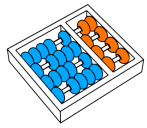
Universidade Estadual de Campinas
Instituto de Computação



Murilo Santos de Lima

Parking Permit and Network Leasing Problems

Problemas de Bilhetes de Estacionamento e Projeto de Redes com Arrendamento

CAMPINAS
2018

Murilo Santos de Lima

Parking Permit and Network Leasing Problems

Problemas de Bilhetes de Estacionamento e Projeto de Redes com Arrendamento

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Orlando Lee
Co-supervisor/Coorientador: Prof. Dr. Mário César San Felice

Este exemplar corresponde à versão final da Tese defendida por Murilo Santos de Lima e orientada pelo Prof. Dr. Orlando Lee.

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): FAPESP, 2014/18781-1; CNPq, 142161/2014-4; CAPES
ORCID: https://orcid.org/0000-0002-2297-811X

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

D379p

De Lima, Murilo Santos, 1987-
Parking permit and network leasing problems / Murilo Santos de Lima. – Campinas, SP : [s.n.], 2018.

Orientador: Orlando Lee.
Coorientador: Mário César San Felice.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Otimização combinatória. 2. Projeto de redes. 3. Algoritmos de aproximação. 4. Algoritmos on-line. I. Lee, Orlando, 1969-. II. San Felice, Mário César, 1985-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Problemas de bilhetes de estacionamento e projeto de redes com arrendamento
Palavras-chave em inglês:
Combinatorial optimization
Network design
Approximation algorithms
Online algorithms
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Orlando Lee [Orientador]
Aritanan Borges Garcia Gruber
Marco Serpa Molinaro
Eduardo Candido Xavier
Lehilton Lelis Chaves Pedrosa
Data de defesa: 11-05-2018
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Murilo Santos de Lima

Parking Permit and Network Leasing Problems

Problemas de Bilhetes de Estacionamento e Projeto de Redes com Arrendamento

Banca Examinadora:

- Prof. Dr. Orlando Lee (Presidente)
  Instituto de Computação – UNICAMP
- Prof. Dr. Aritanan Borges Garcia Gruber
  Centro de Matemática, Computação e Cognição – UFABC
- Prof. Dr. Marco Serpa Molinaro
  Departamento de Informática – PUC-Rio
- Prof. Dr. Eduardo Candido Xavier
  Instituto de Computação – UNICAMP
- Prof. Dr. Lehilton Lelis Chaves Pedrosa
  Instituto de Computação – UNICAMP

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 11 de maio de 2018

em memória do meu avô
Euzébio Bispo dos Santos (1924-2017)

All ya can do is do what you must
You do what you must do and ya do it well
(Bob Dylan)

Agradecimentos

Na minha dissertação de mestrado, eu agradeci a minha mãe por ter colocado meus estudos como prioridade, e a meu avô Euzébio por me ensinar a tocar "Buckets of Rain" do Dylan durante um sonho. Hoje eu entendo que meu avô me ensinou a tocar violão, mas também ensinou minha mãe a priorizar os estudos.

Na comunidade rural onde minha mãe nasceu, a escola só ia até a quinta série (atual sexto ano) do ensino fundamental. Meu avô obrigou minha mãe a repetir a quinta série várias vezes, pra não deixar de ir à escola. (Por fim ela se mudou pra cidade, concluiu o ensino médio e hoje tem duas formações técnicas.) Lembro que ele se gabava de, com apenas seis meses de estudo, ter exercido várias profissões: tropeiro, contador, juiz de paz e, curiosamente pra mim que sou combinatorista, caixeiro-viajante. Tendo feito tanto com tão pouco, desejou pra seus filhos o conhecimento que não pôde ter.

Agradeço então a meu avô, que partiu pra outra classe de complexidade em dez de junho do ano passado (um sábado, como ele tinha planejado), por ter deixado um legado que chegou até mim. Gostaria que ele tivesse gozado, em vida, o orgulho de um neto doutor. E agradeço a minha mãe, por ter se empenhado como pôde e como não pôde em transmitir esse legado.

Agradeço aos demais membros da minha família pelo amor e suporte.

Agradeço a meus tios Elisa e Harry e a Arthur Miranda pelo incentivo a entrar no doutorado.

Agradeço a meus orientadores Orlando Lee e Mário César San Felice pelo trabalho conjunto. Em particular, agradeço ao Lee pela compreensão em diversos momentos, e ao Mário por me tirar da zona de conforto e instigar a busca por demonstrações didáticas e pela compreensão da intuição por trás dos resultados.

Agradeço aos membros da banca examinadora pela disponibilidade e pelas considerações.

Agradeço aos demais professores que passaram pela minha vida e que me inspiraram o gosto pelo conhecimento, aos professores que lhes inspiraram, e assim recursivamente.

Agradeço aos amigos que fiz durante este período, em especial Atílio Gomes, André Silva, Maycon Sambinelli e Yulle Glebbyo. Aos demais amigos que me deram suporte e companhia neste período, em especial Caio Ravagnani, Paulo Ohana, João Moreira, João Alexandre Marson e Adriel Visoto. Agradeço a Fábio Leme pelos bons momentos que passamos juntos, período no qual consegui provar os resultados mais difíceis desta tese.

Agradeço à UNICAMP e ao Instituto de Computação pela oportunidade.

Agradeço ao CNPq (Processo 142161/2014-4) e à FAPESP em conjunto com a CAPES (Processo 2014/18781-1) pelo auxílio financeiro.

Agradeço aos dois últimos presidentes eleitos legitimamente no Brasil, pelo esforço em investir no ensino e na pesquisa. Minha trajetória acadêmica é fruto desse investimento.

E agradeço ao Regente, pelas probabilidades que me favoreceram e pelas que me trouxeram aprendizado.

Resumo

Em problemas de otimização tradicionais, é comum pensar que soluções são construídas adquirindo recursos que perduram no tempo. Em contrapartida, no modelo de otimização com arrendamento se supõe que os recursos podem ser arrendados por diferentes períodos de tempo e que, devido a uma economia de escala, o custo-benefício em arrendar um recurso por períodos mais longos é maior. Esse modelo tem recebido alguma atenção recentemente, por modelar problemas tais como a alocação de recursos na nuvem.

Nesta tese, estudamos o problema dos bilhetes de estacionamento, que é o problema fundamental de arrendamento, e propomos algumas generalizações. A primeira é o problema dos múltiplos bilhetes de estacionamento, que é uma generalização com múltiplos recursos idênticos. Esse problema pode ser resolvido em tempo polinomial. Mostramos também uma redução preservando aproximação para o problema original, que implica em um algoritmo online determinístico e outro probabilístico que são assintoticamente ótimos. A segunda variante proposta é o problema dos bilhetes de estacionamento em grupo, uma generalização do tipo aluguel-ou-compra, para a qual apresentamos uma 8-aproximação e um algoritmo online determinístico competitivo. A complexidade desse problema está em aberto, mas acreditamos que seja fracamente NP-difícil. Por fim, estudamos o problema dos bilhetes de estacionamento 2D, proposto por Hu, Ludwig, Richa e Schmid (2015). Os autores apresentaram um algoritmo com fator de aproximação constante e um algoritmo online determinístico competitivo para a versão hierárquica do problema, mas esses algoritmos consomem tempo pseudopolinomial. Nesta tese, mostramos como transformá-los em algoritmos de tempo polinomial. Mostramos também que o algoritmo de aproximação original funciona para a versão geral do problema, a qual provamos ser NP-difícil.

Esses resultados implicam em algoritmos de aproximação e algoritmos online competitivos para variantes com arrendamento dos problemas da rede de Steiner, do aluguel-ou-compra e do projeto de redes em atacado, através da técnica de aproximação de métricas finitas por métricas arbóreas. Em particular, conseguimos melhorar o fator de aproximação para a versão com arrendamento do problema do aluguel-ou-compra com múltiplos destinos.

Também revisamos algoritmos de aproximação para os problemas da localização de instalações com penalidades e do arrendamento de instalações, e apresentamos uma 3-aproximação para o problema do arrendamento de instalações com penalidades.

Por fim, revisamos algoritmos de aproximação e algoritmos online para o problema da localização de instalações conectadas. Propomos quatro variantes com arredamento desse problema, e apresentamos algoritmos de aproximação e algoritmos online competitivos para o caso em que o (menor) fator de escala é 1. Também discutimos por que alguns dos algoritmos clássicos para o problema da localização de instalações conectadas e as técnicas de análise disponíveis na literatura não são suficientes para obter bons algoritmos para as variantes com arrendamento quando o (menor) fator de escala não é uma constante.

Abstract

In traditional optimization problems, we can think that a solution is built by acquiring resources that persist in time. In contrast, in the leasing optimization model, we assume that resources may be leased for different lengths of time and that, due to economies of scale, it is more cost-effective to lease a resource for longer periods. This model has received some attention recently, since it models problems such as cloud resource allocation.

In this thesis, we study the parking permit problem, which is the seminal leasing problem, and we propose some generalizations. The first is the multi parking permit problem, which is a generalization with multiple identical resources. This problem can be solved in polynomial time, and we show how to reduce it to the parking permit problem, while losing a constant cost factor. This approximation-preserving reduction yields asymptotically optimal deterministic and randomized online algorithms. The second variant we propose is the group parking permit problem, a rent-or-buy generalization for which we give an 8-approximation algorithm and a deterministic competitive online algorithm. The complexity of this problem is open, but we believe it is weakly NP-hard. Finally, we study the 2D parking permit problem, proposed by Hu, Ludwig, Richa and Schmid (2015). They presented a constant approximation algorithm and a deterministic competitive online algorithm for the hierarchical version of the problem, but those algorithms have pseudo-polynomial running time. We show how to turn their algorithms into polynomial time. We also show that their original pseudo-polynomial offline algorithm works for the general version of the 2D parking permit problem, which we prove to be NP-hard.

Those results yield approximation and competitive online algorithms for leasing variants of the Steiner network problem, the rent-or-buy problem, and the buy-at-bulk network design problem, by using the technique of approximating a finite metric by a tree metric. In particular, we improve the previous best approximation algorithm for the leasing version of the multi-commodity rent-or-buy-problem.

We also review approximation algorithms for the facility location problem with penalties and the facility leasing problem, and we propose a 3-approximation algorithm for the facility leasing problem with penalties.

Finally, we review approximation and competitive online algorithms for the connected facility location problem. Then we propose four leasing variants of this problem, and we give approximation and competitive online algorithms for each of them when the (smallest) scale factor is 1. We also discuss why some classical algorithms for the connected facility location problem and the available analysis techniques in the literature do not suffice to obtain good algorithms for the leasing variants when the (smallest) scale factor is not a constant.

List of Figures

6.1 Example of a MPP solution ordered in HTO. 39
6.2 An illustration of the proof of Lemma 6.1. 40
6.3 Lemma 6.1 is no longer valid if we do not assume IM. 40

7.1 An execution of Algorithm 7.1. 44
7.2 An illustration of how we split S'. 46
7.3 An illustration of the Hanoi tower ordering for GPP. 46
7.4 An illustration of overlapping permits. 47
7.5 An illustration of the proof of Lemma 7.3. 48

8.1 Given an instance I, sub-instance $I[k, \hat{t}, r - \lambda]$. 57
8.2 An illustration of the representation of a solution as tuples. 60
8.3 We can split the optimum solution for $I[k-1]$ into layers of height ϕ_k. . . 60
8.4 An illustration of the binary search step. 61

9.1 A graph depicting dependency between parking permit problems. 63

10.1 A graph depicting dependency between the problems we study in Part II. . 73

11.1 Example which shows that the analysis of Algorithm 11.1 is tight. 79

12.1 An illustration of why Algorithm 12.1 produces a feasible solution. 86

A.1 Sequence r^k. 129
A.2 Optimum MPP solution for r^k. 129

List of Tables

9.1 Summary of known results for parking permit problems. 63

10.1 Summary of known results for network leasing problems. 73

14.1 Summary of our results for connected facility leasing problems. 105

List of Algorithms

5.1 Meyerson's deterministic online algorithm for PP [59]. 36

7.1 Approximation algorithm for GPP. 44
7.2 A modified version of Algorithm 7.1, which will be useful for our analysis. . 45

8.1 Algorithm for computing the maximum demand in each interval. 58
8.2 Pseudo-polynomial algorithm for 2DPP. 59

10.1 Online algorithm for SLE [59]. 68

11.1 Primal-dual algorithm for FL [46]. 77
11.2 Primal-dual algorithm for FLP [14]. 80

12.1 Primal-dual algorithm for FLE [60]. 86
12.2 Primal-dual algorithm for FLEP. 90

13.1 A first naïve algorithm for CFL. 95
13.2 A simple randomized sample-and-augment algorithm for CFL. 96
13.3 A more sophisticated randomized algorithm for CFL [35]. 97
13.4 Online algorithm for CFL [65]. 100
13.5 Approximation algorithm for MCFL [35]. 102

14.1 Approximation algorithm for CFLE. 106
14.2 A candidate randomized sample-and-augment algorithm for CFLE. 107
14.3 Online algorithm for CFLE. 108
14.4 Online algorithm for LECFLE. 109
14.5 Approximation algorithm for MCFLE. 111
14.6 Online algorithm for MCFLE. 112

List of Abbreviations and Symbols

$[K]$	$\{1, \ldots, K\}$
$(a)_+$	$\max\{a, 0\}$
2DIM	2D interval model
2DPP	2D parking permit problem
AlgMPP	optimum offline algorithm for multi parking permit problem
AlgOGPP	deterministic online algorithm for group parking permit problem
AlgOPP	deterministic online algorithm for parking permit problem
AlgPP	exact offline algorithm for parking permit problem
AlgRandOPP	randomized online algorithm for parking permit problem
BABND	buy-at-bulk network design problem
CFL	connected facility location problem
CFLE	connected facility leasing problem
CM	change-making problem
$\text{cost}(S)$	cost of solution S (implicit instance)
$\text{cost}(S, I)$	cost of solution S for instance I
FL	facility location problem
FLE	facility leasing problem
FLEP	facility leasing problem with penalties
FLP	facility location problem with penalties
FRT	algorithm of Fakcharoephol, Rao and Talwar for approximating a finite metric by a tree metric [27]
GPP	group parking permit problem
H2DPP	hierarchical 2D parking permit problem
HLEBABND	hierarchical leasing buy-at-bulk network design problem

HLP	hierarchical capacity property
HLP	hierarchical length property
HTO	Hanoi tower ordering
$I[k]$	see Definition 5.7
$I[k, \hat{t}]$	see Definition 5.8
$I[k, \hat{t}, r - \lambda]$	see Definition 8.6
IM	interval model
LeBaBND	leasing buy-at-bulk network design problem
LeCFLe	leasing-connected facility leasing problem
LeRoB	leasing rent-or-buy problem
$m_B(b)$	multiplicity of element b in multiset B
MCFL	multi-commodity connected facility location problem
MCFLe	multi-commodity connected facility leasing problem
MLeCFLe	multi-commodity leasing-connected facility leasing problem
MPP	multi parking permit problem
O2DPP	orthogonal 2D parking permit problem
OFL	online facility location algorithm
OFLe	online facility leasing algorithm
OFLeP	online algorithm for facility leasing problem with penalties
OLeBaBND	orthogonal leasing buy-at-bulk network design problem
opt(I)	value of an optimum solution for instance I
OSF	online Steiner forest algorithm
OST	online Steiner tree problem
PP	parking permit problem
RoB	rent-or-buy problem
SF	Steiner forest problem
SIM	simple interval model
SkiRental	ski rental problem
$S[k, t]$	see Definition 7.1

SN	Steiner network problem
SLE	Steiner leasing problem
SNLE	Steiner network leasing problem
ST	Steiner tree problem
STLE	Steiner tree leasing problem
val(S)	value of solution S (implicit instance)
val(S, I)	value of solution S for instance I

Contents

I Introduction **18**

1 Optimization Problems, Approximation Algorithms **19**

2 Online Algorithms and Competitive Analysis **21**

3 Leasing Optimization **25**

4 Outline of the Text and Contributions **28**

II Parking Permit Problems **30**

5 Ski Rental and Parking Permit **31**
5.1 Ski Rental . . . 31
5.2 Parking Permit . . . 32
5.2.1 The Interval Model . . . 33
5.2.2 A Deterministic Online Algorithm . . . 35

6 Multi Parking Permit **38**

7 Group Parking Permit **42**
7.1 Offline Group Parking Permit . . . 43
7.2 Online Group Parking Permit . . . 49

8 2D Parking Permit **54**
8.1 A Pseudo-Polynomial Algorithm for Generic 2DPP . . . 56
8.2 Hierarchical 2D Parking Permit . . . 59
8.3 General Results via the Covering Problem . . . 62

9 Summary and Discussion **63**

10 Consequences for Network Leasing Problems **65**
10.1 Approximating a Finite Metric by a Tree Metric . . . 65
10.2 Steiner Leasing . . . 66
10.3 Steiner Network Leasing . . . 69
10.4 Leasing Rent-or-Buy . . . 70
10.5 Leasing Buy-at-Bulk Network Design . . . 71
10.6 Summary . . . 73

III Facility Leasing Problems **74**

11 Facility Location, Sometimes with Penalties **75**
11.1 A Simple 3-Approximation Algorithm . . . 75
11.2 Facility Location with Penalties . . . 79
11.3 Online Facility Location, with or without Penalties . . . 82

12 Facility Leasing, Sometimes with Penalties **84**
12.1 Facility Leasing . . . 84
12.2 Facility Leasing with Penalties . . . 88

IV Connected Facility Leasing Problems **93**

13 Connected Facility Location **94**
13.1 Offline Connected Facility Location . . . 94
13.1.1 A First Naïve Algorithm . . . 94
13.1.2 A Simple Sample-and-Augment Algorithm . . . 96
13.1.3 A More Sophisticated Algorithm . . . 96
13.2 Online Connected Facility Location . . . 99
13.3 Multi-Commodity Connected Facility Location . . . 101

14 Connected Facility Leasing **104**
14.1 Connected facility leasing . . . 105
14.2 Leasing-connected facility leasing . . . 108
14.3 Multi-commodity connected facility leasing . . . 110

V Final Remarks **114**

15 Journey **115**

16 List of Results and Publications **118**

17 Open Questions and Further Research Directions **120**

Bibliography **122**

A A Bad Example for a Simple GPP Strategy **128**

Part I

Introduction

Chapter 1

Optimization Problems, Approximation Algorithms

An **optimization problem** consists of a set $\mathcal{I}$ of **instances** and, for each instance I in $\mathcal{I}$, a set $\mathrm{Sol}(I)$ of **feasible solutions** of I, and a function that assigns a value $\mathrm{val}(S, I) \in \mathbb{Q}$ to each solution S in $\mathrm{Sol}(I)$. (We may write $\mathrm{val}(S)$ when the instance I is implicit.) An instance is called **feasible** if it has some feasible solution. In a **minimization problem**, we wish to obtain a feasible solution of minimum value; in **maximization problems**, we wish to obtain a feasible solution of maximum value. In both cases, we use the terms "optimum value", "optimum solution" and "optimization problem" instead of minimum (maximum) value, minimum-cost (maximum-cost) solution, and minimization (maximization) problem. The value of any optimum solution for a feasible instance I is denoted by $\mathrm{opt}(I)$, or simply by opt if instance I is implicit. In this thesis, we consider only minimization problems, even though the definitions are analogous for maximization problems. For minimization problems, it is usual to call the value $\mathrm{val}(S, I)$ of a solution S of I as the **cost** of S, and denote it by $\mathrm{cost}(S, I)$, or $\mathrm{cost}(S)$ if I is implicit.

Let A be an algorithm which, for a minimization problem $\mathcal{P}$ with non-negative solution costs, returns a feasible solution $A(I)$ for each feasible instance I of $\mathcal{P}$. Given $\alpha : \mathcal{I} \mapsto \mathbb{R}$ and a real $d \geq 0$, we say that A is an **asymptotic α-approximation** for $\mathcal{P}$ if, for every feasible instance I, we have that

$$\mathrm{cost}(A(I), I) \leq \alpha(I) \cdot \mathrm{opt}(I) + d.$$

If $d = 0$, then A is an **α-approximation** for $\mathcal{P}$. Then, we say that A is an **approximation algorithm** for $\mathcal{P}$. The infimum of the numbers α that satisfy this inequality is the **approximation factor** of A and, clearly, $\alpha \geq 1$.

A **randomized algorithm** is one which has access to a random bit generator. A random bit generator is an algorithm which, given a rational number $p \in [0, 1]$, returns, in constant time, 1 with probability p and 0 with probability $1 - p$. Approximate implementations of such an algorithm can be found in Knuth [50].

Let A be a randomized algorithm which, for a minimization problem $\mathcal{P}$ with non-

negative solution costs, returns a feasible solution $A(I)$ for each feasible instance I of $\mathcal{P}$. Note that, in this case, $\text{cost}(A(I), I)$ is a random variable, whose probability space is defined by the random calls made by A. Given $\alpha : \mathcal{I} \mapsto \mathbb{R}$ and a real $d \geq 0$, we say that A is an **asymptotic randomized α-approximation** for $\mathcal{P}$ if, for every feasible instance I, we have that

$$\mathbb{E}[\text{cost}(A(I), I)] \leq \alpha(I) \cdot \text{opt}(I) + d.$$

If $d = 0$, then A is a **randomized α-approximation** for $\mathcal{P}$. Then, we say that A is a **randomized approximation algorithm** for $\mathcal{P}$. The infimum of the numbers α that satisfy this inequality is the **expected approximation factor** of A.

Approximation algorithms are of special interest to NP-hard optimization problems, for which there do not exist polynomial-time algorithms that always obtain an optimum solution, unless P = NP. For such problems, a polynomial-time approximation algorithm returns a solution whose cost is bounded by a factor of the optimum value, and this may be good enough for a practical application.

As an example, consider the **Steiner tree problem** (ST).

Problem ST(G, d, D): *Given a graph $G = (V, E)$, a function $d : E \mapsto \mathbb{Q}_+$ assigning a length to each edge, and a subset of the vertices $D \subseteq V$, find a subset of the edges $\mathcal{T} \subseteq E$ such that there exists a path between each pair of vertices of D in the graph $(V, \mathcal{T})$, and such that $\sum_{e \in \mathcal{T}} d(e)$ is minimum.*

If edge lengths are positive, then obviously $G[S]$ is a tree. This is an NP-hard problem [33], and currently there is a 1.39-approximation algorithm by Byrka *et al.* [12].

Theorem 1.1 (Byrka, Grandoni, Rothvoß, and Sanità [12]): *There is a 1.39-approximation algorithm for* ST.

Bibliographical notes. Some parts of the text in this chapter were adapted from the author's Master's thesis [17, Sections 2.3 and 2.4], and are based on Chapters 1 and 6 of de Carvalho *et al.* [16].

Chapter 2

Online Algorithms and Competitive Analysis

The problems described in the previous chapter are **offline** problems, in the sense that the whole input is available to the algorithm at the beginning of the computation, and the whole structure of the input can be evaluated in order to obtain the best possible solution. Furthermore, no causality constraints are assumed on the desired output. It is of practical interest, however, that those results can be extended to models in which the knowledge about the input is limited and/or causality constraints are imposed on partial solutions.

One such model about which there is an extensive literature is the **online computation** model [10]. Informally, in an **online** optimization problem, the input is partially available to the algorithm, which must take a decision without knowing the remaining of the input, and this decision cannot be revoked in the future. An algorithm for an online optimization problem is an **online algorithm**. In contrast, an optimization problem that does not assume these constraints, such as those defined in Chapter 1, is an **offline** problem.

It is not a trivial task to define online optimization problems in a generic manner, such as we did for offline optimization problems in Chapter 1. In the literature, the usual approach is to define each problem in particular. In this chapter we give a definition which is sufficiently generic for the problems we study in this thesis. For a more formal definition, see Borodin and El-Yaniv [10, Chapter 7].

The input of an online problem consists in the following. Consider an optimization problem $\mathcal{P} = (\mathcal{I}, \mathrm{Sol}, \mathrm{cost})$ with instance set $\mathcal{I}$, feasible solution set $\mathrm{Sol}(I)$ for each $I \in \mathcal{I}$, and a function assigning a value $\mathrm{cost}(S, I) \in \mathbb{Q}$ to each solution $S \in \mathrm{Sol}(I)$. In the **online version** of $\mathcal{P}$, we assume that each instance $I \in \mathcal{I}$ is a pair $I = (X, Y)$, where $Y = Y_0, \ldots, Y_{T-1}$ is a sequence for some $T \in \mathbb{Z}_+$. We say that X is the **offline portion** of the input, and $Y_0, \ldots, Y_{T-1}$ is the **request sequence**. We are also given a partial ordering $\preceq$ on the elements of $\bigcup_{I \in \mathcal{I}} \mathrm{Sol}(I)$; intuitively, $\preceq$ indicates which solutions can be built from others. A solution for this instance consists in a sequence $S_0, \ldots, S_{T-1}$ such that, for $0 \leq t < T$, $S_t \in \mathrm{Sol}(X, Y_0, \ldots, Y_t)$, and $S_{t-1} \preceq S_t$ for $0 < t < T$. (I.e., solution S_t must be built from solution S_{t-1}.) We say that $S_0, \ldots, S_{T-1}$ is a $\preceq$**-incremental**

sequence of solutions of $(X, Y_0, \ldots, Y_{T-1})$. An **online algorithm** is one that, given an instance $(X, Y_0, \ldots, Y_{T-1})$ of the online version of $\mathcal{P}$, returns a $\preceq$-incremental sequence $S_0, \ldots, S_{T-1}$ of solutions for that instance, with the constraint that S_t must be built only with the knowledge of $S_0, \ldots, S_{t-1}$ and $X, Y_0, \ldots, Y_t$, for $t = 0, \ldots, T-1$. In this thesis, sometimes we abuse notation and only require that the algorithm returns the final solution S_{T-1}, as long as it is built in an online fashion. We may also abuse terminology and say that an online algorithm for the online version of $\mathcal{P}$ is an online algorithm for $\mathcal{P}$.

As an example, we define the online version of ST, which was defined in Chapter 1. Informally, at each instant, a new set of terminal vertices arrives, and the algorithm must buy edges that connect them to the terminal vertices in the previous solution. The algorithm has no knowledge of the future sets of terminals, and it is not allowed to remove edges that were bought previously. Thus, the offline portion of the input is the graph G and edge lengths d, and the request sequence is a sequence of subsets of vertices $D_0, \ldots, D_{T-1} \subseteq V$. (It is usual to assume that $|D_t| = 1$ for $t = 0, \ldots, T-1$.) An online algorithm begins with an empty set of edges and, given a solution $\mathcal{T}_{t-1}$ for $(G, d, D_0, \ldots, D_{t-1})$, chooses a solution $\mathcal{T}_t$ for $(G, d, D_0, \ldots, D_t)$ such that $\mathcal{T}_{t-1} \subseteq \mathcal{T}_t$. The following is a formal definition of the **online Steiner tree problem** (OST).

Problem OST$(G, d, D_0, \ldots, D_{T-1})$: *Given a graph $G = (V, E)$, a function $d : E \mapsto \mathbb{Q}_+$ assiging a length to each edge, and a sequence $D_0, \ldots, D_{T-1} \subseteq V$ of subsets of vertices, find a sequence $\mathcal{T}_0, \ldots, \mathcal{T}_{T-1} \subseteq E$ of subsets of edges, with $\mathcal{T}_{t-1} \subseteq \mathcal{T}_t$ for $t = 1, \ldots, T-1$, such that there exists a path between each pair of vertices of $D_0 \cup \ldots \cup D_t$ in the graph $(V, \mathcal{T}_t)$, and which minimizes $\sum_{e \in \mathcal{T}_{T-1}} d(e)$. Furthermore, $\mathcal{T}_t$ must be calculated only with the knowledge of $\mathcal{T}_0, \ldots, \mathcal{T}_{t-1}$ and $G, d, D_0, \ldots, D_t$, for $t = 0, \ldots, T-1$.*

The usual way of analyzing the quality of an online algorithm is the **competitive analysis**. The goal is to establish a worst-case guarantee, by comparing the cost of the solution returned by the online algorithm to the **offline optimum cost**, which is the cost of a solution returned by a hypothetical algorithm that knows the whole sequence of requests in advance. This is a rather pessimistic metric; in practice, algorithms with poor competitive factor sometimes obtain satisfactory solutions for typical inputs[1]. Then, another way of analyzing an online algorithm is the **average-case analysis**, in which a probability distribution is assumed on the input sequences, and the expected cost for a solution returned by the algorithm is calculated, given an input chosen according to that probability distribution. However, for most problems it is not easy to identify a typical probability distribution. Moreover, competitive analysis gives a theoretical insight on the hardness the online constraints impose on the problem. In this thesis, we are interested in competitive analysis, which is formalized in the following paragraph.

Given an instance I of an optimization problem $\mathcal{P}$, denote by $\text{opt}(I)$ the offline optimum cost of I. Let $\mathcal{P}$ be a minimization problem with non-negative solution costs, $\mathcal{I}$ the set of instances of $\mathcal{P}$, and $\preceq$ a partial ordering on $\bigcup_{I \in \mathcal{I}} \text{Sol}(I)$. Let A be an online algorithm for $(\mathcal{P}, \preceq)$. Given $\alpha : \mathcal{I} \mapsto \mathbb{R}$ and a real $d \geq 0$, if, for every feasible instance

[1] On this topic, see for example the review by Karlin on experiments with online algorithms for the ski rental problem [48].

$I = (X, Y_0, \ldots, Y_{T-1})$ of $\mathcal{P}$, algorithm A returns a $\preceq$-incremental sequence of feasible solutions $S_0, \ldots, S_{T-1}$ of this instance, such that

$$\text{cost}(S_{T-1}, I) \leq \alpha(I) \cdot \text{opt}(I) + d,$$

then we say that A is α**-competitive**. The infimum of the numbers α that satisfy this inequality is the **competitive factor** of A.

For example, the naïve greedy algorithm for OST, in which each new terminal is connected to the closest previous terminal, is a good online algorithm. Its competitive factor was proven to be $\text{O}(\lg n)$ by Imase and Waxman [43] (for a simpler proof, check [72]), where $n := \left|\bigcup_{t=0}^{T-1} D_t\right|$ is the number of terminals. This is asymptotically optimal since, for every online algorithm for OST, there is an instance for which the competitive factor is $\Omega(\lg n)$ [43].

Theorem 2.1 (Imase and Waxman [43]): *The greedy algorithm for* OST *is* $\text{O}(\lg n)$*-competitive, and any online algorithm for* OST *is* $\Omega(\lg n)$*-competitive, where* n *is the number of terminals.*

Note that the constraint of not knowing the complete instance can lead to a high competitive ratio. One way of trying to overcome this is by the use of randomization. In the following paragraph, we formalize the definition of expected competitive ratio of a randomized online algorithm. For some online problems, it is possible to achieve a randomized algorithm whose expected competitive ratio is asymptotically better than the competitive ratio of the best possible deterministic algorithm.

Let A be an online randomized algorithm for a minimization problem $(\mathcal{P}, \preceq)$ with non-negative solution costs and instance set $\mathcal{I}$. Let $\alpha : \mathcal{I} \mapsto \mathbb{R}$ and a real $d \geq 0$, if, for every feasible instance $I = (X, Y_0, \ldots, Y_{T-1})$ of $\mathcal{P}$, algorithm A returns a $\preceq$-incremental sequence of feasible solutions $S_0, \ldots, S_{T-1}$ of this instance, such that

$$\mathbb{E}[\text{cost}(S_{T-1}, I)] \leq \alpha(I) \cdot \text{opt}(I) + d,$$

then we say that A is α-competitive[2] (in the randomized sense). The infimum of the numbers α that satisfy this inequality is the **expected competitive factor** of A.

In the competitive analysis of online algorithms, in contrast to the study of polynomial-time approximation algorithms, we do not always assume constraints on time and space consumption; the concern on the computational complexity of the algorithms is secondary. In general, it is desirable to obtain efficient algorithms, but the main interest is to understand if the constraints of incrementality and lack-of-knowledge impose a loss in the guarantee of quality of the returned solutions and, in the affirmative case, to bound such loss. In particular, sometimes we have a less efficient algorithm which obtains better solutions, and a more efficient algorithm which obtains solutions of higher cost.

[2]The randomized online algorithm model we assume corresponds to the **oblivious adversary** model described in [10, Section 7.1], in which the adversary knows the code of the algorithm, but does not know the values returned by the random calls the algorithm does in runtime.

Bibliographical notes. Some parts of the text in this chapter were adapted from the author's Master's thesis [17, Section 2.5], and are based on Section 1.1 and Chapter 7 of Borodin and El-Yaniv [10].

Chapter 3

Leasing Optimization

In combinatorial optimization, we usually think of minimization problems as those in which we must buy certain resources to satisfy some objective, by paying the minimum cost possible. In network design problems, in particular, those resources are the nodes or connections in a network infrastructure that must be built to serve user/client requests. For example, we can think that, in the facility location problem [24], we wish to determine where to install servers, given the geographical distribution of the clients that wish to be served.

However, in many scenarios, it is not feasible for a small company, such as a start-up, to install its own servers, due to a small budget. Also, client requests are usually clustered in certain intervals of time. A current trend in those cases is to **lease** a server in the cloud (from a service such as Amazon AWS, Google App Engine or Microsoft Azure) for specific lengths of time. This scenario has recently motivated the proposal of a class of problems known as **leasing optimization** [1, 4, 59, 60]. In those problems, instead of acquiring resources that last for an unlimited period, each resource may be leased for different lengths of time (e.g., a day, a week, a month), after which they expire. In this context, it is cheaper to lease a resource for a longer amount of time than for smaller intervals totalizing an equivalent time; this represents economies of scale. This model may be applied to both offline and online problems.

As an example, we define the leasing version of ST. In this version, we have a fixed root vertex r, and the sets of terminals are distributed along the time, even in the offline setting. Edges are no longer permanent, but can be leased for one of K different lengths of time $\delta_1, \ldots, \delta_K \in \mathbb{N}$, after which edge leases expire. Let $[K] := \{1, \ldots, K\}$. The cost of leasing edge e for length of time δ_k is $d(e) \cdot \gamma_k$, where γ_k is a uniform leasing factor for leasing type $k \in [K]$. If we lease edge e at instant $\hat{t}$ for length of time δ_k, then edge e is **active** for the interval of time $[\hat{t}, \hat{t} + \delta_k)$. The same edge may be leased many times in different instants of time, and more than one lease may be active for the same edge at the same instant. We wish that, from each terminal j arriving at instant t, there is a path from j to r consisting of edges that have an active lease at instant t. The following is a formal definition of the problem, which we call the **Steiner tree leasing problem** (STLE).

