblem *whose competitive ratio depends solely on K can obtain better than $\Omega(\log K)$-competitive ratio.*

Proof. Following Yao's minimax principle [18], which states that given an online problem, the competitive ratio of the best randomized online algorithm against any oblivious adversary is equal to the competitive ratio of the best deterministic online algorithm under some input distribution, Meyerson considered deterministic algorithms operating on

a randomized input instance. There are K lease types such that $c_i = 2^i$ for a lease type i. The duration of a lease type i is assumed to be arbitrarily larger than that of lease type $i-1$. The randomized instance is constructed as follows. We say an interval is *active* if the i-th subinterval of the interval is active with probability $(\frac{1}{2})^{i-1}$. This means that the first sub-interval and the top-level interval are always active. This recursion implies that an active interval corresponding to type 1 has a client. We now look at how a deterministic online algorithm and an optimal offline algorithm behave given this instance.

An online algorithm has to decide whether or not to buy a lease of type k for an active interval. Buying a lease of type k costs $c_k = 2^k$, and if the algorithm chooses to pay separately for each active sub-interval of type $k-1$ instead, it pays an expected cost of

$$\sum_{i=0}^{l_k-1} \frac{2^{k-1}}{2^i} = 2^{k-1} \sum_{i=0}^{l_k-1} 2^{-i} < 2^k. \tag{2.4}$$

This means that it is cheaper for the algorithm to buy shorter leases and this holds for any type of lease. This implies that the algorithm is to buy only type 1 leases every time a client arrives. Thus, the expected cost of the algorithm will be the expected number of clients. Let a_k be the expected number of active intervals of length l_k. We have that $a_k = a_{k-1} \cdot \sum_{i=0}^{l_k-1} 2^{-i}$. If l_k is assumed to be arbitrarily large for any k, then we can say that $a_k \approx 2a_{k-1}$. This implies that $a_K \approx 2^K$, where $a_1 = 1$.

An optimal offline algorithm, on the other hand, can not do worse than buying a lease of type k for every interval containing at least $\log k + 1$ active sub-intervals. Meyerson showed, using induction, that the expected cost of an optimal offline algorithm will thus be upper bounded by $\frac{2^{K+1}}{\log K}$. □

2.3 The Leasing Framework

This section provides a general framework we proposed in [11], to transform a given suitable online problem to its leasing variant. We will later use this framework to

formally describe SetMulticoverLeasing and FacilityLeasing in Chapters 3 and 4, respectively.

The main property, we require, of the original (non-leasing) online problem is a *temporal covering* aspect: *demands* $j \in \mathcal{D}$ from a demand set $\mathcal{D}$ arrive over time and have to be covered by buying a suitable *infrastructure element* $i \in \mathcal{I}$ from some infrastructure set $\mathcal{I}$. We use the notation (j, t) to indicate a demand $j \in \mathcal{D}$ arriving at time $t \in \mathbb{N}$. Buying an infrastructure element i is associated with a (one-time) cost $c_i \in \mathbb{R}_{\geq 0}$. The covering happens on the fly (without knowledge of the number or properties of future demands) and represents an irrevocable decision (we cannot discard an already bought infrastructure element when we perceive a better solution later on). Examples of such problems include online vertex cover (where edges arrive over time and we buy nodes to cover them), online dominating set (where nodes arrive over time and we buy incident nodes to cover them), and online steiner trees (where terminals arrive over time and we buy edges to keep them connected).

Given a problem with such a temporal covering aspect, we transform it into its leasing variant by introducing $K \in \mathbb{N}$ different lease types. Lease type $k \in \{1, \dots, K\}$ is associated with a lease length $l_k \in \mathbb{N}$. We use $l_{\max} \coloneqq \max_k l_k$ to refer to the maximal lease length. Instead of buying an infrastructure element $i \in \mathcal{I}$, an algorithm *leases* i at some time $t \in \mathbb{N}$ using some lease type k. This incurs a *leasing cost* $c_{ik} \in \mathbb{R}_{\geq 0}$ and allows this infrastructure element to cover (suitable) demands during the time window $[t, t + l_k)$. We refer to the triple $\mathcal{I} \times \{1, \dots, K\} \times \mathbb{N}$ as the *infrastructure leasing set* and denote it by $\bar{\mathcal{I}}$. For a time t, we define $\bar{\mathcal{I}}(t) \coloneqq \left\{(i, k, t') \in \bar{\mathcal{I}} \colon t \in [t', t' + l_k)\right\}$ (i.e., all leases covering time t). By setting $K = 1$, $l_1 = \infty$, and $c_{i1} = c_i$, we get the original (non-leasing) online problem. Note that this transformation is independent of other problem aspects. For example, it does not interfere with when an infrastructure element is regarded *suitable* to cover a demand. We will apply such transformations in Chapters 3 and 4 to extend known algorithmic design techniques (e.g., greedy or primal-dual algorithms) to the leasing setting.

Chapter 3

Set Multicover Leasing

Resource allocation has become a major concern in many areas such as economy (resources are allocated by markets), project management (jobs are scheduled to resources), and networks (network nodes are assigned roles to be used as resources). Consider, as an example, a set of files and a number of servers each containing a subset of these files. Users arrive with time and request files among this set. Servers need to be activated in order to give access to users. The difficulty here is that we do not know in advance which files will be requested. Given that, we still need to decide *which* servers to activate, *when*, and for *how long*. Clearly, we want to minimize the cost of activating the servers, keeping in mind that the longer we keep a server active the less we pay for the server per unit time.

As another example, consider a subcontractor who leases expensive equipment from other companies and rents them out to clients. If the subcontractor knows that some equipment will be needed for ten whole years (e.g., each year a new client will come and request the equipment for one year), it is best to lease it *for ten years* from a company offering the equipment. Nevertheless, the subcontractor does not know what will happen in the future and so might lease the equipment *yearly for ten years* every time it is needed and pay a total of more instead.

At the core of these examples, we have a complex optimization problem, SetMulticoverLeasing, which will be our subject of study in this chapter.

Chapter Basis. The results presented in this chapter are based on the following publication.

> Sebastian Abshoff, Christine Markarian, and Friedhelm Meyer auf der Heide. "Randomized Online Algorithms for Set Cover Leasing Problems". In: *Proceedings of the 8th Annual International Conference on Combinatorial Optimization & Applications (COCOA)*, **2014** [8].

Chapter Outline. This chapter starts with an overview of related literature (Section 3.1) along with a summary of results obtained in this chapter. Section 3.2 presents a formal definition of SetMulticoverLeasing, gives some necessary notation, and describes the algorithmic techniques used throughout the chapter. The main results of this chapter comprising of an online algorithm and its analysis are presented in Sections 3.3 and 3.4, respectively. The chapter concludes with a short résume and future work in Section 3.5.

3.1 Related Work & Contribution

Before stating the results obtained in this chapter, let us have an overview of literature evolving around our problem at hand, SetMulticoverLeasing.

The latter roots from the classical $\mathcal{NP}$-hard optimization problem SetCover. Given a universe of elements U ($|U| = n$) and a family F ($|F| = m$) of subsets of U, each associated with a cost. SetCover asks to cover each element while minimizing the total cost of sets. SetCover has an $\mathcal{O}(\log n)$ approximation ratio [19–22], which is the best possible unless $\mathcal{P} = \mathcal{NP}$ [23]. SetCover has been studied as a more general version known as SetMulticover, in which all elements are required to be covered

by $p \geq 1$ different sets and has an $\mathcal{O}(\log \Delta) = \mathcal{O}(\log n)$ approximation ratio [14, 24], where Δ is the maximum cardinality of the sets. This problem is known as *uniform* SetMulticover for a fixed value of p and as *non-uniform* SetMulticover for a possibly different value of p for each set.

SetCover was also studied in the online setting [7, 25–30], the first online variant of which is known as the OnlineSetCover problem. Here, in each time step, an element from the universe U ($|U| = n$) arrives and must be covered by a family F ($|F| = m$) of subsets of U, each associated with a cost. OnlineSetCover asks to minimize the total cost of sets chosen. Alon et al. [25] gave an $\mathcal{O}(\log \delta \log n)$-competitive algorithm for unweighted OnlineSetCover, in which all sets have cost 1, and an $\mathcal{O}(\log m \log n)$-competitive algorithm for weighted OnlineSetCover, where δ is the maximum number of sets an element belongs to. They also showed that these bounds are nearly tight due to a lower bound of $\Omega\left(\frac{\log m \log n}{\log \log m + \log \log n}\right)$ which they give for a wide range of relations among m and n. The $\mathcal{O}(\log \delta \log n)$-competitive ratio for unweighted OnlineSetCover was improved by [27] to $\mathcal{O}(\log n / \mathrm{Opt} \cdot \log \delta)$, where *Opt* is the optimal offline solution to the problem. Korman [26], in his Master's thesis, gave a randomized lower bound of $\Omega(\log m \log n)$ for OnlineSetCover. In a more general form, Alon et al. [7] considered the OnlineSetCoverWithRepetitions problem within the larger context of admissions control in general networks. Here, an element may arrive multiple times and must be covered by a different set at each arrival. Alon et al. gave a randomized $\mathcal{O}(\log^2(m \cdot n))$-competitive algorithm and a deterministic bi-criteria algorithm for the problem. A very similar problem called OnlineSetMulticover, an online variant of SetMulticover, was studied by Berman and DasGupta [6]. As in the offline variant, all (arriving) elements are requested to be covered by p different sets. Motivated by applications in systems biology, Berman and DasGupta focused on tight analysis of approximability by improving results by Alon et al. [25] up to constant factors. Their competitive ratios were given in terms of maximum set size Δ, δ, and p. They also gave lower bounds $\Omega\left(\frac{\log m/p \log n/p}{\log \log m/p + \log \log n/p}\right)$ and $\Omega\left(\frac{\log m \log n}{\log \log m + \log \log n}\right)$ for unweighted and weighted OnlineSetMulticover, respectively.

Later, Anthony et al. [5] generalized Meyerson's ParkingPermitProblem to other

infrastructure problems including OnlineSetCover by introducing its leasing variant SetCoverLeasing and gave offline solutions for these problems. In SetCoverLeasing, sets are leased with K different types and costs. A set S leased with type k ensures that S can be used for the next l_k time steps. The same set can later be used by leasing it again. SetCoverLeasing asks to minimize the total leasing costs. Its connection to OnlineSetCover becomes apparent if we just set $K = 1$ and $l_1 = \infty$. By discovering an interesting relationship between infrastructure leasing problems and stochastic optimization, Anthony et al. [5] achieved $\mathcal{O}(\log n)$-approximation for SetCoverLeasing.

Contribution. In this chapter, we introduce SetMulticoverLeasing in which elements U ($|U| = n$), each with some value p specifying the number of sets to be covered, arrive over time and must be covered by sets from a family F ($|F| = m$) of subsets of U. Each set can be leased for K different periods of time. Leasing a set S for a period k incurs a cost c_{Sk} and allows S to cover its elements for the next l_k time steps. SetMulticoverLeasing asks to minimize the total cost of sets leased, such that each element arriving at time t with value p is covered by p different sets containing it and leased during time t. Then, we present an $\mathcal{O}(\log(\delta \cdot K) \log n)$-competitive algorithm for SetMulticoverLeasing. SetCoverLeasing is a special case of SetMulticoverLeasing if we just set $p = 1$ for all elements. The results obtained imply the first online algorithm for SetCoverLeasing and yield:

- An optimal $\mathcal{O}(\log \delta \log n)$-competitive algorithm for OnlineSetMulticover whose connection to SetMulticoverLeasing becomes apparent if we just set $K = 1$, $l_1 = \infty$, and $c_{S1} = c_S$ for each S, where c_{S1} is the cost incurred by set S leased for a duration l_1.
- An improvement for OnlineSetCoverWithRepetitions from $\mathcal{O}(\log^2(m \cdot n))$ [7] to $\mathcal{O}(\log \delta \log(\delta \cdot n)) = \mathcal{O}(\log m \log(m \cdot n))$.

See Figure 3.1 for an illustration of these models and their corresponding results.

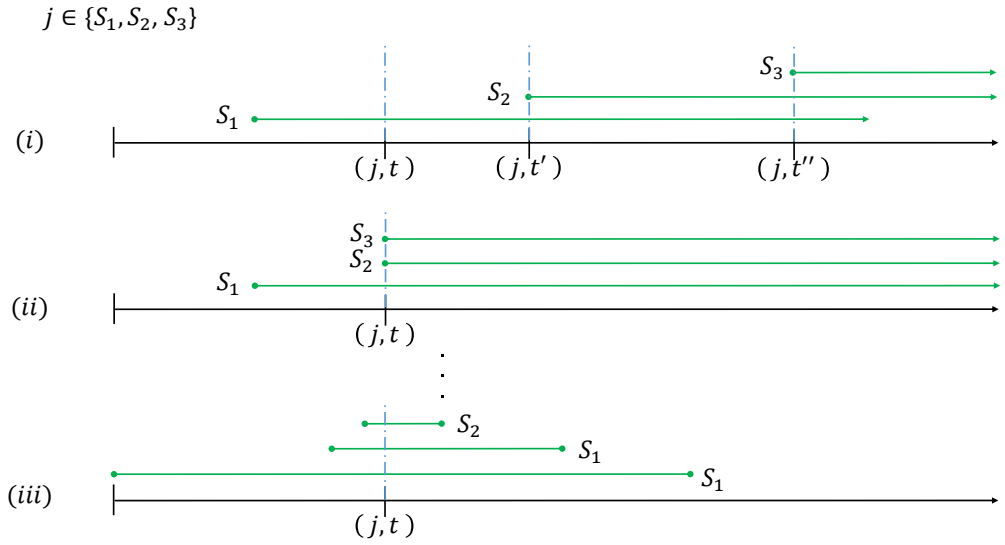


FIGURE 3.1: Set Cover Leasing Models

(i) ONLINESETCOVERWITHREPETITIONS: $\mathcal{O}(\log \delta \log(\delta \cdot n))$; (ii) ONLINESETMULTICOVER: $\mathcal{O}(\log \delta \log n)$; (iii) SETMULTICOVERLEASING: $\mathcal{O}(\log(\delta \cdot K) \log n)$

3.2 Model & Preliminaries

In this section, we start with a formal definition of SETMULTICOVERLEASING and then describe the techniques we use in developing our online algorithm in Section 3.3.