Problem STLE$(G, d, r, K, \delta, \gamma, D_0, \ldots, D_{T-1})$: *We are given a graph $G = (V, E)$, a function $d : E \mapsto \mathbb{Q}_+$ that assigns a length to each edge, a root vertex r, an integer K which represents the number of leasing types, a function $\delta : [K] \mapsto \mathbb{N}$ that assigns a length of time to each leasing type, a function $\gamma : [K] \mapsto \mathbb{Q}_+$ that assigns a leasing factor to each leasing type, and a sequence $D_0, \ldots, D_{T-1} \subseteq V$ of sets of terminals. The goal is to find a set of edge leases $\mathcal{E} \subseteq E \times [K] \times \mathbb{Z}_+$ such that, for every terminal $j \in D_t$, there exists a jr-path P in G such that, for each edge $e \in P$, there is some $(e, k, \hat{t}) \in \mathcal{E}$ such that $t \in [\hat{t}, \hat{t} + \delta_k)$, and we wish to minimize $\sum_{(e,k,\hat{t})\in\mathcal{E}} d(e) \cdot \gamma_k$.*

This is an NP-hard problem, since ST reduces to it if we set $K = 1$, $\delta_1 = \infty$, and $\gamma_1 = 1$. There is a $O(K)$-approximation algorithm for this problem by Anthony and Gupta [4].

In the online version of the problem, we know in advance the graph G, the function d, the root r, the number of leasing types K, leasing lengths δ and leasing factors γ. At each instant of time $t = 0, \ldots, T-1$, we receive a set of terminals D_t that we must connect to the root using edge leases, but we do not know the future sets of terminals, and we cannot remove old edge leases. There is a randomized $O(\lg K \lg |V|)$-competitive online algorithm for the problem, which was proposed by Meyerson for the leasing version of the Steiner forest problem [59]. This algorithm is presented in Section 10.2. More recently, Bienkowski, Kraska and Schmidt presented a deterministic $O(K \lg n)$-competitive online algorithm for STLE [9], where n is the number of terminals.

In this thesis we study approximation and online algorithms for a wide range of problems that arise in the leasing optimization model.

Related Work

The seminal leasing problem is the **parking permit problem**, proposed by Meyerson [59]. This problem corresponds to leasing a single resource (a parking location), and it is presented at Section 5.2. In the offline setting the problem can be solved in polynomial time, and Meyerson presented a deterministic $O(K)$-competitive online algorithm and a randomized $O(\lg K)$-competitive online algorithm, where K is the number of leasing types. He also showed that this is asymptotically the best possible, by showing that any deterministic algorithm for this problem has competitive factor $\Omega(K)$, and any randomized algorithm has competitive factor $\Omega(\lg K)$. In the same paper, Meyerson presented a randomized online algorithm for the leasing version of the Steiner forest problem, which is called the **Steiner leasing problem**. His algorithm is $O(\lg K \lg |V|)$-competitive, where K is the number of leasing types and $|V|$ is the number of vertices in the graph, and it is discussed in Section 10.2.

After that, Anthony and Gupta [4] studied the offline leasing version of some NP-hard network design problems. Specifically, they proved that solving an offline problem with K leasing types reduces, within a constant cost factor, to solving the K-stage stochastic version of the same problem. Thus, based on previous K-stage stochastic algorithms, they obtained $O(K)$-approximation algorithms for STLE and the **facility leasing problem**,

as well as an 8-approximation for the **vertex cover leasing problem**, and a $O(\lg n)$-approximation algorithm for the **set cover leasing problem**, where n is the size of the universe set. Moreover, the authors proposed new algorithms for the stochastic version of the **rent-or-buy problem** and the **buy-at-bulk network design problem**, using the technique of **cost sharing** [36], thus obtaining $O(K)$-approximation algorithms for single-source leasing rent-or-buy and single-source leasing buy-at-bulk network design, and a $O(K \lg n)$-approximation algorithm for multi-source leasing buy-at-bulk network design problem, where n is the number of demand points.

Nagarajan and Williamson [60] improved the result for the offline version of the facility leasing problem, obtaining a 3-approximation which we present in Section 12.1. For the online version, they obtained a $O(K \lg n)$-competitive algorithm, where n is the number of client requests and K is the number of leasing types.

Koutris [53] reviewed several results for network leasing problems, and presented a randomized $O(K \lg n / \lg \lg n)$-competitive algorithm for the online facility leasing problem. Fotakis and Koutris [32] also proposed the **online sum-radii clustering problem**, which is related to the parking permit problem.

Abshoff, Kling, Markarian, auf der Heide and Pietrzyk [1] also improved the result for the online facility leasing problem, this time obtaining a $O(\delta_K \lg \delta_K)$-competitive algorithm, where δ_K is the highest leasing length. This result removes the temporal dependency on the competitive factor of the algorithm. They also presented an online algorithm for the leasing set cover problem which is $O(\lg(Km) \lg n)$-competitive, where n is the size of the universe set, m is the number of cover sets and K is the number of leasing types.

San Felice, Cheung, Lee, Williamson and Fernandes [63] gave a $O(K \lg n)$-competitive algorithm for the online facility leasing problem with penalties.

Bienkowski, Kraska and Schmidt presented a deterministic $O(K \lg n)$-competitive online algorithm for STLE [9].

Hu, Ludwig, Richa and Schmid [41] proposed a bidimensional version of the parking permit problem, and presented an 8-approximation algorithm and a deterministic $O(K)$-competitive algorithm, both however with pseudo-polynomial running time. They conjectured that the problem is NP-hard, which we prove in Chapter 8.

Variants of the parking permit problem and the leasing model have also been proposed. In [55], Li, Mäcker, Markarian, auf der Heide and Riechers discussed the variant in which each demand has a time window to be served after its arrival, and also present an online competitive algorithm for the set cover leasing problem in this model. More recently, Feldkord, Markarian and auf der Heide studied what happens when leasing prices fluctuate along time [28].

Chapter 4

Outline of the Text and Contributions

In this chapter, we present an outline of the text and what are our contributions.

In Part II, which comprises Chapters 5–10, we study the parking permit problem and some generalizations.

In Chapter 5, we review the ski rental problem and the parking permit problem, and we discuss the relationship between them. We also present the interval model, which is an important concept in the development of algorithms for parking permit problems. Finally, we present the deterministic online algorithm for the parking permit problem by Meyerson [59].

In Chapter 6, we propose the multi parking permit problem, which is a generalization of the parking permit problem with multiple identical resources. We show that the problem can be solved in polynomial time, and we present an approximation-preserving reduction to the parking permit problem, which yields deterministic $O(K)$-competitive and randomized $O(\lg K)$-competitive online algorithms. Those results are asympotically optimal due to the lower bounds for the parking permit problem [59].

In Chapter 7, we propose the group parking permit problem, which is a rent-or-buy generalization of the multi parking permit problem. The complexity of this problem is open, but we show that the natural LP relaxation has non-trivial integrality gap. We present an 8-approximation algorithm and a deterministic $O(K)$-competitive online algorithm. The online algorithm is asympotically optimal, due to the deterministic lower bound for the parking permit problem [59]. We consider the results in this chapter to be the most interesting in this thesis. In particular, the analysis of the approximation/competitive factor of the algorithms we propose are unorthodox, in the sense that we do not give a lower bound to the cost of an optimum solution directly, but instead we bound the cost of our algorithms by a hypothetical algorithm that knows some information about the optimal solution.

In Chapter 8, we review the 2D parking permit problem, which was proposed by Hu *et al.* [41]. They presented an 8-approximation algorithm and a deterministic $O(K)$-competitive online algorithm, both however having pseudo-polynomial running time. We show how to turn their algorithms for this problem into polynomial-time. We also show that their original pseudo-polynomial approximation algorithm is a 4-approximation for a less restricted case of the problem, which we prove to be NP-hard. Finally, we discuss

the greedy approximation algorithm for the covering problem by Koufogiannakis and Young [52], which yields a $O(K)$-approximation and a $O(K)$-competitive online algorithm for the general 2D parking permit problem, thus generalizing some of the results we obtained.

In Chapter 9, we summarize the results obtained in the previous chapters, and we discuss their relevance. In particular, see Table 9.1 for a list of results, and Figure 9.1 for a dependency graph between the problems.

Finally, in Chapter 10 we discuss the relationship between those parking permit problems and some network leasing problems. We review the technique of approximating a finite metric by tree metrics [7, 27], and we show how Meyerson applied it to the parking permit problem to obtain approximation and online algorithms for a leasing variant of the Steiner forest problem. We follow this approach and, using the results in Chapters 6–8, we obtain approximation and online algorithms for leasing variants of the Steiner network problem, the rent-or-buy problem and the buy-at-bulk network design problem. In particular, the approximation algorithm for the leasing rent-or-buy problem improves the previous result by Anthony and Gupta [4]. The results are summarized in Table 10.1, and in Figure 10.1 we depict the dependency between the problems we study in Part II.

Part of those results were published in [22], and a full paper was submitted to a journal [23].

In Part III, which comprises Chapters 11 and 12, we discuss facility location and facility leasing problems. In Chapter 11, we review the facility location problem and the facility location problem with penalties. We review some primal-dual approximation algorithms [46, 14], and we discuss the online versions of those problems. In Chapter 12, we review the primal-dual approximation algorithm for the facility leasing problem by Nagarajan and Williamson [60], and we show how to extend it to obtain a 3-approximation algorithm for the facility leasing problem with penalties. (This result was published in [21].) We also discuss results in the literature for the online versions of those problems.

In Part IV, which comprises Chapters 13 and 14, we discuss connected facility location and connected facility leasing problems. In Chapter 13, we review some approximation and online algorithms for connected facility location problems. In Chapter 14, we propose four leasing variants of the connected facility problem. We present approximation and online algorithms for a special case of those problems. For the general case, we also prove that simple techniques that yield good algorithms for the connected facility location problem perform badly in the leasing model, and we discuss why we could not extend more sophisticated techniques for the leasing variants. Those results were published in [19, 20].

In Part V, which consists of Chapters 15–17, we conclude the thesis. We present a brief history of this research in Chapter 15. In Chapter 16, we give a list of results and publications we obtained. Finally, in Chapter 17, we present a list of open questions and other future research directions.

Part II

Parking Permit Problems

Chapter 5

Ski Rental and Parking Permit

In this chapter we review the ski rental problem and the parking permit problem. The parking permit problem is the seminal leasing problem; it corresponds to leasing a single resource (a parking location). The ski rental problem is an older problem which is a particular case of the parking permit problem and is also related to other problems we study in this thesis.

5.1 Ski Rental

In this section, we present the **ski rental problem** [10, 49], a simple problem which is related to the parking permit problem. Although the parking permit problem, which will be discussed in Section 5.2, captures in a more thorough manner the leasing optimization model, we believe that to study the ski rental problem helps to obtain a first glimpse of the structure of these problems. In particular, the ski rental problem has a strong connection with the generalizations of the parking permit problem we present in Chapters 7 and 8.

Imagine the following scenario. Johnny goes to a ski resort. He has the option of renting a pair of skis per day, which costs 1, or buying a pair of skis for cost M to use for the rest of the season. However, Johnny does not know how long the ski season will last, and if he will be able to ski more than $\lfloor M \rfloor$ days. Furthermore, since he lives too far away from the ski resort, it is not feasible to bring the skis home to use them next year. Every day Johnny receives the local weather forecast; if he can ski for one more day, then he has to choose whether he rents or buys skis. Obviously, he wants to spend as little money as possible.

The problem can be formalized in the following manner. Value $r_t = 1$ means that Johnny can ski at day t, and $r_t = 0$ means that the ski season has ended at or before day t. Moreover, $s_t = 0$ means that Johnny rents a pair of skis at day t, and $s_t = 1$ means that Johnny has bought a pair of skis at some day $t' \leq t$.

Problem SkiRental$(M, r_0, \ldots, r_{T-1})$: *The input consists of a constant $M \in \mathbb{Q}_+$ and a sequence $r_0, \ldots, r_{T-1} \in \{0, 1\}$ such that $r_{t-1} \geq r_t$ for $t = 1, \ldots, T-1$. The goal is to find a sequence $s_0, \ldots, s_{T-1} \in \{0, 1\}$ such that $s_t \geq s_{t-1}$ for $t = 1, \ldots, T-1$, which*

minimizes

$$\sum_{t=0}^{T-1} r_t(1-s_t) + M \cdot \max_t r_t s_t.$$

Let $\bar{T} := \max\{t : r_t = 1\}$, i.e., $\bar{T}$ is the number of days the ski season lasts. The offline version of the problem is trivial: simply buy a ski if $\bar{T} \geq M$, and rent a ski for the first $\bar{T}$ days otherwise.

In the online version of the problem, T is unknown, the sequence r is revealed one day at a time, s_t must be calculated without the knowledge of $r_{t+1}, \ldots, r_{T-1}$, and $s_0, \ldots, s_{t-1}$ cannot be modified. The problem admits a simple 2-competitive algorithm: while $r_t = 1$ and $t < M$, rent a pair of skis. If you arrive at day $t \geq M$ with $r_t = 1$, then buy a pair of skis. The optimum solution costs $\min\{\bar{T}, M\}$. The online algorithm pays $\bar{T}$ if $\bar{T} < M$, and pays $M + \lceil M \rceil - 1$ if $\bar{T} \geq M$, so this is a 2-competitive algorithm. We will not prove this fact here, but this is the best possible for a deterministic algorithm. The problem still admits a 2-competitive algorithm even if there are different types of skis, with different durabilities [5, 59]. Also, there is a randomized algorithm with smaller competitive factor [10].

5.2 Parking Permit

Imagine the following scenario: Johnny goes to work everyday and, since he lives close to his job, he may go walking or by car. Since he is aware of global environmental issues, Johnny always goes walking when it does not rain, and he drives otherwise. When he drives, he needs a parking permit. The parking lot has different types of permits, each having a length in days (e.g., daily, weekly, monthly) and a cost. Permits expire even when they are not used; thus, if Johnny buys a weekly permit on Monday, that permit can only be used until Sunday. Johnny wishes to decide which permits to buy, in order to always have a valid permit on a rainy day and to spend as little money as possible. This is the **parking permit problem** (PP), which was proposed by Meyerson [59]. We define the problem formally below. In order to simplify our notation, we write $[K] := \{1, \ldots, K\}$.

Problem PP$(T, K, \delta, \gamma, r)$: *The input consists of a natural number T which represents the number of days, a natural number K which represents the number of permit types, a function $\delta : [K] \mapsto \mathbb{N}$ which assigns a length in days to each permit type, a function $\gamma : [K] \mapsto \mathbb{Q}_+$ which assigns a cost to each permit type, and a sequence $r = (r_0, \ldots, r_{T-1}) \in \{0,1\}^T$, in which $r_t = 1$ indicates that it rains at day t. The goal is to find a set of permits $S \subseteq [K] \times \mathbb{Z}_+$ which covers the rainy days; i.e., there must exist a permit $(k, \hat{t}) \in S$ such that $t \in [\hat{t}, \hat{t} + \delta(k))$ for each day t such that $r_t = 1$. We wish to minimize $\sum_{(k,\hat{t}) \in S} \gamma_k$.*

In the offline setting, the problem can be solved exactly in polynomial time by a dynamic programming algorithm, which we do not present in this thesis. (This is a nice dynamic programming exercise for your students.)

In a more realistic online setting, Johnny does not know the future and weather forecast services are unreliable for long-term predictions. So we assume that T is unknown, and $r_0, \ldots, r_{T-1}$ are revealed one at a time. Johnny begins with an empty set of permits, and

at day t he finds out whether it is raining. If it rains and he does not have a valid permit for day t, he must buy a new permit to cover this day. He does not know $r_{t+1}, \ldots, r_{T-1}$, and he cannot obtain a refund for permits he had bought previously. In this scenario, the problem has a deterministic $O(K)$-competitive algorithm and $\Omega(K)$ lower bound, as well as randomized $O(\lg K)$-competitive algorithm and $\Omega(\lg K)$ lower bound [59].

Note that SkiRental can be reduced to PP. The difference is that, in SkiRental, ski days are consecutive and are concentrated at the beginning of the planning horizon. This subtle difference forces the competitive ratios of the problems to differ by a linear ratio for deterministic algorithms, and by a logarithmic ratio for randomized algorithms.

In Section 5.2.1 we define the Interval Model, which is an important concept in the design of algorithms for parking permit problems. In Section 5.2.2 we present the deterministic algorithm for PP by Meyerson [59].

5.2.1 The Interval Model

In order to give more structure to parking permit problems, it is useful to adopt the following assumptions [4, 59]. Note that the first one is a restriction on the solutions, while the second one is a hypothesis on the instance.

Hypothesis 5.1 (**Simple Interval Model** (SIM)): *Permits of type k can only begin at instants $c \cdot \delta_k$, for $c \in \mathbb{Z}_+$.*

Fact 5.2: *An α-competitive algorithm for* PP *under* SIM *is 2α-competitive for* PP*. Conversely, if there is an α-competitive algorithm for* PP*, then there is a 2α-competitive algorithm for* PP *under* SIM*.*

Proof: Let I be an instance of PP. Let $\mathcal{A}$ be an α-competitive algorithm for PP under SIM, and let $\hat{S}$ be an optimum solution among those that satisfy SIM. We have that $\text{cost}(\mathcal{A}(I)) \leq \alpha \cdot \text{cost}(\hat{S})$. Let S^* be an optimum solution for I when we do not require SIM, and let $\bar{S}$ be a solution that replaces a permit $(k, \hat{t}) \in S^*$ with the permits $(k, \lfloor \hat{t}/\delta_k \rfloor \cdot \delta_k)$ and $(k, \lceil \hat{t}/\delta_k \rceil \cdot \delta_k)$ (these permits may be the same). Note that these permits cover the interval spanned by permit $(k, \hat{t})$, so $\bar{S}$ is a feasible solution for I that satisfies SIM, with $\text{cost}(\bar{S}) \leq 2 \cdot \text{cost}(S^*)$. Also, $\text{cost}(\hat{S}) \leq \text{cost}(\bar{S})$, since $\bar{S}$ satisfies SIM. Thus,

$$\text{cost}(\mathcal{A}(I)) \leq \alpha \cdot \text{cost}(\hat{S}) \leq \alpha \cdot \text{cost}(\bar{S}) \leq 2\alpha \cdot \text{cost}(S^*).$$

Conversely, let $\mathcal{A}'$ be an α-competitive algorithm for PP not requiring SIM. We can obtain a solution S' that satisfies SIM by replacing each permit $(k, \hat{t}) \in \mathcal{A}'(I)$ with the permits $(k, \lfloor \hat{t}/\delta_k \rfloor \cdot \delta_k)$ and $(k, \lceil \hat{t}/\delta_k \rceil \cdot \delta_k)$ (these permits may be the same). Let S^* be an optimum solution for I when we do not require SIM, and let $\hat{S}$ be an optimum solution among those that satisfy SIM. We have that $\text{cost}(S^*) \leq \text{cost}(\hat{S})$, since a solution that satisfies SIM is also a solution for PP not requiring SIM. Thus,

$$\text{cost}(S') \leq 2 \cdot \text{cost}(\mathcal{A}'(I)) \leq 2\alpha \cdot \text{cost}(S^*) \leq 2\alpha \cdot \text{cost}(\hat{S}).$$

□

Hypothesis 5.3 (**Hierarchical Length Property** (HLP)): *For $k = 2, \ldots, K$, it holds that δ_k divides δ_{k-1}.*

Fact 5.4: *If there is an α-competitive algorithm for* PP *under* HLP*, then there is a 2α-competitive algorithm for instances that do not satisfy* HLP*.*

Proof: Take an arbitrary instance $I = (T, K, \delta, \gamma, r)$ not necessarily satisfying HLP, and assume that $\delta_1 \leq \cdots \leq \delta_K$. For $k = 2, \ldots, K$, take $\delta'_k := 2^{\lfloor \lg \delta_k \rfloor}$, i.e., round down each permit length to the closest power of 2. Note that the resulting instance $I' = (T, K, \delta', \gamma, r)$ satisfies HLP. Let $\mathcal{A}$ be an α-competitive algorithm for PP under HLP; we have that $\text{cost}(\mathcal{A}(I')) \leq \alpha \cdot \text{opt}(I')$. Note that $\mathcal{A}(I')$ is a feasible solution for I: if $\mathcal{A}(I')$ uses permit $(k, \hat{t})$, since $\delta'_k \leq \delta_k$, this permit has length δ_k in I, and hence it covers the same period it covers in I' (and maybe more). So we just have to bound $\text{opt}(I')$ with regard to $\text{opt}(I)$. Let S^* be an optimum solution for I. Construct a set $\bar{S}$ of permits of I' as follows: for each permit $(k, \hat{t}) \in S^*$, $\bar{S}$ contains permits $(k, \hat{t})$ and $(k, \hat{t} + \delta'_k)$. Since $\delta'_k = 2^{\lfloor \lg \delta_k \rfloor}$, we have that $\delta_k < 2\delta'_k$, so in I' these two permits cover the interval covered by $(k, \hat{t})$ in I (and maybe more). Thus, $\bar{S}$ is feasible for I', so $\text{opt}(I') \leq \text{cost}(\bar{S})$, and $\text{cost}(\bar{S}) = 2 \cdot \text{cost}(S^*) = 2 \cdot \text{opt}(I)$. Therefore,

$$\text{cost}(\mathcal{A}(I')) \leq \alpha \cdot \text{opt}(I') \leq \alpha \cdot \text{cost}(\bar{S}) = 2\alpha \cdot \text{cost}(S^*) = 2\alpha \cdot \text{opt}(I).$$

(**Remark:** We can easily modify the proof to only use permit lengths that are powers of any integer $\beta \geq 2$, obtaining competitive factor $\alpha\beta$.) □

For most results we present here, we adopt the following assumption.

Hypothesis 5.5 (**Interval Model** (IM)): *We assume* SIM *and* HLP*.*

The following lemma follows from the composition of Facts 5.2 and 5.4.

Lemma 5.6: *If there is an α-competitive algorithm for* PP *under* IM*, then there is a 4α-competitive algorithm for arbitrary instances.*

The main property of the Interval Model is that there is an optimum solution which is the union of optimum solutions for the sub-instances defined by the intervals of length δ_K. This can be formalized as shown in Lemma 5.9.

Consider an instance $I = (T, K, \delta, \gamma, r)$ of PP. For some forthcoming proofs and algorithms, it is convenient to define restricted instances of I in the following manner.

Definition 5.7 ($I[k]$): *Given a permit type $k \in [K]$, let $I[k] := (T, k, \delta, \gamma, r)$ be the sub-instance of I in which we can only use permits of type $1, \ldots, k$.*

Definition 5.8 ($I[k, \hat{t}]$): *Given a permit type $k \in [K]$ and an instant of time $\hat{t} = c \cdot \delta_k$ with $c \in \mathbb{Z}_+$, let $I[k, \hat{t}] := (\delta_k, k, \delta, \gamma, (r_{\hat{t}}, \ldots, r_{\hat{t}+\delta_k-1}))$ be the sub-instance of $I[k]$ restricted to the interval of length δ_k beginning at instant $\hat{t}$.*

Lemma 5.9: *Let I be an instance of* PP *and let $k \in [K]$ be a permit type. Then, under* IM*, we have that*

$$\text{opt}(I[k]) = \sum_{c=0}^{\lceil T/\delta_k \rceil - 1} \text{opt}(I[k, c \cdot \delta_k]).$$

Note that $\lceil T/\delta_k \rceil$ is the number of intervals of length δ_k. So instance $I[k]$ can be seen as a concatenation of $\lceil T/\delta_k \rceil$ instances, corresponding to each of those intervals. Hence, the lemma says that solving $I[k]$ is equivalent to solving independently each sub-instance of length δ_k and taking the union of the corresponding solutions.

Proof: The proof is by induction in $\lceil T/\delta_k \rceil$. If $\lceil T/\delta_k \rceil = 1$, then the lemma holds trivially. So assume the lemma holds for instances in which the number of intervals is smaller than $\lceil T/\delta_k \rceil$. Let S^* be an optimum solution for $I[k]$. Let S' be the set of permits in S^* that cover days in the last interval of length δ_k; i.e., a permit $(k', t') \in S^*$ is also in S' if there is some $t \geq (\lceil T/\delta_k \rceil - 1) \cdot \delta_k$ such that $t \in [t', t' + \delta_{k'})$. Clearly, S' is a feasible solution for the last interval of length δ_k and, due to IM, $S^* \setminus S'$ is a feasible solution for the first $\lceil T/\delta_k \rceil - 1$ intervals. Note that S' and $S^* \setminus S'$ are optimum solutions for those sub-instances. By induction hypothesis, the cost of $S^* \setminus S'$ is the cost solving independently each sub-instance of length δ_k, so the lemma holds. □

Under IM, we can assume without loss of generality that

$$1 = \delta_1 < \delta_2 < \cdots < \delta_K \text{ and } \gamma_k/\delta_k < \gamma_{k'}/\delta_{k'} \text{ for } k > k'; \tag{5.1}$$

i.e., permit costs are sub-additive and represent economies of scale: longer permits cost less per day. If $\delta_1 > 1$, we can simply divide each δ_k by δ_1 and rescale the instance by having a rainy day for each interval of length δ_1 which contains some rainy day. If two permit types k and k' with $k > k'$ do not satisfy Equation (5.1), we can discard type k by replacing each permit of type k with $\delta_k/\delta_{k'}$ permits of type k', and the optimum solution of the new instance is not worse than the original one.

5.2.2 A Deterministic Online Algorithm

In this section we present the online deterministic algorithm for PP by Meyerson [59]. This algorithm is important for understanding the relevance of results we present in other chapters, and the concepts behind them. The algorithm assumes IM.

A pseudocode is presented in Algorithm 5.1. The offline part of the input are parameters K, δ and γ. The algorithm uses as a subroutine an exact algorithm AlgPP for the offline version of the problem under IM. If $r_t = 1$, the algorithm computes an optimum offline solution for request sequence $r_0, \ldots, r_t$, and then uses the same permit as the offline solution to cover day t. Note that Line 5 allows the algorithm to buy a permit whose starting time is earlier than t, and thus it may cover previous days with more than one permit. Note that, in a certain way, the strategy of this algorithm is the same as that of the online algorithm for SkiRental we described in Section 5.1. If at some point

the algorithm realizes that the optimum solution is to buy a longer permit (correspondly, to buy skis), we buy that permit, no matter what we have bought before. However, although this strategy is 2-competitive for SkiRental, even for K types of skis, it is just K-competitive for PP, as we show in Theorem 5.10. It is not possible (deterministically) to do better than this, because any deterministic online algorithm for PP has competitive factor $\Omega(K)$ [59].

Input: (K, δ, γ)
$S \leftarrow \emptyset$;
when r_t arrives **do**
  if $r_t = 1$ **then**
    $S^* \leftarrow \text{AlgPP}(t+1, K, \delta, \gamma, (r_0, \ldots, r_t))$;
    $S \leftarrow S \cup \{(k, \hat{t}) \in S^* : t \in [\hat{t}, \hat{t} + \delta_k)\}$;
return S;

Algorithm 5.1: Meyerson's deterministic online algorithm for PP [59].

Theorem 5.10 (Meyerson [59]): *Algorithm 5.1 is K-competitive under* IM.

Proof: Let $I = (T, K, \delta, \gamma, r)$ be an instance of the problem, and denote by $\text{AlgOPP}(I)$ the solution returned by the algorithm on I. Let $k \in [K]$, and consider instance $I[k]$ as in Definition 5.7. By Lemma 5.9,

$$\text{opt}(I[k]) = \sum_{c=0}^{\lceil T/\delta_k \rceil - 1} \text{opt}(I[k, c \cdot \delta_k]).$$

Since the offline algorithm satisfies IM, the solution returned by the online algorithm is also independent for each interval of length δ_k, so

$$\text{cost}(\text{AlgOPP}(I[k])) = \sum_{c=0}^{\lceil T/\delta_k \rceil - 1} \text{cost}(\text{AlgOPP}(I[k, c \cdot \delta_k])).$$

We prove that, for any $k \in [K]$ and $\hat{t} = c \cdot \delta_k$ for some $c \in \mathbb{Z}_+$, it holds that

$$\text{cost}(\text{AlgOPP}(I[k, \hat{t}])) \leq k \cdot \text{opt}(I[k, \hat{t}]). \tag{5.2}$$

For $k = K$, this proves the lemma. The proof is by induction on k. The base case is trivial, since $\delta_1 = 1$: both the online algorithm and the optimum offline solution buy a permit for each rainy day. So assume Inequality (5.2) holds for $k - 1$. Note that $\text{opt}(I[k, \hat{t}]) \leq \gamma_k$, since permit $(k, \hat{t})$ is a feasible solution of $I[k, \hat{t}]$. We divide the proof in two cases.

1. If $\text{opt}(I[k, \hat{t}]) < \gamma_k$, then the optimum offline solution does not buy $(k, \hat{t})$, and only buys permits of types $1, \ldots, k-1$. Let $d = \delta_k / \delta_{k-1}$ (remember that d is an integer); then
$$\text{opt}(I[k, c \cdot \delta_k]) = \sum_{a=c \cdot d}^{(c+1) \cdot d - 1} \text{opt}(I[k-1, a \cdot \delta_{k-1}]).$$

Since the online algorithm mimics the behavior of the optimum solution, it also never buys $(k, \hat{t})$. Thus,

$$\text{cost}(\text{AlgOPP}(I[k, c \cdot \delta_k])) = \sum_{a=c \cdot d}^{(c+1) \cdot d - 1} \text{cost}(\text{AlgOPP}(I[k-1, a \cdot \delta_{k-1}])).$$

By induction hypothesis, for $a = c \cdot d, \ldots, (c+1) \cdot d - 1$,

$$\text{cost}(\text{AlgOPP}(I[k-1, a \cdot \delta_{k-1}])) \leq (k-1) \cdot \text{opt}(I[k-1, a \cdot \delta_{k-1}]).$$

Thus,

$$\begin{aligned}
\text{cost}(\text{AlgOPP}(I[k, c \cdot \delta_k])) &= \sum_{a=c \cdot d}^{(c+1) \cdot d - 1} \text{cost}(\text{AlgOPP}(I[k-1, a \cdot \delta_{k-1}])) \\
&\leq \sum_{a=c \cdot d}^{(c+1) \cdot d - 1} (k-1) \cdot \text{opt}(I[k-1, a \cdot \delta_{k-1}]) \\
&= (k-1) \cdot \sum_{a=c \cdot d}^{(c+1) \cdot d - 1} \text{opt}(I[k-1, a \cdot \delta_{k-1}]) \\
&= (k-1) \cdot \text{opt}(I[k, c \cdot \delta_k]).
\end{aligned}$$

2. If $\text{opt}(I[k, \hat{t}]) = \gamma_k$, then the optimum offline solution buys only permit $(k, \hat{t})$. For any $t \in [\hat{t}, \hat{t} + \delta_k)$, let $I_t[k, \hat{t}] := (t - \hat{t} + 1, k, \delta, \gamma, (r_{\hat{t}}, \ldots, r_t))$ be the sub-instance of $I[k, \hat{t}]$ that considers only days $\hat{t}, \ldots, t$. Then let t be such that $\text{opt}(I_{t-1}[k, \hat{t}]) < \gamma_k$ but $\text{opt}(I_t[k, \hat{t}]) = \gamma_k$; i.e., t is day in which the online algorithm decides to buy permit $(k, \hat{t})$. Note that the online algorithm will not need to buy any permit after t until the end of this interval, so

$$\text{cost}(\text{AlgOPP}(I[k, \hat{t}])) = \text{cost}(\text{AlgOPP}(I_{t-1}[k, \hat{t}])) + \gamma_k.$$

Due to the analysis of the previous case,

$$\text{cost}(\text{AlgOPP}(I_{t-1}[k, \hat{t}])) \leq (k-1) \cdot \text{opt}(I_{t-1}[k, \hat{t}]) < (k-1) \cdot \gamma_k,$$

so

$$\text{cost}(\text{AlgOPP}(I[k, \hat{t}])) \leq (k-1) \cdot \gamma_k + \gamma_k = k \cdot \gamma_k = k \cdot \text{opt}(I[k, \hat{t}]).$$

□

Meyerson [59] also presented a randomized $O(\lg K)$-competitive online algorithm for PP, which we denote by AlgRandOPP. He also proved that any randomized online algorithm for PP has competitive factor $\Omega(\lg K)$. We do not present those results here; instead, we only use them as black boxes when necessary.

Chapter 6

Multi Parking Permit

In this chapter, we propose the following generalization of PP. Imagine that Johnny and his coworkers decide to share parking permits to save some money. Johnny's means of transportation depend on weather, but Linda goes by car when she has tennis lessons, and so does Ringo when he has to get his kids at school. So, for different days, different employees need a parking permit, and permits can be exchanged so that a same permit can be used by different employees on different days.

This is the **multi parking permit problem** (MPP). For a formal definition, as in PP, the input of MPP contains K types of permits with lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$ and costs $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$. But now a demand greater than one can be given for each instant, i.e., we receive a sequence $r_0, \ldots, r_{T-1} \in \mathbb{Z}_+$, which means that r_t employees need a permit at day t, for $t = 0, \ldots, T-1$. Moreover, multiple copies of the same permit can be bought, so a solution is a multiset of permits. Given a multiset B and an element $b \in B$, we denote by $m_B(b)$ the multiplicity of b in B. We wish to find a multiset of permits $S \subseteq [K] \times \mathbb{Z}_+$ such that

$$\sum_{\substack{(k,\hat{t})\in S \\ t\in[\hat{t},\hat{t}+\delta_k)}} m_S(k,\hat{t}) \geq r_t \quad \text{for each } t,$$

which minimizes

$$\sum_{(k,\hat{t})\in S} m_S(k,\hat{t}) \cdot \gamma_k.$$

The problem has the following formulation as an integer linear program.

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{K}\sum_{\hat{t}=0}^{T-1} x_{k\hat{t}} \cdot \gamma_k \\
\text{subject to} \quad & \sum_{k=1}^{K} \sum_{\substack{\hat{t}=0,\ldots,T-1 \\ t\in[\hat{t},\hat{t}+\delta_k)}} x_{k\hat{t}} \;\geq\; r_t \quad \forall t \in \{0,\ldots,T-1\}, \\
& x_{k\hat{t}} \;\in\; \mathbb{Z}_+ \quad \forall k \in [K], \hat{t} \in \{0,\ldots,T-1\}.
\end{aligned}$$

In this formulation, variable $x_{k\hat{t}}$ indicates the multiplicity of permit $(k,\hat{t})$. (To ensure polynomial size, we may only allow permits that begin at instants t with $r_t > 0$.) The

first constraint ensures that each day is covered by enough permits.