Definition. In compliance with our framework in Section 2.3, we define SETMULTICOVERLEASING as follows. Elements U ($|U| = n$) form the demands set $\mathcal{D}$ and the family F ($|F| = m$) of subsets of U forms the infrastructure set $\bar{\mathcal{S}}$. A demand j arriving at time t with a value p_{jt}, specifying the number of sets it needs to be covered, is denoted as (j,t). A set S with a lease type $k \in \{1,\dots,K\}$ starting at time t is denoted as (S,k,t). The cost of leasing (S,k,t) is c_{Sk}. $\bar{\mathcal{S}}_{t'}$ contains all triples (S,k,t) covering t'. SETMULTICOVERLEASING asks to minimize the total cost of sets leased such that each arriving demand (j,t) is covered by p_{jt} different sets $(S,k,t') \in \bar{\mathcal{S}}_t$ such that $j \in S$. Figure 3.2 shows the integer linear program (ILP) of SETMULTICOVERLEASING. A variable x_{Skt} is assigned to each (S,k,t) indicating whether it is bought or not. The constraint assures that each arriving demand (j,t) is covered by p_{jt} different sets.

$$\begin{aligned} \min \quad & \sum_{(S,k,t)\in\bar{\mathcal{S}}} c_{Sk}x_{Skt} \\ \text{Subject to:} \quad & \sum_{(S,k,t')\in\bar{\mathcal{S}}_t : j\in S} x_{Skt'} \geq p_{jt} \qquad (j,t)\in\mathcal{D} \\ & x_{Skt} \in \{0,1\} \quad (S,k,t)\in\bar{\mathcal{S}}_t \end{aligned}$$

FIGURE 3.2: ILP Formulation of SETMULTICOVERLEASING

Algorithmic Techniques. Before we describe our algorithm, let us have an idea of what techniques are used in designing and analyzing our algorithm. The first technique is a well-known technique, commonly used in randomized algorithms for problems that can be formulated as linear programs, known as *randomized rounding*. The second is based on a greedy strategy and can be used in both deterministic and randomized algorithms - we call it *layering*. We give an informal description of each technique in what follows.

(i) Randomized rounding uses a probabilistic method to convert an optimal solution of a relaxation of a given problem into an approximately optimal solution to the original problem. Basically, it involves the following:

- Formulate the problem at hand into an integer linear program (ILP).
- Compute an optimal fractional solution x to the linear programming relaxation, i.e., the corresponding LP.
- Round the fractional solution x of the LP to an integer solution x' of the ILP.

The challenge here is to choose a suitable integer linear program (ILP), compute an efficient (optimal) fractional solution in polynomial time, and use probabilistic arguments to round the fractional solution at 'low' cost.

(ii) Layering can be useful in designing algorithms for online problems involving demands that can be partitioned into sub-demands such that sub-demands i and $i+1$ are each served by a separate 'entity' of the optimal solution. As an example, consider SETMULTICOVERLEASING, in which each arriving element/demand (j,t) needs to be covered by p_{jt} different sets. Each of $\{1,2,...,p_{jt}\}$ constitutes a sub-demand and a set used to cover (j,t) the first time can not be used to cover it the second time, third time, and so on. Once a problem is defined with such a

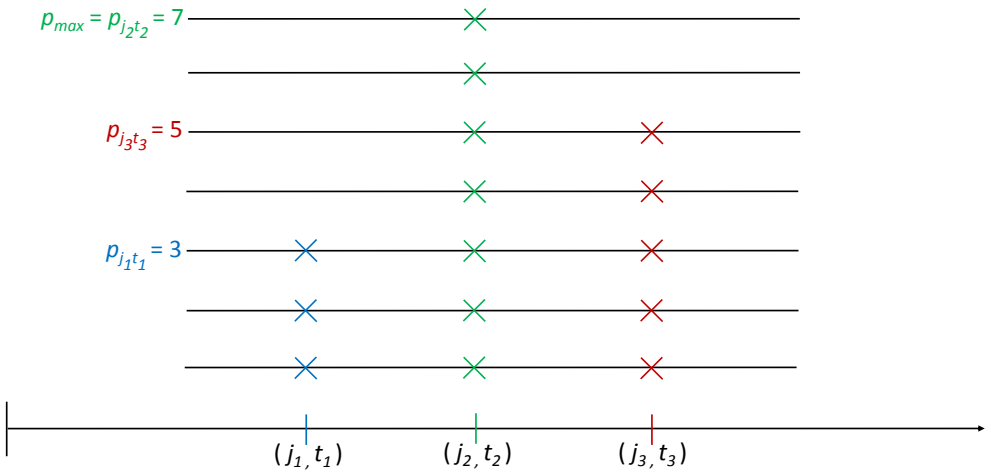


FIGURE 3.3: Layering

Seven layers are formed resulting from sub-demands denoted by 'x' of the demands (j_1, t_1), (j_2, t_2), and (j_3, t_3).

structure, for each arriving demand, we treat each of its sub-demands sequentially as separate demands and serve them using an appropriate greedy strategy.

The term layering emerges from dividing demands into sub-demands, i.e., throughout the overall time period, we will be constructing 'layers' in the following way: a part of a layer, corresponding to time t, is formed for each sub-demand of a demand arriving at time t. In SETMULTICOVERLEASING, p_{max} layers will be constructed, where p_{max} is the maximum p_{jt}: $(j, t) \in \mathcal{D}$ (see Figure 3.3 for an illustrative example).

The main advantage of layering is the ease of analysis it provides, such that each 'entity' of a solution computed by the online algorithm to serve a sub-demand is compared to the corresponding 'entity' of the optimal solution.

3.3 Online Algorithm

In this section, we propose an online randomized algorithm using techniques described in Section 3.2.

The following notation will be used in the algorithm. A triple (S, k, t') is a *candidate* to (j, t) if $j \in S$ and $(S, k, t') \in \bar{\mathcal{S}}_t$. An element is *p-covered* if at least p of its candidates are leased.

Algorithm. Our algorithm maintains a fraction f_{Skt} for each set $(S, k, t) \in \bar{\mathcal{S}}$, initially set to zero and non-decreasing throughout the algorithm. When a demand $(j, t) \in \mathcal{D}$ arrives, we first 1-cover it by some candidate $(S, k, t') \in \bar{\mathcal{S}}$. To do this, we increase the fractions of all candidates until they sum up to 1. Then, using an appropriate randomized rounding, we select at least one candidate and lease it. If $p_{jt} = 1$, we are done. Otherwise, to 2-cover (j, t), we increase the fractions of all candidates excluding the one leased to 1-cover (j, t) (choose arbitrarily any set if there is more than one), until they sum up to 1. Now, we again use randomized rounding to guarantee that a second candidate is leased. We continue with this greedy strategy until (j, t) is p_{jt}-covered.

We first describe an algorithm which i-covers any arriving element (j, t) (Algorithm 3) and then use it in SetMulticoverLeasing (Algorithm 4).

We maintain for each set $(S, k, t) \in \bar{\mathcal{S}}$, $2\lceil \log(n+1) \rceil$ independent random variables $X_{(Skt)(q)}$, $(1 \leq q \leq 2\lceil \log(n+1) \rceil)$, distributed uniformly in the interval $[0, 1]$. We define $\mu_{Skt} := \min\{X_{(Skt)(q)}\}$.

Algorithm 3 i-Cover

Let $Q \subseteq \bar{\mathcal{S}}$ be the collection of candidates of (j, t) not yet leased during one of $i - 1$ previous calls of i-Cover.

(i) (fractional) while $\sum_{(S,k,t) \in Q} f_{Skt} < 1$, do the following *increment*.

$$f_{Skt} = f_{Skt} \cdot (1 + 1/c_{Sk}) + \frac{1}{|Q| \cdot c_{Sk}}$$

(ii) (integer) Lease $(S, k, t) \in Q$ with $f_{Skt} > \mu_{Skt}$.

(iii) If (j, t) is not covered by some set in Q (i.e., there is no leased set in Q), then lease the cheapest $(S, k, t) \in Q$.

Algorithm 4 SetMulticoverLeasing

Whenever an element (j, t) arrives:
  Set i to 0.

  While($i \leq p_{jt}$)
    Run Algorithm 3 (i-Cover).
    Increment i by 1.

3.4 Analysis

We now show that Algorithm 4 is $\mathcal{O}(\log(\delta \cdot K) \log n) = \mathcal{O}(\log(m \cdot K) \log n)$-competitive for SetMulticoverLeasing, thus implying results for OnlineSetMulticover and OnlineSetCoverWithRepetitions as well.

It is easy to see that Algorithm 4 constructs a feasible solution to SetMulticover-Leasing. To compute the total expected cost, we first bound the fractional cost (the fractional cost is the sum of all fractions) by $\mathcal{O}(\log(\delta \cdot K)) \cdot Opt = \mathcal{O}(\log(m \cdot K)) \cdot Opt$, where Opt is the cost of an optimal offline solution. After that, we show that the randomized integer solution has an expected cost of at most $\mathcal{O}(\log n)$ times the fractional cost and hence deduce the expected $\mathcal{O}(\log(\delta \cdot K) \log n) = \mathcal{O}(\log(m \cdot K) \log n)$-competitive ratio of the algorithm. The following lemma below bounds the fractional cost.

Lemma 3.1. *The fractional cost is $\mathcal{O}(\log(\delta \cdot K))$ times the optimal offline solution.*

Proof. We show the following two facts to bound the fractional cost.

1. An increment adds at most two to the fractional cost.
2. The total number of increments in the algorithm is $\mathcal{O}(\log(\delta \cdot K)) \cdot Opt$.

We fix an element j and use S instead of (S, k, t) for simplicity.

Proof of 1: In an increment, the fraction of each candidate $S \in Q$ is increased by $\left(\frac{f_S}{c_S} + \frac{1}{|Q| \cdot c_S}\right)$. Summing up over all candidates in Q, the fractional cost increases, after each increment, by $\sum_{S \in Q} c_S \cdot \left(\frac{f_S}{c_S} + \frac{1}{|Q| \cdot c_S}\right) = \sum_{S \in Q} f_S + 1$. Since an increment is only

done if $\sum_{S \in Q} f_S < 1$, each increment adds at most two to the cost of the fractional solution.

Proof of 2: The total number of increments is the sum of the increments for each $1 \leq i \leq p_{\max}$, where $p_{\max}$ is the maximum number of sets any element needs to be covered. An increment is only done if $\sum_{S \in Q} f_S < 1$. For any $1 \leq i \leq p_{\max}$ and in any increment the algorithm decides to make, at least one set S_{Opt} in the optimal solution is a candidate and therefore increases its fraction (an optimal solution requires $p_{\max}$ sets too). The fraction of S_{Opt} reaches at least $\frac{1}{|Q|}$ after $c_{S_{Opt}}$ increments due to the second part of the increment $\frac{1}{|Q| \cdot c_{S_{Opt}}}$. After this, the first part of the increment keeps multiplying the fraction by $(1 + 1/c_{S_{Opt}})$ and stops when the fraction is larger than 1. Hence, after $\mathcal{O}(c_{S_{Opt}} \cdot \log |Q|)$ increments, $\sum_{S \in Q} f_S$ will be greater than 1. Since $|Q| \leq \delta K$, 2 holds.

Since the fractions increase only during an increment, the lemma follows. □

Lemma 3.2. *The randomized integer solution has an expected cost of at most* $\mathcal{O}(\log n)$ *times the fractional cost.*

Proof. We fix a set S. The expected cost of choosing S throughout the algorithm is $c_S \cdot Pr(f_S > \mu_S) \leq 2 \log(n+1) \cdot c_S \cdot f_S$. Thus, the total expected cost is upper bounded by $\sum_{S \in F} 2 \log(n+1) \cdot c_S \cdot f_S$. Furthermore, to guarantee a feasible solution, Algorithm 3 adds the cheapest candidate to the solution if an element j is not covered after randomization (this is a lower bound to the optimal solution Opt). Nevertheless, we show that this only happens with probability at most $1/n^2$ and adds an unnoticed additional cost to the competitive ratio.

For a single $1 \leq q \leq 2 \lceil \log(n+1) \rceil$, the probability that j is not covered is $\leq \prod_{S \in Q} (1 - f_S) \leq e^{-\sum_{S \in Q} f_S} \leq 1/e$. The last inequality holds because $\sum_{S \in Q} f_S \geq 1$. Thus, the probability that j is not covered, for all $1 \leq q \leq 2 \lceil \log(n+1) \rceil$, is at most $1/n^2$. The additional expected cost is thus upper bounded by $n \cdot 1/n^2 \cdot Opt$. □

From the two lemmas 3.1 and 3.2, we can deduce the following theorem.