Let $P := \{\mathbf{x} \in \mathbb{R}^{K \times T} | \mathbf{A} \cdot \mathbf{x} \geq \mathbf{r}, \mathbf{x} \geq \mathbf{0}\}$ be the polytope of the relaxation of this linear program. Note that, for a fixed column in matrix $\mathbf{A}$, ones are consecutive, because a permit covers a consecutive sequence of days. Such a matrix is totally unimodular [71] and, since $\mathbf{r}$ is integer, if follows that all extreme points of P are integral [66]. Thus, MPP can be solved in polynomial time.

Now we present another result which helps us to solve MPP in the online setting, where T is unknown and $r_0, \ldots, r_{T-1}$ are revealed one at a time. Note that Lemmas 5.6 and 5.9 are also true for MPP, and that under IM we can assume that permits satisfy Equation (5.1) without loss of generality.

We define an ordering of the permits in a solution and a unique assignment between demands and permits, from which our next result follows straightforwardly. Given a multiset of permits, we sort them in non-increasing order of permit type; permits of same type are sorted in non-decreasing order of starting time, breaking ties arbitrarily. We call this the **Hanoi tower ordering** (HTO), since if we represent permits as rectangles as in Figure 6.1(a), larger permits are under smaller permits, as in the Hanoi tower problem. Then, we assign demands of the input to permits in a solution so that each demand is **covered** by the earliest possible permit in HTO. Note that this defines a level to each demand and each permit; see Figure 6.1(b). Due to IM, a demand from instant t is assigned to a smaller permit only if all larger permits that cover instant t have some other demand assigned to them.

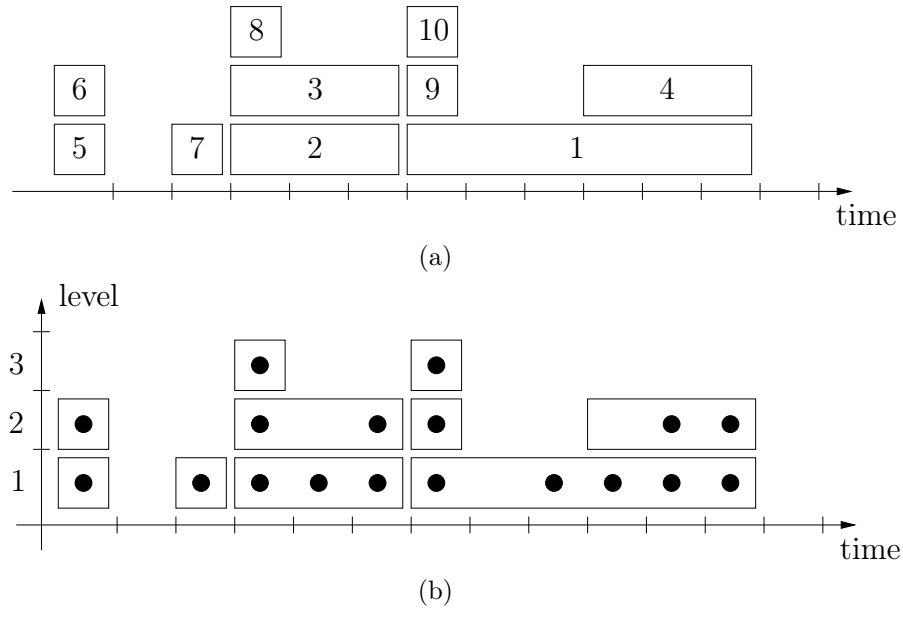


Figure 6.1: (a) Example of a MPP solution ordered in HTO, given an instance with demand sequence $r = (2, 0, 1, 3, 1, 2, 3, 0, 1, 1, 2, 2)$. (b) Demands of this instance assigned to those permits.

Thus, under IM and HTO, MPP has the following property: each level of an optimum solution is an optimum solution of the corresponding PP instance. Let $I = (T, K, \delta, \gamma, r)$ be an instance of MPP. Let $R := \max_{t=0,\ldots,T-1} r_t$ be the maximum demand and, for $j = 1, \ldots, R$, let $I^j := (T, K, \delta, \gamma, r^j)$ be an instance of PP such that, for $t = 0, \ldots, T-1$,

$r_t^j = 1$ if $r_t \geq j$, and $r_t^j = 0$ otherwise. (I^j is the PP instance corresponding to level j.)

Lemma 6.1: *Assume* IM. *Given an* MPP *instance* I*, there exists an optimum solution of* I *which is the union of optimum solutions of* PP *instances* $I^1, I^2, \ldots, I^R$.

Proof: Let S^* be an optimum solution of I and, for each $j \in [R]$, let S^{j*} be an optimum PP solution of I^j. Note that $\bigcup_{j=1}^{R} S^{j*}$ is a feasible solution for I. Suppose by contradiction that $\text{cost}(S^*) < \sum_{j=1}^{R} \text{cost}(S^{j*})$. Sort permits in S^* and assign demands to permits as in HTO. Demands of instant t must be assigned to levels $1, \ldots, r_t$ since, due to IM, if permits $(k, \hat{t})$ and (k', t') are such that $k < k'$ and $\hat{t} \in [t', t' + \delta_{k'})$, then $[\hat{t}, \hat{t} + \delta_k) \subseteq [t', t' + \delta_{k'})$. (See Figure 6.2.) Thus, S^* can be partitioned into feasible solutions of $I^1, \ldots, I^R$, a contradiction to the fact that $S^{1*}, \ldots, S^{R*}$ are optimum. □

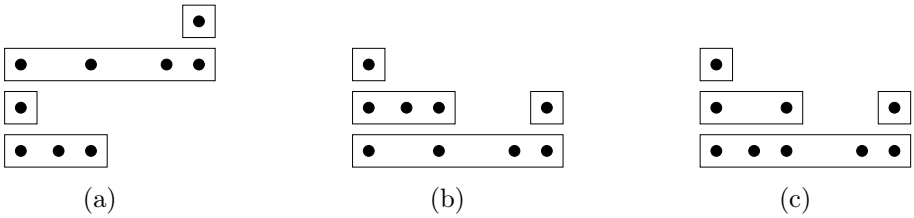


Figure 6.2: An illustration of the proof of Lemma 6.1. Consider an instance with $T = 6$, $K = 3$, $\delta = (1, 3, 6)$, $\gamma = (1, 5/2, 4)$ and $r = (3, 1, 2, 0, 1, 2)$. (a) The optimum solution is $\{(2, 0), (1, 0), (3, 0), (1, 5)\}$. (b) Optimum solution after reordering permits. (c) Optimum solution after reassigning demands; permits correspond to optimum solutions for PP instances.

So under IM, from Lemma 6.1, MPP reduces to solving R instances of PP. Thus, given an α-competitive online algorithm for PP, we obtain an α-competitive online algorithm for MPP.[1] By Lemma 5.6, we have deterministic $O(K)$-competitive and randomized $O(\lg K)$-competitive online algorithms for MPP. Note that Lemma 6.1 is valid only if we assume IM. (See a counterexample if we do not assume IM in Figure 6.3.) However, this reduction is pseudo-polynomial, since it runs in time $\Omega(R)$ and the input size is proportional to $O(\lg R)$. We show how to overcome this in the following lemma, which is inspired on the online algorithm for SN [72].

Figure 6.3: Lemma 6.1 is no longer valid if we do not assume IM. Consider an instance I with $T = 5$, $K = 3$, $\delta = (1, 2, 4)$, $\gamma = (2, 3, 5)$ and $r = (1, 2, 1, 1, 1)$. (a) The optimum solution is $\{(2, 0), (3, 1)\}$, which costs 8. (b) The union of optimum solutions for PP instances I^1 and I^2 is $\{(1, 0), (3, 1), (1, 1)\}$, which costs 9.

[1]Since an online algorithm does not know the value of R in advance, we must run new instances of the online algorithm for PP as the maximum demand increases.

Lemma 6.2: *Assume* IM. *Given an α-competitive algorithm for* PP, *there exists a strictly polynomial-time 2α-competitive algorithm for* MPP.

Proof: Let $L := \lfloor \lg R \rfloor$; we define $L+1$ instances $\hat{I}^0, \hat{I}^1, \ldots, \hat{I}^L$ of PP. For each instant t and $\ell = 0, \ldots, L$, the demand of day t in $\hat{I}^\ell$ is 1 if $\ell \leq \lfloor \lg r_t \rfloor$, and 0 otherwise. Run the α-competitive algorithm for PP on each of $\hat{I}^0, \ldots, \hat{I}^L$, and buy 2^ℓ copies of the permits bought by PP on $\hat{I}^\ell$. This is feasible since $\sum_{\ell=0}^{\lfloor \lg r_t \rfloor} 2^\ell = 2^{\lfloor \lg r_t \rfloor + 1} - 1$ and $r_t < 2^{\lfloor \lg r_t \rfloor + 1}$.

Consider instances $I^1, \ldots, I^R$ as defined before Lemma 6.1. By Lemma 6.1, we have that $\mathrm{opt}(I) = \sum_{j=1}^{R} \mathrm{opt}(I^j)$. For $1 \leq j < R$, we have that $r_t^{j+1} \leq r_t^j$ for every t; thus, an optimum solution for I^j is feasible for I^{j+1}, and hence $\mathrm{opt}(I^{j+1}) \leq \mathrm{opt}(I^j)$, for $1 \leq j < R$.

Note that, for $\ell = 0, \ldots, L$, we have that $\hat{I}^\ell = I^{2^\ell}$. Let $\hat{S}^\ell$ be the solution obtained by the α-competitive algorithm for PP on instance $\hat{I}^\ell$; we have that $\mathrm{cost}(\hat{S}^\ell) \leq \alpha \cdot \mathrm{opt}(\hat{I}^\ell)$. Let S be the returned MPP solution. We have that

$$\begin{aligned}
\mathrm{cost}(S) &= \sum_{\ell=0}^{L} 2^\ell \cdot \mathrm{cost}(\hat{S}^\ell) \leq \sum_{\ell=0}^{L} 2^\ell \cdot \alpha \cdot \mathrm{opt}(\hat{I}^\ell) = \alpha \cdot \left(\mathrm{opt}(I^1) + 2 \cdot \sum_{\ell=1}^{L} 2^{\ell-1} \mathrm{opt}(I^{2^\ell}) \right) \\
&\leq \alpha \cdot \left(\mathrm{opt}(I^1) + 2 \cdot \sum_{\ell=1}^{L} \sum_{j=2^{\ell-1}+1}^{2^\ell} \mathrm{opt}(I^j) \right) = \alpha \cdot \left(\mathrm{opt}(I^1) + 2 \cdot \sum_{j=2}^{2^L} \mathrm{opt}(I^j) \right) \\
&\leq 2\alpha \cdot \sum_{j=1}^{R} \mathrm{opt}(I^j) = 2\alpha \cdot \mathrm{opt}(I),
\end{aligned}$$

where the second inequality follows because interval $[2^{\ell-1}+1, 2^\ell]$ contains $2^{\ell-1}$ integers, and $\mathrm{opt}(I^{j+1}) \leq \mathrm{opt}(I^j)$ for $1 \leq j < R$. Thus, the lemma follows. □

Due to the online algorithms for PP by Meyerson [59], we have the following result.

Theorem 6.3: *There are polynomial-time deterministic* $\mathrm{O}(K)$*-competitive and randomized* $\mathrm{O}(\lg K)$*-competitive online algorithms for* MPP.

Chapter 7

Group Parking Permit

We propose the following generalization of MPP, which we call the **group parking permit problem** (GPP). Every day a tourism agency receives a group of guests willing to visit a museum. The agency has an agreement with the museum, for which the agency can buy permits (or tickets) that last for different periods (e.g., a day, a week, a month). Moreover, the agency can buy a special permit of type $k \in [K]$ beginning at day $\hat{t}$ that costs $M \cdot \gamma_k$, which can be used by an unlimited number of guests on the period $[\hat{t}, \hat{t} + \delta_k)$. We call this a **group permit**, and a usual permit for a single guest a **single permit**. The agency wishes to buy a minimum-cost multiset of single and group permits. More formally, we have K types of permits with lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$ and costs $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, and we are given a sequence $r_0, \ldots, r_{T-1} \in \mathbb{Z}_+$ and a constant $M \geq 1$. We wish to find a multiset $S \subseteq [K] \times \mathbb{Z}_+$ of single permits, and a set $Q \subseteq [K] \times \mathbb{Z}_+$ of group permits, such that S and Q meet the demand. I.e., for $t = 0, \ldots, T-1$, either

$$\sum_{\substack{(k,\hat{t}) \in S \\ t \in [\hat{t}, \hat{t} + \delta_k)}} m_S(k, \hat{t}) \geq r_t,$$

or there is some group permit $(k, \hat{t}) \in Q$ such that $t \in [\hat{t}, \hat{t} + \delta_k)$. We wish to minimize

$$\sum_{(k,\hat{t}) \in S} m_S(k, \hat{t}) \cdot \gamma_k + M \cdot \sum_{(k,\hat{t}) \in Q} \gamma_k.$$

In order to simplify our notation, we denote the total cost of a multiset of permits S by $\text{cost}(S) := \sum_{(k,\hat{t}) \in S} m_S(k, \hat{t}) \cdot \gamma_k$. Thus, the cost of a solution (S, Q) is $\text{cost}(S) + M \cdot \text{cost}(Q)$.

This problem reduces to PP if $r_t \leq 1$ for every day t, to MPP if $M = \infty$, and to SkiRental if $K = 1$ and $\delta_1 = \infty$.

Note that Lemmas 5.6 and 5.9, regarding IM, are also true for GPP, and that under IM we can assume that permits satisfy Equation (5.1) without loss of generality.

7.1 Offline Group Parking Permit

Consider the following formulation of GPP as an integer linear program.

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{K} \sum_{\hat{t}=0}^{T-1} (x_{k\hat{t}} + M \cdot y_{k\hat{t}}) \cdot \gamma_k \\
\text{subject to} \quad & \sum_{k=1}^{K} \sum_{\substack{\hat{t}=0,\ldots,T-1 \\ t \in [\hat{t}, \hat{t}+\delta_k)}} (x_{k\hat{t}} + r_t \cdot y_{k\hat{t}}) \;\geq\; r_t \quad \forall t \in \{0, \ldots, T-1\}, \\
& x_{k\hat{t}} \in \mathbb{Z}_+, y_{k\hat{t}} \in \{0,1\} \qquad \forall k \in [K], \hat{t} \in \{0, \ldots, T-1\}.
\end{aligned}$$

Variable $x_{k\hat{t}}$ indicates how many copies of single permit $(k, \hat{t})$ we must buy, and $y_{k\hat{t}}$ indicates whether we must buy a group permit $(k, \hat{t})$. The first constraint ensures that each day is covered by a group permit or by enough single permits. The matrix of this linear program is no longer totally unimodular, since it is not a 0/1 matrix. It is interesting to remark that, after running some experiments, we found some random instances for which the integrality gap is greater than 1, even under IM.[1] We believe that GPP is weakly NP-hard even under IM, but this is an open question. (Otherwise, it is one of those few interesting problems which are polynomially solvable but have integrality gap.) Under IM, there exists a pseudo-polynomial exact algorithm for GPP, which we discuss in Section 8.1.

In this section, we present a polynomial-time 2-approximation algorithm for GPP under IM. Before we present the algorithm, let us define some notation.

Definition 7.1 ($S[k, \hat{t}]$): *Given a multiset of permits $S \subseteq [K] \times \mathbb{Z}_+$, a permit type $k \in [K]$ and an instant of time $\hat{t} = c \cdot \delta_k$ with $c \in \mathbb{Z}_+$, let*

$$S[k, \hat{t}] := \{(k', t') \in S : k' \leq k \text{ and } t' \in [\hat{t}, \hat{t} + \delta_k)\}$$

be the submultiset of permits of S of types $1, \ldots, k$ that are contained in the interval of length δ_k beginning at instant $\hat{t}$.

Note that this definition applies both to multisets of single permits and sets of group permits.

The pseudocode of our algorithm is presented in Algorithm 7.1. Roughly speaking, first we run a polynomial-time algorithm for MPP under IM, which we denote by AlgMPP, on the corresponding instance (ignoring M); this is the initial solution. Then, for $k = 1, \ldots, K$, we consider an interval of type k, and we check if we improve the current solution by replacing permits of types $1, \ldots, k$ chosen so far for this interval with a group permit of type k.

The intuition behind this algorithm is the following: we collect contributions of permits in the solution obtained by AlgMPP to decide when to buy a group permit in GPP. Single

[1] One such simple instance has $T = 8$, $K = 4$, $M = 10$, $\delta = (1, 2, 4, 8)$, $\gamma = (20, 39, 77, 152)$, and $r = (1, 1, 1, 1, 1, 12, 1, 1)$. While the optimum solution under IM costs 336, the optimum fractional solution under IM costs $335 + 1/3$.

Input: $(T, K, \delta, \gamma, r, M)$
1 $S_0 \leftarrow \text{AlgMPP}(T, K, \delta, \gamma, r)$, $Q_0 \leftarrow \emptyset$;
2 **for** $k \leftarrow 1$ **to** K **do**
3 $\quad S_k \leftarrow S_{k-1}$, $Q_k \leftarrow Q_{k-1}$;
4 $\quad$ **for** $\hat{t} \leftarrow 0$ **to** $T - 1$ **step** δ_k **do**
5 $\quad\quad$ **if** $\text{cost}(S_{k-1}[k, \hat{t}]) + M \cdot \text{cost}(Q_{k-1}[k, \hat{t}]) \geq M \cdot \gamma_k$ **then**
6 $\quad\quad\quad S_k \leftarrow S_k \setminus S_{k-1}[k, \hat{t}]$;
7 $\quad\quad\quad Q_k \leftarrow (Q_k \setminus Q_{k-1}[k, \hat{t}]) \cup \{(k, \hat{t})\}$;
8 **return** (S_K, Q_K);

Algorithm 7.1: Approximation algorithm for GPP.

permits of types $1, \ldots, k$ contribute to group permits of type k. However, we limit the contributions of types $1, \ldots, k$ in a single interval of type k to $M \cdot \gamma_k$. This prevents us from buying a group permit of a larger type when most contributions are clustered in an interval of smaller type. Note that the algorithm always returns a feasible solution, since every time we replace a subsolution with a group permit, it covers all demands in that interval. We illustrate the execution of the algorithm in Figure 7.1.

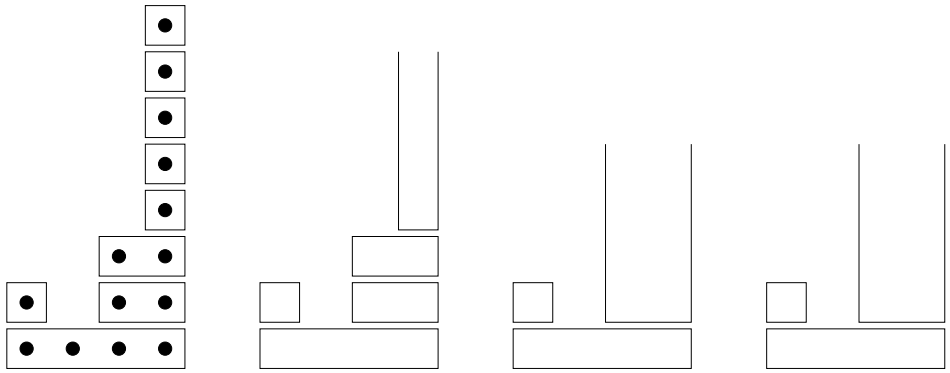


Figure 7.1: An execution of Algorithm 7.1 on an instance with $T = 4$, $K = 3$, $M = 4$, $\delta = (1, 2, 4)$, $\gamma = (4, 6, 11)$, and $r = (2, 1, 3, 8)$. (a) S_0 is the optimum solution for the corresponding MPP instance. (b) In S_1 we replace type-1 permits with type-1 group permits because this improves the solution. (c) In S_2 we replace permits of types 1 and 2 in S_1 with group permits of type 2 because this improves the solution. (d) In this case $S_3 = S_2$ since $\text{cost}(S_2) = 39$ and $M \cdot \gamma_3 = 44$. Note that $\text{cost}(S_0) = 47$, which costs more than a type-3 group permit, but we can obtain a better solution using intermediary group permits.

In order to analyze the algorithm, we define a modified version (Algorithm 7.2) in which we also receive a set of group permits $Q \subseteq [K] \times \mathbb{Z}_+$, and we only execute Lines 5-7 of Algorithm 7.1 if the considered interval is contained in the interval defined by some group permit in Q. We present a pseudocode in Algorithm 7.2. Note that Algorithm 7.1 is equivalent to running Algorithm 7.2 with $Q = \{(K, t) : t = c \cdot \delta_K, c \in \mathbb{Z}_+\}$.

Input: $(T, K, \delta, \gamma, r, M, Q)$
1 $S'_0 \leftarrow \text{AlgMPP}(T, K, \delta, \gamma, r)$, $Q_0 \leftarrow \emptyset$;
2 **for** $k \leftarrow 1$ **to** K **do**
3 $\quad S'_k \leftarrow S'_{k-1}$, $Q'_k \leftarrow Q'_{k-1}$;
4 $\quad$ **for** $\hat{t} \leftarrow 0$ **to** $T-1$ **step** δ_k **do**
5 $\quad\quad$ **if** $k \le k'$ **and** $\hat{t} \in [t', t'+\delta_{k'})$ **for some** $(k', t') \in Q$ **then**
6 $\quad\quad\quad$ **if** $\text{cost}(S'_{k-1}[k, \hat{t}]) + M \cdot \text{cost}(Q'_{k-1}[k, \hat{t}]) \ge M \cdot \gamma_k$ **then**
7 $\quad\quad\quad\quad S'_k \leftarrow S'_k \setminus S'_{k-1}[k, \hat{t}]$;
8 $\quad\quad\quad\quad Q'_k \leftarrow (Q'_k \setminus Q'_{k-1}[k, \hat{t}]) \cup \{(k, \hat{t})\}$;
9 **return** (S'_K, Q'_K);

Algorithm 7.2: A modified version of Algorithm 7.1, which will be useful for our analysis.

Lemma 7.2: *Given a* GPP *instance I, for any $Q \subseteq [K] \times \mathbb{Z}_+$, the cost of the solution returned by running Algorithm 7.1 on I is at most the cost of the solution returned by running Algorithm 7.2 on (I, Q).*

Proof: Due to IM, if Algorithm 7.1 executes Lines 5-7 for some k and $\hat{t}$, then it has executed Lines 5-7 before for each sub-interval. (The same holds for Lines 6-8 of Algorithm 7.2.) Thus it is easy to prove, by induction on k, that for any pair $(k, \hat{t})$ for which both Algorithm 7.1 and Algorithm 7.2 run this step, we have that

$$\text{cost}(S_k[k, \hat{t}]) = \text{cost}(S'_k[k, \hat{t}]) \text{ and } \text{cost}(Q_k[k, \hat{t}]) = \text{cost}(Q'_k[k, \hat{t}]).$$

Note that an execution of Lines 5-7 in Algorithm 7.1 never increases the cost of the returned solution. Since Algorithm 7.1 considers each interval that Algorithm 7.2 does, the lemma holds. □

Our goal is to prove that Algorithm 7.1 is a 2-approximation for GPP under IM, which implies that there exists an 8-approximation for arbitrary instances. Let $I = (T, K, \delta, \gamma, r, M)$ be a GPP instance, and let (S, Q) be the solution returned by Algorithm 7.1 on I. Let (S^*, Q^*) be an optimum solution of I which always utilizes a group permit whenever possible, and let (S', Q') be the solution returned by Algorithm 7.2 on (I, Q^*). We claim that

$$\text{cost}(S) + M \cdot \text{cost}(Q) \le \text{cost}(S') + M \cdot \text{cost}(Q') \le \text{cost}(S^*) + 2M \cdot \text{cost}(Q^*) \le 2 \cdot \text{opt}(I).$$

The first inequality follows from Lemma 7.2, so it suffices to prove the second inequality.

We partition S' in two multisets, $S'_{\le}$ and $S'_{>}$. Let $S'_{\le} := \bigcup_{(k,\hat{t}) \in Q^*} S'[k, \hat{t}]$, i.e., $S'_{\le}$ is the multiset of single permits that are contained in the interval defined by some group permit in Q^*. Let $S'_{>} := S' \setminus S'_{\le}$. (See Figure 7.2.) Since Algorithm 7.2 executes Lines 6-8 for each interval defined by a permit $(k, \hat{t}) \in Q^*$, it considers buying group permit $(k, \hat{t})$. Moreover, every group permit in Q' is contained in the interval defined by some group permit in Q^*. Thus,

$$\text{cost}(S'_{\le}) + M \cdot \text{cost}(Q') \le M \cdot \text{cost}(Q^*).$$

It remains to prove that $\text{cost}(S'_>) \leq \text{opt}(I) = \text{cost}(S^*) + M \cdot \text{cost}(Q^*)$. Before that, we need some auxiliary definitions and results.

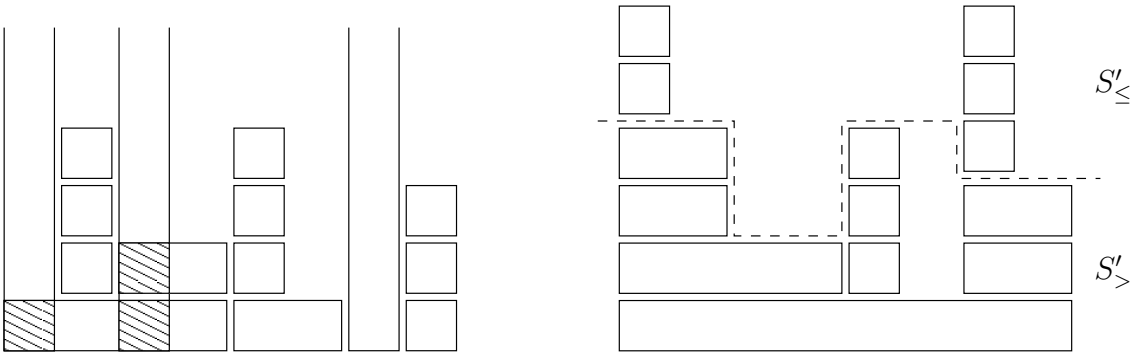


Figure 7.2: An illustration of how we split S'. (a) Optimum solution; (b) permits in $S'_\leq$ are contained in the interval defined by some group permit in the optimum solution.

Due to IM, there is no overlap in group permits in Q^*. In order to simplify our argument, we extend the Hanoi tower ordering to include group permits and assume that, in the optimum solution, a group permit never overlaps with a single permit. We can shift up a group permit so that it does not overlap with single permits, as shown in Figure 7.3. Note that, due to IM, a group permit can only overlap with single permits of larger length; otherwise, we could remove the single permit.

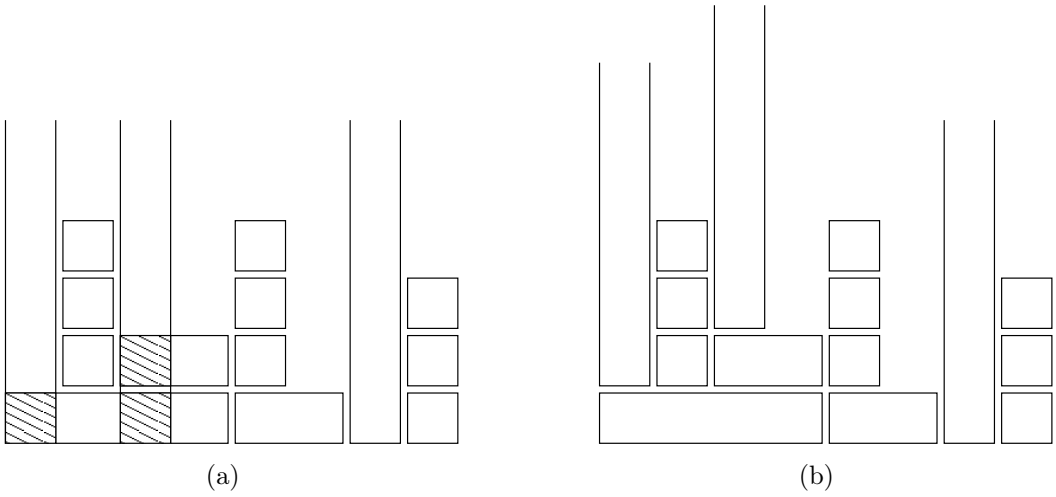


Figure 7.3: An illustration of the Hanoi tower ordering for GPP. (a) A solution of GPP; (b) Solution (a) reorganized so that group and single permits do not overlap.

We say that a group permit in Q^* and a permit in $S'_>$ **overlap** if, after reorganizing both solutions in the Hanoi tower ordering, those permits are used to cover some common demand. See Figure 7.4 for an example.

Lemma 7.3: *At most* $\lceil M \rceil - 1$ *permits in* $S'_>$ *overlap with each group permit in* Q^*.

Proof: We assume that M is an integer; otherwise, it is possible to modify the proof to cover that case as well, as we explain later.

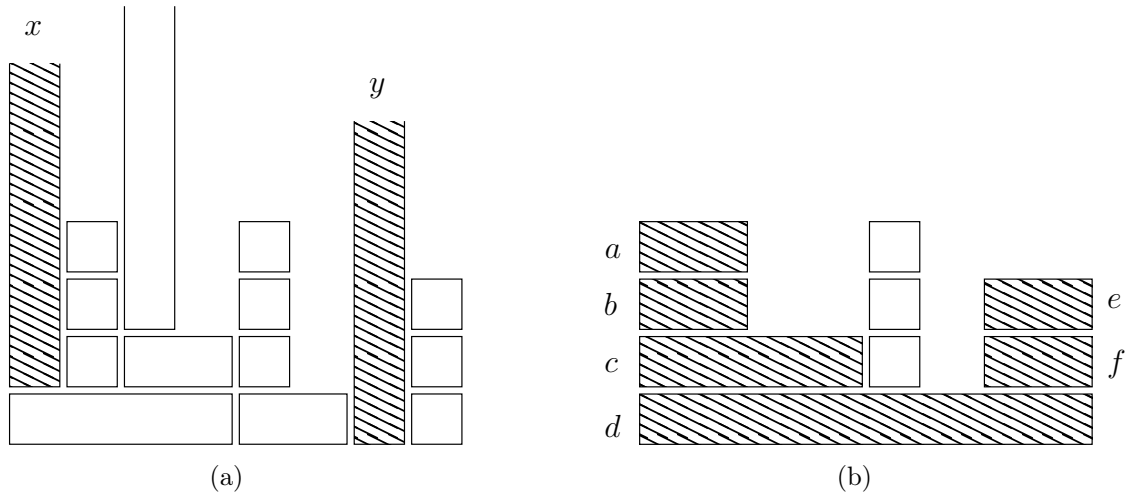


Figure 7.4: An illustration of overlapping permits. (a) Optimum solution; (b) $S_{>}$. Group permit x overlaps with permits a, b and c, and y overlaps with d, e and f.

Given a group permit $(k, \hat{t}) \in Q^*$, let $\lambda_{S^*}(k, \hat{t})$ be the number of permits in S^* of type $\tilde{k} > k$ that cover instant $\hat{t}$; i.e., $\lambda_{S^*}(k, \hat{t}) := |\{(\tilde{k}, \tilde{t}) \in S^* : \tilde{k} > k, \hat{t} \in [\tilde{t}, \tilde{t} + \delta_{\tilde{k}})\}|$. Thus, given a group permit $(k, \hat{t}) \in Q^*$, we have that $(k, \hat{t})$ is used to cover levels greater than $\lambda_{S^*}(k, \hat{t})$. (See Figure 7.5(a).)

Suppose, by contradiction, that a permit $(k, \hat{t}) \in Q^*$ overlaps with at least M permits in $S'_{>}$; if there is more than one such permit in Q^*, consider one with smallest $\lambda_{S^*}(k, \hat{t})$. Let (k', t') be the permit in $S'_{>}$ that overlaps with $(k, \hat{t})$ at level $\lambda_{S^*}(k, \hat{t}) + M$. (See Figure 7.5(a).) Note that $k' > k$, and that only permits of types smaller than k' are used, in the optimum solution, in interval $[t', t' + \delta_{k'})$ to cover demands at levels greater than $\lambda_{S^*}(k, \hat{t})$ (or those permits would overlap with $(k, \hat{t})$).

We replace, in the optimum solution, each group permit in Q^* with M single permits of the corresponding type. Let $\hat{S}$ be the resulting multiset of permits; we have that $\text{cost}(\hat{S}) = \text{opt}(I)$ (see Figure 7.5(b)). Note that all demands covered by (k', t') are covered by single permits in $\hat{S}$, and thus the total cost of those permits in $\hat{S}$ is at least $\gamma_{k'}$, otherwise S'_0 would not be an optimum solution for the MPP instance in Line 1 of Algorithm 7.2. (If M is not an integer, we use a fraction $M - \lfloor M \rfloor$ of a single permit $(k, \hat{t})$ for level $\lambda_{S^*}(k, \hat{t}) + \lceil M \rceil$, and this combined with the other permits in $\hat{S}$ that overlap with (k', t') must cost at least $(M - \lfloor M \rfloor) \cdot \gamma_{k'}$.) Due to the Hanoi tower ordering, for each of the other $M - 1$ lower levels, the total cost of the permits in $\hat{S}$ at interval $[t', t' + \delta_{k'})$ must also be at least $\gamma_{k'}$. Therefore, we conclude that it is better to buy a group permit of type k', which costs $M \cdot \gamma_{k'}$, to cover this interval, but this contradicts the choice of the optimum solution, which always buys a group permit whenever possible. □

Lemma 7.4: $\text{cost}(S'_{>}) \leq \text{opt}(I)$.