Theorem 3.3. *There is an online randomized algorithm for* SetMulticoverLeasing *that is* $\mathcal{O}(\log(\delta \cdot K)\log n) = \mathcal{O}(\log(m \cdot K)\log n)$*-competitive.*

As mentioned earlier, OnlineSetMulticover is a special case of SetMulticoverLeasing. Thus, one can easily see that our algorithm yields an $(\mathcal{O}(\log \delta \log n) = \mathcal{O}(\log m \log n)$-competitive ratio for OnlineSetMulticover. This matches the randomized $\Omega(\log m \log n)$ lower bound by Korman [26] for the problem. Hence, we deduce the following.

Corollary 3.4. *There is an online randomized optimal algorithm for* OnlineSetMulticover *that has a competitive ratio of* $\mathcal{O}(\log \delta \log n) = \mathcal{O}(\log m \log n)$.

Furthermore, by slightly modifying our algorithm, we manage to improve the competitive ratio for OnlineSetCoverWithRepetitions by Alon et al. [7]. This can be done as follows. Rather than maintaining for each set $2\lceil \log(n+1) \rceil$ independent random variables, we maintain $2\lceil \log((\delta n)+1) \rceil$ independent random variables. Therefore, the randomized integer solution is at most $\mathcal{O}(\log n)$ times the fractional cost (the additional expected cost is now $(n\delta \cdot 1/(n\delta)^2) \cdot Opt$). Hence, we deduce the following.

Corollary 3.5. *There is an online randomized algorithm for* OnlineSetCoverWithRepetitions *that has a competitive ratio of* $\mathcal{O}(\log \delta \log(\delta \cdot n)) = \mathcal{O}(\log m \log(m \cdot n))$.

3.5 Conclusion & Outlook

This chapter presented the first online algorithm for the leasing variant of the well-known SetCover which has always been of both theoretical and practical interest. With the results in this chapter, we open a research room for a wide range of covering problems (e.g., vertex cover, edge cover), that were not previously studied with a leasing aspect, thus shortening the distance between theory and practice.

While the techniques used in our approach proved to yield (optimal) efficient algorithms, designing algorithms for more sophisticated covering problems may not be so

easy. Nevertheless, our techniques may give the first insights to solve the leasing variants of these problems. Speaking of which, it is interesting to note that these techniques are mainly a combination of techniques used in the non-leasing variants of the problems studied. From here, we wonder whether leasing inherits a difficulty in addition to the difficulty incurred by the infrastructure nature of the problems, or we could just treat infrastructure leasing problems as their non-leasing variants - like we more or less do here in this chapter. One way to estimate this difficulty is through lower bounds. The only lower bounds we have for SetCoverLeasing, for example, are the deterministic $\Omega\Big(K + \frac{\log m \log n}{\log\log m + \log\log n}\Big)$ and the randomized $\Omega(\log K + \log m \log n)$. While $\Omega(K)$ and $\Omega(\log K)$ are deterministic and randomized lower bounds for the ParkingPermitProblem, respectively, $\Omega\Big(\frac{\log m \log n}{\log\log m + \log\log n}\Big)$ and $\Omega(\log m \log n)$ are deterministic and randomized lower bounds for OnlineSetCover, respectively. It is still not known whether we can prove stronger lower bounds or we can close the gaps by designing new algorithms.

Another important observation is the difference between the factors in the competitive ratio of the ParkingPermitProblem and SetCoverLeasing. While the competitive ratio of the ParkingPermitProblem depends solely on K, that of SetCoverLeasing depends on n, m, and K. Obviously, this difference emerges from the infrastructure nature of SetCoverLeasing which makes the competitive ratio depend on m and n in addition to K. Nevertheless, it turns out that one may get rid of this dependency on n as we shall see in Chapter 5. This does not merely mean a competitive ratio depending on simply different factors - like is known for many classical optimization problems (e.g., fixed parameter tractable (FPT) problems), but it also means in-dependency on time. The latter is a crucial property for problems modeled in the online setting in which n may be unbounded, i.e., demands continue to arrive as long as the business is running.

SetCoverLeasing captures the basic leasing model one may think of while moving from buying to leasing resources. In fact, we assume in the ParkingPermitProblem that we do not know demands in advance - since we can not know the weather forecast. But what if we collect data from previous years and assume demands are given according to some probability distribution. While such extensions are not within the scope of this

thesis, we do actually extend the ParkingPermitProblem to a more general model for demands in Chapter 5.

Chapter 4

Facility Leasing

Consider a company that runs a distributed service on a network. In order to provide the service, the company has to choose a set of nodes to become service providers in such a way that they are easily accessible by customer nodes. The nodes in the network do not belong to the company and must therefore be leased before they can be used to provide the service. There are various leases of different duration and cost. Once a lease expires, the node is no longer able to provide the service. In order to use the node again, a new lease must be bought. The customer nodes can freely use any node's service as long as there is an active lease for this node. The costs of using it are proportional to the distance (latency) between customer node and service-providing node. This means that on the one hand the company wants to buy leases for nodes as seldom as possible, while on the other hand it wants to make sure that customer nodes are not too far away from currently leased nodes. Things would have been easy if the company knows in advance which customers will request the service - but since it does not, it must face the difficulty incurred by the uncertainty of the future.

As another example, suppose that a soft-drink company plans to place its vending machines in a city. The company has already identified potential sites for the machines in a number of different neighborhoods and knows the cost of renting each potential site. Its aim is to minimize the renting costs and the average traveling distance of customers,

without knowing which customers will arrive.

At the core of these examples, we have a complex optimization problem, FacilityLeasing, which will be our subject of study in this chapter.

Chapter Basis. The results in this part were achieved by Kling et al. in [10] and later incorporated into the following publication.

> Sebastian Abshoff, Peter Kling, Christine Markarian, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. "Towards the Price of Leasing Online". To appear in: *Journal of Combinatorial Optimization (JOCO)*, **2015** [11].

Chapter Outline. This chapter starts with an overview of related literature (Section 4.1) along with a summary of results obtained in this chapter. Section 4.2 presents a formal definition of FacilityLeasing, gives some necessary notation, and describes the algorithmic techniques used throughout the chapter. The main results of this chapter comprising of an online algorithm and its analysis are presented in Sections 4.3 and 4.4, respectively. The chapter concludes with a short résume and future work in Section 4.5.

4.1 Related Work & Contribution

Before stating the results obtained in this chapter, let us have an overview of literature evolving around our problem at hand, FacilityLeasing.

The latter originates from one of the most popular $\mathcal{NP}$-hard optimization problems, FacilityLocation. Not only is FacilityLocation widely known in operations research [31, 32] and many other applications, it has also attracted the attention of computer scientists and particularly theorists. A survey of its applications and methods handling it can be found in [33]. Given a complete bipartite graph $G = (F \cup C, E)$, where F ($|F| = m$) refers to facilities and C ($|C| = n$) refers to clients. FacilityLocation asks to open facilities in order to assign each client to an open facility. Opening

a facility incurs an *opening cost* and assigning a client i to facility j yields a *connection cost*. The goal is to minimize the total opening and connection costs. FacilityLocation has many variants most of which have constant approximation [34–41].

FacilityLocation was also known as an online version OnlineFacilityLocation. Here, in each step, a subset of clients arrive and need to be assigned to open facilities. OnlineFacilityLocation has been studied in both the metric setting [42, 43], where connection costs satisfy the triangle inequality and the non-metric setting [44, 45]. In the metric setting, Meyerson [45] presented a randomized $\mathcal{O}(\log n)$-competitive algorithm which was improved into a deterministic $\mathcal{O}(\log n / \log\log n)$ - competitive algorithm by Fotakis [44] who also showed that this bound is optimal.

Inspired by Meyerson's ParkingPermitProblem, Anthony et al. [5] extended OnlineFacilityLocation to FacilityLeasing. Here, unlike OnlineFacilityLocation in which facilities can be used forever once they are open, we must pick one of K different lease types when opening a facility. A lease type k has a certain lease length l_k. Leasing a facility i using lease type k incurs a cost c_{ik} and ensures that i is open for the next l_k time steps. The same facility can later be used by leasing it again. FacilityLeasing asks to minimize the total leasing and connection cost. Its connection to OnlineFacilityLocation becomes apparent if we just set $K = 1$ and $l_1 = \infty$. Anthony et al. [5] showed an interesting relationship between infrastructure leasing problems and stochastic optimization that led to an $\mathcal{O}(K)$-approximation for FacilityLeasing (offline setting). Nagarajan and Williamson [42] later improved the $\mathcal{O}(K)$-approximation into a 3-approximation and gave an $\mathcal{O}(K \log n)$-competitive algorithm for metric FacilityLeasing (online setting).

Contribution. In this chapter, we study FacilityLeasing introduced by Anthony et al. [5] who studied the problem in the offline setting. Clients D ($|D| = n$) arrive over time and must be connected to open facilities F ($|F| = m$). Each facility can be leased for K different periods of time. Leasing a facility i for a period k incurs a cost c_{ik} and ensures that i is open for the next l_k time steps. Connecting a client j to facility i incurs a connecting cost d_{ij}. FacilityLeasing asks to connect each client to an open

facility while minimizing the total leasing and connecting costs. Nagarajan et al. [9] gave the first online algorithm, with an $\mathcal{O}(K \log n)$-competitive factor for the problem. The results in this chapter extend those by Nagarajan et al. [9] and include:

- The first online algorithm, with a time-independent competitive factor $\mathcal{O}(l_{\max} \log(l_{\max}))$, where $l_{\max}$ denotes the maximum lease length.
- An $\mathcal{O}(\log^2(l_{\max}))$-competitive factor for many 'natural' cases, such as situations where the number of clients arriving in each time step does not vary too much, or is non-increasing, or is polynomially bounded in $l_{\max}$.

4.2 Model & Preliminaries

In this section, we start with a formal definition of FacilityLeasing and then describe the techniques used in developing the online algorithm in Section 4.3.

Definition. In compliance with our framework in Section 2.3, we define FacilityLeasing as follows. The demand set is the set of clients D ($|D| = n$) and the set F ($|F| = m$) of facilities forms the infrastructure set $\bar{\mathcal{F}}$. More than one client may arrive in a time step and $\mathcal{D}_t$ denotes the set of clients arriving at time step t. A client j arriving at time t is denoted as (j, t) and a facility i with lease type $k \in \{1, \ldots, K\}$ starting at time t is denoted as (i, k, t). The cost of leasing (i, k, t) is c_{ik}. Each (j, t) must be connected to a leased (i, k, t), and this incurs a connection cost d_{ij}. Clients and facilities reside in a metric space such that connection costs satisfy the triangle inequality: $\forall i, i' \in \mathcal{F}, j, j' \in \mathcal{D} : d_{i'j} \leq d_{ij} + d_{ij'} + d_{i'j'}$. FacilityLeasing asks to minimize the total leasing and connection costs. The series $H_q \coloneqq \sum_{i=1}^{q}(|\mathcal{D}_i|/(\sum_{j=1}^{i}|\mathcal{D}_j|))$ can be used to describe the clients' arriving pattern, and we will see that it is tightly connected to the algorithm's competitiveness in Section 4.4.

Figure 4.1 shows the integer linear program (ILP) of FacilityLeasing. The first sum in the objective function represents the costs incurred by leasing facilities. A variable

$$
\begin{aligned}
\min \quad & \sum_{(i,k,t)\in\bar{\mathcal{F}}} c_{ik}x_{ikt} + \sum_{(j,t)\in\mathcal{D}}\sum_{i\in\mathcal{F}} d_{ij}y_{ijt} \\
\text{Subject to:} \quad & \sum_{i\in\mathcal{F}} y_{ijt} \geq 1 && (j,t)\in\mathcal{D} \\
& \sum_{(i,k,t')\in\bar{\mathcal{F}}_t} x_{ikt'} - y_{ijt} \geq 0 && i\in\mathcal{F}, (j,t)\in\mathcal{D} \\
& y_{ijt} \in \{0,1\} && i\in\mathcal{F}, (j,t)\in\mathcal{D} \\
& x_{ikt} \in \{0,1\} && (i,k,t)\in\bar{\mathcal{F}}
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & \sum_{(j,t)\in\mathcal{D}} \alpha_{jt} \\
\text{Subject to:} \quad & \alpha_{jt} - \beta_{ijt} \leq d_{ij} && i\in\mathcal{F}, (j,t)\in\mathcal{D} \\
& \sum_{(j,t')\in\mathcal{D}:\, t'\in[t,t+l_k)} \beta_{ijt'} \leq c_{ik} && (i,k,t)\in\bar{\mathcal{F}} \\
& \beta_{ijt} \geq 0 && i\in\mathcal{F}, (j,t)\in\mathcal{D} \\
& \alpha_{jt} \geq 0 && (j,t)\in\mathcal{D}
\end{aligned}
$$

FIGURE 4.1: ILP Formulation of FACILITYLEASING

x_{ikt} is assigned to each (i,k,t) indicating whether it is bought or not. The remaining part of the objective function represents the costs incurred by connecting each client to a facility, where variable y_{ijt} indicates whether a client j that arrived at time step t is connected to facility i. While the first primal constraint guarantees that each client is connected to at least one facility, the second makes sure that each client is only connected to a facility that is leased during the time step of the client's arrival. Let $\bar{\mathcal{F}}_t$ be the set of all facilities covering time step t.