Proof: We replace, in the optimum solution, each group permit in Q^* with $\lceil M \rceil - 1$ single permits of the corresponding type; let $\tilde{S}$ be the resulting multiset of permits, and note that $\text{cost}(\tilde{S}) \leq \text{opt}(I)$. Since at most $\lceil M \rceil - 1$ permits in $S'_{>}$ overlap some group

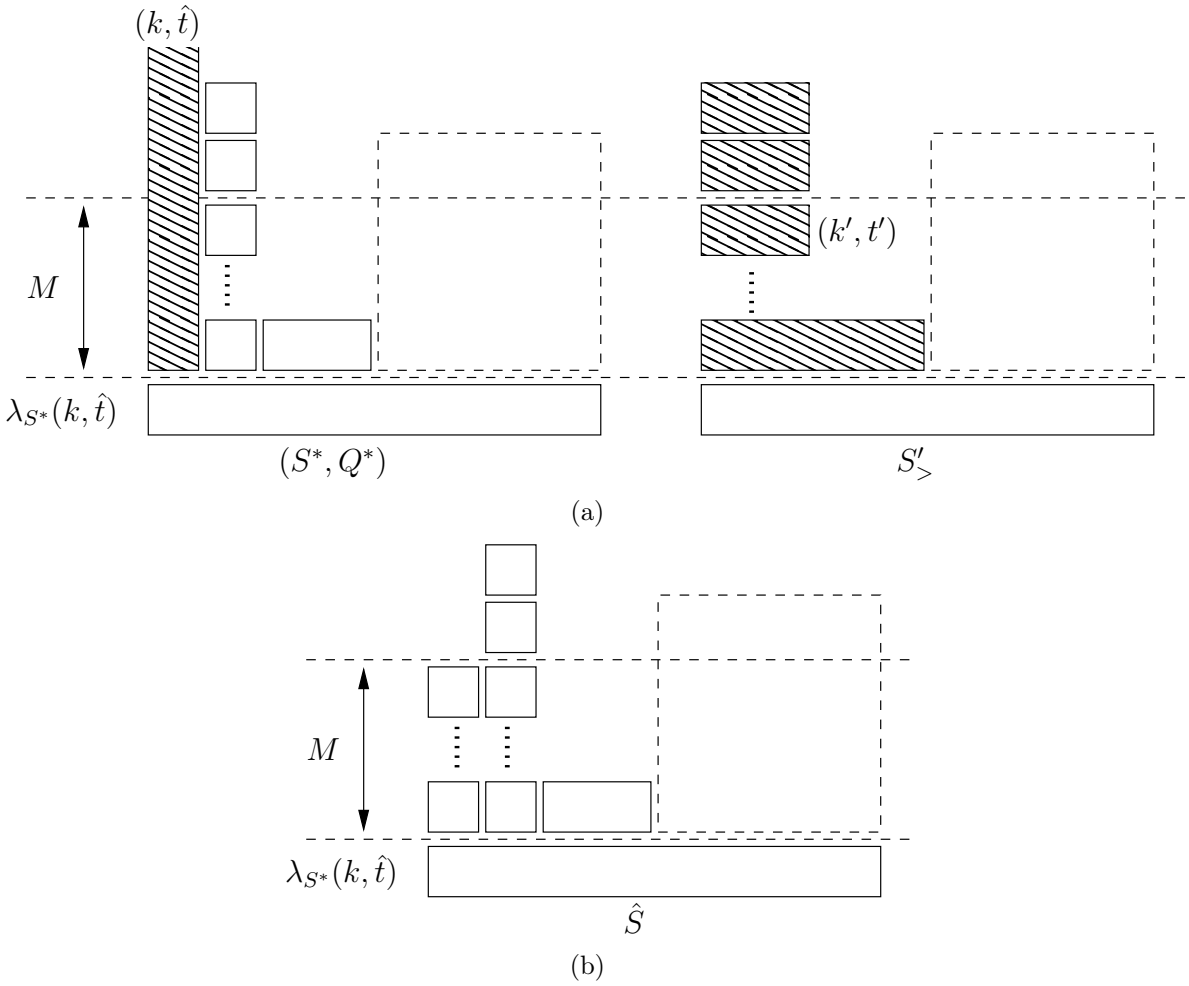


Figure 7.5: An illustration of the proof of Lemma 7.3. (a) Suppose by contradiction that $(k,\hat{t})$ overlaps at least M permits in $S'_>$; consider the M lower levels and let (k',t') be the permit that overlaps with $(k,\hat{t})$ at level $\lambda_{S^*}(k,\hat{t})+M$. (b) Replace each group permit with M single permits of the corresponding type.

permit in Q^*, every demand covered by some permit in $S'_>$ is covered by some permit in $\tilde{S}$. Also, note that $S'_>$ is the optimum solution for the MPP instance defined by the demands covered by those permits. Indeed, suppose by contradiction that there is a better solution $S^*_>$; then $S^*_> \cup (S'_0 \setminus S'_>)$ costs less than S'_0, a contradiction since S'_0 is optimum. Thus, we must have that $\text{cost}(S'_>) \leq \text{cost}(\tilde{S})$. □

Thus, we obtain the following theorem.

Theorem 7.5: *Algorithm 7.1 is a 2-approximation for* GPP *under* IM.

To conclude this section, we present an instance which shows that Algorithm 7.1 has approximation factor at least 4/3 under IM. Take an arbitrary K, $M \geq 2$ integer, $0 < \epsilon \ll 1$, $\delta_k = M^{k-1}$, $\gamma_k = M^{k-1} - (M^{k-1}-1)\epsilon$ and demands such that AlgMPP buys $M-1$ permits $(k,0)$, for $k = 1,\ldots,K-1$, plus one permit $(K,0)$. It is easy to check that Algorithm 7.1 does not buy any group permit; on the other hand, the optimum buys a group permit of type $K-1$ plus $M-1$ single permits of type $K-1$. The cost of the

solution returned by the algorithm is

$$\begin{aligned} & \sum_{k=1}^{K-1}(M-1)\cdot\left(M^{k-1}-(M^{k-1}-1)\epsilon\right)+M^{K-1}-(M^{K-1}-1)\epsilon \\ = \;& (M-1)\cdot\sum_{k=1}^{K-1}M^{k-1}+M^{K-1}-E=2M^{K-1}-1-E, \end{aligned}$$

where $E \ll 1$ is a small number. The optimum solution costs

$$(M+M-1)\cdot\left(M^{K-2}-(M^{K-2}-1)\epsilon\right)\leq(2M-1)\cdot M^{K-2}.$$

Thus, the approximation factor approaches $4/3$ as M approaches 2. An open question is whether Algorithm 7.1 has approximation factor smaller than 2 under IM. Experiments on random instances never attained a factor greater than $4/3$. Another open question is whether we can obtain approximation factor better than 8 if we do not assume IM.

7.2 Online Group Parking Permit

In the online version of GPP, we are given K, δ, γ and M in advance, but the values of T and r are unknown. We begin with an empty multiset of permits and, at each instant of time $t \in \{0,\ldots,T-1\}$, we receive r_t demands and we must buy some permits to ensure that a group permit or at least r_t single permits cover day t. We cannot remove any previously bought permits. In order to simplify our argumentation, given an instance $I=(T,K,\delta,\gamma,r,M)$, we denote by I_t the instance up to day t, i.e., $(t+1,K,\delta,\gamma,(r_0,\ldots,r_t),M)$. In this section we denote a solution for I_t by (S_t,Q_t); note that this is not the same as S_k and Q_k as described in Algorithms 7.1 and 7.2.

One obvious online algorithm, in the spirit of AlgOPP (Section 5.2.2), is the following: at each instant t, we run a (pseudo-polynomial) offline algorithm that obtains an optimum solution (S_t^*,Q_t^*) for I_t, and based on this we buy enough permits from (S_t^*,Q_t^*) to cover day t. However, we do not know how to bound the competitive factor of such algorithm. Instead, we propose the following algorithm, based on a similar principle.

Algorithm AlgOGPP: for each day $t=0,\ldots,T-1$,

Step 1. Run Algorithm 7.1 on I_t to obtain a solution (S_t,Q_t);

Step 2. If Q_t contains some group permit $(k,\hat{t})$ such that no group permit (k^*,t^*) was bought until now with $k^*\geq k$ and $\hat{t}\in[t^*,t^*+\delta_{k^*})$ (i.e., $(k,\hat{t})$ is contained in the interval defined by (k^*,t^*)), then buy group permit $(k,\hat{t})$;

Step 3. Let κ_t be the number of demands from day t that are covered by permits bought until now;

Step 4. Buy permits that, respecting the Hanoi tower ordering, are at levels $\kappa_t+1,\ldots,r_t$ on day t in (S_t,Q_t).[2]

[2]This step can be done in polynomial time by using the techniques we describe in Section 8.2.

The main result of this section is the following. We claim that, under IM, for $t = 0, \ldots, T-1$,

$$\text{cost}(\text{AlgOGPP}(I_t)) \leq 4K \cdot \text{opt}(I_t). \tag{7.1}$$

We actually prove the following stronger result:

$$\text{cost}(\text{AlgOGPP}(I_t)) \leq 2K \cdot (\text{cost}(S_t) + M \cdot \text{cost}(Q_t)), \tag{7.2}$$

where (S_t, Q_t) is the solution obtained by running Algorithm 7.1 on I_t. Thus, Equation (7.1) follows from Theorem 7.5 and Equation (7.2).

The following notation will be useful. Assume permits in (S_t, Q_t) are in the Hanoi tower ordering. We represent a single permit $(k, \hat{t}) \in S_t$ that covers demands of level $\lambda \in \mathbb{Z}_+$ by $(k, \hat{t}, \lambda, \text{SINGLE})$, and a group permit $(k, \hat{t}) \in Q_t$ that covers demands of levels $\lambda, \lambda+1, \ldots$ by $(k, \hat{t}, \lambda, \text{GROUP})$. Thus, we can represent (S_t, Q_t) by a set $\mathcal{S}_t$ of permits in this format. For each $p = (k, \hat{t}, \lambda, g) \in \mathcal{S}_t$, we denote its cost by

$$\text{cost}(p) := \begin{cases} \gamma_k, & \text{if } g = \text{SINGLE}, \\ M \cdot \gamma_k, & \text{if } g = \text{GROUP}. \end{cases}$$

In order to simplify notation, given a set of permits $\mathcal{S}$, we write $\text{cost}(\mathcal{S}) := \sum_{p \in \mathcal{S}} \text{cost}(p)$.

In order to prove Equation (7.2), we share the cost of $\text{AlgOGPP}(I_t)$ among the permits in $\mathcal{S}_t$. More precisely, we define an assignment $\Phi_t : \mathcal{S}_t \mapsto \mathbb{Q}_+$ such that $\sum_{p \in \mathcal{S}_t} \Phi_t(p) = \text{cost}(\text{AlgOGPP}(I_t))$. In order to simplify notation, given a set of permits $\mathcal{S}$ and an assignment Φ, we write $\Phi(\mathcal{S}) := \sum_{p \in \mathcal{S}} \Phi(p)$. We are going to define Φ_t in such a way that $\Phi_t(p) \leq 2K \cdot \text{cost}(p)$ (Lemma 7.9) for every $p \in \mathcal{S}_t$. This implies that

$$\text{cost}(\text{AlgOGPP}(I_t)) = \Phi_t(\mathcal{S}_t) \leq 2K \cdot \text{cost}(\mathcal{S}_t),$$

and so Equation (7.2) holds. We define Φ_t in and inductive manner. For $t = 0$, take $\Phi_0(p) := \text{cost}(p)$ for each $p \in \mathcal{S}_0$; clearly $\Phi_0(\mathcal{S}_0) = \text{cost}(\text{AlgOGPP}(I_0))$. Now suppose Φ_{t-1} is defined, and consider the solution $\mathcal{S}'_{t-1}$ obtained by running Algorithm 7.2 on (I_{t-1}, Q_t). Before defining Φ_t, we define an assignment $\Phi'_{t-1} : \mathcal{S}'_{t-1} \mapsto \mathbb{Q}_+$ which satisfies $\Phi'_{t-1}(\mathcal{S}'_{t-1}) = \Phi_{t-1}(\mathcal{S}_{t-1})$. We need a new concept first.

We say that a single permit p' from a set $\mathcal{S}'$ is **contained** in a permit p from a set $\mathcal{S}$, and we write $p' \subseteq p$ if, after ordering both $\mathcal{S}$ and $\mathcal{S}'$ in HTO, we have that p covers every demand p' covers. Also, we say that a group permit $p' = (k', t', \lambda', \text{GROUP})$ from a set $\mathcal{S}'$ is contained in a group permit $p = (k, \hat{t}, \lambda, \text{GROUP})$ from a set $\mathcal{S}$ if $k' \leq k$ and $t' \in [\hat{t}, \hat{t} + \delta_k)$, i.e., if the interval defined by p contains p'. In order to simplify notation, we denote by $\mathcal{S}'(p) := \{p' \in \mathcal{S}' : p' \subseteq p\}$ the set of permits in $\mathcal{S}'$ that are contained in p. Note that, due to HTO, if $p \in \mathcal{S}'$, then $\mathcal{S}'(p) = \{p\}$.

Now let us define Φ'_{t-1}. Note that each permit in $\mathcal{S}_{t-1}$ contains some permit in $\mathcal{S}'_{t-1}$, since both $\mathcal{S}'_{t-1}$ and $\mathcal{S}_{t-1}$ are built from the same MPP solution for I_{t-1}, and Algorithm 7.1 executes Lines 5-7 for each pair $(k, \hat{t})$ for which Algorithm 7.2 executes Lines 6-8. Also, each permit in $\mathcal{S}'_{t-1}$ is contained in a unique permit in $\mathcal{S}_{t-1}$, due to HTO. For each $p \in \mathcal{S}_{t-1}$

and each $p' \in \mathcal{S}'_{t-1}(p)$, let

$$\Phi'_{t-1}(p') := \Phi_{t-1}(p) \cdot \frac{\text{cost}(p')}{\text{cost}(\mathcal{S}'_{t-1}(p))}.$$

I.e., we split the cost share of p among the permits it contains in $\mathcal{S}'_{t-1}$ in proportion to their cost. Clearly $\Phi'_{t-1}(\mathcal{S}'_{t-1}) = \Phi_{t-1}(\mathcal{S}_{t-1})$, and since $\Phi_{t-1}(\mathcal{S}_{t-1}) = \text{cost}(\text{AlgOGPP}(I_{t-1}))$ by induction hypothesis, we have that $\Phi'_{t-1}(\mathcal{S}'_{t-1}) = \text{cost}(\text{AlgOGPP}(I_{t-1}))$.

Fact 7.6: *For every $p \in \mathcal{S}_{t-1}$ and every $p' \in \mathcal{S}'_{t-1}(p)$, if $\Phi_{t-1}(p) \leq \alpha \cdot \text{cost}(p)$ for some $\alpha \geq 0$, then we have that $\Phi'_{t-1}(p') \leq \alpha \cdot \text{cost}(p')$.*

Proof: We have that $\text{cost}(p) \leq \text{cost}(\mathcal{S}'_{t-1}(p))$, by a similar argument to the proof of Lemma 7.2. Thus,

$$\Phi'_{t-1}(p') = \Phi_{t-1}(p) \cdot \frac{\text{cost}(p')}{\text{cost}(\mathcal{S}'_{t-1}(p))} \leq \alpha \cdot \text{cost}(p) \cdot \frac{\text{cost}(p')}{\text{cost}(\mathcal{S}'_{t-1}(p))} \leq \alpha \cdot \text{cost}(p').$$

□

Now we can relate $\mathcal{S}'_{t-1}$ and $\mathcal{S}_t$.

Fact 7.7: *Every permit in $\mathcal{S}'_{t-1}$ is contained in a unique permit in $\mathcal{S}_t$.*

Proof: First note, due to the definition of Algorithm 7.2, that $\mathcal{S}'_{t-1}$ cannot have group permits where $\mathcal{S}_t$ does not. Thus, every group permit in $\mathcal{S}'_{t-1}$ is contained in some group permit in $\mathcal{S}_t$. If a single permit in $\mathcal{S}'_{t-1}$ is not contained in a group permit in $\mathcal{S}_t$, then it must be contained in some single permit in $\mathcal{S}_t$ because, due to the optimality of AlgMPP, every permit in $\text{AlgMPP}(I_{t-1})$ is contained in some permit in $\text{AlgMPP}(I_t)$. The uniqueness simply follows from HTO. □

Now we define Φ_t. For each $p \in \mathcal{S}_t$,

$$\Phi_t(p) := \begin{cases} \Phi_{t-1}(S'_{t-1}(p)) + \text{cost}(p), & \text{if AlgOGPP buys } p \text{ at instant } t, \\ \Phi_{t-1}(S'_{t-1}(p)), & \text{otherwise.} \end{cases}$$

Since $\Phi'_{t-1}(\mathcal{S}'_{t-1}) = \text{cost}(\text{AlgOGPP}(I_{t-1}))$, clearly $\Phi_t(\mathcal{S}_t) = \text{cost}(\text{AlgOGPP}(I_t))$.

Fact 7.8: *For every $p \in \mathcal{S}_t$, we have that $\text{cost}(\mathcal{S}'_{t-1}(p)) \leq \text{cost}(p)$.*

Proof: If p is a group permit, then Algorithm 7.2 considers buying p, so $\text{cost}(\mathcal{S}'_{t-1}(p)) \leq \text{cost}(p)$. If p is a single permit, then the inequality follows from the optimality of AlgMPP. □

Lemma 7.9: *For each $p \in \mathcal{S}_t$, we have that $\Phi_t(p) \leq 2K \cdot \text{cost}(p)$.*

Proof: We actually prove a stronger claim[3]: for every $p = (k, \hat{t}, \lambda, g) \in \mathcal{S}_t$,

(i) $\Phi_t(p) \leq 2k^* \cdot \text{cost}(p)$ for some $k^* \geq k$ and, at some instant $\tilde{t} \leq t$, AlgOGPP bought a group permit (k^*, t^*) such that $\hat{t} \in [t^*, t^* + \delta_{k^*})$; i.e., (k^*, t^*) contains p; or

(ii) $\Phi_t(p) \leq (2k-1) \cdot \text{cost}(p)$ and $g = \text{SINGLE}$.

We prove this claim by induction on t. The claim is trivial for $t = 0$. So assume that $t > 0$ and the claim is valid for $t-1$. Note that, due to Lemma 7.6 and the fact that each permit in $\mathcal{S}'_{t-1}$ is contained in a unique permit in $\mathcal{S}_{t-1}$, the induction hypothesis implies that, for every $p' = (k', t', \lambda', g') \in \mathcal{S}'_{t-1}$,

(i') $\Phi'_{t-1}(p') \leq 2k^* \cdot \text{cost}(p')$ for some $k^* \geq k'$ and, at some instant $\tilde{t} \leq t-1$, AlgOGPP bought a group permit (k^*, t^*) such that $t' \in [t^*, t^* + \delta_{k^*})$; or

(ii') $\Phi'_{t-1}(p') \leq (2k'-1) \cdot \text{cost}(p')$ and $g' = \text{SINGLE}$.

Now consider a permit $p = (k, \hat{t}, \lambda, g) \in \mathcal{S}_t$, and let us prove that one of (i) or (ii) holds. We divide the proof in two cases.

1. Suppose there is some $p' = (k', t', \lambda', g') \in \mathcal{S}'_{t-1}(p)$ which satisfies (i') with one additional condition: $\Phi'_{t-1}(p') \leq 2k^* \cdot \text{cost}(p')$ for some $k^* \geq k'$, at some instant $\tilde{t} \leq t-1$ AlgOGPP bought a group permit (k^*, t^*) such that $t' \in [t^*, t^* + \delta_{k^*})$, *and* $k^* \geq k$. If there is more than one such permit in $\mathcal{S}'_{t-1}(p)$, choose one with largest k^*. Note that AlgOGPP does not buy p at instant t, since group permit (k^*, t^*) contains p. Furthermore, for any $p' \in \mathcal{S}'_{t-1}(p)$, we have that $\Phi'_{t-1}(p') \leq 2k^* \cdot \text{cost}(p')$, since we chose k^* as large as possible. Then,

$$\Phi_t(p) = \Phi'_{t-1}(\mathcal{S}'_{t-1}(p)) \leq 2k^* \cdot \text{cost}(\mathcal{S}'_{t-1}(p)) \leq 2k^* \cdot \text{cost}(p),$$

where the equality holds by definition of Φ_t, and the last inequality follows from Lemma 7.8. Hence, (i) holds for p since group permit (k^*, t^*) contains p.

2. So assume that each permit in $\mathcal{S}'_{t-1}(p)$ satisfies (ii'), or satisfies (i') with $k^* < k$. We have two subcases.

   (2a) If $g = \text{GROUP}$, then permits of type k in $\mathcal{S}'_{t-1}(p)$ satisfy (ii'). Permits of types $1, \ldots, k-1$ can satisfy (i') or (ii'). Either way, for each $p' \in \mathcal{S}'_{t-1}(p)$, we have that $\Phi'_{t-1}(p') \leq (2k-1) \cdot \text{cost}(p')$. Note that, in Step 2, AlgOGPP buys group permit p. Therefore,

$$\begin{aligned} \Phi_t(p) &= \Phi'_{t-1}(\mathcal{S}'_{t-1}(p)) + \text{cost}(p) \leq (2k-1) \cdot \text{cost}(\mathcal{S}'_{t-1}(p)) + \text{cost}(p) \\ &\leq (2k-1) \cdot \text{cost}(p) + \text{cost}(p) = 2k \cdot \text{cost}(p). \end{aligned}$$

[3]Although this claim seems too strong, it is necessary because, even though every single permit in $\mathcal{S}_{t-1}$ is contained in some permit in $\mathcal{S}_t$, that is not true for group permits. A demand that is covered by a large group permit in $\mathcal{S}_{t-1}$ may be covered by a smaller permit in $\mathcal{S}_t$ because AlgMPP(I_t) decides to buy longer permits for the lower demands.

So p satisfies (i) with $(k^*, t^*) = (k, \hat{t})$ and $\tilde{t} = t$.

(2b) If $g =$ SINGLE, we have two more cases.

(2b.1) $\mathcal{S}'_{t-1}(p) = \{p\}$. Then p satisfies (ii') with $k' = k$. AlgOGPP does not buy p at instant t, since at Step 4 we ensure that we only buy permits for uncovered demands according to HTO. So, $\Phi_t(p) = \Phi'_{t-1}(p) \leq (2k-1) \cdot \text{cost}(p)$ and p satisfies (ii).

(2b.2) $\mathcal{S}'_{t-1}(p)$ consists of single permits of types $1, \ldots, k-1$ that satisfy (i') or (ii'). Either way, for each $p' \in \mathcal{S}'_{t-1}(p)$, we have that $\Phi'_{t-1}(p') \leq 2(k-1) \cdot \text{cost}(p')$. Thus,

$$\begin{aligned} \Phi_t(p) &\leq \Phi'_{t-1}(\mathcal{S}'_{t-1}(p)) + \text{cost}(p) \leq 2(k-1) \cdot \text{cost}(\mathcal{S}'_{t-1}(p)) + \text{cost}(p) \\ &\leq 2(k-1) \cdot \text{cost}(p) + \text{cost}(p) = (2k-1) \cdot \text{cost}(p), \end{aligned}$$

and p satisfies (ii).

□

Theorem 7.10: *Under* IM, AlgOGPP *is* $4K$*-competitive for* GPP.

Therefore, there exists a $16K$-competitive online algorithm for arbitrary instances. This is asymptotically optimal since the $\Omega(K)$ deterministic lower bound for PP [59] also applies to GPP. A $2K$-competitive online algorithm for GPP under IM can be obtained via the greedy algorithm for the covering problem by Koufogiannakis and Young [52], which we discuss in Section 8.3. We did not know of this result until the thesis defense, so we obtained Theorem 7.10 independently, and we decided to keep the presentation of our proof since we believe the analysis technique we developed is interesting.

An open question is whether there exists a randomized $o(K)$-competitive online algorithm for GPP.

Chapter 8

2D Parking Permit

In the **2D parking permit problem** (2DPP), as in MPP and GPP, at each instant we receive a demand. However, in this problem each type of permit has a capacity, besides the length in time. The objective, as usual, is to cover demands with permits whose total cost is minimum. Formally, we have K types of permits with lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$, capacities $\phi_1, \ldots, \phi_K \in \mathbb{N}$, and costs $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, respectively, and we are given a sequence of demands $r_0, \ldots, r_{T-1} \in \mathbb{Z}_+$. The goal is to find a minimum-cost multiset of permits $S \subseteq [K] \times \mathbb{Z}_+$ such that

$$\sum_{\substack{(k,\hat{t})\in S \\ t\in[\hat{t},\hat{t}+\delta_k)}} m_S(k,\hat{t}) \cdot \phi_k \geq r_t \quad \text{for each } t,$$

where $m_S(k,\hat{t})$ is the multiplicity of $(k,\hat{t})$ in S.

This problem admits a $O(K)$-approximation algorithm and a $O(K)$-competitive online algorithm, both by Koufogiannakis and Young [52], which we discuss in Section 8.3. We show that this is an NP-hard problem via a reduction from the **change-making problem** (CM) [57], which is a variant of the unbounded integer knapsack problem.[1] We define it below.

Problem CM(K, ϕ, γ, R): *Given K types of coins with values $\phi_1, \ldots, \phi_K \in \mathbb{Z}_+$ and weights $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, and a change value $R \in \mathbb{Z}_+$, find a minimum-weight multiset of coins, with as many coins of each type as we wish, whose value is at least R.*

Note that this is the unbounded knapsack problem with inverted signs. CM reduces to 2DPP by taking $T = 1$, $r_0 = R$ and $\delta_1 = \cdots = \delta_K = 1$. CM admits an FPTAS, which was proposed for the unbounded knapsack problem [42].

An extension of IM can be defined for the 2D case, in order to give more structure to permits. In addition to IM hypotheses, we assume that there is an ordering of the permits for which capacities divide each other.

Hypothesis 8.1 (**Hierarchical Capacity Property** (HCP)): *There is a permutation $\pi : [K] \mapsto [K]$ such that $\phi_{\pi(k)}$ divides $\phi_{\pi(k-1)}$ for $k = 2, \ldots, K$.*

[1] 2DPP is a leasing variant of BABND on a single edge, and Awerbuch and Azar [5] pointed that BABND on a single edge corresponds to CM.

Fact 8.2: *If there is an α-competitive algorithm for* 2DPP *under* HCP*, then there is a 2α-competitive algorithm for instances that do not satisfy* HCP*.*

Proof: Take an arbitrary instance $I = (T, K, \delta, \phi, \gamma, r)$. Reorder permit types such that $\phi_1 \leq \cdots \leq \phi_K$. For $k = 2, \ldots, K$, take $\phi'_k := 2^{\lfloor \lg \phi_k \rfloor}$, i.e., round down each permit capacity to the closest power of 2. Note that instance $I' = (T, K, \delta, \phi', \gamma, r)$ satisfies HCP. Let $\mathcal{A}$ be an α-competitive algorithm for 2DPP under HCP; $\text{cost}(\mathcal{A}(I')) \leq \alpha \cdot \text{opt}(I')$. Note that $\mathcal{A}(I')$ is a feasible solution for I: if $\mathcal{A}(I')$ uses permit $(k, \hat{t})$, since $\phi'_k \leq \phi_k$, this permit covers in I more demands than it covers in I'. So we just have to bound $\text{opt}(I')$ with regard to $\text{opt}(I)$. Let S^* be an optimum solution of I. Consider a solution $\bar{S}$ which, for each permit $(k, \hat{t}) \in S^*$, contains $2 \cdot m_{S^*}(k, \hat{t})$ copies of $(k, \hat{t})$. Since $\phi'_k = 2^{\lfloor \lg \phi_k \rfloor}$, we have that $\phi_k < 2\phi'_k$, so $\bar{S}$ covers more demands than S^*. Thus, $\bar{S}$ is feasible for I', so $\text{opt}(I') \leq \text{cost}(\bar{S})$, and $\text{cost}(\bar{S}) = 2 \cdot \text{cost}(S^*) = 2 \cdot \text{opt}(I)$. Therefore,

$$\text{cost}(\mathcal{A}(I')) \leq \alpha \cdot \text{opt}(I') \leq \alpha \cdot \text{cost}(\bar{S}) = 2\alpha \cdot \text{cost}(S^*) = 2\alpha \cdot \text{opt}(I).$$

(We can easily modify the proof to only use permit capacities that are powers of any integer $\beta \geq 2$, obtaining competitive factor $\alpha\beta$.) □

Hypothesis 8.3 (**2D Interval Model (**2DIM**)**): *We assume* IM *and* HCP*.*

The following Lemma follows by composing Lemma 5.6 and Fact 8.2.

Lemma 8.4: *If there is an α-competitive algorithm for* 2DPP *under* 2DIM*, then there is an 8α-competitive algorithm for arbitrary instances.*

Note that Lemma 5.9 is also true for 2DPP, even if we only assume IM.

Fact 8.5: CM *is polynomially solvable if we assume* HCP*.*

Proof: Consider a greedy algorithm that chooses the largest-value coin which is at most the remaining change. Suppose by contradiction that there exists some optimum solution S^* which does not use a coin of maximum value $\phi_k \leq R$. Since HCP requires that coin values divide each other, we can find a subset of coins $S' \subseteq S^*$ whose value is exactly ϕ_k. Under HCP, we can assume without loss of generality that $\gamma_k/\phi_k < \gamma_{k'}/\phi_{k'}$ for $k > k'$ (otherwise, we can replace each coin of type k by $\phi_k/\phi_{k'}$ coins of type k' and obtain a lighter solution), so $\gamma(S') > \gamma_k$. Thus, $(S^* \setminus S') \cup \{k\}$ is a solution which is lighter than S^*, a contradiction. □

We can also define an extension of the Hanoi tower ordering for 2DPP under IM. For permits that overlap in time, permits of larger length are under permits of smaller length. For permits of same length that overlap in time, permits of larger capacity are under permits of smaller capacity. Again, demands are assigned to permits in the lowest possible level.

We identify two important particular cases of 2DPP.

The first is what we call the **hierarchical 2D parking permit problem** (H2DPP), in which we assume that $\delta_1 \leq \cdots \leq \delta_K$ and $\phi_1 \leq \cdots \leq \phi_K$; i.e., a permit of type k' always fits into a permit of type $k > k'$. Note that GPP is not a particular case of this

problem, because group permits of smaller length do not fit into single permits of larger length. Under 2DIM we can assume without loss of generality that

$$1 = \delta_1 \cdot \phi_1 < \delta_2 \cdot \phi_2 < \cdots < \delta_K \cdot \phi_K \text{ and } \gamma_k/(\delta_k \cdot \phi_k) < \gamma_{k'}/(\delta_{k'} \cdot \phi_{k'}) \text{ for } k > k'.$$

This version of the problem was proposed by Hu *et al.* [41], and they gave a constant-approximation algorithm and a deterministic $O(K)$-competitive online algorithm. Their result in the online setting is asymptotically optimal due to the lower bound of $\Omega(K)$ on the competitive ratio for any deterministic online algorithm for PP [59], which is a particular case of this problem. However, their algorithms are pseudo-polynomial. In Section 8.2, we show how to turn their algorithms into polynomial time. We also show, in Section 8.1, that their offline algorithm is a pseudo-polynomial exact algorithm under IM for generic 2DPP.

We call the second case the **orthogonal 2D parking permit problem** (O2DPP). In this problem, we have $K \cdot L$ types of permits, each defined by a length of time and a capacity. There are K lengths of time $\delta_1, \ldots, \delta_K \in \mathbb{N}$ with corresponding time scaling costs $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, and L capacities $\phi_1, \ldots, \phi_L \in \mathbb{N}$ with corresponding capacity scaling costs $\mu_1, \ldots, \mu_L \in \mathbb{Q}_+$. A permit with length of time δ_k and capacity ϕ_ℓ costs $\gamma_k \cdot \mu_\ell$. Under 2DIM, we can assume without loss of generality that Equation (5.1) holds and that

$$1 = \phi_1 < \phi_2 < \cdots < \phi_L \text{ and } \mu_\ell/\phi_\ell < \mu_{\ell'}/\phi_{\ell'} \text{ for } \ell > \ell'.$$

Another form of defining the problem is supposing we are given a sub-additive function $\mu' : \mathbb{N} \mapsto \mathbb{Q}_+$ which, given an arbitrary integer capacity, returns a capacity scaling cost in polynomial time. Thus, a permit of length δ_k and capacity $\phi \in \mathbb{N}$ costs $\gamma_k \cdot \mu'(\phi)$. This version of the problem is equivalent up to a constant to the previous definition [70]. There is a $O(K)$-approximation algorithm for this problem, which was given for a more general problem by Anthony and Gupta [4] (we discuss this in Section 10.5). Also, the algorithm by Koufogiannakis and Young [52] for the covering problem is a $O(KL)$-competitive online algorithm for O2DPP.

Note that the reduction from CM proves that both H2DPP and O2DPP are NP-hard, even if we assume IM but do not assume HCP. We do not know how to reduce H2DPP to O2DPP (or vice versa) while losing only a constant factor, so we believe the problems are independent. It turns out that O2DPP generalizes GPP, but H2DPP only generalizes MPP. Note that, if GPP is proven weakly NP-hard under IM, so is O2DPP even under 2DIM while, in Section 8.2, we present a polynomial-time algorithm for H2DPP which is exact under 2DIM.

8.1 A Pseudo-Polynomial Algorithm for Generic 2DPP

In this section we show that the pseudo-polynomial offline algorithm that Hu *et al.* [41] proposed for H2DPP is indeed exact for generic 2DPP under IM. Moreover, our version of the algorithm is simpler, we present it in a clearer way, and our implementation is more efficient. Note that this is also a pseudo-polynomial exact algorithm for GPP under IM.

Let $I = (T, K, \delta, \phi, \gamma, r)$ be an instance of 2DPP.

Definition 8.6 ($I[k, \hat{t}, r - \lambda]$): *For $k \in [K]$, $\hat{t} = c \cdot \delta_k$ for some $c \in \mathbb{Z}_+$, and $\lambda \in \mathbb{Z}_+$, let $I[k, \hat{t}, r - \lambda] := (\delta_k, k, \delta, \phi, \gamma, (r'_{\hat{t}}, \ldots, r'_{\hat{t}+\delta_k-1}))$, where $r'_t = \max\{r_t - \lambda, 0\}$, be the corresponding instance in which only permits of types $1, \ldots, k$ can be used, we only consider interval $[\hat{t}, \hat{t} + \delta_k)$ and we remove λ from the demand of each day in this interval.*

We illustrate this definition in Figure 8.1.

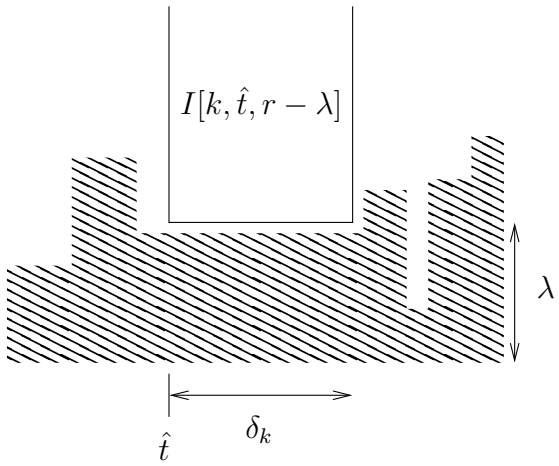


Figure 8.1: Given an instance I, $I[k, \hat{t}, r - \lambda]$ is the sub-instance in which we only allow permits of types $1, \ldots, k$, we only consider interval $[\hat{t}, \hat{t} + \delta_k)$, and we remove λ from the demand of each day in this interval.

Under IM, we claim that 2DPP can be solved via the following recurrence. (Note that we do not require HCP.) We sort permits in non-decreasing order of length of time, and permits of same length of time in increasing order of capacity.