Algorithmic Techniques. Before we describe the algorithm, let us have an idea of what techniques are used in designing and analyzing the algorithm.

While the latter is a primal-dual algorithm exploiting the properties of linear programs discussed in Chapter 2, it does not, however, follow the standard approach of constructing *feasible* primal and dual solutions. Instead, it allows itself to violate the dual constraints, thus constructing an infeasible dual solution. The infeasible dual solution is then used to construct a feasible primal solution. The challenge is now to analyze the

algorithm. Obviously, weak duality alone no more yields the competitive ratio since the dual solution constructed is not feasible. Hence, a technique that proved to be successful in analyzing many algorithms is adopted. The latter is based on scaling the dual solution constructed with some factor $f()$ (i.e., multiplying the dual solution by $f()$)) to yield a dual solution that is feasible. Obviously, this factor will be incorporated into the competitive ratio:

- primal $\leq f'() \cdot$ infeasible dual (the primal solution is bounded in terms of the dual solution constructed). See Figure 4.2 for an illustration.
- feasible dual $= f() \cdot$ infeasible dual. This part will be the most difficult: finding a suitable value for $f()$.
- primal $\leq \frac{f'(k)}{f(k)} \cdot$ feasible dual $\Rightarrow \mathcal{O}(\frac{f'(k)}{f(k)})$-competitive ratio (weak duality).

These primal-dual algorithms which construct a feasible primal and an infeasible dual are called *dual-fitting* algorithms. For examples of dual-fitting algorithms, see [38, 40, 46]. The idea of dual-fitting algorithms is in fact not new - it has always been there in the literature, but mostly implicitly: e.g., to prove the approximation of SetCover [19, 22].

Another prominent property highly contributing in the analysis is the *triangle inequality* with which the metric version of the problem is defined. The latter has proved to be of significant help in achieving the competitiveness, as we shall see in Section 4.4.

4.3 Online Algorithm

In this section, we present an online deterministic algorithm using techniques described in Section 4.2.

A triple (i, k, t') is a *candidate* to (j, t) if $(i, k, t') \in \bar{\mathcal{F}}_t$. The algorithm is formulated such that it creates a solution that adheres to the interval model defined in Lemma 2.6 in which lease lengths l_k are powers of two and leases of the same type do not overlap. At

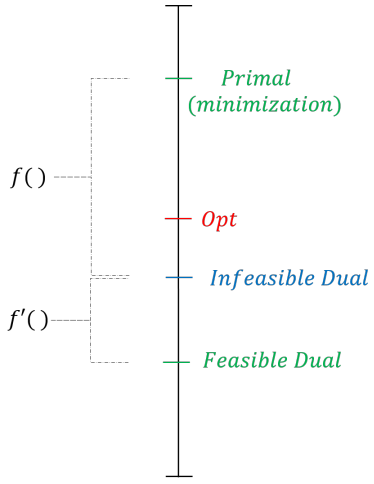


FIGURE 4.2: Dual-fitting

the beginning, all facilities are closed. At the arrival of the client set $\mathcal{D}_t$ at time t, these clients are assigned to open facilities to satisfy their demands by opening new facilities if necessary. The costs charged for this step comprise the corresponding connection cost d_{ij} for assigning the clients $j \in \mathcal{D}_t$ to open facilities $(i,k,t) \in \bar{\mathcal{F}}$ and the opening cost of newly opened facilities. Complying with the interval model, facilities can be opened using lease type k only at times t that are a multiple of the corresponding lease length l_k. That is, only at times t with $t \equiv 0 \mod l_k$. In particular, this might cause us to open a facility $(i,k,t') \in \bar{\mathcal{F}}$ belatedly at the current time $t \geq t'$. Note that the algorithm uses (i,k,t') only to cover clients arriving at or after time $t \geq t'$ - former clients must have been covered by other facilities at their arrival. The interval model ensures that for each k and i there is exactly one candidate $(i,k,t') \in \bar{\mathcal{F}}$ which covers t (i.e., $(i,k,t') \in \bar{\mathcal{F}}_t$). Thus, an algorithm merely has to specify which pair (i,k) is chosen to satisfy a client's demand. Due to this, a facility is sometimes denoted as a tuple (i,k), meaning facility i with lease type k covering the current time step.

Algorithm. The algorithm is based on an approximation algorithm by Jain and Vazirani [38] for the classical facility location problem. Their algorithm uses a primal-dual approach to compute a 3-approximation, and makes use of a similar approach in each single time step. In each time step t the algorithm operates in two phases, similar to the algorithm by Nagarajan and Williamson [9] for the static facility location problem. In the first phase, clients essentially bid towards the facilities - or better said, towards the

tuples (i, k), representing for each k the facility of type k covering the current time step. In the second phase, triangle inequality is used to choose a cheap subset of facilities to actually open and assign clients to. In contrast to [38], the challenge here is to cope with the problem to build a good solution for a facility location problem starting from a partial solution (earlier arrived clients). This is similar to [9] but more complex since here all newly arrived clients are considered simultaneously (instead of one after the other).

First Phase: For each client $j \in \mathcal{D}_{\leq t} := \bigcup_{t' \leq t} \mathcal{D}_{t'}$ that arrived at time t or before, a potential α_{jkt} is introduced, starting at zero and continuously increasing (concurrently and at the same rate for each client). Each of these potentials is reset to zero in each round. To simplify notation, let us define $(x)_+ := \max(x, 0)$. For any facility i of lease type k the invariant $c_{ik} \geq \sum_{j \in \mathcal{D}_{\leq t}} (\alpha_{jkt} - d_{ij})_+$ (INV1) is maintained. Whenever equality is reached for some facility i and lease type k, i is temporarily opened using lease type k. As soon as $\alpha_{jkt} \geq d_{ij}$ for a client $j \in \mathcal{D}_{\leq t}$ and a (temporarily or permanently) open facility i of lease type k, α_{jkt} stops increasing. If $j \in \mathcal{D}_t$ (i.e., j is a newly arrived client), j is connected to i and furthermore $\hat{\alpha_j}$: is set to α_{jtk} (these $\hat{\alpha_j}$ correspond to the dual variables α_{jt} in the ILP from Figure 4.1, the t given implicitly by the relation $j \in \mathcal{D}_t$). The second invariant ensures that in no time step t' an $\alpha_{jkt'}$ is increased beyond $\hat{\alpha_j}$ (INV2).

Second Phase: In this phase K different conflict graphs are built, one for each lease type k. The nodes of the graph for lease type k are given by temporarily and permanently opened facilities i of lease type k. There is an edge between two nodes i and i' if and only if there is some client $j \in \mathcal{D}_{\leq t}$ with $\alpha_{jkt} > \max(d_{ij}, d_{i'j})$. Facilities i and i' are said to be in conflict. Now, for each conflict graph a maximal independent set (MIS) is computed and facilities in the MIS are permanently opened (while closing the remaining temporarily opened facilities). If for a client $j \in \mathcal{D}_t$ (i.e., newly arrived clients) the facility i it was connected to during the first phase is not a member of a MIS, j is reconnected to a neighbor of i that is a member of a MIS (i.e., permanently open).

Note that the terms "temporarily open" and "permanently open" do not refer to the lease length, but only whether the algorithm's decision to open a new facility i with lease type k that covers the current time step is final or not.

4.4 Analysis

We now show that the algorithm in the previous section is $(3+K)H_{l_{\max}}$-competitive for FacilityLeasing.

Note that it is sufficient to consider the first $l_{\max}$ time steps: at time $l_{\max}$ all facilities must be closed, since for any $k \in \{1, 2, \ldots, K\}$ we have $l_{\max} \equiv 0 \mod l_k$. Let us partition the time horizon into rounds $\tau_i := \{(i-1)l_{\max}, \ldots, il_{\max} - 1\}$ of length $l_{\max}$. By the above observation, these rounds yield independent sub-problems, each of length $l_{\max}$. Then we show that the solution of the algorithm is $(3+K)H_{l_{\max}}$-competitive in each such round. Since the costs over all rounds are additive, this yields an overall $(3+K)H_{l_{\max}}$-competitive factor.

The values $\hat{\alpha}_j$ computed by the algorithm correspond to the dual variables of the ILP formulation in Figure 4.1. First of all, we show that the sum of all $\hat{\alpha}_j$ times $(3+K)$ is an upper bound for the cost of the solution produced by the algorithm (Lemma 4.1). Next, we consider the $\hat{\alpha}_j$ as a (possibly infeasible) solution to the dual program of the FacilityLeasing ILP. We show that by scaling this solution down by a suitable factor, we get a feasible solution to the dual program (Lemma 4.4). By the weak duality theorem, multiplying both factors yields the final competitive factor (Theorem 4.5).

Upper Bounding the Solution. The following lemma upper bounds the cost of the solution produced by the algorithm by $(3+K)\sum_{j\in\mathcal{D}}\hat{\alpha}_j$. The basic idea is to exploit the triangle inequality to show that $3\sum_{j\in\mathcal{D}}\hat{\alpha}_j$ is a bound on the total connection cost and that the algorithm ensures that each $\hat{\alpha}_j$ is used at most K times to cover the complete costs for opening facilities.

Lemma 4.1. *The cost of the primal solution produced by the algorithm can be bounded from above by* $(3 + K) \sum_{j \in \mathcal{D}} \hat{\alpha}_j$.

Proof. We bound the connection costs of the clients and the opening costs of facilities separately. The $\hat{\alpha}_j$ value of a client j is computed in step t of j's arrival during the first phase of the algorithm. During this phase, j is either connected to a facility i that was already (permanently) opened at time $t' < t$ or one that was temporarily opened at the current time t. In both cases its $\hat{\alpha}_j$ value was set such that it can cover at least the distance d_{ij} between i and j. If i remains in one of the MIS computed in the second phase of the algorithm, this guarantees that j is assigned to a facility i' such that $\hat{\alpha}_j$ is an upper bound on the client's connection costs $d_{i'j}$. Otherwise, if i is no longer in any MIS at the end of phase two, Proposition 4.2 (see below) exploits the metric property of the facility location problem and yields that j is assigned to a facility i' such that $3\hat{\alpha}_j$ is an upper bound on the client's connection costs $d_{i'j}$.

Now, consider the facility costs and fix a facility i of lease type k that is permanently opened at some time t by the algorithm. As (i, k) is opened permanently at time t in the second phase, it must have been temporarily opened in the first phase. Thus, by definition of the algorithm, invariant INV1 must hold with equality, that is $c_{ik} = \sum_{j \in \mathcal{D}_{\leq t}} (\alpha_{jkt} - d_{ij})_+$. Consider these *bids* $(\alpha_{jkt} - d_{ij})_+$ of clients $j \in \mathcal{D}_{\leq t}$ to facility i of lease type k. Note that all non-zero bids of clients j at the current time are guaranteed to be used by facility (i, k) only, as (i, k) must have been in the corresponding MIS for lease type k. Moreover, note that for a single client that arrived at time t, all its bids given to (and used by) facilities of type k sum up to at most $\hat{\alpha}_j$, as any $\alpha_{jkt'}$ with $t \geq t'$ stops increasing as soon as a corresponding open facility (or $\hat{\alpha}_j$) is reached. Together, this yields that the total costs for opening facilities of type k in the solution produced by the algorithm is upper bounded by $\sum_{j \in \mathcal{D}} \hat{\alpha}_j$. As there are K different lease types, together with the bound on the connection costs we get the lemma's statement. □

The following proposition exploits the triangle inequality of our metric facility location problem and its proof is completely analogous to ([38], Lemma 5).

Proposition 4.2. *For each client j that is reconnected in the second phase to a facility i, we have $\hat{\alpha}_j \geq \frac{1}{3} d_{ij}$.*

Proof. Let i' be the facility that client j was connected to in the first phase of the algorithm. There must be a client j' that is responsible for the conflict between facility i and i'. We have that $\alpha_{j'} \geq d_{i'j}$, $\alpha_{j'} \geq d_{ij}$ and $\alpha_j \geq d_{i'j}$. Let s_i resp. $s_{i'}$be the points in time where i resp. i' are temporarily opened. We know that $\alpha_{j'} \leq \min(s_i, s_{i'})$ since j' was contributing to both facilities, and that $\hat{\alpha}_j \geq s_{i'}$ since j was connected to i'. Plugging this information into the triangle inequality $d_{ij} \leq d_{i'j} + d_{i'j'} + d_{ij'}$ yields the proposition. □

Scaling the Dual Variables for Feasibility. For the second part of the proof, it remains to scale down the dual solution represented by $\hat{\alpha}_j$ such that we obtain a feasible solution. Before we do so, we need another proposition based on the triangle inequality. In spirit, it is similar to ([9], Lemma 5).