Lemma 8.7:

$$\mathrm{opt}(I[k, \hat{t}, r - \lambda]) = \min \left\{ \begin{array}{l} \gamma_k + \mathrm{opt}(I[k, \hat{t}, r - \lambda - \phi_k]), \\ \sum_{c=0}^{\delta_k/\delta_{k-1}-1} \mathrm{opt}(I[k-1, \hat{t} + c \cdot \delta_{k-1}, r - \lambda]) \end{array} \right\}.$$

I.e., we either buy a permit of type k and combine that with an optimum solution for the remaining demand, or we use an optimum solution that only uses permits of types $1, \ldots, k-1$.

Proof: The proof is by double induction on k and on the total demand of the input, $\sum_{t=0}^{T-1} r_t$. The base case with $k = 1$ is trivial.

Now suppose $k > 1$ and $\max_{t \in [\hat{t}, \hat{t}+\delta_k)} \max\{r_t - \lambda, 0\} > 0$, i.e., the $I[k, \hat{t}, r - \lambda]$ has some positive demand. Let S^* be an optimum solution for $I[k, \hat{t}, r - \lambda]$. If S^* does not use a permit of type k then, due to Lemma 5.9, it is composed of optimum solutions for $I[k-1, \hat{t}, r-\lambda], \ldots, I[k-1, \hat{t}+\delta_k-\delta_{k-1}, r-\lambda]$, which is the second case of the recurrence. Otherwise, if S^* uses a copy of $(k, \hat{t})$, let S' be a solution obtained from S^* by removing one copy of $(k, \hat{t})$. Then, S' must be optimum for $I[k, \hat{t}, r - \lambda - \phi_k]$; otherwise, if $\hat{S}$ is

an optimum solution for $I[k, \hat{t}, r - \lambda - \phi_k]$, then $\hat{S} \cup \{(k, \hat{t})\}$ is a feasible solution for $I[k, \hat{t}, r - \lambda]$ which costs less than S^*, a contradiction which proves the first case of the recurrence. □

Now we show how to implement this recurrence efficiently. This is adapted from the description by Hu *et al.* [41, Algorithms 2–3]. Their algorithms consume time $\mathrm{O}(T^2 \cdot R)$, where $R = \max_{t=0,\ldots,T-1} r_t$, while our implementation consumes time $\mathrm{O}(K \cdot T \cdot R)$, which is usually better since $K = \mathrm{o}(T)$ for typical inputs. We also show that our implementation consumes time $\mathrm{O}(T \cdot R)$ when $\delta_1 < \cdots < \delta_K$. We assume that $\delta_1 = 1$.

First, we run Algorithm 8.1 to compute an auxiliary matrix MAX. For every $k \in [K]$ and every $\hat{t} = c \cdot \delta_k < T$ for some $c \in \mathbb{Z}_+$, $MAX[k, \hat{t}]$ indicates the maximum demand in interval $[\hat{t}, \hat{t} + \delta_k)$.

```
   Input: (T, K, δ, r)
1  for t̂ ← 0 to T − 1 do
2  |  MAX[1, t̂] ← r_t;
3  for k ← 2 to K do
4  |  for t̂ ← 0 to T − 1 step δ_k do
5  |  |  MAX[k, t̂] ← MAX[k − 1, t̂];
6  |  |  for t ← t̂ to min{T − 1, t + δ_k − 1} step δ_{k−1} do
7  |  |  |  if MAX[k − 1, t] > MAX[k, t̂] then
8  |  |  |  |  MAX[k, t̂] ← MAX[k − 1, t];
```

Algorithm 8.1: Algorithm for computing the maximum demand in each interval.

Note that Lines 4–8 consume time $\mathrm{O}(T/\delta_{k-1})$, since Line 4 is executed $\mathrm{O}(T/\delta_k)$ times and, for each time Line 4 is executed, Line 6 is executed $\mathrm{O}(\delta_k/\delta_{k-1})$ times. Thus, the total consumption of the algorithm is $\mathrm{O}(K \cdot T)$. If $\delta_1 < \cdots < \delta_K$, the bound is

$$\mathrm{O}(T) + \sum_{k=2}^{K} \mathrm{O}\left(\frac{T}{\delta_{k-1}}\right) = \mathrm{O}(T) \cdot \left(1 + \frac{1}{\delta_1} + \cdots + \frac{1}{\delta_{K-1}}\right) = \mathrm{O}(T),$$

where the last equality follows since δ_k strictly divides δ_{k-1} for $k = 2, \ldots, K$.

Then, we run Algorithm 8.2 to fill, in a bottom-up fashion, a table with the value of $\mathrm{opt}(I[k, \hat{t}, r - \lambda])$ for every $k \in [K]$, every $\hat{t} = c \cdot \delta_k < T$ with $c \in \mathbb{Z}_+$, and every $\lambda \in \{0, \ldots, MAX[k, \hat{t}] - 1\}$. For $k = 1$, the algorithm simply buys as many type-1 permits as necessary; for $k > 1$, it follows the recurrence. Note that the algorithm computes the value of an optimum solution; the actual solution can be obtained in a top-down fashion after the value is computed.

It is easy to check that Algorithm 8.2 consumes time $\mathrm{O}(K \cdot T \cdot R)$. From the same argument above, it consumes time $\mathrm{O}(T \cdot R)$ if $\delta_1 < \cdots < \delta_K$. Let $\bar{T}$ be the number of days with positive demand; the algorithm may be implemented in time $\mathrm{O}(K \cdot \bar{T} \cdot R)$ by using linked lists. Note that the input can be encoded in $\mathrm{O}(K \cdot \lg(\delta_K \cdot \phi_K \cdot \gamma_K) + \bar{T} \cdot \lg(T \cdot R))$ bits, so this algorithm is pseudo-polynomial.

Since this algorithm is exact under IM, then there exists a pseudo-polynomial 4-approximation for arbitrary instances. Moreover, this algorithm implies that 2DPP is

Input: $(T, K, \delta, \phi, \gamma, r, MAX)$
for $\hat{t} \leftarrow 0$ **to** $T-1$ **do**
  for $\lambda \leftarrow 0$ **to** $r_{\hat{t}} - 1$ **do**
    $\text{opt}(I[1, \hat{t}, r-\lambda]) \leftarrow \lceil (r_{\hat{t}} - \lambda)/\phi_1 \rceil \cdot \gamma_1$;
for $k \leftarrow 2$ **to** K **do**
  for $\hat{t} \leftarrow 0$ **to** $T-1$ **step** δ_k **do**
    for $\lambda \leftarrow MAX[k, \hat{t}] - 1$ **downto** 0 **do**
      $\text{opt}(I[k, \hat{t}, r-\lambda]) \leftarrow \sum_{c=0}^{\delta_k/\delta_{k-1}-1} \text{opt}(I[k-1, \hat{t} + c \cdot \delta_{k-1}, r-\lambda])$;
      if $\text{opt}(I[k, \hat{t}, r-\lambda]) > \gamma_k + \text{opt}(I[k, \hat{t}, r-\lambda-\phi_k])$ **then**
        $\text{opt}(I[k, \hat{t}, r-\lambda]) \leftarrow \gamma_k + \text{opt}(I[k, \hat{t}, r-\lambda-\phi_k])$;

Algorithm 8.2: Pseudo-polynomial algorithm for 2DPP.

weakly NP-hard if we assume IM but do not assume HCP.

8.2 Hierarchical 2D Parking Permit

In this section we show how to turn the algorithm of Section 8.1 into an exact polynomial-time algorithm for H2DPP under 2DIM.

The algorithm represents the output as a set of tuples in the form (t, ℓ, k, q), where t is a starting time, ℓ is the first demand level covered, k is a permit type and q is a multiplicity. A tuple (t, ℓ, k, q) means that q copies of permit (k, t) are stacked to supply the demand of levels $[\ell, \ell + q \cdot \phi_k)$. In some sense, (t, ℓ) is the coordinate of the bottom-left corner of the tuple, and k and q encode their width and height, respectively. (See Figure 8.2.) This representation of the output is essential to guarantee that the algorithm runs in polynomial time.

Let $I = (T, K, \delta, \phi, \gamma, r)$ be an instance of the problem. We sort permit types in non-decreasing order of capacity, breaking ties in increasing order of length. For $k = 1, \ldots, K$, let $I[k]$ be the instance in which only permits of types $1, \ldots, k$ can be used. The algorithm finds an optimum solution for $I[k]$ in the following manner. For $k = 1$, we simply buy r_t copies of permit $(1, t)$, represented by a tuple $(t, 1, 1, r_t)$, for $t = 0, \ldots, T$.

Now suppose $k > 1$ and that we have an optimum solution for $I[k-1]$. We find the optimum solution for each interval of length δ_k of $I[k]$ independently. Note that, due to the optimal substructure presented in Section 8.1, we can assume that the optimum solution for $I[k]$ in the considered interval uses permits of type k to supply levels $1, \ldots, \ell$, together with the optimum solution of $I[k-1]$ for levels $\ell+1, \ldots, R$, for some $\ell \in \{0, \ldots, R\}$ which is a multiple of ϕ_k. So, we just have to find the ℓ which minimizes the cost of the total solution for the considered interval. Due to 2DIM, we can split the optimum solution of $I[k-1]$ in layers of height ϕ_k (see Figure 8.3). Due to HTO, demand decreases as level increases, so we can perform a binary search to find the highest layer for which the cost of the optimum for $I[k-1]$ is greater than γ_k. Also due to HTO, permits of same k and t are used to serve contiguous levels, so we can ensure that we have at most one tuple for each (k, t) and each subproblem. Thus, we can perform the following operations in

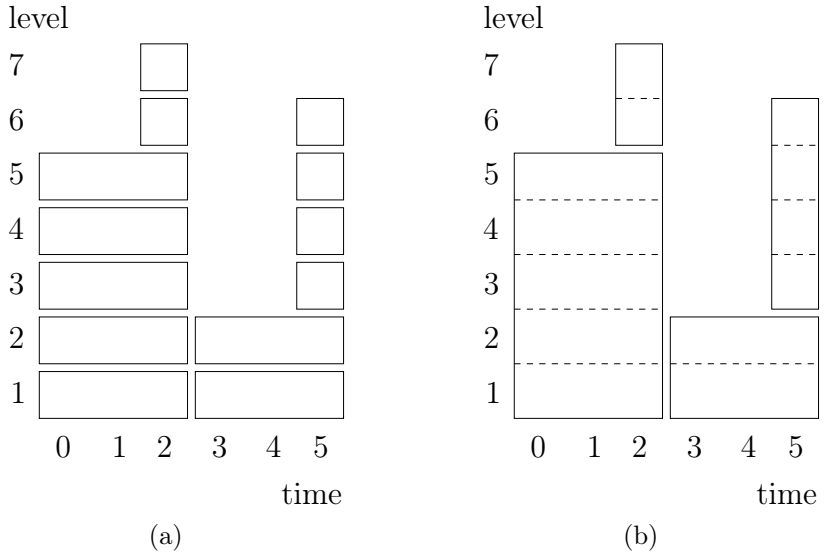


Figure 8.2: An illustration of the representation of a solution as tuples. Consider an instance with $\delta = (1, 3, 6)$ and $\phi = (1, 1, 1)$. (a) Given a multiset of permits with 5 copies of $(2, 0)$, 2 copies of $(1, 2)$, 2 copies of $(2, 3)$ and 4 copies of $(1, 5)$, (b) we represent them as tuples $(0, 1, 2, 5)$, $(2, 6, 1, 2)$, $(3, 1, 2, 2)$ and $(5, 3, 1, 4)$.

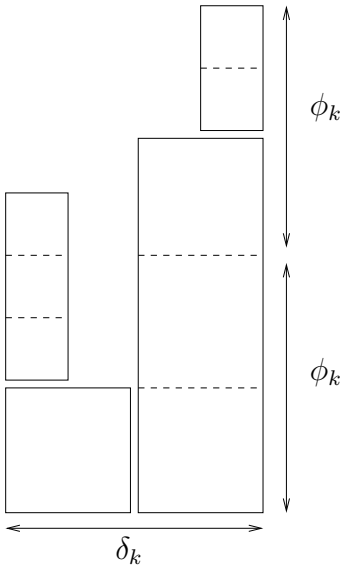


Figure 8.3: We can split the optimum solution for $I[k-1]$ into layers of height ϕ_k. We may split a tuple in two or more parts but, due to 2DIM, we never split a permit.

polynomial time: (i) compute the cost of a given layer of the optimum of $I[k-1]$, as well as (ii) separate the solution in two halves by splitting a tuple evenly among the levels it covers. After we find the correct ℓ, we merge tuples with same (k, t), to guarantee that we have only one tuple for each (k, t). This can be done in polynomial time since the binary search splits the problem into $O(\lg R)$ subproblems. (See Figure 8.4.)

Thus, we have an algorithm that runs in polynomial time and is optimum under 2DIM, so there exists an 8-approximation for arbitrary instances. Note that this algorithm is also a more efficient way to solve MPP exactly if we can assume IM.

The online algorithm by Hu *et al.* [41] utilizes their offline algorithm as a black box

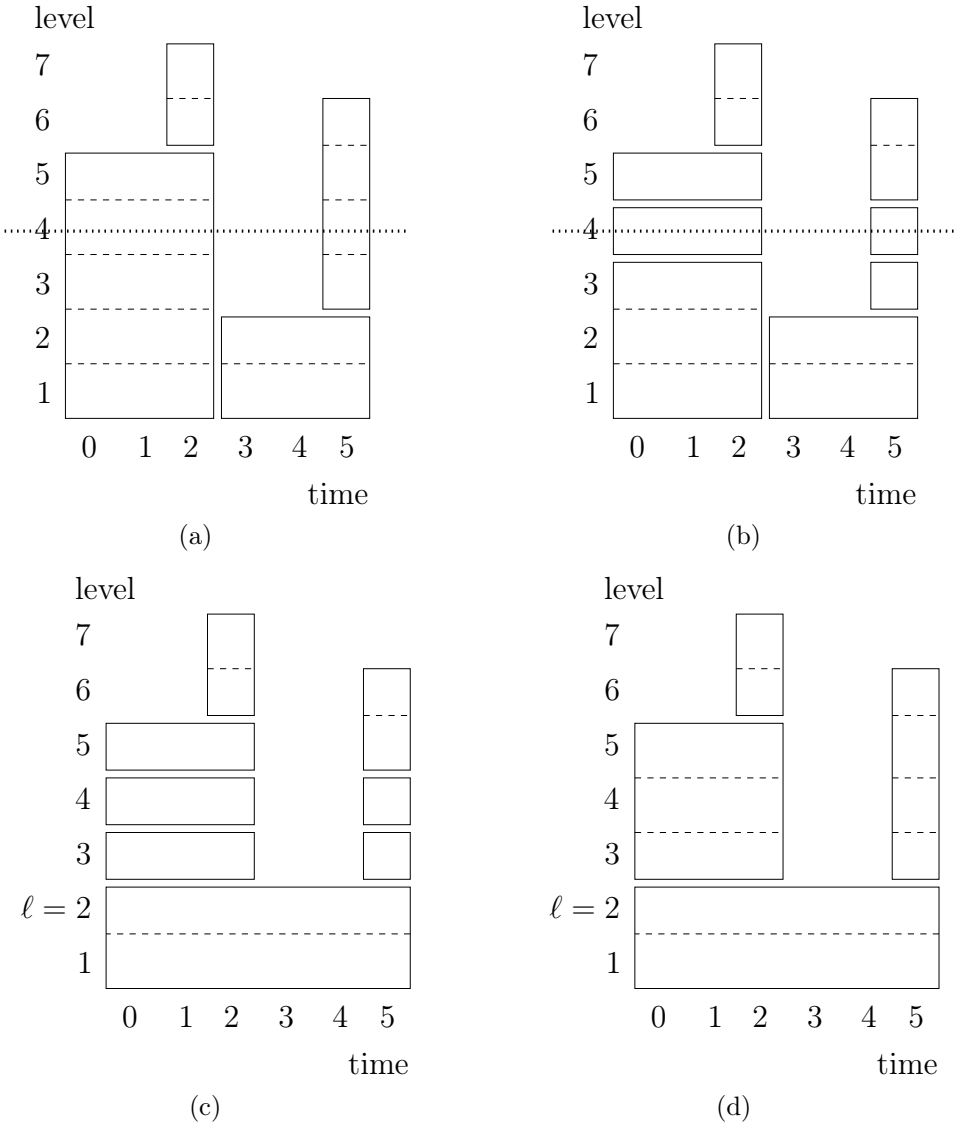


Figure 8.4: An illustration of the binary search step. Consider an instance with $\delta = (1, 3, 6)$ and $\phi = (1, 1, 1)$, and let (a) be an optimum solution for $I[2]$. In order to compute the optimum for $I[3]$, we begin by splitting the solution at level 4. (b) We obtain a solution with tuples $(0, 1, 2, 3)$, $(3, 1, 2, 2)$ and $(5, 3, 1, 1)$ for levels 1–3, tuples $(0, 4, 2, 1)$ and $(5, 4, 1, 1)$ for level 4, and tuples $(0, 5, 2, 1)$, $(2, 6, 1, 2)$ and $(5, 5, 1, 2)$ for levels 5–7. (c) Suppose that, at the end of the binary search we find $\ell = 2$, so we buy a tuple $(0, 1, 3, 2)$ for levels 1–2 and the optimum of $I[2]$ for levels 3–7. (d) Finally, we merge intermediate tuples after the binary search.

and, at each new instant t, it buys the permits bought by the offline solution for sequence $r_0, \ldots, r_t$; this is similar to Meyerson's deterministic algorithm for PP. Under 2DIM, their algorithm is $O(K)$-competitive.[2] By simply replacing their offline algorithm with ours, we obtain a polynomial-time algorithm which is also $O(K)$-competitive. This result can also be obtained via the algorithm for the covering problem by Koufogiannakis and Young [52], which we discuss in the next section.

[2]Indeed, it is K-competitive under 2DIM, so for MPP it has smaller constant hidden factor than the algorithm obtained via the reduction we presented in Chapter 6.

Hu *et al.* [41] also discuss generalizing H2DPP to more dimensions. Their online algorithm is still K-competitive for D dimensions under a D-dimension version of IM. Thus, if D is a constant, then there is a $O(K)$-competitive algorithm for the problem. Moreover, the technique we presented in this section can be extended to D dimensions, and the resulting algorithms run in polynomial time if D is a constant.

8.3 General Results via the Covering Problem

Some general results for 2DPP can be obtained via the greedy algorithm for the covering problem by Koufogiannakis and Young [52]. We restrict our attention to what they call the problem of **covering linear programs with upper bounds on the variables**, which are problems of the form $\min\{\mathbf{c} \cdot \mathbf{x} | \mathbf{x} \in \mathbb{Z}_+^n, \mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}, \mathbf{x} \leq \mathbf{u}\}$, where c_i, b_j and A_{ij} are non-negative for every i, j. They show that a simple greedy algorithm is a Δ-approximation, where Δ is the maximum number of positive coefficients in a constraint. This greedy algorithm can also be used to solve the online version of the problem, thus it is a Δ-competitive online algorithm.

Consider the following formulation of 2DPP as an integer linear program.

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{K} \sum_{\hat{t}=0}^{T-1} x_{k\hat{t}} \cdot \gamma_k \\
\text{subject to} \quad & \sum_{k=1}^{K} \sum_{\substack{\hat{t}=0,\ldots,T-1 \\ t \in [\hat{t}, \hat{t}+\delta_k)}} x_{k\hat{t}} \cdot \phi_k \;\geq\; r_t \quad \forall t \in \{0, \ldots, T-1\}, \\
& x_{k\hat{t}} \in \mathbb{Z}_+, x_{k\hat{t}} \leq R, \qquad \forall k \in [K], \hat{t} \in \{0, \ldots, T-1\}.
\end{aligned}$$

Variable $x_{k\hat{t}}$ indicates how many copies of permit $(k, \hat{t})$ we must buy. The first constraint ensures that each day is covered by enough permits. Note that, if we assume IM, each constraint contains exactly K positive coefficients, so the algorithm by Koufogiannakis and Young [52] yields a $O(K)$-approximation algorithm and a $O(K)$-competitive online algorithm for 2DPP, thus generalizing the online results we obtained for GPP and H2DPP, and also yielding a $O(KL)$-competitive online algorithm for O2DPP.

Chapter 9

Summary and Discussion

In Table 9.1, we summarize known approximation and competitive online results about the parking permit problems we study in Chapters 5–8.

problem	offline setting		online setting	
	general case	under IM (or 2DIM)	general case	under IM (or 2DIM)
PP	1 [59]		$4K$ [59]	K [59]
			randomized $O(\lg K)$ [59]	
MPP	1 (Chapter 6)		$4K$	K (Sec. 8.2), [52]
			randomized $O(\lg K)$ (Chapter 6)	
GPP	8	2 (Sec. 7.1)	$16K$	$4K$ (Sec. 7.2)
			$8K$	$2K$ [52]
O2DPP	$O(K)$ [4]		$4KL$	KL [52]
H2DPP	pseudo-8 [41]	pseudo-1 [41]	pseudo-$(8K)$ [41]	pseudo-K [41]
	8	1 (Sec. 8.2)	$4K$	K (Sec. 8.2), [52]
2DPP	pseudo-4	pseudo-1 (Sec. 8.1)	$4K$	K [52]
	$4K$	K [52]		

Table 9.1: Summary of known approximation and competitive online results for parking permit problems. A '1' means an exact algorithm. A "pseudo-α" means a pseudo-polynomial α-approximation (α-competitive) algorithm.

In Figure 9.1, we depict the dependency between those problems.

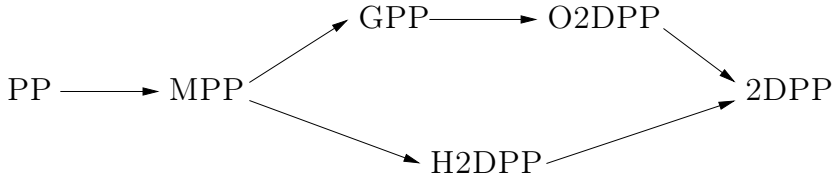


Figure 9.1: A graph depicting dependency between the parking permit problems we study in Chapters 5–8. An arrow $A \to B$ indicates that problem A is a particular case of problem B.

In Chapters 6–8, we addressed generalizations of PP. In particular, the problems we study combine time dynamicity, which is central to PP, with the idea of **capacity dynamicity**.

The capacity dynamicity issue is well-solved for a fixed resource, since it consists in SkiRental. (Even though SkiRental is usually defined with time as a parameter, we note that capacity is a better interpretation for it.) As we discussed in Section 5.1, SkiRental has a trivial exact solution in the offline setting, and a simple 2-competitive online algorithm. We may interpret GPP as a combination of PP with SkiRental, the first dealing with time dynamicity, and the second dealing with capacity dynamicity.

Going one step further, SkiRental can be generalized to multiple types of skis, each lasting for a different amount of time (capacity). In the offline setting, this consists in CM, which we discussed in Chapter 8 and has an FPTAS algorithm. In the online setting, the problem has a 2-competitive algorithm [5, 59]. The combination of PP with this generalization of SkiRental is, then, 2DPP.

Our first guess, when we started this study, was that time dynamicity and capacity dynamicity were independent issues. This would be true if we obtained a polynomial-time exact algorithm for GPP, and an FPTAS for O2DPP. Similarly, if this independency holds, it would be possible to obtain randomized $O(\lg K)$-competitive online algorithms for GPP and 2DPP, which is an open question that seems to be difficult to answer.

Chapter 10

Consequences for Network Leasing Problems

In this chapter we discuss how results on parking permit problems can be used to solve leasing variants of classical network design problems, by using the technique of approximating a finite metric by a tree metric [7, 27].

10.1 Approximating a Finite Metric by a Tree Metric

A metric is a pair (V, d), where V is a set and d is a function $d : V \times V \mapsto \mathbb{Q}_+$ which satisfies, for all $u, v, w \in V$,

(i) $d(u, v) = 0$ if and only if $u = v$ (identity of indiscernibles),

(ii) $d(u, v) = d(v, u)$ (symmetry),

(iii) $d(u, v) \leq d(u, w) + d(w, v)$ (triangle inequality).

We may also say that d is a **metric function**. If V is finite, then (V, d) is a **finite metric**.

Lemma 10.1: *Given a connected graph $G = (V, E)$ and a function $c : E \mapsto \mathbb{Q}_+^*$ that assigns a positive length to each edge, consider the function $d : V \times V \mapsto \mathbb{Q}_+$ where $d(u, v)$ is the length of a shortest uv-path for every $u, v \in V$. Then, (V, d) is a metric.*

A proof of this fact can be found in [17, Example 3.4]. Conversely, every finite metric can be represented by a complete graph with positive edge lengths corresponding to the distances in the metric, which is called a **metric graph**.

Definition 10.2: *We say that a metric (V', d')* **dominates** *a metric (V, d) if $V' \supseteq V$ and $d'(u, v) \geq d(u, v)$ for any $u, v \in V$.*

Definition 10.3: *We say that a metric (V', d')* **α-approximates** *a metric (V, d) if (V', d') dominates (V, d) and, for any $u, v \in V$, $d'(u, v) \leq \alpha \cdot d(u, v)$.*

Definition 10.4: *If $\mathcal{D}$ is a probability distribution on a family $\mathcal{S}$ of metrics that dominate a metric (V, d), then $(\mathcal{S}, \mathcal{D})$* **probabilistically α-approximates** *(V, d) if, for every $u, v \in V$, it holds that $\mathbb{E}_{(V', d') \in_{\mathcal{D}} \mathcal{S}}[d'(u, v)] \leq \alpha \cdot d(u, v)$.*

We call the factor α the **distortion** of the metric approximation. Bartal [7] proved the following result, which states that, if we know how to approximate a metric by a metric of a particular type with low distortion, then we can reduce some problems in arbitrary metrics to the problem restricted to that class of metrics by losing some guarantee of quality.

Lemma 10.5 (Bartal [7]): *Given a minimization problem on a finite metric (V, d) whose objective function is a non-negative linear combination of distances in d, if (V, d) can be α-approximated by a metric in a class $\mathcal{C}$ and there is an β-competitive algorithm for the special case of metrics in $\mathcal{C}$, then there is a randomized $\mathrm{O}(\alpha\beta)$-competitive algorithm for the general case.*

A proof of this fact can be found in [17, Theorem 3.11]. One type of metric of particular interest for this reduction is the class of **tree metrics**, which are metrics induced by shortest paths in a tree. Since it is usually easier to find good algorithms for trees, approximating a finite metric by a tree metric is a very powerful technique. Unfortunately, there are some finite metrics for which every deterministic approximation by a tree metric has distortion $\Omega(|V|)$ [61]. However, if we look for a probabilistic approximation, the lower bound is $\Omega(\lg |V|)$ [7] and, indeed, there is a randomized algorithm by Fakcharoenphol, Rao and Talwar [27] that obtains a probabilistic approximation of any finite metric by tree metrics with expected distortion $\mathrm{O}(\lg |V|)$.

Theorem 10.6 (Fakcharoenphol, Rao and Talwar [27]): *There exists a polynomial-time randomized algorithm which, given a finite metric (V, d), returns a tree metric (V', d') which dominates (V, d) and, for every $u, v \in V$, it holds that $\mathbb{E}[d'(u, v)] \leq \mathrm{O}(\lg |V|) \cdot d(u, v)$.*

For nice presentations of this result, see [17, Chapter 4] and [74, Section 8.5].

In this thesis, we are interested in approximating a metric (V, d) by a tree metric (V', d') with the extra property that $V' = V$. This can be done by losing a constant extra factor on the distortion [51] (see [74, Theorem 8.19] for a nice presentation). We denote by FRT the resulting randomized algorithm which, given a finite metric (V, d) returns a tree $\mathcal{T} = (V, \mathcal{E})$ and a function $c_{\mathcal{T}} : \mathcal{E} \mapsto \mathbb{Q}_+^*$, such that $(V, d_{\mathcal{T}})$ probabilistically $\mathrm{O}(\lg |V|)$-approximates (V, d), where $d_{\mathcal{T}} : V \times V \mapsto \mathbb{Q}_+$ represents the length of shortest paths in $\mathcal{T}$ with edge lengths according to $c_{\mathcal{T}}$.

10.2 Steiner Leasing

In this section we are interested in a leasing variant of the **Steiner forest problem** (SF), which we formally define below.

Problem $\mathrm{SF}(G, d, D_0, \ldots, D_{T-1})$: *Given a graph $G = (V, E)$, a function $d : E \mapsto \mathbb{Q}_+$ assigning a length to each edge, and a sequence $D_0, \ldots, D_{T-1} \subseteq V$ of subsets of the vertices, find a subset of the edges $\mathcal{E} \subseteq E$ such that, for $t = 0, \ldots, T-1$, there exists a path between each pair of vertices of D_t in the graph $(V, \mathcal{E})$, and such that $\sum_{e \in \mathcal{E}} d(e)$ is minimum.*

Note that ST reduces to this problem if $T = 1$, so SF is NP-hard. The best known algorithm for SF is a 2-approximation, due to Agrawal, Klein and Ravi [2]. In the online

version of SF, T is unknown, $D_0, \ldots, D_{T-1}$ are given one at a time, and we cannot remove edges previously bought. The problem admits a $O(\lg n)$-competitive online algorithm by Berman and Coulston [8], where $n := \left|\bigcup_{t=0}^{T-1} D_t\right|$. The reduction from ST also shows that SF has a lower bound of $\Omega(\lg n)$ on the competitive factor.

It is common to assume that $|D_t| = 2$ for every t; this case is polynomially equivalent to the general case. It is also practical to assume that (G, d) is a metric graph; this is without loss of generality [73].

Lemma 10.7: SF *is equivalent to the case that* (G, d) *is a metric graph.*

Proof: If (G, d) is not a metric graph, assume that d is positive (we can force this by contracting zero-length edges). Then, consider the **metric closure** of (G, d), which is a pair (G', d') where $G' = (V, E')$ is a complete graph and $d' : E' \mapsto \mathbb{Q}_+^*$ is given by taking shortest paths in (G, d). For any optimum solution in (G, d), each edge must be a shortest path between its ends, or instead we could use the edges in the shortest path and obtain a better solution. Similarly, given a solution for (G', d'), we obtain a solution of same cost for (G, d) by replacing each edge by the shortest path between its ends in (G, d). □

Now we define a leasing variant of SF, the **Steiner leasing problem** (SLE).

Problem SLE$(G, d, K, \delta, \gamma, (u_0, v_0), \ldots, (u_{T-1}, v_{T-1}))$: *We are given a complete graph* $G = (V, E)$*, a metric function* $d : V \times V \mapsto \mathbb{Q}_+$*, an integer* K *which represents the number of leasing types, a function* $\delta : [K] \mapsto \mathbb{N}$ *that assigns a length of time to each leasing type, a function* $\gamma : [K] \mapsto \mathbb{Q}_+$ *that assigns a leasing factor to each leasing type, and a sequence* $(u_0, v_0), \ldots, (u_{T-1}, v_{T-1}) \in V \times V$ *of pairs of vertices. The goal is to find a set of edge leases* $\mathcal{E} \subseteq E \times [K] \times \mathbb{Z}_+$ *such that, for every* t*, there exists a* (u_t, v_t)*-path* P *in* G *such that, for each edge* $e \in P$*, there is some* $(e, k, \hat{t}) \in \mathcal{E}$ *such that* $t \in [\hat{t}, \hat{t} + \delta_k)$*, and we wish to minimize* $\sum_{(e,k,\hat{t})\in\mathcal{E}} d(e) \cdot \gamma_k$*.*

This problem was proposed by Meyerson [59]. Clearly SF reduces to SLE if $K = 1$, $\delta_1 = \infty$ and $\gamma_1 = 1$, so SLE is NP-hard. Note that PP also reduces to SLE if $|V| = 2$. In the online version of SLE, T is unknown, $(u_0, v_0), \ldots, (u_{T-1}, v_{T-1})$ are revealed one at a time, and we cannot remove previous edge leases. From the reductions we describe above, we have that the competitive factor of SLE has lower bound $\Omega(\lg K + \lg n')$, where $n' := \min\{n, \delta_K\}$.[1] Moreover, Lemma 10.7 is also true for SLE, since leasing types are uniform among the edges.

Meyerson gave a $O(\lg K \lg |V|)$-competitive online algorithm for SLE, which we present in Algorithm 10.1. First, the algorithm obtains, via algorithm FRT, a tree $\mathcal{T}$ such that $V_{\mathcal{T}} = V$, and whose distances probabilistically $O(\lg |V|)$-approximate the distances in G. Then, for each edge e in $\mathcal{T}$, it maintains an instance of algorithm AlgRandOPP. When a pair of vertices (u_t, v_t) arrives, it finds the unique (u_t, v_t)-path in $\mathcal{T}$, and requires a parking permit for day t to the instance corresponding to each edge in this path. The algorithm then leases edge e using the corresponding permit type AlgRandOPP$[e]$ chooses.

[1] We use n' instead of n due to Lemma 5.9.

Input: $(G, d, K, \delta, \gamma)$
$\mathcal{E} \leftarrow \emptyset$;
$(\mathcal{T}, c_{\mathcal{T}}) \leftarrow \text{FRT}(V, d)$;
foreach edge e of $\mathcal{T}$ **do**
  initialize an instance AlgRandOPP[e] with (K, δ, γ);
when (u_t, v_t) arrives **do**
  $P \leftarrow \text{path}_{\mathcal{T}}(u_t, v_t)$;
  foreach edge $e \in P$ **do**
    send 1 at instant t to AlgRandOPP[e], obtaining a permit (k_e, t_e);
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(e, k_e, t_e)\}$;
return $\mathcal{E}$;

Algorithm 10.1: Online algorithm for SLE [59].

Theorem 10.8 (Meyerson [59]): *Algorithm 10.1 is* $O(\lg K \lg |V|)$*-competitive for* SLE.

Proof: Let S be the solution returned Algorithm 10.1, S^* be an optimal solution, and S' be an optimal solution on $\mathcal{T}$. We have that

$$\begin{aligned}
\mathbb{E}[\text{cost}(S)] &= \mathbb{E}\left[\sum_{(e,k,t)\in S} \gamma_k \cdot d(e)\right] \\
&= \mathbb{E}\left[\sum_{e\in\mathcal{T}} d(e) \cdot \sum_{(e,k,t)\in S} \gamma_k\right] \\
&\leq \mathbb{E}\left[\sum_{e\in\mathcal{T}} d_{\mathcal{T}}(e) \cdot \sum_{(e,k,t)\in S} \gamma_k\right] \\
&\leq \mathbb{E}\left[\sum_{e\in\mathcal{T}} d_{\mathcal{T}}(e) \cdot O(\lg K) \cdot \sum_{(e,k,t)\in S'} \gamma_k\right] \\
&= O(\lg K) \cdot \mathbb{E}[\text{cost}(S')] \\
&\leq O(\lg K) \cdot \mathbb{E}\left[\sum_{(e,k,t)\in S^*} \gamma_k \cdot d_{\mathcal{T}}(e)\right] \\
&= O(\lg K) \cdot \sum_{(e,k,t)\in S^*} \gamma_k \cdot \mathbb{E}[d_{\mathcal{T}}(e)] \\
&\leq O(\lg K) \cdot \sum_{(e,k,t)\in S^*} \gamma_k \cdot O(\lg |V|) \cdot d(e) \\
&= O(\lg K \lg |V|) \cdot \text{cost}(S^*).
\end{aligned}$$

The second equality follows from the fact that we only lease edges in $\mathcal{T}$. The second inequality follows from the fact that SLE on $\mathcal{T}$ reduces to solve MPP on each edge. The third inequality follows from the fact that we can build a feasible instance on $\mathcal{T}$ from S^* in the following manner: for each $(e, k, t) \in S^*$ with $e = uv$, we lease (e', k, t), for each e' in the uv-path in $\mathcal{T}$. □

The same technique can be used to obtain an offline $O(\lg n)$-approximation algorithm. Note that we do not need to approximate the entire metric (V, d); instead, we may only approximate the metric induced by the requested points. The proof of the approximation guarantee is identical to that of Theorem 10.8. If we restrict the problem to the tree case, when we have a fixed root vertex r so that $v_t = r$ for every t, then there are a $O(K)$-approximation algorithm [4] and a $O(K \lg n)$-competitive online algorithm [9].

10.3 Steiner Network Leasing

A simple extension of SF is the **Steiner network problem** (with edge duplication) (SN), in which we receive pairs of vertices $(u_0, v_0), \ldots, (u_{T-1}, v_{T-1})$, and a demand $r_t \in \mathbb{Z}_+$ for each pair (u_t, v_t). We wish to find a minimum-cost multiset of the edges, and we are allowed to use as many copies of each edge as we need, so that u_t and v_t are connected by r_t *edge-disjoint* paths in the multigraph induced by the multiset of edges. SF reduces to SN if $r_t = 1$ for each t, so this is an NP-hard problem. It was proposed by Jain [44], who gave a 2-approximation algorithm. Lemma 10.7 is also true for this problem.

In the online version of SN, the graph and edge lengths are given in advance, and the algorithm must maintain a multiset of edges, which is initially empty. At each instant of time t, we receive a pair (u_t, v_t) of vertices with a demand r_t. Additional edges may be bought, in order to guarantee that there exist r_t edge-disjoint paths between u_t and v_t. We cannot remove edges that were bought previously, and we wish to minimize the cost of the final multiset of edges. Umboh [72] gave a $O(\lg n)$-competitive online algorithm for this problem, where n is the number of vertices that are in some request pair. From the reduction from SF, SN has online lower bound of $\Omega(\lg n)$.

We propose the following leasing variant of SN. The input for the **Steiner network leasing problem** (SNLe) consists of a complete graph $G = (V, E)$, a metric distance function $d : V \times V \mapsto \mathbb{Q}_+$, K leasing types with lengths of time $\delta_1, \ldots, \delta_K \in \mathbb{N}$ and scaling costs $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, and a sequence of triples $(u_0, v_0, r_0), \ldots, (u_{T-1}, v_{T-1}, r_{T-1})$ in which $u_t, v_t \in V$ and $r_t \in \mathbb{Z}_+$ for every t. A solution consists of a multiset of edge leases $S \subseteq E \times [K] \times \mathbb{Z}_+$ such that, for $t = 0, \ldots, T-1$, the multigraph induced by multiset $\{e \in E : (e, k, \hat{t}) \in S \text{ and } t \in [\hat{t}, \hat{t} + \delta_k)\}$ contains r_t edge-disjoint (u_t, v_t)-paths. The goal is to minimize $\sum_{(e,k,\hat{t}) \in S} m_S(e, k, \hat{t}) \cdot d_e \cdot \gamma_k$.

If G is a tree, then there is a unique path between each pair of vertices. In this case, SNLe reduces to solving MPP in each edge, in order to decide how many copies and which leasing types to use to serve each input triple. Since we have a constant-approximation algorithm and an online $O(\lg K)$-competitive algorithm for MPP, we obtain a $O(\lg n)$-approximation algorithm and a $O(\lg K \lg |V|)$-competitive online algorithm for SNLe by using the same technique of Algorithm 10.1 for SLe. The proof of the quality of the returned solution is identical, so we omit it.

10.4 Leasing Rent-or-Buy

Another well-studied generalization of SF is the **rent-or-buy problem** (RoB), which was proposed by Karger and Minkoff [47]. In this problem, the input consists of a graph $G = (V, E)$ with edge lengths $d : E \mapsto \mathbb{Q}_+$, a scaling factor $M \geq 1$ and a sequence $(u_0, v_0), \ldots, (u_{T-1}, v_{T-1}) \in V \times V$ of pairs of vertices, which we call the **terminals**. The goal is to find a set of edges to **buy** and, for each pair of terminals, a set of edges to **rent**, so that there exists a path between each pair that uses only bought edges or edges that were rented by the pair. A bought edge e can be used to serve an unlimited number of requests and costs $M \cdot d(e)$, while a rented edge e costs $d(e)$ for each pair that decides to rent it. We wish to minimize the total cost of buying and renting edges. Note that this problem is NP-hard, since SF reduces to it if $M = 1$. Fleischer *et al.* [29] gave a 5-approximation algorithm for RoB. A particular case of the problem that has been given much attention in the literature is the **single-source** rent-or-buy problem, in which one of the vertices in each pair is a fixed vertex r, which we call the **root**. Note that, in this case, the problem becomes a generalization of ST, and it admits a 3.55-approximation algorithm, due to Gupta *et al.* [36]. Lemma 10.7 is also true for RoB.

In the online version of RoB, the graph, edge lengths and the scaling factor M are known in advance, and the pairs of terminals are given one at a time. The algorithm must maintain a set of bought edges, which is initially empty and, for each pair of terminals that arrive, the algorithm must choose some edges to buy and some edges to rent, so that there exists a path between the pair of terminals composed of bought edges and edges that are rented for this pair. We wish to minimize the cost of the bought edges, plus the sum of the cost of the edges that are rented for each pair. $O(\lg n)$-competitive online algorithms were given for this problem [6, 72], where n is the number of vertices that are in some request pair. From the reduction from SF, RoB has online lower bound of $\Omega(\lg n)$.

In the **leasing rent-or-buy problem** (LeRoB), we are given a complete graph $G = (V, E)$ with a metric distance function $d : V \times V \mapsto \mathbb{Q}_+$, K leasing types with lengths of time $\delta_1, \ldots, \delta_K \in \mathbb{N}$ and scaling costs $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, a constant $M \geq 1$, and a sequence $D_0, \ldots, D_{T-1} \subseteq V \times V$ of pairs of vertices. We wish to find a multiset of single edge leases $S \subseteq E \times [K] \times \mathbb{Z}_+$ and a set of group edge leases $Q \subseteq E \times [K] \times \mathbb{Z}_+$ such that, for each $(u, v) \in D_t$ with $t \in \{0, \ldots, T-1\}$, there exists some (u, v)-path P_{uvt} in G such that, for every $t' \in \{0, \ldots, T-1\}$ and every edge $e \in G$, we have that $\sum_{\substack{(e,k,\hat{t}) \in S \\ t' \in [\hat{t}, \hat{t}+\delta_k)}} m_S(e, k, \hat{t}) \geq |\{(u, v) \in D_{t'} : e \in P_{uvt'}\}|$, i.e., we have a different single edge lease for each path that uses e at instant t', or we have some group edge lease $(e, k, \hat{t}) \in Q$ with $t' \in [\hat{t}, \hat{t} + \delta_k)$. We wish to minimize

$$\sum_{(e,k,\hat{t}) \in S} m_S(e, k, \hat{t}) \cdot c_e \cdot \gamma_k + M \cdot \sum_{(e,k,\hat{t}) \in Q} c_e \cdot \gamma_k.$$

Note that, if $|D_t| = 1$ for every t, then it is never useful to obtain a group lease, and the problem reduces to SLe. Also, LeRoB is equivalent to the variant in which a single pair of vertices and an integer demand are received at each instant of time. Thus, SNLe is a

particular case of LeRoB when $M = \infty$, even though online SN is not a particular case of online RoB: in the former, edges are permanent while, in the latter, rented edges are temporary.

LeRoB reduces to solve GPP in each edge if the input metric is a tree. Thus, by approximating the input metric by a tree metric, we obtain a randomized $O(\lg n)$-approximation algorithm and a randomized $O(K \lg |V|)$-competitive online algorithm. For the single-source case of LeRoB (in which one of the vertices in every pair is a fixed vertex r), Anthony and Gupta presented a $O(K)$-approximation [4], which is usually better than our result since the approximation factor does not depend on the temporal dimension. However, for multiple sources, our result improves the previous best algorithm, which was their $O(K \lg n)$-approximation for orthogonal LeBaBND. Orthogonal LeBaBND generalizes LeRoB, and we discuss it in Section 10.5.

10.5 Leasing Buy-at-Bulk Network Design

A generalization of RoB is the **buy-at-bulk network design problem** (BaBND). We are given a graph $G = (V, E)$, with an edge length function $d : E \mapsto \mathbb{Q}_+$. We are given a number L of types of cables and, for each type of cable ℓ, we are given a capacity $\phi_\ell \in \mathbb{N}$ and a cost per unit of length $\mu_\ell \in \mathbb{Q}_+$. We are then given a sequence $(u_0, v_0), \ldots, (u_{T-1}, v_{T-1}) \in V \times V$ of pair of vertices, and a demand $r_t \in \mathbb{Z}_+$ for each pair (u_t, v_t). We wish to find (i) a function $\lambda : E \times \{1, \ldots, L\} \mapsto \mathbb{Z}_+$ that assigns a number of cables of each type to each edge of G and, (ii) for each pair (u_t, v_t), a path $P(u_t, v_t)$ in G, such that

$$\sum_{(u_t,v_t):e \in P(u_t,v_t)} r_t \leq \sum_{\ell=1}^{L} \phi_\ell \cdot \lambda(e, \ell)$$

for each edge $e \in E$. The goal is to minimize the cost of installing the cables, which is given by

$$\sum_{\ell=1}^{L} \sum_{e \in E} \mu_\ell \cdot d(e) \cdot \lambda(e, \ell).$$

Intuitively, the goal is to connect the pairs by paths with sufficient capacity to flow the demand, while minimizing the cost of the network. Note that the definition of the problem requires that the flow between each pair goes through a single path. We can assume without loss of generality that cables with larger capacity have smaller cost per unit of length, which represents an economy of scale and allows us to take advantage of the subadditivity property. This problem is NP-hard since RoB reduces to it if $K = 2$, $\phi = (1, \infty)$ and $\mu = (1, M)$. BaBND is NP-hard even if the graph is a single edge, in which case it is equivalent to CM. BaBND was proposed by Awerbuch and Azar [5], who gave a $O(\lg n)$-approximation algorithm that uses Lemma 10.5 and Theorem 10.6. The problem does not admit a $O(\lg^{\frac{1}{4}-\epsilon} n)$-approximation for any constant $\epsilon > 0$, unless NP $\subseteq$ ZPTIME($n^{\text{polylog } n}$) [3]. It is usual to assume that one of the vertices in each pair

is a fixed root vertex; in this case we call the problem **single-source** BABND.[2] For this case, Grandoni and Italiano [34] gave a 24.92-approximation algorithm. Lemma 10.7 is also true for BABND.

In the online version of BABND, the graph, edge lenghts and types of cables are known in advance, and the pairs of vertices and their demands are given in an online manner. For each new pair (u_t, v_t) that arrives, we must define a path connecting (u_t, v_t), and new cables must be bought so that there is enough capacity to flow the current total load of the network. The goal is to minimize the total cost for installing the cables. Awerbuch and Azar [5] gave a randomized $O(\lg |V|)$-competitive online algorithm for this problem, which is also based on Lemma 10.5 and Theorem 10.6. Recently, Gupta *et al.* [37] gave a deterministic $O(\lg n)$-competitive online algorithm for the single-source case, where n is the number of vertices that are in some request pair. Due to the reduction from ROB, BABND has online lower bound of $\Omega(\lg n)$.

In the **leasing buy-at-bulk network design problem** (LEBABND), we are given a metric distance function d between the vertices in a complete graph $G = (V, E)$, K types of cables with lengths of time $\delta_1, \ldots, \delta_K \in \mathbb{N}$, capacities $\phi_1, \ldots, \phi_K \in \mathbb{N}$ and costs per unit of distance $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, and at every instant t we receive a pair (u_t, v_t) of vertices with an associated demand r_t. We must lease cables so that there is a path connecting u_t and v_t in G whose edges have leased capacity at least r_t at instant t. We wish to minimize the cost of the leased cables, where leasing a cable of type k for edge e costs $\gamma_k \cdot d_e$. The problem reduces to solve 2DPP in each edge if the input metric is a tree metric so, by approximating the input metric by a tree, we have a pseudo-polynomial $O(\lg n)$-approximation algorithm, and there are a polynomial-time $O(K \lg n)$-approximation algorithm and a $O(K \lg |V|)$-competitive online algorithm, where n is the number of requested pairs.

As we did for 2DPP, we can define hierarchical and orthogonal versions of LEBABND. In the hierarchical version, we have that $\delta_1 \leq \cdots \leq \delta_K$ and $\phi_1 \leq \cdots \leq \phi_K$, and the problem reduces to solve H2DPP in each edge if the input metric is a tree. Thus, by approximating the input metric by a tree metric, we obtain a randomized $O(\lg n)$-approximation algorithm, and a randomized $O(K \lg |V|)$-competitive online algorithm.

In the orthogonal version, we have $K \cdot L$ types of cables, each defined by a length of time and a capacity. There are K lengths of time $\delta_1, \ldots, \delta_K \in \mathbb{N}$ with corresponding time scaling costs $\gamma_1, \ldots, \gamma_K \in \mathbb{Q}_+$, and L capacities $\phi_1, \ldots, \phi_L \in \mathbb{N}$ with corresponding capacity scaling costs $\mu_1, \ldots, \mu_L \in \mathbb{Q}_+$. A cable with length of time δ_k and capacity ϕ_ℓ for edge e costs $\gamma_k \cdot \mu_\ell \cdot d_e$. This problem was addressed in the offline setting by Anthony and Gupta[3] [4], and they gave a $O(K)$-approximation algorithm for the case with a single source (if $v_t = r$ for a fixed vertex r). For multiple sources, they gave a $O(K)$-approximation algorithm for the case in which the input metric is a tree[4]; this yields a randomized $O(K \lg n)$-approximation for arbitrary metrics by Lemma 10.5

[2] In the literature, it is usually known as **single-sink** BABND, but we opt for a unified nomenclature among the problems we study.

[3] They define the problem in terms of a sub-additive capacity scaling cost function; as we mentioned in Chapter 8, this is equivalent to our definition up to a constant cost factor.

[4] Since this problem on a single edge corresponds to O2DPP, this turns out to be a $O(K)$-approximation for O2DPP.

and Theorem 10.6. By combining the algorithm by Koufogiannakis and Young [52] with Lemma 10.5 and Theorem 10.6, we obtain a $O(KL \lg |V|)$-competitive online algorithm.

10.6 Summary

In Table 10.1, we summarize known approximation and competitive online results about the network leasing problems we study in this chapter.

problem	offline setting		online setting	
	single-source	multi-source	single-source	multi-source
SLe	$O(K)$ [4]	$O(\lg n)$ [59]	deterministic $O(K \lg n)$ [9]	randomized $O(\lg K \lg \lvert V\rvert)$ [59]
SNLe	$O(K)$ [4]	$O(\lg n)$ (Sec. 10.3)	$O(\lg K \lg \lvert V\rvert)$ (Sec. 10.3)	
LeRoB	$O(K)$ [4]	$O(\lg n)$ (Sec. 10.4)	$O(K \lg \lvert V\rvert)$ (Sec. 10.4), [27, 52]	
OLeBaBND	$O(K)$ [4]	$O(K \lg n)$ [4]	$O(KL \lg \lvert V\rvert)$ [27, 52]	
HLeBaBND	$O(\lg n)$ (Sec. 10.5)		$O(K \lg \lvert V\rvert)$ (Sec. 10.5), [27, 52]	
LeBaBND	pseudo-$O(\lg n)$ (Sec. 10.5)		$O(K \lg \lvert V\rvert)$ [27, 52]	
	$O(K \lg n)$ [27, 52]			

Table 10.1: Summary of known approximation and competitive online results for network leasing problems. A "pseudo-α" means a pseudo-polynomial α-approximation (α-competitive) algorithm.

In Figure 10.1, we extend the dependency graph in Figure 9.1 to include the problems we discuss in this chapter.

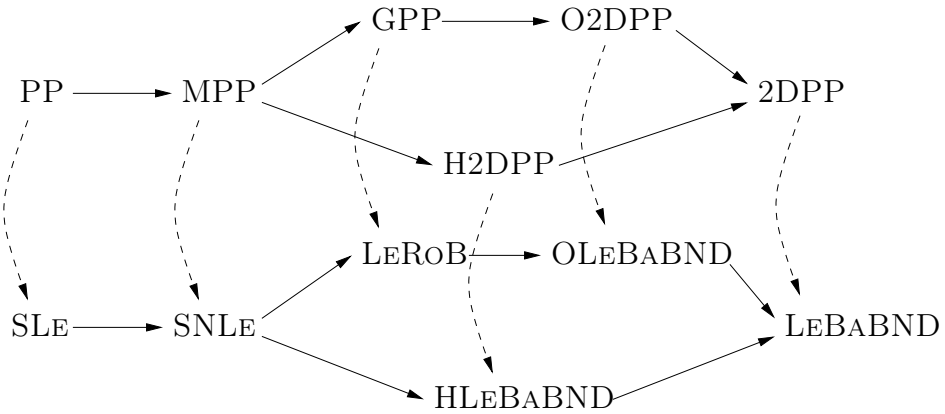


Figure 10.1: An extension of the dependency graph of Figure 9.1 to include the network leasing problems we discuss in this chapter. A full arrow $A \to B$ indicates that problem A is a particular case of problem B. A dashed arrow $A \dashrightarrow B$ indicates that parking permit problem A is a particular case of network leasing problem B, and B reduces to solving A for each edge if the input metric is a tree.

Part III

Facility Leasing Problems

Chapter 11

Facility Location, Sometimes with Penalties

In the following chapters, we are interested in variations of the (uncapacitated) **facility location problem** (FL), which is formally defined below.

Problem FL(G, d, F, γ, D): *The input consists of a complete graph $G = (V, E)$, a function $d : V \times V \mapsto \mathbb{Q}_+$ that assigns distances between vertices, a set $F \subseteq V$ of potential facilities, a function $\gamma : F \mapsto \mathbb{Q}_+$ that assigns an opening cost to each potential facility, and a set $D \subseteq V$ of clients. The goal is to obtain a set $X \subseteq F$ of open facilities and a function $a : D \mapsto X$ that assigns each client to an open facility, and we wish to minimize*

$$\sum_{f \in X} \gamma_f + \sum_{j \in D} d(j, a(j)).$$

This is an NP-hard problem. If no property is assumed on the distance function d, then there is a $O(\lg n)$-approximation algorithm, where $n := |D|$ is the number of clients, and this is the best possible unless P = NP, since there is an approximation-preserving reduction from the set cover problem to FL [40]. However, if (V, d) is a metric, then the problem can be approximated within a constant factor. Currently there is a 1.488-approximation algorithm by Li [54], and there is no algorithm with approximation factor smaller than 1.463, unless P = NP [68]. For the remainder of the text, we assume that d is metric.

11.1 A Simple 3-Approximation Algorithm

In this section we review a simple 3-approximation algorithm for FL, proposed by Jain and Vazirani [46]. The algorithm is based on the following formulation of the problem as an integer linear program.

$$\begin{array}{rrcll} \text{minimize} & \sum_{f \in F} \gamma_f \cdot y_f + \sum_{j \in D} \sum_{f \in F} d(j, f) \cdot x_{jf} & & & \\ \text{subject to} & \sum_{f \in F} x_{jf} & \geq & 1 & \forall j \in D, \\ & x_{jf} & \leq & y_f & \forall j \in D, f \in F, \\ & x_{jf}, y_f & \in & \{0, 1\} & \forall j \in D, f \in F. \end{array}$$

In this formulation, the binary variable y_f indicates whether facility f is open, and the binary variable x_{jf} indicates whether client j is served by facility f. The first constraint ensures that each client is connected to at least one facility, and the second constraint ensures that clients are connected to open facilities.

The dual of the relaxation of this problem consists in the following.

$$\begin{array}{lrcll} \text{maximize} & \sum_{j \in D} \alpha_j & & & \\ \text{subject to} & \sum_{j \in D} \beta_{jf} & \leq & \gamma_f & \forall f \in F, \\ & \alpha_j - \beta_{jf} & \leq & d(j,f) & \forall j \in D, f \in F, \\ & \alpha_j, \beta_{jf} & \geq & 0 & \forall j \in D, f \in F. \end{array}$$

Given a rational number x, let $(x)_+ := \max\{x, 0\}$. The following is an equivalent simplified formulation of the dual of the relaxation. The equivalence is due to the fact that the second constraint ensures that $\alpha_j - d(j,f) \leq \beta_{jf}$.

$$\begin{array}{lrcll} \text{maximize} & \sum_{j \in D} \alpha_j & & & \\ \text{subject to} & \sum_{j \in D} (\alpha_j - d(j,f))_+ & \leq & \gamma_f & \forall f \in F, \\ & \alpha_j & \geq & 0 & \forall j \in D. \end{array}$$

This linear program has the following economic interpretation: each client j is willing to pay α_j to be connected to some facility. A fraction of this value covers the distance between the client and the facility; the other portion is a contribution on the cost of opening the facility.

The algorithm proposed by Jain and Vazirani is inspired in both primal and dual formulations of the problem; thus, it is called a **primal-dual** algorithm. Informally, the algorithm increases uniformly each variable of the dual problem until some dual constraint is tight, and the primal variable corresponding to that constraint is set to 1. The dual variables that contribute to the tight constraint stop increasing, and the process continues for the remaining variables until all dual variables contribute to some tight constraint. At the end of the process, we obtain feasible primal and dual solutions. Due to weak duality, the cost of the dual solution is a lower bound on the cost of an optimum primal solution. For the facility location problem, however, the cost of the primal solution obtained cannot be bound from the cost of the dual solution. This can be overcome by a simple "clean-up" phase that removes some of the facilities, and the cost of the solution using the remaining facilities can be bound by a constant times the cost of the dual solution.

Now we describe the algorithm in more detail. A pseudocode is presented in Algorithm 11.1. The algorithm maintains a dual variable α_j for each client j, a set X of temporarily open facilities, and a set S of the clients whose dual variable still is being increased, which initially is the whole set of clients D. We say that client j **reaches** facility f if $\alpha_j \geq d(j,f)$. Increasing pauses when: (a) a client reaches an already temporarily open facility, or (b) the sum of the contributions towards a facility pays for its opening cost. We then add to X the facilities that reach condition (b), and remove from S the clients that reach some temporarily open facility. We proceed increasing the remaining

dual variables until S becomes empty.

After this phase, we build an interference graph G_X between the facilities in X. Graph G_X has vertex set X and has an edge between facilities f and f' if there is some client that reaches both f and f'. Then, we compute greedily a maximal independent set X' in G_X; this will be our final set of open facilities. Each client will be served by the closest facility in X'.

Input: (G, d, F, γ, D)
$X \leftarrow \emptyset$, $S \leftarrow D$, $\alpha_j \leftarrow 0$ for every $j \in D$;
while $S \neq \emptyset$ **do**
  increase α_j uniformly for every $j \in S$ **until**
    (a) $\alpha_j = d(j, f)$ for some $j \in S$ and $f \in X$, **or**
    (b) $\gamma_f = \sum_{j \in D} (\alpha_j - d(j, f))_+$ for some $f \in F \setminus X$;
  $X \leftarrow X \cup \{f \in F \setminus X : f \text{ satisfies (b)}\}$;
  $S \leftarrow S \setminus \{j \in S : j \text{ reaches some } f \in X\}$;
build the graph G_X with
  $V[G_X] \leftarrow X$,
  $E[G_X] \leftarrow \{(f, f') : \exists j \in D : j \text{ reaches both } f \text{ and } f'\}$;
build a maximal independent set X' of G_X;
foreach $j \in D$ **do**
  $a(j) \leftarrow \arg\min_{f' \in X'} \{d(j, f')\}$;
return (X', a);

Algorithm 11.1: Primal-dual algorithm for FL [46].

Theorem 11.1 (Jain and Vazirani [46]): *Algorithm 11.1 is a 3-approximation to* FL.

Proof: First note that, since conditions (a) and (b) correspond to the constraints of the dual program, we have that α is a feasible dual solution. Therefore, by weak duality, we have that

$$\sum_{j \in D} \alpha_j \leq \text{opt}(G, d, F, \gamma, D).$$

We show that the cost of the primal solution (X', a) returned by the algorithm is at most 3 times the cost of the dual solution, and thus the algorithm is a 3-approximation to FL.

For every client $j \in D$, we define numbers α_j^C and α_j^F, in the following manner:

1. If j reaches some $f \in X'$, then let

$$\alpha_j^C := d(j, f), \quad \alpha_j^F := \alpha_j - d(j, f);$$

2. Else, let

$$\alpha_j^C := \alpha_j, \quad \alpha_j^F := 0.$$

Note that, either case, we have that

$$\alpha_j = \alpha_j^C + \alpha_j^F.$$

First we bound the facility opening cost. Note that, by construction, each client reaches at most one facility in X'. Also, by case (b) of the algorithm, the opening cost of each $f \in X'$ is totally paid by contributions from clients that reach f. Therefore, we have that

$$\sum_{f \in X'} \gamma_f = \sum_{j \in D} \alpha_j^F.$$

Now we bound the client connection cost. Let $D_{X'}$ be the set of clients that reach some facility in X'. For each $j \in D_{X'}$, let $f \in X'$ be the facility reached by j. Note that j is connected to the closest facility in X', so

$$d(j, a(j)) \leq d(j, f) = \alpha_j^C.$$

Now let j be some client that reaches some $f \in X$ but does not reach any facility in X'. There must be some $f' \in X'$ and some $j' \in D$ that reaches both f and f', by construction of X'. We must have that $\alpha_j \geq \alpha_{j'}$ since, when $\alpha_{j'}$ stopped increasing, j' had reached both f and f', and j had reached f when α_j stopped increasing. Since j' reaches both f and f', we have that

$$\alpha_{j'} \geq d(j', f) \quad \text{and} \quad \alpha_{j'} \geq d(j', f').$$

Thus, by triangle inequality, we have that

$$d(j, a(j)) \leq d(j, f') \leq d(j, f) + d(j', f) + d(j', f') \leq \alpha_j + \alpha_{j'} + \alpha_{j'} \leq 3 \cdot \alpha_j = 3 \cdot \alpha_j^C.$$

Summing up the previous inequalities, we have that

$$\sum_{f \in X'} \gamma_f + \sum_{j \in D} d(j, a(j)) \leq \sum_{j \in D} \alpha_j^F + 3 \cdot \sum_{j \in D} \alpha_j^C \leq 3 \cdot \sum_{j \in D} \alpha_j \leq 3 \cdot \text{opt}(G, d, F, \gamma, D).$$

□

There is a family of examples which shows that this bound is tight; i.e., the cost of the solution returned by the algorithm is close to 3 times the cost of the optimum solution. The metric is obtained by taking shortest paths in the graph of Figure 11.1. We have $F = \{f_1, f_2\}$, $\gamma(f_1) = \epsilon$, $\gamma(f_2) = (n+1)\epsilon$, and $D = \{c_1, c_2, c_3, \ldots, c_n\}$. The optimum solution opens f_2 and connects all the clients to it, for a total cost of $n + (n+1)\epsilon$. Now consider the behavior of Algorithm 11.1. The algorithm raises all dual variables up to $1+\epsilon$, when c_1 reaches f_1 and f_1 is temporarily open. Then the algorithm raises $\alpha_{c_2}, \ldots, \alpha_{c_n}$ up to $1 + \epsilon + \epsilon/n$, and temporarily opens f_2 too. Note that c_1 now reaches both f_1 and f_2, which are thus adjacent in G_X. We can force the algorithm to choose f_1 when building independent set X', obtaining a solution of cost $3(n-1) + 1 + \epsilon$.

Note that most of the cost of the solution lies in the client connection cost. There are other techniques that impose a greater penalty on the facility opening cost, and thus reduce the client connection cost. One example is the **dual-fitting** algorithm by Jain, Mahdian, Markakis, Saberi, and Vazirani [45], which only opens a facility when the contributions pay a factor greater than 1 of the opening cost, and has approximation

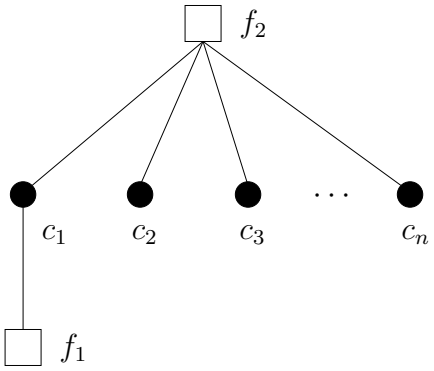


Figure 11.1: Example which shows that the analysis of Algorithm 11.1 is tight.

factor 1.61. (The algorithm has other details that do not matter to our discussion.)

The current best algorithm for FL is a 1.488-approximation [54]. We do not present this algorithm here since we do not use any of its techniques in this thesis. Instead, we use this result as a black box in other sections.

11.2 Facility Location with Penalties

One simple generalization of FL is if we allow a client j not to be connected to any facility if we pay a penalty $\pi_j \in \mathbb{Q}_+$. This is called the **facility location problem with penalties** (FLP), which we define formally below.

Problem FLP$(G, d, F, \gamma, D, \pi)$: *The input consists of a complete graph $G = (V, E)$, a metric distance function $d : V \times V \mapsto \mathbb{Q}_+$ between the vertices, a set $F \subseteq V$ of potential facilities, a function $\gamma : F \mapsto \mathbb{Q}_+$ that assigns an opening cost to each potential facility, a set $D \subseteq V$ of clients, and a function $\pi : D \mapsto \mathbb{Q}_+$ assigning a penalty to each client. The goal is to obtain a set $X \subseteq F$ of open facilities and a function $a : D \mapsto X \cup \{\mathsf{null}\}$ that assigns each client to an open facility or to* null*, and we wish to minimize*

$$\sum_{f \in X} \gamma_f + \sum_{\substack{j \in D \\ a(j) \neq \mathsf{null}}} d(j, a(j)) + \sum_{\substack{j \in D \\ a(j) = \mathsf{null}}} \pi_j.$$

FL reduces to FLP if we set $\pi_j = \infty$ for all $j \in D$, so FLP is also NP-hard.

In this section we present a 3-approximation primal-dual algorithm for FLP, which is very similar to Algorithm 11.1 for FL. This algorithm was proposed by Charikar, Khuller, Mount and Narasimhan [14].

The integer linear program formulation is very similar to that of FL.

$$\begin{array}{rrcll} \text{minimize} & \sum_{f \in F} \gamma_f \cdot y_f + \sum_{j \in D} \sum_{f \in F} d(j,f) \cdot x_{fj} + \sum_{j \in D} \pi_j \cdot z_j & & & \\ \text{subject to} & \sum_{f \in F} x_{jf} + z_j & \geq & 1 & \forall j \in D, \\ & x_{jf} & \leq & y_f & \forall j \in D, f \in F, \\ & x_{jf}, y_f, z_j & \in & \{0,1\} & \forall j \in D, f \in F. \end{array}$$

The difference is that we have an extra variable z_j for each client j, which indicates

whether we pay the penalty for this client. In the first constraint, we have an extra option: not to connect a client to any facility, but to pay for its penalty. The objective function also includes the penalty cost of those clients. The simplified version of the dual of the relaxation of this program is the following.

$$\begin{array}{lrcll} \text{maximize} & \sum_{j \in D} \alpha_j & & & \\ \text{subject to} & \sum_{j \in D} (\alpha_j - d(j,f))_+ & \leq & \gamma_f & \forall f \in F, \\ & \alpha_j & \leq & \pi_j & \forall j \in D, \\ & \alpha_j & \geq & 0 & \forall j \in D. \end{array}$$

Note that we have an extra constraint, which can be interpreted as that a client will not pay a value higher than its penalty.

The pseudocode is presented in Algorithm 11.2. The difference is that we stop increasing the dual variable α_j for a client j when it exceeds its penalty π_j. Also, in Line 10, we ensure that such clients are not assigned to any facility. Again, we say that client j **reaches** facility f if $\alpha_j \geq d(j,f)$.

Input: $(G, d, F, \gamma, D, \pi)$
1 $X \leftarrow \emptyset$, $S \leftarrow D$, $\alpha_j \leftarrow 0$ for every $j \in D$;
2 **while** $S \neq \emptyset$ **do**
3 increase α_j uniformly for every $j \in S$ **until**
(a) $\alpha_j = d(j,f)$ for some $j \in S$ and $f \in X$, **or**
(b) $\gamma_f = \sum_{j \in D} (\alpha_j - d(j,f))_+$ for some $f \in F \setminus X$, **or**
(c) $\alpha_j = \pi_j$ for some $j \in S$;
4 $X \leftarrow X \cup \{f \in F \setminus X : f \text{ satisfies (b)}\}$;
5 $S \leftarrow S \setminus \{j \in S : \alpha_j \geq \pi_j$ **or** j reaches some $f \in X\}$;
6 build the graph G_X with
$V[G_X] \leftarrow X$,
$E[G_X] \leftarrow \{(f, f') : \exists j \in D : j \text{ reaches both } f \text{ and } f'\}$;
7 build a maximal independent set X' of G_X;
8 **foreach** $j \in D$ **do**
9 **if** j reaches some $f \in X$ **then** $a(j) \leftarrow \arg\min_{f' \in X'} \{d(j,f')\}$;
10 **else** $a(j) \leftarrow$ null;
11 **return** (X', a);

Algorithm 11.2: Primal-dual algorithm for FLP [14].

Theorem 11.2 (Charikar, Khuller, Mount and Narasimhan [14]): *Algorithm 11.2 is a 3-approximation to* FLP.

Proof: First note that, since conditions (a), (b) and (c) correspond to the constraints of the dual program, we have that α is a feasible dual solution. Therefore, by weak duality, we have that

$$\sum_{j \in D} \alpha_j \leq \text{opt}(G, d, F, \gamma, D, \pi).$$

We show that the cost of the primal solution (X', a) returned by the algorithm is at

most 3 times the cost of the dual solution, and thus the algorithm is a 3-approximation to FLP.