Proposition 4.3. *Given a client l that arrived in time step t and a facility i of type k for any client j that arrived before time t, we have $\alpha_{jkt} - d_{ij} \geq \hat{\alpha}_l - 2d_{ij} - d_{il}$.*

Proof. Showing that $\alpha_{jkt} + d_{ij} + d_{il} \geq \hat{\alpha}_l$ proves the proposition. Since for $\alpha_{jkt} \geq \hat{\alpha}_l$ the statement trivially holds, we assume the contrary. This means that client j reached an open facility i' (and thus its α_{jkt} stopped increasing) before $\hat{\alpha}_l$ was fixed (i.e., α_{lkt} stopped increasing). Since α_{lkt} stops increasing once it is large enough to cover the distance between i' and l and this distance is at most $\alpha_{jkt} + d_{ij} + d_{il}$, the proposition follows. □

Before we continue with the Lemma 4.4 and its proof, let us define N_t to be the number of clients that have arrived until time t (i.e., $N_t := |\mathcal{D}_1| + |\mathcal{D}_2| + \ldots + |\mathcal{D}_{t-1}|$) and note that

$$N_t = N_t \frac{|\mathcal{D}_t|}{|\mathcal{D}_t|} = \sum_{j \in \mathcal{D}_t} \frac{N_t}{|\mathcal{D}_t|} = \frac{N_t}{|\mathcal{D}_t|} \sum_{j \in \mathcal{D}_t} 1. \tag{4.1}$$

For the previously defined series H_q, it holds that

$$\sum_{t=1}^{t^*} 2h_t \sum_{t'=1}^{t-1} \sum_{j \in \mathcal{D}_{t'}} d_{ij} = \sum_{t=1}^{t^*} 2 \sum_{j \in \mathcal{D}_t} d_{ij}(H_{t^*} - H_t), \tag{4.2}$$

which can be easily seen by observing that $\sum_{i=1}^{q} \sum_{j=1}^{i-1} x_j h_i = \sum_{i=1}^{q} x_i(H_q - H_i)$ holds for any series and any coefficients x_i. Given these tools, we are now ready to formulate and prove Lemma 4.4, which essentially shows that we get a feasible solution to the dual program of our problem if we scale the $\hat{\alpha}_j$ by a factor of $\frac{1}{H_{l_{\max}}}$. To ease notation in the following, whenever we consider a facility i of lease type k at time t^*, any time steps t we speak of are assumed to lie in the corresponding time interval of (i, k, t^*) (all other time steps are of no interest with respect to the constraints of the dual program).

Lemma 4.4. *For any facility i and lease type k at time t^* we have*

$$\sum_{t=1}^{t^*} \sum_{j \in \mathcal{D}_t} (\hat{\alpha}_j / 2H_{t^*} - d_{ij}) \leq c_{ik},$$

where $H_q := \sum_{i=1}^{q} \frac{|\mathcal{D}_i|}{\sum_{j=1}^{i} |\mathcal{D}_j|}$.

Proof. Recall the INV1 of the algorithm which states that the sum of bids towards a facility at any point in time never exceeds its opening cost. Thus, for any time step t^* we have

$$\begin{aligned}
c_{ik} &\geq \sum_{j \in \mathcal{D}_{\leq t^*}} (\alpha_{jkt^*} - d_{ij})_+ = \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il})_+ + \sum_{j \in \mathcal{D}_{<t^*}} (\alpha_{jkt^*} - d_{ij})_+ \\
&\geq \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il}) + \sum_{j \in \mathcal{D}_{<t^*}} (\alpha_{jkt^*} - d_{ij}) \\
&= \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il}) + \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} (\alpha_{jkt^*} - d_{ij}) \\
&\geq \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il}) + \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} (\hat{\alpha}_{l^*} - d_{il^*} - 2d_{ij}) \\
&\overset{*}{=} \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il}) + \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} (\hat{\alpha}_{l^*} - d_{il^*}) - 2 \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} d_{ij} \\
&= \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il}) + (\hat{\alpha}_{l^*} - d_{il^*}) \frac{N_{t^*}}{|\mathcal{D}_{t^*}|} \sum_{j \in \mathcal{D}_{t^*}} 1 - 2 \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} d_{ij}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il}) + \frac{N_{t^*}}{|\mathcal{D}_{t^*}|} \sum_{j \in \mathcal{D}_{t^*}} (\hat{\alpha}_{l^*} - d_{il^*}) - 2 \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} d_{ij} \\
&\geq \sum_{l \in \mathcal{D}_{t^*}} (\hat{\alpha}_l - d_{il}) + \frac{N_{t^*}}{|\mathcal{D}_{t^*}|} \sum_{j \in \mathcal{D}_{t^*}} (\hat{\alpha}_j - d_{ij}) - 2 \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} d_{ij} \\
&= \Big(1 + \frac{N_{t^*}}{|\mathcal{D}_{t^*}|}\Big) \sum_{j \in \mathcal{D}_{t^*}} (\hat{\alpha}_j - d_{ij}) - 2 \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} d_{ij} \\
&= \Big(\frac{N_{t^*} + |\mathcal{D}_{t^*}|}{|\mathcal{D}_{t^*}|}\Big) \sum_{j \in \mathcal{D}_{t^*}} (\hat{\alpha}_j - d_{ij}) - 2 \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} d_{ij}.
\end{aligned}
$$

(* follows from Proposition 4.3 and $l^* := \arg\max(\hat{\alpha}_l - d_{il})$)

The above inequality holds for each $t \in \{1, \ldots, t^*\}$. Dividing each such inequality by $\frac{N_t + |\mathcal{D}_t|}{|\mathcal{D}_t|}$ yields the following set of inequalities:

$$
\begin{aligned}
\frac{|\mathcal{D}_{t^*}|}{N_{t^*} + |\mathcal{D}_{t^*}|} c_{ik} &\geq \sum_{j \in \mathcal{D}_{t^*}} (\hat{\alpha}_j - d_{ij}) - 2 \frac{|\mathcal{D}_{t^*}|}{N_{t^*} + |\mathcal{D}_{t^*}|} \sum_{t=1}^{t^*-1} \sum_{j \in \mathcal{D}_t} d_{ij} \\
\frac{|\mathcal{D}_{t^*-1}|}{N_{t^*-1} + |\mathcal{D}_{t^*-1}|} c_{ik} &\geq \sum_{j \in \mathcal{D}_{t^*-1}} (\hat{\alpha}_j - d_{ij}) - 2 \frac{|\mathcal{D}_{t^*-1}|}{N_{t^*-1} + |\mathcal{D}_{t^*-1}|} \sum_{t=1}^{t^*-2} \sum_{j \in \mathcal{D}_t} d_{ij} \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\frac{|\mathcal{D}_2|}{N_2 + |\mathcal{D}_2|} c_{ik} &\geq \sum_{j \in \mathcal{D}_2} (\hat{\alpha}_j - d_{ij}) - 2 \frac{|\mathcal{D}_2|}{N_2 + |\mathcal{D}_2|} \sum_{t=1}^{1} \sum_{j \in \mathcal{D}_t} d_{ij} \\
\frac{|\mathcal{D}_1|}{N_1 + |\mathcal{D}_1|} c_{ik} &\geq \sum_{j \in \mathcal{D}_1} (\hat{\alpha}_j - d_{ij}) - 0.
\end{aligned}
$$

Adding up these t^* inequalities yields

$$
\begin{aligned}
&\left(\frac{|\mathcal{D}_1|}{N_1 + |\mathcal{D}_1|} + \ldots + \frac{|\mathcal{D}_{t^*}|}{N_{t^*} + |\mathcal{D}_{t^*}|} \right) c_{ik} \\
\geq &\sum_{t=1}^{t^*} \sum_{j \in \mathcal{D}_t} (\hat{\alpha}_j - d_{ij}) - \sum_{t=1}^{t^*} 2 \frac{|\mathcal{D}_t|}{N_t + |\mathcal{D}_t|} \sum_{t'=1}^{t-1} \sum_{j \in \mathcal{D}_{t'}} d_{ij}.
\end{aligned}
$$

Due to Inequality (4.2) we have

$$\begin{aligned} H_{t^*} c_{ik} &\geq \sum_{t=1}^{t^*} \sum_{j \in \mathcal{D}_t} (\hat{\alpha}_j - d_{ij}) - \sum_{t=1}^{t^*} 2 \sum_{j \in \mathcal{D}_t} d_{ij}(H_{t^*} - H_t) \\ &= \sum_{t=1}^{t^*} \sum_{j \in \mathcal{D}_t} (\hat{\alpha}_j - 2H_{t^*} d_{ij}) + \sum_{t=1}^{t^*} \sum_{j \in \mathcal{D}_t} 2d_{ij} \left(H_t - \frac{1}{2} \right) \\ &\geq \sum_{t=1}^{t^*} \sum_{j \in \mathcal{D}_t} (\hat{\alpha}_j - 2H_{t^*} d_{ij}) \end{aligned}$$

Dividing by $2H_{t^*}$ yields $\sum_{t=1}^{t^*} \sum_{j \in \mathcal{D}_t} \left(\frac{\hat{\alpha}_j}{2H_{t^*}} - d_{ij} \right) \leq \frac{c_{ik}}{2} \leq c_{ik}$. □

Finally, by combining the results from Lemma 2.6, Lemma 4.1 and Lemma 4.4, and using that our time horizon is at most $l_{\max}$ (i.e., $t^* \leq l_{\max}$), the weak duality theorem implies a competitive factor depending on the series H_k.

Theorem 4.5. *(an upper bound for* FacilityLeasing*) The algorithm is at most* $4(3+K)H_{l_{max}}$*-competitive for* FacilityLeasing*. Here, the series* H_q *is defined by*

$$H_q \coloneqq \sum_{i=1}^{q} \frac{|\mathcal{D}_i|}{\sum_{j=1}^{i} |\mathcal{D}_j|} \tag{4.3}$$

and describes the relationship between the number of clients that arrive in each step.

The following simple corollaries bound $H_{l_{\max}}$ and bring the competitive factor guaranteed by Theorem 4.5 into a more concrete and compact form.

Corollary 4.6. *The algorithm is at most* $4(3+K)l_{max} = \mathcal{O}(\log(l_{max})l_{max})$*-competitive for* FacilityLeasing.

Corollary 4.7. *If for each round, the number of clients at any time* t *does vary by at most a constant factor, is non-increasing, or bounded from above by a polynomial in* l_{max}*, the algorithm becomes at most* $\mathcal{O}(K \log(l_{max})) = \mathcal{O}(\log^2(l_{max}))$*-competitive for* FacilityLeasing.

While Corollary 4.7 arguably covers the most interesting and realistic cases, it seems probable that one can in fact construct an instance where the bound given in Corollary 4.6 is tight. Consider an arrival pattern where we have an exponential increase in

the number of clients in each round: $D_i = 2^i$. Intuitively, such arrival patterns seem to feature a unique hardness not only for this algorithm but for any online algorithm: At any time t, the number of arriving clients essentially matches the total number of clients that arrived up to now. Thus, in each single time step we have to solve a problem as hard as the complete problem up to the current time. It remains an interesting problem, whether such instances are inherently difficult to handle for online algorithms, or whether this conjectured lower bound is merely limited to this online algorithm.

4.5 Conclusion & Outlook

This chapter presented the first online algorithm for FacilityLeasing that has a competitive ratio independent on the input length and thereby on time. One can easily argue for the need of having such competitive factors specially when it comes to online problems.

The competitive bounds presented in this chapter can be written as $\mathcal{O}(Kl_{\max})$ and $\mathcal{O}(K\log(l_{\max}))$, respectively, since $K \leq \log(l_{\max}/l_{min})$. Furthermore, as the deterministic lower bound $\Omega(K)$ and the randomized lower bound $\Omega(\log K)$ for the ParkingPermitProblem carry over immediately to FacilityLeasing, one may hope to improve these bounds to $\mathcal{O}(l_{\max}\log(K))$ and $\mathcal{O}(\log(K)\log(l_{\max}))$, respectively, using randomization. Preliminary ideas about how to achieve this can be found in [47].

Another interesting direction, suggested by Pietrzyk [47] in his thesis, includes distributed algorithms, similar in spirit to [34, 48]. Such distributed and local implementations, where a solution is computed not by a central authority but a network of distributed sensor nodes (e.g., in our case, the facilities and clients), have attracted much interest in recent years, and the primal-dual approach the algorithm here uses has proven to be compatible with such a distributed model in other scenarios.

While previous work, including the results in this chapter, were focused on the leasing variant of the most *basic* model of FacilityLocation which is referred to as the

uncapacitated FacilityLocation, one may want to have a look at other variants of FacilityLocation. A first step could be to study the leasing variant of *capacitated* FacilityLocation in which facilities can serve limited number of clients per time step. A recent paper [49] by An et al. solves capacitated FacilityLocation using LP-based methodologies. The latter can be a strong starting point, by finding out whether these methodologies also carry over to the online/leasing variant of the problem.

Capacitated FacilityLocation is tightly connected to *scheduling*. In order to see this connection, let *machines* be the facilities and *jobs* be the clients. A machine can only serve a limited number of jobs per time step. Consequently, studying the leasing variant of FacilityLocation would mean studying the scheduling problem in which machines are *rented* rather than *bought*. Hence, it will be exciting to combine techniques from the two well-studied areas: scheduling and FacilityLocation, not only from a technical point of view but also practical. This also involves investigating different scheduling models including jobs with different execution durations, values, and additional precedence constraints between jobs. Furthermore, machines in capacitated FacilityLocation are identical. One may want to consider *heterogeneous* machines with different functionality or include set up costs for renting machines.

Chapter 5

Flexible Demands

Consider a travel agency that offers guided tours to tourists in a city. Each day, new tourists who want to attend the tour *before leaving the city* may arrive. The travel agency is willing to pay for each time a guide/tour is needed. To optimize its profit, it must make wise decisions regarding when to hire a guide and for how long since the longer (more consecutive days) a guide is hired, the lower the costs per day will be. Furthermore, once it hires a guide for some period of time, it cannot change its mind and tell the guide to stay for a shorter period.