For every client $j \in D$, we define numbers α_j^C, α_j^F, and α_j^P in the following manner:

1. If j reaches some $f \in X'$, then let

$$\alpha_j^C := d(j, f), \quad \alpha_j^F := \alpha_j - d(j, f), \quad \alpha_j^P := 0;$$

2. If j does not reach any facility in X' but reaches some $f \in X$, then let

$$\alpha_j^C := \alpha_j, \quad \alpha_j^F := 0, \quad \alpha_j^P := 0;$$

3. Finally, if j does not reach any facility in X, let

$$\alpha_j^C := 0, \quad \alpha_j^F := 0, \quad \alpha_j^P := \alpha_j.$$

Note that, either case, we have that

$$\alpha_j = \alpha_j^C + \alpha_j^F + \alpha_j^P.$$

First we bound the facility opening cost. Note that, by construction, each client reaches at most one facility in X'. Also, by case (b) of the algorithm, the opening cost of each $f \in X'$ is totally paid by contributions from clients that reach f. Therefore, we have that

$$\sum_{f \in X'} \gamma_f = \sum_{j \in D} \alpha_j^F.$$

Now we bound the penalty cost. We have that a client j has $a(j)$ set to null if, and only if, it does not reach any facility in X, in which case $\alpha_j = \alpha_j^P$. Also, due to case (c) of the algorithm, we have that $\alpha_j = \pi_j$. Thus, it is straightforward to conclude that

$$\sum_{\substack{j \in D \\ a(j) = \mathsf{null}}} \pi_j = \sum_{j \in D} \alpha_j^P.$$

Finally, we bound the client connection cost. Let $D_{X'}$ be the set of clients that reach some facility in X'. For each $j \in D_{X'}$, let $f \in X'$ be the facility reached by j. Note that j is connected to the closest facility in X', so

$$d(j, a(j)) \leq d(j, f) = \alpha_j^C.$$

Now let j be some client that reaches some $f \in X$ but does not reach any facility in X'. There must be some $f' \in X'$ and some $j' \in D$ that reaches both f and f', by construction of X'. We must have that $\alpha_j \geq \alpha_{j'}$ since, when $\alpha_{j'}$ stopped increasing, j' had reached both f and f', and j had reached f when α_j stopped increasing. Since j' reaches both f

and f', we have that

$$\alpha_{j'} \geq d(j', f) \quad \text{and} \quad \alpha_{j'} \geq d(j', f').$$

Thus, by triangle inequality, we have that

$$d(j, a(j)) \leq d(j, f') \leq d(j, f) + d(j', f) + d(j', f') \leq \alpha_j + \alpha_{j'} + \alpha_{j'} \leq 3 \cdot \alpha_j = 3 \cdot \alpha_j^C.$$

Summing up the previous inequalities, we have that

$$\begin{aligned} & \sum_{f \in X'} \gamma_f + \sum_{\substack{j \in D \\ a(j) \neq \mathsf{null}}} d(j, a(j)) + \sum_{\substack{j \in D \\ a(j) = \mathsf{null}}} \pi_j \\ \leq \; & \sum_{j \in D} \alpha_j^F + 3 \cdot \sum_{j \in D} \alpha_j^C + \sum_{j \in D} \alpha_j^P \\ \leq \; & 3 \cdot \sum_{j \in D} \alpha_j \\ \leq \; & 3 \cdot \mathrm{opt}(G, d, F, \gamma, D). \end{aligned}$$

□

The same example from the previous section shows that this analysis is tight.

Currently, the best algorithm for FLP is a 1.5148-approximation, by Li, Du, Xiu, and Xu [56]. This algorithm is based on an LP-rounding technique, but we do not describe it here since we do not use any of its techniques in this thesis. The authors also have the following general result, which we do not prove here.

Theorem 11.3 (Li, Du, Xiu, and Xu [56]): *For any covering problem that admits an α-approximation algorithm, the corresponding problem with* **submodular** *penalties admits a $(1 - e^{-1/\alpha})^{-1}$-approximation algorithm.*

Note that the facility location problem with submodular penalties is more general than FLP, which the authors call the facility location problem with *linear* penalties.

11.3 Online Facility Location, with or without Penalties

In the online version of FL, only the set of potential facilities F and opening costs γ are known in advance. At each instant of time, we receive a client j, along with the distance between j and each potential facility in F. Then, we must decide if we open some new facilities, and we must assign an open facility to client j. We cannot close previously open facilities nor change the assignment of previous clients. The goal is to minimize the cost of the final FL solution.

This problem admits $O(\lg n / \lg \lg n)$-competitive online algorithms [31, 58], and any online algorithm has competitive factor $\Omega(\lg n / \lg \lg n)$ [31], where $n := |D|$. We do not prove these results, but we use them as black boxes.

In the online version of FLP, penalties are received in an online fashion as well. I.e., we known F and γ in advance, and at each instant we receive a client j along with π_j and the distance between j and each $f \in F$. We cannot close previously open facilities nor change the assignment of previous clients. The goal is to minimize the cost of the final FLP solution. This problem admits $O(\lg n)$-competitive online algorithms [26, 62]. The $\Omega(\lg n / \lg \lg n)$ lower bound for FL also applies to this problem, since FL is a particular case of FLP. Thus, it is an open question to find a $O(\lg n / \lg \lg n)$-competitive online algorithm or a $\Omega(\lg n)$ lower bound for FLP.

Chapter 12

Facility Leasing, Sometimes with Penalties

In this chapter we present leasing variants of FL and FLP.

12.1 Facility Leasing

In the **facility leasing problem** (FLE), clients are distributed along time, and instead of opening facilities permanently, we may lease each facility for one of K different durations $\delta_1, \ldots, \delta_K$. The cost for leasing a facility f for δ_k units of time is $\gamma_k^f \in \mathbb{Q}_+$; it depends on the facility position, as in FL, but also on leasing type k. We may assume that leasing costs respect economies of scale, i.e., it is more cost-effective to lease facilities for longer periods. We wish to select a set of facility leases that serve the clients and minimizes the leasing costs plus the sum of the distances from each client to the facility lease that serves it. We define the problem formally below.

Problem FLE$(G, d, F, K, \delta, \gamma, D_0, \ldots, D_{T-1})$: *The input consists of a complete graph $G = (V, E)$, a metric distance function $d : V \times V \mapsto \mathbb{Q}_+$, a set $F \subseteq V$ of potential facilities, an integer $K > 0$ that represents the number of lease types, a function $\delta : [K] \mapsto \mathbb{N}$ that maps each leasing type to a length of time, a cost $\gamma_k^f \in \mathbb{Q}_+$ for leasing facility $f \in F$ with leasing type $k \in [K]$, and a sequence $D_0, \ldots, D_{T-1} \subseteq V$ of clients. Let $\mathcal{D} := \{(j, t) : j \in D_t, t \in \{0, \ldots, T-1\}\}$ be the set of client requests. A solution consists of a set $X \subseteq \mathcal{F} := F \times [K] \times \mathbb{Z}_+$ of facility leases, and a function $a : \mathcal{D} \mapsto X$ that maps each client request (j, t) to some $(f, k, \hat{t}) \in X$ such that $t \in [\hat{t}, \hat{t} + \delta_k)$. The goal is to find a solution (X, a) which minimizes*

$$\sum_{(f,k,\hat{t}) \in X} \gamma_k^f + \sum_{(j,t) \in \mathcal{D}} d(j, a(j, t)).$$

Note that FL reduces to FLE if $K = 1$ and $\delta_1 = \infty$, so FLE is also NP-hard. This problem has a 3-approximation primal-dual algorithm due to Nagarajan and Williamson [60], which we present in the sequel. It is based on the algorithm presented in Section 11.1.

In this chapter, we use the following notation, which avoids clutter. (The notation

above, however, will be used again in Chapter 14 when we discuss connected facility leasing problems.) We denote a client request by a pair $j = (p_j, t_j)$, where $p_j \in V$ is a vertex and $t_j \in \{0, \ldots, T-1\}$ is the instant of time j arrives. We also denote a facility lease by a triple $f = (p_f, k_f, t_f)$, where $p_f \in F$ is a potential facility, $k_f \in [K]$ is a leasing type, and $t_f \in \mathbb{Z}_+$ is the instant of time the leasing begins. We then write δ_f instead of $\delta(k_f)$, and γ_f instead of $\gamma_{k_f}^{p_f}$.

We say that a facility lease f **covers** client request j if $t_j \in [t_f, t_f + \delta_f)$. In order to simplify notation, for $f \in \mathcal{F}$ and $j \in \mathcal{D}$, we define the distance from j to f as

$$d(j, f) := \begin{cases} d(p_j, p_f) & \text{if } t_j \in [t_f, t_f + \delta_f), \\ \infty & \text{otherwise.} \end{cases} \tag{12.1}$$

I.e., the distance is infinite if f does not cover t_j.

The problem has the following integer linear program formulation.

$$\begin{array}{rrcll} \text{minimize} & \sum_{f\in\mathcal{F}} \gamma_f \cdot y_f + \sum_{j\in\mathcal{D}} \sum_{f\in\mathcal{F}} d(j,f) \cdot x_{jf} & & & \\ \text{subject to} & \sum_{f\in\mathcal{F}} x_{jf} & \geq & 1 & \forall j \in \mathcal{D}, \\ & x_{jf} & \leq & y_f & \forall j \in \mathcal{D}, f \in \mathcal{F}, \\ & x_{jf}, y_f & \in & \{0,1\} & \forall j \in \mathcal{D}, f \in \mathcal{F}. \end{array}$$

Note that this formulation is almost identical to the one of FL, except that Equation (12.1) forces facility leases to serve only client requests they cover. Again, we consider a simplified form of the dual of the relaxation of this program.

$$\begin{array}{rrcll} \text{maximize} & \sum_{j\in\mathcal{D}} \alpha_j & & & \\ \text{subject to} & \sum_{h\in\mathcal{D}} (\alpha_h - d(j,f))_+ & \leq & \gamma_f & \forall f \in \mathcal{F}, \\ & \alpha_j & \geq & 0 & \forall j \in \mathcal{D}. \end{array}$$

We present a pseudocode in Algorithm 12.1. We say that client request $j \in \mathcal{D}$ **reaches** facility lease $j \in \mathcal{F}$ if $\alpha_j \geq d(j, f)$. The first phase of the algorithm is identical to that of Algorithm 11.1. When we build the independent set X' of the interference graph, however, we prioritize facility leases of larger length of time. This ensures that:

1. Every client request reaches at most one facility lease in X';
2. If f and f' in X are reached by the same client request j, and if $f' \in X'$, then $\delta_f \leq \delta_{f'}$.

Note that there may be some client request j that reaches some f in X but is not covered by any facility lease in X'. However, remember that some $f' \in X'$ shares a reaching client with f, thus $\delta_f \leq \delta_{f'}$ and the intervals covered by facility leases f and f' overlap. Then, we buy $\hat{X}$, which leases $p_{f'}$ three times, at instants $t_{f'} - \delta_{f'}$, $t_{f'}$ and $t_{f'} + \delta_{f'}$. Thus, the interval formed by those three facility leases, which is $[t_{f'} - \delta_{f'}, t_{f'} + 2\delta_{f'})$, is a superset of interval $[t_f, t_f + \delta_f)$, and therefore one of the facility leases covers j. This is illustrated in Figure 12.1.

Input: $(G, d, F, K, \delta, \gamma, \delta, D_0, \ldots, D_{T-1})$
$X \leftarrow \emptyset$, $S \leftarrow \mathcal{D}$, $\alpha_j \leftarrow 0$ for every $j \in \mathcal{D}$;
while $S \neq \emptyset$ **do**
  increase α_j uniformly for every $j \in S$ **until**
    (a) $\alpha_j = d(j, f)$ for some $j \in S$ and $f \in X$, **or**
    (b) $\gamma_f = \sum_{j \in \mathcal{D}} (\alpha_j - d(j, f))_+$ for some $f \in \mathcal{F} \setminus X$;
  $X \leftarrow X \cup \{f \in \mathcal{F} \setminus X : f$ satisfies (b)$\}$;
  $S \leftarrow S \setminus \{j \in S : j$ reaches some $f \in X\}$;
build the graph G_X with
  $V[G_X] \leftarrow X, \quad E[G_X] \leftarrow \{(f, f') : \exists j \in D : j$ reaches both f and $f'\}$;
build a maximal independent set X' in G_X greedily in non-increasing order of δ;
$\hat{X} \leftarrow \{(p_{f'}, k_{f'}, t_{f'} - \delta_{f'}), f', (p_{f'}, k_{f'}, t_{f'} + \delta_{f'}) : f' \in X'\}$;
foreach $j \in \mathcal{D}$ **do**
  $a(j) \leftarrow \arg\min_{f' \in \hat{X}} \{d(j, f')\}$;
return $(\hat{X}, a)$;

Algorithm 12.1: Primal-dual algorithm for FLE [60].

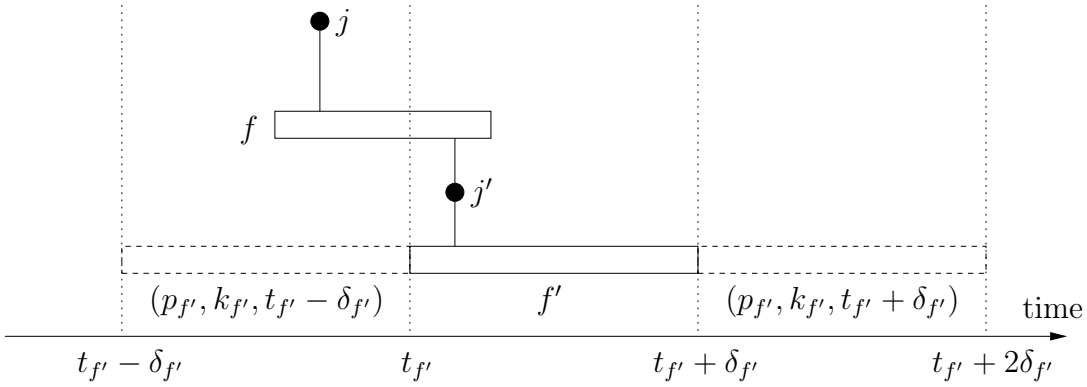


Figure 12.1: An illustration of why Algorithm 12.1 produces a feasible solution. (Reproduced from [60] with permission.)

Theorem 12.1 (Nagarajan and Williamson [60]): *Algorithm 12.1 is a 3-approximation to* FLE.

Proof: First note that, since conditions (a) and (b) correspond to the constraints of the dual program, we have that α is a feasible dual solution. Therefore, by weak duality, we have that

$$\sum_{j \in \mathcal{D}} \alpha_j \leq \text{opt}(G, d, F, K, \delta, \gamma, \delta, D_0, \ldots, D_{T-1}).$$

We show that the cost of the primal solution $(\hat{X}, a)$ returned by the algorithm is at most 3 times the cost of the dual solution, and thus the algorithm is a 3-approximation to FLE.

For every client request $j \in \mathcal{D}$, we define numbers α_j^C and α_j^F, in the following manner:

1. If j reaches some $f \in X'$, then let
$$\alpha_j^C := d(j, f), \quad \alpha_j^F := \alpha_j - d(j, f);$$
2. Else, let
$$\alpha_j^C := \alpha_j, \quad \alpha_j^F := 0.$$

Note that, either case, we have that

$$\alpha_j = \alpha_j^C + \alpha_j^F.$$

First we bound the facility leasing cost. Note that, by construction, each client request reaches at most one facility lease in X'. Also, by case (b) of the algorithm, the leasing cost of each $f \in X'$ is totally paid by contributions from clients that reach it. Therefore, we have that

$$\sum_{f \in X'} \gamma_f = \sum_{j \in \mathcal{D}} \alpha_j^F.$$

Since $\hat{X}$, which is the set of facility leases actually bought by the algorithm, consists of three copies of each facility lease in X', we have that

$$\sum_{f \in \hat{X}} \gamma_f \leq 3 \cdot \sum_{j \in \mathcal{D}} \alpha_j^F.$$

Now we bound the client connection cost. Let $\mathcal{D}_{X'}$ be the set of client requests that reach some facility lease in X'. For each $j \in \mathcal{D}_{X'}$, let $f \in X'$ be the facility lease reached by j. Note that j is connected to the closest active facility lease in $\hat{X}$, so

$$d(j, a(j)) \leq d(j, f) = \alpha_j^C.$$

Now let j be some client request that reaches some $f \in X$ but does not reach any facility lease in X'. There must be some $f' \in X'$ and some $j' \in \mathcal{D}$ that reaches both f and f', by construction of X'. We must have that $\alpha_j \geq \alpha_{j'}$ since, when $\alpha_{j'}$ stopped increasing, j' had reached both f and f', and j had reached f when α_j stopped increasing. Since j' reaches both f and f', we have that

$$\alpha_{j'} \geq d(j', f) \quad \text{and} \quad \alpha_{j'} \geq d(j', f').$$

Thus, by triangle inequality, we have that

$$d(j, a(j)) \leq d(j, f') \leq d(j, f) + d(j', f) + d(j', f') \leq \alpha_j + \alpha_{j'} + \alpha_{j'} \leq 3 \cdot \alpha_j = 3 \cdot \alpha_j^C.$$

Summing up the previous inequalities, we have that

$$\begin{aligned}\sum_{f\in\hat{X}}\gamma_f+\sum_{j\in\mathcal{D}}d(j,a(j)) &\leq 3\cdot\sum_{j\in\mathcal{D}}\alpha_j^F+3\cdot\sum_{j\in\mathcal{D}}\alpha_j^C\\ &= 3\cdot\sum_{j\in\mathcal{D}}\alpha_j\\ &\leq 3\cdot\mathrm{opt}(G,d,F,K,\delta,\gamma,\delta,D_0,\ldots,D_{T-1}).\end{aligned}$$

□

Note that the same example of Figure 11.1 shows that this analysis is tight. Indeed, this algorithm cannot be improved with the dual-fitting technique, since we have a factor of 3 both on facility leasing and client connection costs.

For the online version of FLE, we know in advance G, F, K, δ and γ. At each instant of time t, we receive a set of clients D_t along with the distances between each $j \in D_t$ and each $f \in F$. We must decide if we acquire some new facility leases, and we must assign an active facility lease to each client request in D_t. We cannot remove previous facility leases nor change the assignment of previous clients. The goal is to minimize the cost of the final FLE solution. This problem admits a $\mathrm{O}(K \lg n/\lg\lg n)$-competitive online algorithm [53], and a $\mathrm{O}(\delta_K \lg \delta_K)$-competitive online algorithm [1], whose competitive factor is independent of the time dimension. The problem has a lower bound of $\Omega(\lg K + \lg n'/\lg\lg n')$, where $n' := \min\{n, \delta_K\}$, due to the lower bounds for the online versions of PP [59] and FL [31]. Even if it is not the best online algorithm, in this text we use as a black box the algorithm by Nagarajan and Williamson, which is $\mathrm{O}(K \lg n)$-competitive [60]; this is sufficient for our purposes.

12.2 Facility Leasing with Penalties

In this section, we study the combination of FLP and FLE. As in FLE, instead of opening facilities that last for the whole planning horizon, we consider K facility leasing types. Also, as in FLP, we may choose not to connect a client request (j,t) to any facility lease by paying a penalty π_{jt}. We define the problem formally below, which we call the **facility leasing problem with penalties** (FLEP).

Problem FLEP$(G,d,F,K,\delta,\gamma,D_0,\ldots,D_{T-1},\pi)$: *We are given a complete graph $G=(V,E)$, a metric distance function $d: V\times V\mapsto\mathbb{Q}_+$, a set $F\subseteq V$ of potential facilities, an integer $K>0$ that represents the number of lease types, a function $\delta:[K]\mapsto\mathbb{N}$ that maps each leasing type to a length of time, a cost $\gamma_k^f\in\mathbb{Q}_+$ for leasing facility $f\in F$ with leasing type $k\in[K]$, and a sequence $D_0,\ldots,D_{T-1}\subseteq V$ of clients. Let $\mathcal{D}:=\{(j,t): j\in D_t, t\in\{0,\ldots,T-1\}\}$ be the set of client requests. We are also given a function $\pi:\mathcal{D}\mapsto\mathbb{Q}_+$ that assigns a penalty to each client request. A solution consists of a set $X\subseteq\mathcal{F}:=F\times[K]\times\mathbb{Z}_+$ of facility leases, and a function $a:\mathcal{D}\mapsto X\cup\{\mathsf{null}\}$ that maps each client request (j,t) to* null *or to some $(f,k,\hat{t})\in X$ such that $t\in[\hat{t},\hat{t}+\delta_k)$. The*

goal is to find a solution (X, a) *which minimizes*

$$\sum_{(f,k,\hat{t})\in X} \gamma_k^f + \sum_{\substack{(j,t)\in\mathcal{D}\\ a(j,t)\neq\mathsf{null}}} d(j, a(j,t)) + \sum_{\substack{(j,t)\in\mathcal{D}\\ a(j,t)=\mathsf{null}}} \pi_{jt}.$$

This is an NP-hard problem since FLE reduces to FLEP if $\pi_{jt} = \infty$ for all $(j,t) \in \mathcal{D}$. In the following we present a 3-approximation algorithm for FLEP, which is based on Algorithms 11.2 and 12.1. This result was published in [21].

Again, in order to avoid clutter, we use the following simplified notation in this section, but the notation above will be used again in Chapter 14. We denote a client request by a triple $j = (p_j, t_j, \pi_j)$, where $p_j \in V$ is a vertex, $t_j \in \{0, \ldots, T-1\}$ is the instant of time j arrives and $\pi_j \in \mathbb{Q}_+$ is the penalty for not assigning a facility lease to j. We also denote a facility lease by a triple $f = (p_f, k_f, t_f)$, where $p_f \in F$ is a potential facility, $k_f \in [K]$ is a leasing type, and $t_f \in \mathbb{Z}_+$ is the instant of time the leasing begins. We then write δ_f instead of $\delta(k_f)$, and γ_f instead of $\gamma_{k_f}^{p_f}$. We also define distances between client requests and facility leases as in Equation (12.1).

The problem has the following integer linear program formulation.

$$\begin{array}{rrcll}
\text{minimize} & \sum_{f\in\mathcal{F}} \gamma_f \cdot y_f + \sum_{j\in\mathcal{D}} \sum_{f\in\mathcal{F}} d(j,f) \cdot x_{jf} + \sum_{j\in\mathcal{D}} \pi_j \cdot z_j & & & \\
\text{subject to} & \sum_{f\in\mathcal{F}} x_{jf} + z_j & \geq & 1 & \forall j \in \mathcal{D}, \\
& x_{jf} & \leq & y_f & \forall j \in \mathcal{D}, f \in \mathcal{F}, \\
& x_{jf}, y_f, z_j & \in & \{0,1\} & \forall j \in \mathcal{D}, f \in \mathcal{F}.
\end{array}$$

Note that the formulation follows from those of FLP and FLE. The simplified dual of the relaxation of this program is the following.

$$\begin{array}{rrcll}
\text{maximize} & \sum_{j\in\mathcal{D}} \alpha_j & & & \\
\text{subject to} & \sum_{j\in\mathcal{D}} (\alpha_j - d(j,f))_+ & \leq & \gamma_f & \forall f \in \mathcal{F}, \\
& \alpha_j & \leq & \pi_j & \forall j \in \mathcal{D}, \\
& \alpha_j & \geq & 0 & \forall j \in \mathcal{D}.
\end{array}$$

A pseudocode is presented in Algorithm 12.2. The main difference from Algorithm 12.1 is that we stop increasing the dual variable for j when $\alpha_j = \pi_j$. The feasibility of the returned solution follows from the same arguments for Algorithm 12.1.

Theorem 12.2: *Algorithm 12.2 is a 3-approximation to* FLEP.

Proof: First note that, since conditions (a), (b) and (c) correspond to the constraints of the dual program, we have that α is a feasible dual solution. Therefore, by weak duality, we have that $\sum_{j\in\mathcal{D}} \alpha_j \leq \text{opt}(G, d, F, K, \delta, \gamma, \delta, D_0, \ldots, D_{T-1}, \pi)$.

We show that the cost of the primal solution $(\hat{X}, a)$ returned by the algorithm is at most 3 times the cost of the dual solution, and thus the algorithm is a 3-approximation to FLEP.

Input: $(G, d, F, K, \delta, \gamma, \delta, D_0, \ldots, D_{T-1}, \pi)$

$X \leftarrow \emptyset$, $S \leftarrow \mathcal{D}$, $\alpha_j \leftarrow 0$ for every $j \in \mathcal{D}$;
while $S \neq \emptyset$ **do**
  increase α_j uniformly for every $j \in S$ **until**
    (a) $\alpha_j = d(j, f)$ for some $j \in S$ and $f \in X$, **or**
    (b) $\gamma_f = \sum_{j \in \mathcal{D}} (\alpha_j - d(j, f))_+$ for some $f \in \mathcal{F} \setminus X$, **or**
    (c) $\alpha_j = \pi_j$ for some $j \in S$;
  $X \leftarrow X \cup \{f \in \mathcal{F} \setminus X : f$ satisfies (b)$\}$;
  $S \leftarrow S \setminus \{j \in S : \alpha_j \geq \pi_j$ **or** j reaches some $f \in X\}$;
build the graph G_X with
  $V[G_X] \leftarrow X, \quad E[G_X] \leftarrow \{(f, f') : \exists j \in D : j$ reaches both f and $f'\}$;
build a maximal independent set X' in G_X greedily in non-increasing order of δ;
$\hat{X} \leftarrow \{(p_{f'}, k_{f'}, t_{f'} - \delta_{f'}), f', (p_{f'}, k_{f'}, t_{f'} + \delta_{f'}) : f' \in X'\}$;
foreach $j \in \mathcal{D}$ **do**
  if j reaches some $f \in X'$ **then** $a(j) \leftarrow \arg\min_{f' \in \hat{X}} \{d(j, f')\}$;
  else $a(j) \leftarrow$ null;
return $(\hat{X}, a)$;

Algorithm 12.2: Primal-dual algorithm for FLeP.

For every client request $j \in \mathcal{D}$, we define numbers α_j^C, α_j^F, and α_j^P in the following manner:

1. If j reaches some $f \in X'$, then let

$$\alpha_j^C := d(j, f), \quad \alpha_j^F := \alpha_j - d(j, f), \quad \alpha_j^P := 0;$$

2. If j does not reach any facility lease in X' but reaches some $f \in X$, then let

$$\alpha_j^C := \alpha_j, \quad \alpha_j^F := 0, \quad \alpha_j^P := 0;$$

3. Finally, if j does not reach any facility lease in X, let

$$\alpha_j^C := 0, \quad \alpha_j^F := 0, \quad \alpha_j^P := \alpha_j.$$

Note that, either case, we have that

$$\alpha_j = \alpha_j^C + \alpha_j^F + \alpha_j^P.$$

First we bound the facility leasing cost. Note that, by construction, each client request reaches at most one facility lease in X'. Also, by case (b) of the algorithm, the leasing cost of each $f \in X'$ is totally paid by contributions from clients that reach it. Therefore, we have that

$$\sum_{f \in X'} \gamma_f = \sum_{j \in \mathcal{D}} \alpha_j^F.$$

Since $\hat{X}$, which is the set of facility leases actually bought by the algorithm, consists of

three copies of each facility lease in X', we have that

$$\sum_{f \in \hat{X}} \gamma_f \leq 3 \cdot \sum_{j \in \mathcal{D}} \alpha_j^F.$$

Now we bound the penalty cost. We have that a client request j has $a(j)$ set to null if, and only if, it does not reach any facility lease in X, in which case $\alpha_j = \alpha_j^P$. Also, due to case (c) of the algorithm, we have that $\alpha_j = \pi_j$. Thus, it is straightforward to conclude that

$$\sum_{\substack{j \in \mathcal{D} \\ a(j)=\text{null}}} \pi_j = \sum_{j \in \mathcal{D}} \alpha_j^P.$$

Finally, we bound the client connection cost. Let $\mathcal{D}_{X'}$ be the set of client requests that reach some facility lease in X'. For each $j \in \mathcal{D}_{X'}$, let $f \in X'$ be the facility lease reached by j. Note that j is connected to the closest active facility lease in $\hat{X}$, so

$$d(j, a(j)) \leq d(j, f) = \alpha_j^C.$$

Now let j be some client request that reaches some $f \in X$ but does not reach any facility lease in X'. There must be some $f' \in X'$ and some $j' \in \mathcal{D}_{X'}$ that reaches both f and f', by construction of X'. We must have that $\alpha_j \geq \alpha_{j'}$ since, when $\alpha_{j'}$ stopped increasing, j' had reached both f and f', and j had reached f when α_j stopped increasing. Since j' reaches both f and f', we have that

$$\alpha_{j'} \geq d(j', f) \quad \text{and} \quad \alpha_{j'} \geq d(j', f').$$

Thus, by triangle inequality, we have that

$$d(j, a(j)) \leq d(j, f') \leq d(j, f) + d(j', f) + d(j', f') \leq \alpha_j + \alpha_{j'} + \alpha_{j'} \leq 3 \cdot \alpha_j = 3 \cdot \alpha_j^C.$$

Summing up the previous inequalities, we have that

$$\begin{aligned}
& \sum_{f \in \hat{X}} \gamma_f + \sum_{\substack{j \in \mathcal{D} \\ a(j) \neq \text{null}}} d(j, a(j)) + \sum_{\substack{j \in \mathcal{D} \\ a(j)=\text{null}}} \pi_j \\
\leq \; & 3 \cdot \sum_{j \in \mathcal{D}} \alpha_j^F + 3 \cdot \sum_{j \in \mathcal{D}} \alpha_j^C + \sum_{j \in \mathcal{D}} \alpha_j^P \\
\leq \; & 3 \cdot \sum_{j \in \mathcal{D}} \alpha_j \\
\leq \; & 3 \cdot \text{opt}(G, d, F, K, \delta, \gamma, \delta, D_0, \ldots, D_{T-1}, \pi).
\end{aligned}$$

□

Again, this analysis is tight due to the example of Figure 11.1, and this algorithm cannot benefit from the dual-fitting technique.

If we combine Theorem 11.3 with Algorithm 12.1, we obtain a 3.5277-approximation

algorithm for the facility leasing problem with submodular penalties. Note that FLeP is a particular case of this problem, but our algorithm has smaller approximation ratio.

For the online version of FLeP, we know in advance G, F, K, δ and γ. At each instant of time t, we receive a set of clients D_t along with the distances between each $j \in D_t$ and each $f \in F$ and a new penalty for each $j \in D_t$. We must decide if we acquire some new facility leases, and we must choose to pay a penalty or assign an active facility lease to each client request in D_t. We cannot remove previous facility leases nor change the assignment of previous clients. The goal is to minimize the cost of the final FLeP solution. This problem admits a $O(K \lg n)$-competitive algorithm [63]. The problem has the same lower bound as FLe, of $\Omega(\lg K + \lg n' / \lg \lg n')$, where $n' := \min\{n, \delta_K\}$.

Part IV

Connected Facility Leasing Problems

Chapter 13

Connected Facility Location

In this chapter we review some approximation and online algorithms for the **connected facility location problem** (CFL), which is a two-layer network design problem in which we connect clients via facilities connected through a core tree. We give a formal definition of the problem below.

Problem CFL(G, d, F, γ, M, D): *The input consists of a complete graph $G = (V, E)$, a metric function $d : V \times V \mapsto \mathbb{Q}_+$ that assigns distances between vertices, a set $F \subseteq V$ of potential facilities, a function $\gamma : F \mapsto \mathbb{Q}_+$ that assigns an opening cost to each potential facility, a rational scaling constant $M \geq 1$, and a set $D \subseteq V$ of clients. The goal is to obtain a set $X \subseteq F$ of open facilities, a function $a : D \mapsto X$ that assigns each client to an open facility, and a set of core edges $\mathcal{T} \subseteq E$ which connects X, and we wish to minimize*

$$\sum_{f \in X} \gamma_f + \sum_{j \in D} d(j, a(j)) + M \cdot \sum_{e \in \mathcal{T}} d(e).$$

This is an NP-hard problem, since ST reduces to it if we take $F = D$, $\gamma_f = 0$ for all $f \in F$ and $M = 1$.

In Sections 13.1 and 13.2 we review some approximation and online algorithms for CFL, respectively. In Section 13.3, we discuss the multi-commodity version of the problem.

13.1 Offline Connected Facility Location

In this section, we review some approximation algorithms for CFL, which will be useful for our discussion in the next chapter.

13.1.1 A First Naïve Algorithm

Let us consider the following algorithm for CFL. The algorithm utilizes as black boxes approximation algorithms for FL and ST.

A common assumption in CFL algorithms is that a root facility $r \in F$ is given at zero cost. This is without loss of generality since we can run the algorithm $|F|$ times and return the best solution.

The algorithm runs the FL algorithm on the corresponding instance (without M), then builds a Steiner tree connecting the clients to the root, and finally adds a core edge between each client and its assigned facility.

Input: $(G, d, F, r, \gamma, M, D)$
1 **set** $\gamma_r \leftarrow 0$;
2 $(X, a) \leftarrow \mathrm{FL}(G, d, F, \gamma, D)$;
3 $\mathcal{T} \leftarrow \mathrm{ST}(G, d, D \cup \{r\})$;
4 $\mathcal{T} \leftarrow \mathcal{T} \cup \{(j, a(j)) : j \in D\}$;
5 **return** $(X, a, \mathcal{T})$;

Algorithm 13.1: A first naïve algorithm for CFL.

Unfortunately, this algorithm has unbounded approximation factor. Take an instance with a root r with $\gamma_r = 0$, and another facility f with $\gamma_f = 0$ and $d(f, r) = 1$. Then take one client at the same point as f. The algorithm will open f and connect it to the root, for a total cost of M, while the optimum solution will connect the client to the root by paying 1. Thus, the approximation factor is $\Omega(M)$.

However, we show that this algorithm performs well if $M = 1$. Indeed, it is a 4.36-approximation. Let $\alpha_{\mathrm{FL}} := 1.488$ and $\alpha_{\mathrm{ST}} := 1.39$ be the current best approximation factors for FL and ST, respectively.

Theorem 13.1 (Swamy and Kumar [69]): *Algorithm 13.1 is a* $(2 \cdot \alpha_{\mathrm{FL}} + \alpha_{\mathrm{ST}})$*-approximation to* CFL *if* $M = 1$.

Proof: Given a solution returned by the algorithm, let O be the facility opening cost, C the client connection cost, and S the core tree cost. Similarly, let O^*, C^* and S^* be those costs on an optimum solution.

Let O' be the facility opening cost and C' be the client connection cost of the solution returned by the FL algorithm. We have that $O + C = O' + C' \leq \alpha_{\mathrm{FL}} \cdot \mathrm{opt}_{\mathrm{FL}}$. Since an optimum solution for CFL induces a feasible solution for FL, $\mathrm{opt}_{\mathrm{FL}} \leq O^* + C^*$, so $O + C \leq \alpha_{\mathrm{FL}} \cdot (O^* + C^*)$.