As another example, consider clients who are flexible regarding when to use certain resources offered by a subcontracting company (e.g., any day *within two weeks* will do) and will be happy to be offered better resource prices for a later day. Since it does not own the resources and despite having more freedom regarding when to provide the service, the subcontracting company needs to make critical decisions of how long to wait before serving a client. Since the subcontractor does not know of future clients in advance, it may postpone a service to some day just to realize later on that it would have been cheaper on another day.

At the core of these examples, we have leasing decisions that include *deadlines*, which will be our subject of study in this chapter.

Chapter Basis. The results presented in this chapter are based on the following publication.

> Shouwei Li, Alexander Mäcker, Christine Markarian, Friedhelm Meyer auf der Heide, and Sören Riechers. "Towards Flexible Demands in Online Leasing Problems". In: *Proceedings of the 21st International Computing and Combinatorics Conference (COCOON)*, **2015** [12].

Chapter Outline. This chapter introduces a new leasing model in which demands have deadlines. Section 5.1 gives an overview of related literature along with a summary of results obtained in this chapter. Section 5.2 introduces the new model and gives a formal definition of the latter. Sections 5.3 and 5.4 present a deterministic algorithm and its analysis, respectively. Section 5.5 introduces SetCoverLeasingWithDeadlines and gives an algorithm for the latter. The chapter concludes with a short résume and future work in Section 5.6.

5.1 Related Work & Contribution

A standard assumption in most models for infrastructure problems is the permanence of the infrastructure purchased. Once a resource is bought, it is assumed it can be used any time in the future without inducing further costs that can be influenced by time or number of uses. In pursuit of better economies of scale, a number of models were introduced. In the Buy-at-Bulk model [50], number of uses matter, such that cost varies with the capacity a resource provides (the larger the capacity the cheaper per unit). Another well studied model is the Rent-or-Buy model [51], where apart from buying resources for various capacities, a resource can be bought and used forever at a larger cost. None of these models, however, consider the effect of time on the cost. In fact, deploying a server for a long period, for example, incurs maintenance and update costs which must be taken care of.

In the light of realizing influence of time on the cost of resources, Meyerson introduced the leasing model [2] with the ParkingPermitProblem. In the same paper, he introduced SteinerTreeLeasing, the leasing variant of SteinerTree. Given an undirected graph $G = (V, E)$ ($|V| = n$) and a cost for each edge $e \in E$. Pairs of communicating nodes announce themselves in each step. SteinerTreeLeasing asks to lease edges, in order to maintain at each step a path between each pair, while minimizing the total costs. Edges can be leased for K different periods of time with different costs. Meyerson gave an $\mathcal{O}(\log n \log K)$-competitive algorithm for the problem. Anthony et al. [5] generalized the ParkingPermitProblem to infrastructure leasing problems including SetCoverLeasing and FacilityLeasing. All related literature to these two variants were discussed in Chapters 3 and 4, respectively.

A common feature in all these models is that demands need to be served *on the spot*. However, this need not be always true. In many cases, although we do not know future demands in advance, we might in fact have demands which are *flexible*, meaning they have *deadlines* and can be served any time before their deadline. It is important to point out that these deadlines only make sense in problems in which resources are *leased* rather than *bought* since otherwise, it would always be better to postpone serving a demand until its last deadline.

Contribution. In this chapter, we introduce a new model where, in contrast to related work, demands do not have to be served *immediately*. As a natural extension, demands can be postponed up to some fixed period of time resulting in a *deadline* for each demand. Similar to the leasing model by Meyerson, a resource can be leased for K different periods of time each incurring a different cost, such that longer leases cost less per unit time. Each demand j can be served anytime between its arrival a_j and its deadline $a_j + d_j$. The objective is to meet all deadlines while minimizing the total leasing costs. This model is a natural generalization of Meyerson's ParkingPermitProblem [2] in which $d_j = 0$ for all j. Apart from introducing the model, we:

- give an online algorithm for the proposed model, with a $\Theta(K + \frac{d_{max}}{l_{min}})$-competitive

factor where d_{max} and l_{min} denote the largest d_j and the shortest available lease length, respectively.

- introduce the SetCoverLeasingWithDeadlines problem, an extension of SetCoverLeasing that includes deadlines.
- give an online competitive algorithm for SetCoverLeasingWithDeadlines, which also improves results for SetCoverLeasing.

5.2 The Leasing Framework with Deadlines

In this section, we will define the leasing framework with deadlines by introducing the OnlineLeasingWithDeadlines problem (OLD).

On each day t, a number of clients with deadlines $t + d_i$ (we say a client with *interval* $[t, t + d_i]$, where each day corresponds to a distance of 1 in the interval) arrives. There are K different types of leases, each with its own duration and cost. Longer leases tend to cost less per day. A client arriving on day t with deadline $t + d$ is *served* if there is a lease which covers at least one day of its interval. This also implies that we can replace all clients arriving on a day t by only the client with the lowest deadline that arrives on that day. Thus, we can assume without loss of generality that on every day t, either (i) no client or (ii) only one client with deadline $t + d$ arrives. The goal is to buy a set of leases such that all arriving clients are served while minimizing the total cost of purchases.

We distinguish between *uniform* OLD and *non-uniform* OLD as follows. All clients in *uniform* OLD have the same interval length, whereas clients in *non-uniform* OLD have different interval lengths.

A lease of type k has cost c_k and length l_k. l_{min} and l_{max} denote the shortest and the longest lease length, respectively. We denote by d_{max} and d_{min} the longest and the shortest interval length of the clients, respectively. An online algorithm now does not only need to serve clients while minimizing cost, but also needs to decide when to serve

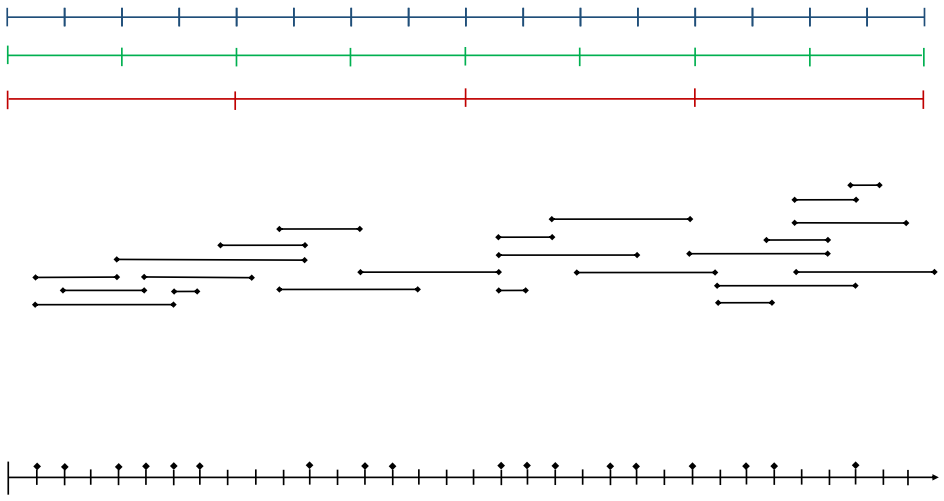


FIGURE 5.1: The Leasing Model with Deadlines

At the top, you see the leases in the interval model. The middle part represents the OLD model and the lower part, on the time line, is a special case of OLD: the PARKINGPERMITPROBLEM.

a client. Since resources expire after some time, decisions regarding when to serve a client are critical. An online algorithm may decide to serve a client on some day just to realize later on that postponing it would have been a better choice because a later lease could have served more clients. Or, the opposite is true, where an online algorithm may decide to postpone serving a client whereas serving it earlier by enlarging a lease that has been bought would have cost less.

The PARKINGPERMITPROBLEM is a special case of OLD if we just set d_{max} to 0. See Figure 5.1 for an illustration of our model compared to the PARKINGPERMITPROBLEM.

We formulate OLD using an integer linear program (ILP) (see Figure 5.2). We refer to a type k lease starting at time t as (k, t), a client arriving at time t with deadline $t + d$ as (t, d), and an interval $[a, a + b]$ as I_a^b. The collection of all leases is L and the collection of all clients is D. We denote by L_t all leases covering day t. We say a lease $(k, t') \in L$ is a *candidate* to client $(t, d) \in D$ if $I_t^d \cap I_{t'}^{l_k} \neq \emptyset$. The sum in the objective function represents the costs of buying the leases. The indicator variable $X_{(k,t)}$ tells us whether lease (k, t) is bought or not. The primal constraints guarantee that each client

$$\min \sum_{(k,t)\in L} X_{(k,t)} \cdot c_k$$

$$\text{Subject to: } \forall (t,d) \in D : \sum_{(k,t')\in L, I_t^d \bigcap I_{t'}^{l_k} \neq \emptyset} X_{(k,t')} \geq 1$$

$$\forall (k,t) \in L : X_{(k,t)} \in \{0,1\}$$

$$\max \sum_{(t,d)\in D} Y_{(t,d)}$$

$$\text{Subject to: } \forall (k,t) \in L : \sum_{(t',d)\in D, I_{t'}^d \bigcap I_t^{l_k} \neq \emptyset} Y_{(t',d)} \leq c_k$$

$$\forall (t,d) \in D : Y_{(t,d)} \geq 0$$

FIGURE 5.2: ILP Formulation of OLD

$(t,d) \in D$ is served. A dual variable $Y(t,d)$ is assigned to each client (t,d).

5.3 Online Algorithm

In this section, we present a deterministic primal-dual algorithm for OLD whose analysis will follow in Section 5.4.

We adopt the *interval model* (Lemma 2.6) in which leases of type k are available only at times t that are a multiple of the corresponding lease length l_k. Thus, any day t can be covered by exactly K different leases. Therefore, when a client $(t,d) \in D$ arrives, our algorithm needs to decide on which day $t' \in [t, t+d]$ to serve it and to specify one of the K leases in $L_{t'}$. For every lease $(k,t) \in L$, we define its *contribution* to be the sum of values of the dual variables corresponding to clients having (k,t) as a candidate. We say (t',d) *contributes* to (k,t) if (k,t) is a candidate of (t',d) and $Y_{(t',d)} > 0$. Two clients (t',d') and (t,d) with $t' < t$ *intersect* if their corresponding intervals $I_{t'}^{d'}$ and I_t^d intersect at $t' + d'$.

Algorithm. When a client (t, d) arrives, if it does not intersect any client (t', d') with a non-zero dual variable where $t' < t$, we perform the following two steps.

Step 1: We increase the dual variable $Y_{(t,d)}$ of the client until the constraint of some candidate (k, t') becomes tight, i.e.,

$$\sum_{(t,d)\in D: I_t^d \cap I_{t'}^{l_k} \neq \emptyset} Y_{(t,d)} = c_k$$

We then buy all the leases in L_t with a tight constraint (we set their primal variable to 1). At this point, the following proposition holds.

Proposition 5.1. *There exists at least one lease with a tight constraint that covers t.*

Proof. Assume, for contradiction, that there is no lease with a tight constraint in L_t. Then according to the algorithm, there must be a lease with a tight constraint in L_j, $j \in [t+1, t+d]$ (we do not stop increasing the client's dual variable until some constraint becomes tight). Moreover, before (t, d) arrives, the contribution to every lease in L_t is at least the contribution to its corresponding lease in L_j, $j \in [t+1, t+d]$. To show that the latter is true, assume, for contradiction, that there is a lease (k, t') in L_j, $j \in [t+1, t+d]$, with a contribution greater than that of its corresponding lease (k, t'') in L_t. Then, there must be a client which has contributed to (k, t') and not to (k, t'') (a client contributes the same to all its candidates). This is only possible if this client has arrived after (t, d), which is a contradiction. Hence, if the constraint of some lease in L_j, $j \in [t + 1, t + d]$, becomes tight when (t, d) arrives, then the constraint of its corresponding lease in L_t must become tight as well (at any day, there are exactly K lease types). □

Step 2: By the proposition above, we have that the algorithm buys at least one lease in L_t. Even though the client is now served, we do one more step. We buy the lease(s) from L_{t+d} which correspond(s) to what is bought in Step 1 from L_t (we set the primal variable(s) to 1).

5.4 Analysis

We now show that the primal-dual algorithm above is $\mathcal{O}(K)$-competitive for *uniform* OLD and $\mathcal{O}(K + \frac{d_{max}}{l_{min}})$-competitive for *non-uniform* OLD. We also show that the analysis of our algorithm is tight. This also implies an $\mathcal{O}(K)$-competitive factor for the ParkingPermitProblem ($d_{max} = 0$) which coincides with the tight result given by Meyerson [2].

Proposition 5.2. *Both the primal and the dual solutions constructed by the algorithm are feasible.*

Proof. It is easy to see that the dual constraints are never violated since the algorithm stops increasing the dual variables as soon as some constraint becomes tight. As for the primal solution, we show that each client $(t, d) \in D$ is served. When a client (t, d) arrives, we have two possibilities: either (t, d) intersects a previous client or it does not. If it does not, then our algorithm makes sure it is served in Step 1. Otherwise if it intersects a previous client (t', d) with $Y_{(t',d)}$ being zero, our algorithm makes sure it serves (t, d) in Step 1. If $Y_{(t',d)}$ is greater than zero, then our algorithm already covered days t' and $t'+d$ to serve (t', d). Since (t, d) and (t', d) intersect at $t'+d$, (t, d) is therefore served as well. □

Theorem 5.3. *The primal-dual algorithm achieves an optimal $\mathcal{O}(K)$- and an $\mathcal{O}(K + \frac{d_{max}}{l_{min}})$-competitive ratio for uniform and non-uniform OLD respectively.*

Proof. Let $P \subseteq L$ denote the primal solution constructed by the algorithm. Because the dual constraint is tight for each $(k, t) \in P$, we have

$$c_k = \sum_{(t',d) \in D : I^d_{t'} \cap I^{l_k}_t \neq \emptyset} Y_{(t',d)}.$$

Hence,

$$\sum_{(k,t) \in P} c_k = \sum_{(k,t) \in P} \sum_{(t',d) \in D : I^d_{t'} \cap I^{l_k}_t \neq \emptyset} Y_{(t',d)} = \sum_{(t',d) \in D} Y_{(t',d)} \sum_{(k,t) \in P : I^d_{t'} \cap I^{l_k}_t \neq \emptyset} 1.$$

Whenever the algorithm buys leases to serve $(t', d) \in D$, it only buys candidates from $L_{t'}$ (Step 1) and $L_{t'+d}$ (Step 2). Since there are exactly K leases at any day, it therefore buys at most $2K$ candidates. If the algorithm does not buy any further candidates of (t', d), we get an $\mathcal{O}(K)$-competitive ratio by weak duality theorem (both primal and dual solutions are feasible) since

$$\sum_{(k,t)\in P: I_{t'}^{d} \cap I_{t}^{l_k} \neq \emptyset} 1 \leq 2K.$$

This will be the case for *uniform* OLD since any client sharing common candidates with (t', d) intersects (t', d) at $t' + d$ thus being served at $t' + d$ and the algorithm does not buy any further candidates of (t', d). As for *non-uniform* OLD, the algorithm may buy more of (t', d)'s candidates when new clients sharing common candidates with (t', d) arrive in the coming days. We upper bound the total number of these candidates as follows.

$$\sum_{(k,t)\in P: I_{t'}^{d} \cap I_{t}^{l_k} \neq \emptyset} 1 \leq \sum_{i=t'}^{t'+d} |L_i| \leq \sum_{j=1}^{K} \left\lceil \frac{d_{max}}{l_j} \right\rceil$$

By Lemma 2.6 we have that l_j's are increasing and powers of two. Hence, the right sum above can be bounded by the sum of a geometric series with ratio half.

$$\sum_{j=1}^{K} \left\lceil \tfrac{d_{max}}{l_j} \right\rceil \leq K + d_{max} \left[\tfrac{1}{l_1} \left(\tfrac{1-(1/2)^K}{1-1/2} \right) \right] = K + d_{max} \left[\tfrac{2}{l_1} \left(1 - (1/2)^K \right) \right].$$

Since $K \geq 1$ we have

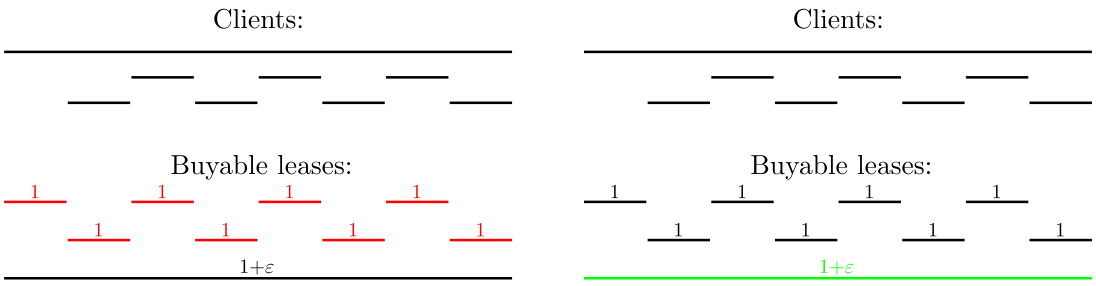
$$K + d_{max} \left[\tfrac{2}{l_1} \left(1 - (1/2)^K \right) \right] \leq K + \tfrac{d_{max}}{l_1}.$$

Therefore,

$$\sum_{(k,t)\in P: I_{t'}^{d} \cap I_{t}^{l_k} \neq \emptyset} 1 \leq K + \frac{d_{max}}{l_{min}}$$

since the algorithm can not buy more than $K + \frac{d_{max}}{l_{min}}$ candidates. □

Proposition 5.4. *The analysis of the aforementioned algorithm is tight.*



(A) Leases bought by our primal-dual algorithm are marked in red.

(B) Leases bought by an optimal algorithm are marked in green.

Comparison of our primal-dual and an optimal algorithm for a specific instance of our problem. It is easy to see that the primal-dual algorithm pays almost d_{max}/l_{min} times what the optimal algorithm would pay.

FIGURE 5.3: Tight Example

Proof. A lower bound of $\Omega(K)$ follows immediately from the lower bound of $\Omega(K)$ for the PARKINGPERMITPROBLEM by setting $d_{max} = 0$. We now give a tight example for $\Omega(d_{max}/l_{min})$ for the non-uniform case. Let d_{max} and l_{min} be arbitrary. For our problem instance, we start with a client $(0, d_{max})$ and add clients $((i-1) \cdot l_{min}, i \cdot l_{min})$ for $i \in \{2, \dots, \lfloor d_{max}/l_{min} \rfloor\}$. Similarly, we add 2 different lease types, one with length l_{min} and cost 1, and one with length $2^{\lceil \log_2(d_{max}) \rceil}$ and cost $1 + \varepsilon$. See Figure 5.3 for a visualization. Now, in order to cover client $(0, d_{max})$, the dual variable of this client is increased until

$$\sum_{(t,d) \in D : I_t^d \cap I_{t'}^{l_k} \neq \emptyset} Y_{(t,d)} = c_k \tag{5.1}$$

and this happens at the same time for all leases of length l_{min} in the interval $I_0^{d_{max}}$. The algorithm then only buys the leases at the start and at the end point. However, to cover clients $((i-1) \cdot l_{min}, i \cdot l_{min})$ for $i \in \{2, \dots, \lfloor d_{max}/l_{min} \rfloor\}$, the algorithm buys all the short leases, as constraint (5.1) is already tight from the prior step. This leads to an overall cost of at least $\lfloor d_{max}/l_{min} \rfloor$, whereas the optimal algorithm only buys the long lease with cost $1 + \varepsilon$. □

5.5 Application to Set Cover Leasing

In this section, we introduce the SETCOVERLEASINGWITHDEADLINES problem (SCLD) and give an $\mathcal{O}(\log(m \cdot (K + \frac{d_{max}}{l_{min}})) \log l_{max})$-competitive algorithm.

$$\min \sum_{(S,k,t)\in F} X_{(S,k,t)} \cdot c_S^k$$

$$\text{Subject to: } \forall (e,t,d) \in U : \sum_{(S,k,t')\in F, I_t^d \bigcap I_{t'}^{l_k} \neq \emptyset, e\in S} X_{(S,k,t')} \geq 1$$

$$\forall (S,k,t) \in F : X_{(S,k,t)} \in \{0,1\}$$

FIGURE 5.4: ILP Formulation of SCLD

5.5.1 Problem Definition

SCLD is a generalization of SETCOVERLEASING in which elements arrive over time and must be covered by sets from a family of subsets of these elements. Each set can be leased for K different periods of time. Leasing a set S for a period k incurs a cost c_S^k and allows S to cover its elements for the next l_k time steps. The objective is to minimize the total cost of the sets leased, such that elements arriving at any time t are covered by sets which contain them and are leased during time t. SCLD extends SETCOVERLEASING by allowing elements to have deadlines and be covered any time before their deadline. We define SCLD analogously to *non-uniform* OLD and formulate it using ILP (see Figure 5.4).

We denote by δ the maximum number of sets an element belongs to, by n the number of elements, and by m the number of sets. We refer to a set S with lease type k starting on day t as (S,k,t) and an element e arriving on day t with deadline $t+d$ as (e,t,d). The collection of all set triples is F and the collection of all element triples is U. We say $(S,k,t') \in F$ is a *candidate* to $(e,t,d) \in U$ if $e \in S$ and $I_t^d \cap I_{t'}^{l_k} \neq \emptyset$. The sum in the objective function represents the costs of buying the sets. The indicator variable $X_{(S,k,t)}$ tells us whether (S,k,t) is bought or not. An element is *covered* if at least one of its candidates is bought. The primal constraints guarantee that each $(e,t,d) \in U$ is covered.

5.5.2 Online Algorithm

In this section, we present a randomized algorithm for SCLD. We denote by $F_{(e,t,d)}$ the collection of all candidates of (e,t,d). Our algorithm first solves the LP of SCLD and then rounds it to solve its ILP. The algorithm maintains for each set $(S,k,t) \in U$, $2\lceil\log(l_{max})\rceil$ independent random variables $r_{(Skt)(q)}$, $1 \leq q \leq 2\lceil\log(l_{max})\rceil$, distributed uniformly in the interval $[0,1]$. We define $\mu_{Skt} \coloneqq \min\{r_{(Skt)(q)}\}$.

Algorithm 5 SetCoverLeasingWithDeadlines

When an element (e,t,d) arrives,
(i) (LP solution) *while* $\sum_{(S,k,t)\in F_{(e,t,d)}} X_{(S,k,t)} < 1$;

$$X_{(S,k,t)} = X_{(S,k,t)} \cdot (1 + 1/c_S^k) + \frac{1}{|F_{(e,t,d)}| \cdot c_S^k}$$

(ii) (ILP solution) Round $X_{(S,k,t)}$ to 1 if $X_{(S,k,t)} > \mu_{Skt}$ and if (e,t,d) is not yet covered, buy the cheapest $(S,k,t) \in F_{(e,t,d)}$ (set its primal variable to 1).

5.5.3 Analysis

We show that Algorithm 5 above is $\mathcal{O}(\log(\delta \cdot (K + \frac{d_{max}}{l_{min}}))\log l_{max}) = \mathcal{O}(\log(m \cdot (K + \frac{d_{max}}{l_{min}}))\log l_{max})$-competitive for SCLD.

It is easy to see that Algorithm 5 constructs a feasible solution ILP to SetCoverLeasing. To compute the total expected cost C_{ILP} of ILP, we first bound the cost of the LP solution C_{LP} by $\mathcal{O}(\log(\delta \cdot (K + \frac{d_{max}}{l_{min}}))) = \mathcal{O}(\log(m \cdot (K + \frac{d_{max}}{l_{min}}))) \cdot \text{Opt}$, where Opt is the optimal solution cost of ILP. Then, we show that C_{ILP} is at most $\mathcal{O}(\log l_{max})$ times C_{LP} and hence deduce the expected $\mathcal{O}(\log(m \cdot (K + \frac{d_{max}}{l_{min}}))\log l_{max})$-competitive factor of the algorithm.

To do so, we partition the time horizon into intervals of length l_{max}. Due to the interval model (Lemma 2.6), all leases of all sets end on days $i : i = 0 \mod l_{max}$. Hence, we bound C_{ILP} over any interval of length l_{max} by $\mathcal{O}(\log(m \cdot (K + \frac{d_{max}}{l_{min}}))\log l_{max})) \cdot \text{Opt}_{l_{max}}$, where $\text{Opt}_{l_{max}}$ is the optimum over the corresponding interval of length l_{max}. Summing up over all such intervals yields our competitive factor for SCLD.

Lemma 5.5. *The cost $C_{LP(l_{max})}$ of the LP solution over an interval of length l_{max} is at most $\mathcal{O}(\log(\delta \cdot (K + \frac{d_{max}}{l_{min}}))) \cdot \text{OPT}_{l_{max}} = \mathcal{O}(\log(m \cdot (K + \frac{d_{max}}{l_{min}}))) \cdot \text{OPT}_{l_{max}}$ where $\text{OPT}_{l_{max}}$ is the cost of the optimal solution over this interval.*

Proof. We fix any interval of length l_{max} from our partition. Any set (S_{OPT}, k, t') in the optimum solution over this interval has been a candidate for some element (e, t, d). When (e, t, d) arrives, our algorithm increases the primal variables of (e, t, d)'s candidates until they sum up to one. After $\mathcal{O}(c^k_{S_{OPT}} \cdot \log \left|F_{(e,t,d)}\right|)$ increases, $X_{(S_{OPT},k,t')}$ becomes greater than one and the algorithm makes no further increases. Furthermore, these increases never add a total of more than 2 to the primal variables. This is because

$$\sum_{(S,k,t) \in F_{(e,t,d)}} c^k_S \cdot \left(\frac{X_{(S,k,t)}}{c^k_S} + \frac{1}{c^k_S} \cdot \left|F_{(e,t,d)}\right| \right) \leq 2,$$

since $\sum_{(S,k,t) \in F_{(e,t,d)}} X_{(S,k,t)} < 1$ before the increase. The same holds for any other set in the optimum solution over this interval. Using a similar argument as in OLD, we can bound $\left|F_{(e,t,d)}\right|$ by $\delta \cdot (K + \frac{d_{max}}{l_{min}})$ (there are at most $K + \frac{d_{max}}{l_{min}}$ leases for each of the at most δ candidate sets). This completes the proof of the lemma. □

Lemma 5.6. *The cost $C_{ILP(l_{max})}$ of the ILP solution over an interval of length l_{max} is at most $\mathcal{O}(\log l_{max}) \cdot C_{LP(l_{max})}$, where $C_{LP(l_{max})}$ is the cost of the LP solution over this interval.*

Proof. We fix any interval of length l_{max} from our partition. The probability to buy a set $(S, k, t) \in F$ in this interval is proportional to the value of its primal variable. Hence, $C_{ILP(l_{max})}$ is upper bounded by

$$\sum_{(S,k,t) \in F} 2 \log(l_{max} + 1) \cdot c^k_S \cdot X_{(S,k,t)}$$

To guarantee feasibility, every time an element is not covered, the algorithm buys the cheapest candidate, which is a lower bound to $\text{OPT}_{l_{max}}$. The probability that an element

is not covered is at most $1/(l_{max})^2$. Since the random variables are drawn independently, we can add the expected costs incurred by the corresponding at most l_{max} elements and deduce a negligible expected cost of $l_{max} \cdot 1/(l_{max})^2 \cdot \text{OPT}_{l_{max}}$ which concludes the proof of the lemma. □

From the two lemmas above, we deduce the following theorem.

Theorem 5.7. *There is an online randomized algorithm for SCLD with a competitive factor of*

$$\mathcal{O}(\log(\delta \cdot (K + \frac{d_{max}}{l_{min}})) \log l_{max}) = \mathcal{O}(\log(m \cdot (K + \frac{d_{max}}{l_{min}})) \log l_{max})$$

SetCoverLeasing is nothing but a special case of SCLD if we set $d_{max} = 0$. Hence, we deduce the following corollary thereby improving the previous result for SetCoverLeasing [8] from $\mathcal{O}(\log(m \cdot K) \log n)$ to $\mathcal{O}(\log(m \cdot K) \log l_{max})$ by removing the dependency on n and therefore on time.

Corollary 5.8. *There is an online randomized algorithm for* SetCoverLeasing *that has a time-independent* $\mathcal{O}(\log(\delta \cdot K) \log l_{max}) = \mathcal{O}(\log(m \cdot K) \log l_{max})$*-competitive factor.*

5.6 Conclusion & Outlook

In this chapter, we extended the line of leasing by introducing a new model for online leasing problems, and as a first infrastructure leasing problem, we studied SetCoverLeasing with this model. Proceeding in this direction, one may want to look at other infrastructure leasing problems starting, for instance, with FacilityLeasing and SteinerTreeLeasing.

Our model introduces *flexibility* to demands with the aim to capture more general applications. Demands in our model have the flexibility of having a deadline. It will be interesting to extend this work to include models that handle other flexibilities (e.g., can be served on specific days within some period of time).

Furthermore, demands in our model require a single day to be served. Allowing demands that require more than one day to be served will be a natural extension of our model. Even though the techniques used in this chapter do not carry over directly to this extension, they still give the first insights. Along this direction, one may want to consider demands with *weights* and leases with *capacities*, such that a weight represents some load required to serve the corresponding demand, and a capacity represents how much load a lease can bear per unit time step.

Along the same line of leasing lies an important unanswered question of what happens if demands and/or their deadlines are given according to some probability distribution. The assumption made in all existing leasing models is indeed relatively strong: in many real-life scenarios, it is possible to predict what the future hides based on past events. While it is difficult to have an optimal degree of abstraction, one may still want to look at applications from actual markets in order to extend leasing models accordingly.

Another direction will be to consider lease prices changing over time, or in other words, prices also given according to some probability distribution. This clearly makes sense for many scenarios in which *fixed* prices are often hard to find.

Bibliography

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[2] Adam Meyerson. The parking permit problem. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 274–284, 2005. doi: 10.1109/SFCS.2005.72. URL http://dx.doi.org/10.1109/SFCS.2005.72.

[3] Christine Markarian and Friedhelm Meyer auf der Heide. Online resource leasing. To appear in 34th ACM Symposium on Principles of Distributed Computing (PODC'15), 2015.

[4] Sebastian Kniesburges, Christine Markarian, Friedhelm Meyer auf der Heide, and Christian Scheideler. Algorithmic aspects of resource management in the cloud. In *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, pages 1–13, 2014. doi: 10.1007/978-3-319-09620-9_1. URL http://dx.doi.org/10.1007/978-3-319-09620-9_1.

[5] Barbara M. Anthony and Anupam Gupta. Infrastructure leasing problems. In *IPCO*, pages 424–438, 2007.

[6] Piotr Berman and Bhaskar DasGupta. Approximating the online set multicover problems via randomized winnowing. *Theor. Comput. Sci.*, 393(1-3):54–71, 2008. doi: 10.1016/j.tcs.2007.10.047. URL http://dx.doi.org/10.1016/j.tcs.2007.10.047.

[7] Noga Alon, Yossi Azar, and Shai Gutner. Admission control to minimize rejections and online set cover with repetitions. *ACM Transactions on Algorithms*, 6 (1), 2009. doi: 10.1145/1644015.1644026. URL http://doi.acm.org/10.1145/1644015.1644026.

[8] Sebastian Abshoff, Christine Markarian, and Friedhelm Meyer auf der Heide. Randomized online algorithms for set cover leasing problems. In *Combinatorial Optimization and Applications - 8th International Conference, COCOA 2014, Wailea, Maui, HI, USA, December 19-21, 2014, Proceedings*, pages 25–34, 2014. doi: 10.1007/978-3-319-12691-3_3. URL http://dx.doi.org/10.1007/978-3-319-12691-3_3.

[9] Chandrashekhar Nagarajan and David P. Williamson. Offline and online facility leasing. *Discrete Optimization*, 10(4):361–370, 2013. doi: 10.1016/j.disopt.2013.10.001. URL http://dx.doi.org/10.1016/j.disopt.2013.10.001.

[10] Peter Kling, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. An algorithm for online facility leasing. In *Structural Information and Communication Complexity - 19th International Colloquium, SIROCCO 2012, Reykjavik, Iceland, June 30-July 2, 2012, Revised Selected Papers*, pages 61–72, 2012. doi: 10.1007/978-3-642-31104-8_6. URL http://dx.doi.org/10.1007/978-3-642-31104-8_6.

[11] Sebastian Abshoff, Peter Kling, Christine Markarian, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. Towards the price of leasing online. To appear in Journal of Combinatorial Optimization (JOCO).

[12] Shouwei Li, Alexander Mäcker, Christine Markarian, Friedhelm Meyer auf der Heide, and Sören Riechers. Towards flexible demands in online leasing problems. In *Computing and Combinatorics - 21st International Conference, COCOON 2015, Beijing, China, August 4-6, 2015, Proceedings*, pages 277–288, 2015. doi: 10.1007/978-3-319-21398-9_22. URL http://dx.doi.org/10.1007/978-3-319-21398-9_22.

[13] Vineet Jain. Security: The elephant in the cloud, 2015. URL http://www.forbes.com/sites/netapp/2015/04/16/security-elephant-cloud/.

[14] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001. ISBN 3-540-65367-8.

[15] Amos Fiat and Gerhard J. Woeginger, editors. *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*, 1998. Springer. ISBN 3-540-64917-4.

[16] George B. Dantzig and Mukund N. Thapa. *Linear Programming 1: Introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997. ISBN 0-387-94833-3.

[17] Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997. ISBN 0-534-94968-1.

[18] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of

complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 222–227, Washington, DC, USA, 1977. IEEE Computer Society. doi: 10.1109/SFCS.1977.24. URL http://dx.doi.org/10.1109/SFCS.1977.24.

[19] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. doi: 10.1287/moor.4.3.233. URL http://dx.doi.org/10.1287/moor.4.3.233.

[20] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi: 10.1145/285055.285059. URL http://doi.acm.org/10.1145/285055.285059.

[21] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974. doi: 10.1016/S0022-0000(74)80044-9. URL http://dx.doi.org/10.1016/S0022-0000(74)80044-9.

[22] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13 (4):383–390, January 1975. ISSN 0012-365X. doi: 10.1016/0012-365X(75)90058-8. URL http://dx.doi.org/10.1016/0012-365X(75)90058-8.

[23] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, 2(2):153–177, 2006. doi: 10.1145/1150334.1150336. URL http://doi.acm.org/10.1145/1150334.1150336.

[24] Piotr Berman, Bhaskar DasGupta, and Eduardo D. Sontag. Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene networks. *Discrete Applied Mathematics*, 155(6-7):733–749, 2007. doi: 10.1016/j.dam.2004.11.009. URL http://dx.doi.org/10.1016/j.dam.

2004.11.009.

[25] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009. doi: 10.1137/060661946. URL http://dx.doi.org/10.1137/060661946.

[26] Simon Korman. On the use of randomization in the online set cover problem. Master's thesis, Weizmann Institute of Science, Rehovot, Israel, December 2004.

[27] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009. doi: 10.1287/moor.1080.0363. URL http://dx.doi.org/10.1287/moor.1080.0363.

[28] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009. doi: 10.1287/moor.1080.0363. URL http://dx.doi.org/10.1287/moor.1080.0363.

[29] Giorgio Ausiello, Aristotelis Giannakos, and Vangelis Th. Paschos. Greedy algorithms for on-line set-covering and related problems, 2006.

[30] Kshipra Bhawalkar, Sreenivas Gollapudi, and Debmalya Panigrahi. Online set cover with set requests. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 64–79, 2014. doi: 10.4230/LIPIcs.APPROX-RANDOM.2014.64. URL http://dx.doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.64.

[31] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963. URL http://EconPapers.repec.org/RePEc:inm:ormnsc:v:9:y:1963:i:4:p:643-666.

[32] Alan S. Manne. Plant Location Under Economies-of-Scale–Decentralization and Computation. *Management Science*, 11:213–235, 1964. doi: 10.1287/mnsc.11.2.213.

[33] Nancy Casey and Michael R. Fellows. *This is Mega-Mathematics!* Los Alamos National Labs, 1992. Available at http://www.c3.lanl..gov/ captors/mega-math. See also www.ccs3.lanl.gov/mega-math/write.html.

[34] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 265–274, New York, NY, USA, 1997. ACM. ISBN 0-89791-888-6. doi: 10.1145/258533.258600. URL http://doi.acm.org/10.1145/258533.258600.

[35] Fabián A. Chudak and David P. Williamson. Improved approximation algorithms for capacitated facility location problems. *Math. Program.*, 102(2):207–222, March 2005. ISSN 0025-5610. doi: 10.1007/s10107-004-0524-9. URL http://dx.doi.org/10.1007/s10107-004-0524-9.

[36] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'11, pages 77–88, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22011-1. URL http://dl.acm.org/citation.cfm?id=2027223.2027230.

[37] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 649–657, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics. ISBN 0-89871-410-9. URL http://dl.acm.org/citation.cfm?id=314613.315037.

[38] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, March 2001. ISSN 0004-5411. doi: 10.1145/375827.375845. URL http://doi.acm.org/10.1145/375827.375845.

[39] Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM J. Comput.*, 32(3):816–832, March 2003. ISSN 0097-5397. doi: 10.1137/S0097539701383443. URL http://dx.doi.org/10.1137/S0097539701383443.

[40] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 50(6):795–824, November 2003. ISSN 0004-5411. doi: 10.1145/950620.950621. URL http://doi.acm.org/10.1145/950620.950621.

[41] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Improved approximation algorithms for metric facility location problems. In *In Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 229–242, 2002.

[42] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. A general approach to online network optimization problems. *ACM Trans. Algorithms*, 2(4):640–660, October 2006. ISSN 1549-6325. doi: 10.1145/1198513.1198522. URL http://doi.acm.org/10.1145/1198513.1198522.

[43] Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *J. of Discrete Algorithms*, 5(1):141–148, March 2007. ISSN 1570-8667. doi: 10.1016/j.jda.2006.03.001. URL http://dx.doi.org/10.1016/j.jda.2006.03.001.

[44] Dimitris Fotakis. On the competitive ratio for online facility location. In *Proceedings*

of the 30th International Conference on Automata, Languages and Programming, ICALP'03, pages 637–652, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-40493-7. URL http://dl.acm.org/citation.cfm?id=1759210.1759273.

[45] A. Meyerson. Online facility location. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 426–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1390-5. URL http://dl.acm.org/citation.cfm?id=874063.875567.

[46] Ari Freund and Dror Rawitz. Combinatorial interpretations of dual fitting and primal fitting. In *Approximation and Online Algorithms, First International Workshop, WAOA 2003, Budapest, Hungary, September 16-18, 2003, Revised Papers*, pages 137–150, 2003. doi: 10.1007/978-3-540-24592-6_11. URL http://dx.doi.org/10.1007/978-3-540-24592-6_11.

[47] Peter Pietrzyk. *Local and Online Algorithms for Facility Location*. PhD thesis, Department of Computer Science, University of Paderborn, Paderborn, Germany, September 2013.

[48] Saurav Pandit and Sriram V. Pemmaraju. Rapid randomized pruning for fast greedy distributed algorithms. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '10, pages 325–334, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-888-9. doi: 10.1145/1835698.1835777. URL http://doi.acm.org/10.1145/1835698.1835777.

[49] Hyung-Chan An, Mohit Singh, and Ola Svensson. Lp-based algorithms for capacitated facility location. *CoRR*, abs/1407.3263, 2014. URL http://arxiv.org/abs/1407.3263.

[50] Matthew Andrews and Lisa Zhang. Wavelength assignment in optical networks with fixed fiber capacity. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, pages 134–145, 2004. doi: 10.1007/978-3-540-27836-8_14. URL http://dx.doi.org/10.1007/978-3-540-27836-8_14.

[51] Anupam Gupta, Amit Kumar, Martin P´al, and Tim Roughgarden. Approximation via cost sharing: Simpler and better approximation algorithms for network design. *J. ACM*, 54(3), June 2007. ISSN 0004-5411. doi: 10.1145/1236457.1236458. URL http://doi.acm.org/10.1145/1236457.1236458.