Since $M = 1$, we bound S by the cost of solving ST on $D \cup \{r\}$, plus the cost of connecting each client to its assigned facility. Thus $S \leq \alpha_{\mathrm{ST}} \cdot \mathrm{opt}_{\mathrm{ST}} + C$. Since the optimum core tree combined with an edge between each client and its optimum facility induces a feasible solution for ST on $D \cup \{r\}$, we have that $\mathrm{opt}_{\mathrm{ST}} \leq S^* + C^*$.

Combining the previous inequalities, we have that

$$\begin{aligned} O + C + S &\leq \alpha_{\mathrm{FL}} \cdot (O^* + C^*) + \alpha_{\mathrm{ST}} \cdot (S^* + C^*) + C \\ &\leq \alpha_{\mathrm{FL}} \cdot (O^* + C^*) + \alpha_{\mathrm{ST}} \cdot (S^* + C^*) + \alpha_{\mathrm{FL}} \cdot (O^* + C^*) \\ &\leq (2 \cdot \alpha_{\mathrm{FL}} + \alpha_{ST}) \cdot (O^* + C^* + S^*). \end{aligned}$$

□

13.1.2 A Simple Sample-and-Augment Algorithm

In this section we present a simple randomized sample-and-augment algorithm for CFL, which avoids the unbounded approximation factor of the previous algorithm. It is a variation of the algorithm by Gupta, Srinivasan and Tardos [38].

A pseudocode is presented in Algorithm 13.2. The idea is that, in order to avoid a few clients to open a facility, which incurs in buying core edges M times more expensive than an edge connecting clients to already open facilities, we sample each client with probability $1/M$. Then we run the previous algorithm on the sampled clients, and connect the remaining clients to the closest open facility.

Input: $(G, d, F, r, \gamma, M, D)$
1 **set** $\gamma_r \leftarrow 0, \quad D' \leftarrow \emptyset$;
2 **foreach** $j \in D$ **do**
3 | add j to D' with probability $1/M$;
4 $(X, a) \leftarrow \mathrm{FL}(G, d, F, \gamma, D')$;
5 $\mathcal{T} \leftarrow \mathrm{ST}(G, d, D' \cup \{r\})$;
6 $\mathcal{T} \leftarrow \mathcal{T} \cup \{(j, a(j)) : j \in D'\}$;
7 **foreach** $j \notin D'$ **do**
8 | $a(j) \leftarrow \arg\min_{f \in X} d(j, f)$;
9 **return** $(X, a, \mathcal{T})$;

Algorithm 13.2: A simple randomized sample-and-augment algorithm for CFL.

We do not prove the following theorem, since the analysis is not important to our discussion. It is a simple modification of the one in [38, Section 6.1].

Theorem 13.2: *Algorithm 13.2 is a 8.94-approximation to* CFL.

Eisenbrand, Grandoni, Rothvoß and Schäfer [25] obtained a better result by doing a simple modification in the previous algorithm. They obtain a FL solution (X, a) on the whole set of clients D, open a sampled subset $X' \subseteq X$ of the facilities, and connect each client to the closest open facility. Their algorithm is a 4-approximation. We do not present the analysis of this algorithm here.

13.1.3 A More Sophisticated Algorithm

In this section we present the algorithm for CFL by Grandoni and Rothvoß [35]. This is the current best approximation algorithm for CFL, and it has a rather simple description.

The algorithm uses a parameter $\rho \in \mathbb{Q}_+^*$ whose value is obtained in the analysis, in order to minimize the approximation factor. The algorithm utilizes as black boxes an α_{ST}-approximation for ST and a bifactor algorithm for FL which is an (α_F, α_D)-approximation, i.e., which returns a solution whose cost is at most $\alpha_F \cdot \sum_{f \in X^*} \gamma_f + \alpha_D \cdot \sum_{j \in D} d(j, a^*(j))$, where (X^*, a^*) is an optimum solution.

The algorithm samples each client with probability ρ/M, and builds a Steiner tree connecting the sampled clients to the root. Then, it runs a modified instance of FL, in which the cost of opening facility f is the original cost γ_f plus the distance between f

and the terminals of the core tree. Then, the algorithm completes the tree with edges to open facilities.

Input: $(G, d, F, r, \gamma, M, D, \rho)$
1 **set** $\gamma_r \leftarrow 0, \quad D' \leftarrow \emptyset$;
2 **foreach** $j \in D$ **do**
3 | add j to D' with probability ρ/M;
4 $\mathcal{T}' \leftarrow \mathrm{ST}(G, d, D' \cup \{r\})$;
5 **foreach** $f \in F$ **do**
6 | **let** $\gamma'_f := \gamma_f + M \cdot d(f, D' \cup \{r\})$;
7 $(X, a) \leftarrow \mathrm{FL}(G, d, F, \gamma', D)$;
8 $\mathcal{T} \leftarrow \mathcal{T}' \cup \{\text{min-path}(f, \mathcal{T}') : f \in X\}$;
9 **foreach** $j \notin D'$ **do**
10 | $a(j) \leftarrow \arg\min_{f \in X} d(j, f)$;
11 **return** $(X, a, \mathcal{T})$;

Algorithm 13.3: A more sophisticated randomized algorithm for CFL [35].

The analysis considers $\alpha_{\mathrm{ST}} = 1.39$ [12] and the result by Byrka and Aardal [11], which shows that, for every $\alpha_F > 1.67$, there exists an $(\alpha_F, 1 + 2e^{-\alpha_F})$-approximation for FL.

The analysis also utilizes the following lemma [25], which we do not prove here.

Lemma 13.3 (Eisenbrand *et al.* [25]): *Consider a complete graph $G = (V, E)$ with a metric distance function $d : V \times V \mapsto \mathbb{Q}_+$, a set of clients $D \subseteq V$, a subtree $\mathcal{T}'$ of G with a root r, an assignment $a : D \mapsto V(\mathcal{T}')$, and a probability $p \in (0, 1]$. Sample each client in D independently with probability p, and let D' be sampled set. Let $a(D') := \bigcup_{j \in D'} a(j)$. Then*

$$\mathbb{E}\left[\sum_{j \in D} d\left(j, a(D') \cup \{r\}\right)\right] \leq \frac{0.807}{p} \cdot \sum_{e \in \mathcal{T}'} d(e) + \sum_{j \in D} d(j, a(j)).$$

Let $(X^*, a^*, \mathcal{T}^*)$ be an optimum CFL solution, and let O^* be the facility opening cost, C^* be the client connection cost, and S^* be the core tree cost in this solution. Let also $O' := \sum_{f \in X} \gamma'_f = \sum_{f \in X} (\gamma_f + M \cdot d(f, D' \cup \{r\}))$ be the facility opening cost and $C' := \sum_{j \in D} d(j, a(j))$ be the client connection cost of the solution obtained at Line 7 for the modified FL problem.

Lemma 13.4: *The cost of the solution returned by Algorithm 13.3 is at most $M \cdot \sum_{e \in \mathcal{T}'} d(e) + O' + C'$.*

Proof: The client connection cost is the same in both problems, and the definition of γ' pays for the facility opening cost in the CFL instance and for the edges added at Line 8 of the algorithm. □

Lemma 13.5:

$$\mathbb{E}\left[M \cdot \sum_{e \in \mathcal{T}'} d(e)\right] \leq \alpha_{\mathrm{ST}} \cdot (S^* + \rho \cdot C^*).$$

Proof: It is possible to transform $\mathcal{T}^*$ into a feasible solution to ST on $D' \cup \{r\}$ simply

by adding an edge between each $j \in D'$ and $a^*(j)$. The expected cost of this solution is

$$\sum_{e \in \mathcal{T}^*} d(e) + \mathbb{E}\left[\sum_{j \in D'} d(j, a^*(j)\right] = \sum_{e \in \mathcal{T}^*} d(e) + \frac{\rho}{M} \sum_{j \in D} d(j, a^*(j) = \sum_{e \in \mathcal{T}^*} d(e) + \frac{\rho}{M} \cdot C^*.$$

From the analysis of the ST algorithm,

$$\begin{aligned} \mathbb{E}\left[M \cdot \sum_{e \in \mathcal{T}'} d(e)\right] &\leq M \cdot \alpha_{\mathrm{ST}} \cdot \mathbb{E}[\mathrm{opt}_{\mathrm{ST}}(D' \cup \{r\})] \\ &\leq M \cdot \alpha_{\mathrm{ST}} \cdot \left(\sum_{e \in \mathcal{T}^*} d(e) + \frac{\rho}{M} \cdot C^*\right) \\ &= \alpha_{\mathrm{ST}} \cdot (S^* + \rho \cdot C^*). \end{aligned}$$

□

Lemma 13.6:

$$\mathbb{E}[O' + C'] \leq \alpha_F \cdot (O^* + \rho \cdot C^*) + \alpha_C \cdot \left(C^* + \frac{0.807}{\rho} \cdot S^*\right).$$

Proof: Given an optimum solution $(X^*, a^*, \mathcal{T}^*)$, we can obtain a feasible solution for FL instance (G, d, F, γ', D) in the following manner. Open facility set $a^*(D') \cup \{r\}$, where $a^*(D') = \bigcup_{j \in D'} a^*(j)$. The expected opening cost of this set is

$$\begin{aligned} \mathbb{E}\left[\sum_{f \in a^*(D') \cup \{r\}} \gamma'_f\right] &= \mathbb{E}\left[\sum_{f \in a^*(D') \cup \{r\}} \gamma_f\right] + M \cdot \mathbb{E}\left[\sum_{f \in a^*(D') \cup \{r\}} d(f, D' \cup \{r\})\right] \\ &\leq \mathbb{E}\left[\sum_{f \in X^*} \gamma_f\right] + M \cdot \mathbb{E}\left[\sum_{j \in D'} d(j, a^*(j))\right] \\ &= O^* + M \cdot \frac{\rho}{M} \cdot \sum_{j \in D} d(j, a^*(j)) = O^* + \rho \cdot C^*. \end{aligned}$$

In order to bound the expected client connection cost in this solution, we apply Lemma 13.3 with client set D, tree $\mathcal{T}^*$, assignment a^*, root r and probability ρ/M:

$$\mathbb{E}\left[\sum_{j \in D} d(j, a^*(D') \cup \{r\})\right] \leq \frac{0.807}{\rho/M} \cdot \sum_{e \in \mathcal{T}^*} d(e) + \sum_{j \in D} d(j, a^*(j)) = \frac{0.807}{\rho} \cdot S^* + C^*.$$

□

In particular, note that the cost of the returned solution is bounded by the cost of a solution which utilizes only facilities that serve clients in D' in the optimum solution. This is important for our discussion in Chapter 14.

Theorem 13.7 (Grandoni and Rothvoß [35]): *Algorithm 13.3 is a 3.19-approximation to* CFL.

Proof: Take $\alpha_{\mathrm{ST}} = 1.39$, $\rho = 0.539$, $\alpha_F = 2.294$ and $\alpha_C \leq 1 + 2e^{-\alpha_F}$. Combining the previous lemmas, we have that the expected cost of the solution returned by the algorithm is at most

$$\begin{aligned} & \alpha_{\mathrm{ST}} \cdot (S^* + \rho \cdot C^*) + \alpha_F \cdot (O^* + \rho \cdot C^*) + \alpha_C \cdot \left(C^* + \frac{0.807}{\rho} S^* \right) \\ = \; & \alpha_F \cdot O^* + \left(\alpha_{\mathrm{ST}} + \frac{0.807}{\rho} \cdot \alpha_C \right) \cdot S^* + (\alpha_{\mathrm{ST}} \cdot \rho + \alpha_F \cdot \rho + \alpha_C) \cdot C^* \\ \leq \; & 2.30 \cdot O^* + 3.19 \cdot S^* + 3.19 \cdot C^* \leq 3.19 \cdot \mathrm{opt}(G, d, F, r, \gamma, M, D). \end{aligned}$$

□

13.2 Online Connected Facility Location

In this section we review online algorithms for CFL.

In the online version of CFL, we known in advance G, F, γ and M, as well as root facility $r \in F$ with $\gamma_r = 0$, and at each instant of time $t = 0, \ldots, T-1$ we receive a set of clients D_t, along with the distances between each client in D_t and each facility in F.[1] We must decide if we open new facilities, connect them to r, and assign an open facility to each client in D_t. We cannot close previously open facilities, remove core edges or change the assignment of previous clients. Due to the reduction from ST to CFL shown at the beginning of the chapter, we have that CFL has online lower bound of $\Omega(\lg n)$, where $n := \left| \bigcup_{t=0}^{T-1} D_t \right|$.

The problem admits a deterministic $\mathrm{O}(\lg n)$-competitive online algorithm, due to Umboh [72]. This algorithm, however, is rather complicated and its description does not add much to our discussion in this thesis. Just to give a glimpse, it is based on the technique of approximating a metric by a tree metric which we described at Section 10.1. However, this algorithm does not make a simple use of Lemma 10.5; indeed, the proof in [31] shows that FL has an online lower bound of $\Omega(\lg n / \lg \lg n)$ even on tree metrics. Actually, Umboh used the concept of hierarchical decompositions to devise his algorithm, and used Theorem 10.6 in the analysis of its competitive ratio. He used the same technique to develop good online algorithms for other problems, such as SN and RoB.

Another $\mathrm{O}(\lg n)$-competitive online algorithm for CFL, this time a randomized one, was given by San Felice, Williamson and Lee [65]. It is based on the primal-dual online algorithm for FL by Fotakis [30], and on the randomized online algorithm for RoB by Awerbuch, Azar and Bartal [6], both of which are $\mathrm{O}(\lg n)$-competitive. In some sense, it is similar to Algorithm 13.2.

We present a pseudocode in Algorithm 13.4. We denote the online algorithm of Fotakis [30] for FL by OFL, whose offline parameters are G, d, F and γ, and whose request sequence is a sequence of clients; this algorithm is $\mathrm{O}(\lg n)$-competitive. The online algo-

[1] Note that, in the online version of problem, a root facility r is necessarily assumed.

rithm for CFL maintains a set $\mathcal{D}$ of client requests, and a set $\mathcal{D}^{\mathcal{T}}$ of clients which are connected to the core tree. Each client request that arrives is forwarded to algorithm OFL, and the corresponding FL virtual solution is updated. Then, the client is added to the core tree with probability $1/M$. If this happens, then the algorithm adds to the core tree the shortest path between the client and the tree, opens the facility indicated by the virtual solution, and adds another core tree edge between the client and this new facility. Otherwise, if the client is not added to the core tree, then it is served by the closest open facility.

Input: (G, d, F, r, γ, M)
$X \leftarrow \emptyset, \quad \mathcal{D} \leftarrow \emptyset, \quad \mathcal{D}^{\mathcal{T}} \leftarrow \emptyset, \quad \mathcal{T} \leftarrow (\{r\}, \emptyset)$;
set $\gamma_r \leftarrow 0$;
initialize OFL with (G, d, F, γ) and let (X', a') be the virtual solution;
when D_t arrives **do**
  foreach $j \in D_t$ **do**
    send (j, t) to OFL and update virtual solution (X', a');
    add j to $\mathcal{D}^{\mathcal{T}}$ with probability $1/M$;
    if $j \in \mathcal{D}^{\mathcal{T}}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \text{min-path}(j, \mathcal{T})$;
      $f \leftarrow a'(j, t)$;
      if $f \notin X$ **then**
        $X \leftarrow X \cup \{f\}$;
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{(f, j)\}$;
    let $f \in X$ be the closest open facility to j;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(j, t)\}, \quad a(j, t) \leftarrow f$;
return $(X, a, \mathcal{T})$;

Algorithm 13.4: Online algorithm for CFL [65].

This algorithm has expected competitive ratio $\mathrm{O}(\lg n)$. However, we only prove this fact for $M = 1$, in which case the algorithm becomes deterministic. The proof for generic $M \geq 1$ is a bit complicated and is not necessary to our discussion.

Theorem 13.8 (San Felice, Williamson and Lee [65]): *Algorithm 13.4 has competitive factor* $\mathrm{O}(\lg n)$ *if* $M = 1$.

Proof: Given a solution returned by the algorithm, let O be the facility opening cost, C the client connection cost, and S the core tree cost. Similarly, let O^*, C^* and S^* be those costs on an optimum solution.

Let O' be the facility opening cost and C' be the client connection cost of the virtual solution produced by OFL. Since $M = 1$, the algorithm is deterministic and always mimics the virtual solution; thus we have that $O + C = O' + C' = \mathrm{O}(\lg n) \cdot \mathrm{opt}_{\mathrm{FL}}$. Since an optimum solution for CFL induces a feasible solution for FL, $\mathrm{opt}_{\mathrm{FL}} \leq O^* + C^*$, so $O + C = \mathrm{O}(\lg n) \cdot (O^* + C^*)$.

Let $\mathcal{D} := \bigcup_{t=0}^{T-1} D_t$. We bound S by the cost of running the greedy ST online algorithm on $\mathcal{D} \cup \{r\}$, plus the cost of an edge between each client and its assigned facility, which is the connection cost for that client since $M = 1$. Thus, $S \leq \mathrm{O}(\lg n) \cdot \mathrm{opt}_{\mathrm{ST}} + C$. Since the

optimum core tree combined with an edge between each client and its optimum facility induces a feasible solution for ST on $\mathcal{D} \cup \{r\}$, we have that $\text{opt}_{\text{ST}} \leq S^* + C^*$ and the theorem follows. □

13.3 Multi-Commodity Connected Facility Location

In the multi-commodity version of CFL, instead of connecting clients to the core network via open facilities, we wish to connect pairs of clients. The pair may be connected directly, or via facilities connected by core edges. To be more precise, the path between the pair of clients may be composed of simple edges, which are temporary and charged to this pair, and of core edges, which cost M times the cost of a simple edge but are permanent and have infinite capacity. However, the path must enter and leave the core tree through open facilities. A good analogy for this is a person who wishes to go to her job, and can walk part of the way, but can also use subway lines to save some time and energy. Obviously, she can only enter or leave the subway through open stations. We define the problem formally below, which we call the **multi-commodity connected facility location problem** (MCFL).

Problem MCFL(G, d, F, γ, M, P): *The input consists of a complete graph $G = (V, E)$, a metric distance $d : V \times V \mapsto \mathbb{Q}_+$, a set $F \subseteq V$ of potential facilities, a function $\gamma : F \mapsto \mathbb{Q}_+$ that assigns an opening cost to each potential facility, a rational scaling constant $M \geq 1$, and a set $P \subseteq V \times V$ of pairs of clients. Let $X \subseteq F$, $\mathcal{T} \subseteq E$, and $u, v \in V$; we denote by $d_{X,\mathcal{T}}(u, v)$ the distance between u and v in the graph in which we add to G edges of cost zero between each pair of facilities in X that are in the same tree in $\mathcal{T}$. The goal is to find a set $X \subseteq F$ of open facilities and a forest $\mathcal{T} \subseteq E$ which minimize*

$$\sum_{f \in X} \gamma_f + \sum_{(u,v) \in P} d_{X,\mathcal{T}}(u, v) + M \cdot \sum_{e \in \mathcal{T}} d(e).$$

This problem is NP-hard since it reduces to CFL if one of the vertices in each pair is some root facility $r \in F$. The problem was proposed by Grandoni and Rothvoß, who gave a 16.2-approximation [35]. We present this algorithm in this section.

A pseudocode is presented in Algorithm 13.5. It uses as black boxes an α_{FLP}-approximation algorithm for FLP, and an α_{SF}-approximation for SF. Currently we have $\alpha_{\text{FLP}} = 1.5148$ [56] and $\alpha_{\text{SF}} = 2$ [2]. For each pair $(u, v) \in P$, we assign a penalty of $d(u, v)/2$ to both u and v, and we run the FLP algorithm on the corresponding set of clients. If the FLP algorithm does not assign a facility to one of (u, v), then the algorithm will connect u and v directly. Otherwise, the algorithm includes a set consisting of these two clients in D' with probability $1/M$. Then the algorithm runs the SF algorithm to find a forest which connects the pairs in D', and adds an edge between each client in D' and the facility to which it is assigned. The algorithm returns only facilities that serve clients in D'.

We prove that the algorithm is a 5.03-approximation when $M = 1$. The proof that it is a 16.2-approximation for $M \geq 1$ is unnecessary to our discussion.

Input: (G, d, F, γ, M, P)
1 **foreach** $(u, v) \in P$ **do set** $\pi_u \leftarrow \pi_v \leftarrow d(u, v)/2$;
2 **let** $D \leftarrow \bigcup_{(u,v)\in P}\{u, v\}$;
3 $(X', a') \leftarrow \mathrm{FLP}(G, d, F, \gamma, D, \pi)$;
4 $D' \leftarrow \emptyset$;
5 **foreach** $(u, v) \in P : a'(u) \neq \mathsf{null}$ **and** $a'(v) \neq \mathsf{null}$ **do**
6 | add terminal set $\{u, v\}$ to D' with probability $1/M$;
7 $X \leftarrow \{f \in X' : \exists u \in D' : a'(u) = f\}$;
8 $\mathcal{T} \leftarrow \mathrm{SF}(G, d, D') \cup \{(u, a'(u)) : u \in D'\}$;
9 **return** $(X, \mathcal{T})$;

Algorithm 13.5: Approximation algorithm for MCFL [35].

Theorem 13.9: *Algorithm 13.5 is a* $(2 \cdot \alpha_{\mathrm{FLP}} + \alpha_{\mathrm{ST}})$*-approximation to* MCFL *if* $M = 1$.

Proof: Given a solution returned by the algorithm, let O be the facility opening cost, C the client connection cost, and S the core tree cost. Similarly, let O^*, C^* and S^* be those costs on an optimum solution.

Let O' be the facility opening cost, C' be the client connection cost, and Π' be the penalty cost of the solution returned by the FLP algorithm on D. We have that $O' + C' + \Pi' \leq \alpha_{\mathrm{FLP}} \cdot \mathrm{opt}_{\mathrm{FLP}}$. Given an optimum solution for MCFL on P, we obtain a feasible solution for FLP on D by paying the penalties for the clients in the pairs that are connected directly, and by assigning the other clients to the closest facility in the path connecting the pair. Thus, $\mathrm{opt}_{\mathrm{FLP}} \leq O^* + C^*$ and $O' + C' + \Pi' \leq \alpha_{\mathrm{FLP}} \cdot (O^* + C^*)$.

Since Algorithm 13.5 opens a subset of the facilities in X', clearly $O \leq O'$.

For a pair (u, v) connected directly by Algorithm 13.5, we have that one of u and v pays for its penalty cost. Suppose w.l.o.g. that it is u; then $d_{X,\mathcal{T}}(u, v) \leq 2 \cdot \pi_u$. For a pair (u, v) connected through the core tree, since $M = 1$ and the algorithms always adds $\{u, v\}$ to D', a whole path between $a'(u)$ and $a'(v)$ is added to the tree, so $d_{X,\mathcal{T}}(u, v) \leq d(u, a'(u)) + d(v, a'(v))$. Thus $C \leq 2 \cdot \Pi' + C'$.

Since $M = 1$, we bound S by the cost of solving SF on D', plus the cost of connecting each client in D' to its assigned facility. Thus, $S \leq \alpha_{\mathrm{SF}} \cdot \mathrm{opt}_{\mathrm{SF}}(D') + C' \leq \alpha_{\mathrm{SF}} \cdot \mathrm{opt}_{\mathrm{SF}}(P) + C'$. Since the optimum core tree combined with the optimum client connection edges induces a feasible solution for SF on P, we have that $\mathrm{opt}_{\mathrm{SF}}(P) \leq S^* + C^*$. Substituting the previous inequalities, we have that

$$\begin{aligned} O + C + S &\leq O' + 2 \cdot \Pi' + C' + \alpha_{\mathrm{SF}} \cdot (S^* + C^*) + C' \\ &\leq \alpha_{\mathrm{SF}} \cdot (S^* + C^*) + 2 \cdot (O' + \Pi' + C') \\ &\leq \alpha_{\mathrm{SF}} \cdot (S^* + C^*) + 2 \cdot \alpha_{\mathrm{FLP}} \cdot (O^* + C^*) \\ &\leq (2 \cdot \alpha_{\mathrm{FLP}} + \alpha_{\mathrm{ST}}) \cdot (O^* + C^* + S^*). \end{aligned}$$

□

In the online version of MCFL, the offline part of the input consists of G, F, γ and M, and at each instant of time we are given a set of pairs of clients along with their distances to F. We must decide if we open new facilities and if we buy new core edges, from which a path between each new pair of clients is fixed. We cannot close previous open facilities or remove core edges. The problem admits a $O(\lg^2 n)$-competitive online algorithm, and a $O(\lg n)$-competitive online algorithm when $M = 1$, both by San Felice, Fernandes and Lintzmayer [64]. The problem has online lower bound of $\Omega(\lg n)$, due to the reduction from CFL we discussed above. Those algorithms are similar to Algorithm 13.5, so we do not present them here.

Chapter 14

Connected Facility Leasing

In this chapter we propose leasing variants of CFL and MCFL, and we give approximation and competitive online algorithms for a special case of these problems. The leasing variants we propose model the problem of a network service provider, who has to install cables between routers to serve clients, but resources, such as routers and backbone cables, have different lifetimes which are not negligible. The results in this chapter were published in [19, 20].

The first leasing variant of CFL we study is the **connected facility leasing problem** (CFLE). The input for CFLE is similar to that for CFL, but we no longer open facilities for unlimited time. Instead, we lease each facility for one of K different lengths of time, which we denote by $\delta_1, \ldots, \delta_K \in \mathbb{N}$. The cost γ_k^f of leasing a facility $f \in F$ depends on f, but also on the leasing type $k \in [K]$. There is a special root facility r which has cost zero for any leasing length. If we lease a facility f with leasing type k at instant $\hat{t}$, then we say that **facility lease** $(f, k, \hat{t})$ is **active** during interval $[\hat{t}, \hat{t} + \delta_k)$. The goal is to minimize the sum of the costs of the facility leases, plus the sum of the distances from each client to its assigned facility lease, plus M times the cost of a set of edges connecting leased facilities to r. Both FLE and ST reduce to CFLE, so this is an NP-hard problem. In Section 14.1, we present a 7.39-approximation algorithm for CFLE when $M = 1$, which combines Algorithm 12.1, which is a 3-approximation for FLE [60], and the 1.39-approximation algorithm for ST [12]. We also present a $O(K \lg n)$-competitive online algorithm for CFLE when $M = 1$. This algorithm combines the $O(K \lg n)$-competitive online algorithm for FLE [60] with the $O(\lg n)$-competitive online algorithm for ST [43]. We also discuss why we could not obtain good results for $M > 1$ in Section 14.1.

The second variant we study is the **leasing-connected facility leasing problem** (LECFLE), in which we lease both facilities and edges connecting facilities to the root. We have K^F types of facility leases, and K^E types of core edge leases. Facilities and core edges may have different leasing lengths. Instead of a single scaling factor M, we have K^E scaling factors $\gamma_1^E, \ldots, \gamma_{K_E}^E$, one for each core edge leasing length. Thus, to lease an edge e with leasing type k_e costs $\gamma_{k_e}^E \cdot d(e)$. We wish to assign an active facility lease to each client and, for each instant in which a facility serves a client, there must exist a path of active edge leases from that facility to the root. We wish to minimize the cost of leasing facilities, plus the cost of connecting clients to facilities, plus the cost of leasing core edges. For the case in which the smallest edge leasing scaling factor γ_1^E is

equal to 1, we give a $O(K^E)$-approximation algorithm and a $O(K^F \lg n + \lg K^E \lg |V|)$-competitive online algorithm. The approach is similar to the one we use for CFLE, but we change the building block algorithm of the core tree. The offline algorithm uses the $O(K)$-approximation algorithm for STLE [4], and the online algorithm uses the $O(\lg K \lg |V|)$-competitive online algorithm for SLE [59]. This problem is addressed in Section 14.2.

We also study two leasing variants of MCFL: the **multi-commodity connected facility leasing problem** (MCFLE), in which facilities are leased but core edges are permanent, and the **multi-commodity leasing-connected facility leasing problem** (MLECFLE), in which both facilities and core edges are leased. For MCFLE with $M = 1$, we give an 8-approximation algorithm, which combines the 3-approximation for FLEP we presented at Section 12.2 with the 2-approximation for SF [2]. For its online version, we give a $O(K \lg n)$-competitive algorithm, which combines the $O(K \lg n)$-competitive online algorithm for FLEP [63] with the $O(\lg n)$-competitive online algorithm for SF [8]. Note that both facility leasing and Steiner problems are different from the ones we use to solve CFLE. For MLECFLE with $\gamma_1^E = 1$, we give a $O(\lg n)$-approximation algorithm and a $O(K^F \lg n + \lg K^E \lg |V|)$-competitive online algorithm. Here the underlying facility leasing problem is FLEP, as in MCFLE, and the underlying Steiner problem is SLE, as in LECFLE. These results are detailed in Section 14.3.

We summarize the approximation/competitive factors we obtain in Table 14.1. Our technique for solving these problems for the case with $M = 1$ ($\gamma_1^E = 1$), both in offline and online settings, consists in solving the associated facility leasing problem, buying (leasing) a core network that connects the clients, and then buying (leasing) core edges between each client and its corresponding facility lease. The guarantee follows from the analysis of the corresponding building block algorithms, and from reducing the optimal solution of the connected facility leasing problem to a feasible solution of the building block problems. This is the approach of Algorithms 13.1, 13.4 and 13.5 for CFL and MCFL when $M = 1$; it is a common approach in the literature [64, 65, 69]. The general cases ($M > 1$ and $\gamma_1^E > 1$, respectively) for the four problems we study, both in offline and online settings, are open.

problem	offline algorithm	online algorithm
CFLE	7.39 if $M = 1$	$O(K \lg n)$ if $M = 1$ [19]
LECFLE	$O(K^E)$ if $\gamma_1^E = 1$	$O(K^F \lg n + \lg K^E \lg \|V\|)$ if $\gamma_1^E = 1$
MCFLE	8 if $M = 1$	$O(K \lg n)$ if $M = 1$
MLECFLE	$O(\lg n)$ if $\gamma_1^E = 1$	$O(K^F \lg n + \lg K^E \lg \|V\|)$ if $\gamma_1^E = 1$

Table 14.1: Summary of our results for connected facility leasing problems.

14.1 Connected facility leasing

In CFLE, we are given a complete graph $G = (V, E)$ with a metric distance between vertices $d : V \times V \mapsto \mathbb{Q}_+$, a set $F \subseteq V$ of potential facilities, a root facility $r \in F$, K leasing lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$, a cost $\gamma_k^f \in \mathbb{Q}_+$ for leasing facility $f \in F$ with leasing type $k \in [K]$ (ensuring $\gamma_k^r = 0$ for any k), a rational constant $M \geq 1$ and, for $t = 0, \ldots, T-1$, a

set $D_t \subseteq V$ of clients arriving at instant t. Let $\mathcal{D} := \{(j,t) : j \in D_t \text{ for } t \in \{0,\ldots,T-1\}\}$ be the set of client requests. The goal is to find a set $X \subseteq F \times [K] \times \mathbb{Z}_+$ of facility leases, a function $a : \mathcal{D} \mapsto X$ that assigns each client request (j,t) to a facility lease $(f,k,\hat{t})$ such that $t \in [\hat{t}, \hat{t}+\delta_k)$, and a set $\mathcal{T} \subseteq E$ of edges connecting X to r; and we wish to minimize

$$\sum_{(f,k,\hat{t})\in X} \gamma_k^f + \sum_{(j,t)\in\mathcal{D}} d(j,a(j,t)) + M \cdot \sum_{e\in\mathcal{T}} d(e).$$

Again, in order to simplify notation, for $(f,k,\hat{t}) \in F \times [K] \times \mathbb{Z}_+$ and $(j,t) \in \mathcal{D}$, we define the distance from (j,t) to $(f,k,\hat{t})$ as in Equation (12.1).

Algorithm 14.1 is a 7.39-approximation for CFLE when $M = 1$. It is similar to Algorithm 13.1: first, we obtain a solution of FLE on the clients, by using Algorithm 12.1, which is a 3-approximation [60]. Then we build a tree connecting the clients to the root, using an approximation algorithm for ST, and finally we add an edge in the tree between each client and the facility lease that serves it. The approximation factor of our algorithm can be expressed as $6+\alpha_{\text{ST}}$, where $\alpha_{\text{ST}} \approx 1.39$ is the best approximation factor for ST [12].

Input: $(G, d, F, r, K, \delta, \gamma, M, D_0, \ldots, D_{T-1})$
1 **set** $\gamma_K^r \leftarrow 0$;
2 $(X, a) \leftarrow \text{FLE}(G, d, F, K, \delta, \gamma, D_0, \ldots, D_{T-1})$;
3 $\mathcal{T} \leftarrow \text{ST}(G, d, \mathcal{D} \cup \{r\})$;
4 $\mathcal{T} \leftarrow \mathcal{T} \cup \{(j, a(j,t)) : (j,t) \in \mathcal{D}\}$;
5 **return** $(X, a, \mathcal{T})$;

Algorithm 14.1: Approximation algorithm for CFLE.

Theorem 14.1: *Algorithm 14.1 is a $(6+\alpha_{\text{ST}})$-approximation when $M = 1$.*

Proof: Given a solution returned by the algorithm, let L be the facility leasing cost, C the client connection cost, and S the core tree cost. Similarly, let L^*, C^* and S^* be those costs on an optimum solution.

Let L' be the facility leasing cost and C' be the client connection cost of the solution returned by the FLE algorithm. We have that $L + C = L' + C' \leq 3 \cdot \text{opt}_{\text{FLe}}$. Since an optimum solution for CFLE induces a feasible solution for FLE, $\text{opt}_{\text{FLe}} \leq L^* + C^*$, so $L + C \leq 3 \cdot (L^* + C^*)$.

Since $M = 1$, we bound S by the cost of solving ST on $\mathcal{D}$, plus the cost of connecting each client to its assigned facility lease. Thus, $S \leq \alpha_{\text{ST}} \cdot \text{opt}_{\text{ST}} + C$. Since the optimum core tree combined with an edge between each client and its optimum facility induces a feasible solution for ST on $\mathcal{D}$, we have that $\text{opt}_{\text{ST}} \leq S^* + C^*$ and the theorem follows. □

This algorithm has approximation factor $\Omega(M)$ if $M > 1$, due to the same example we presented in Section 13.1.1 to show that Algorithm 13.1 is a $\Omega(M)$-approximation for CFL if $M > 1$.

We do not know if there is a constant-approximation algorithm for CFLE if $M > 1$. In particular, consider Algorithm 14.2, which is a same sample-and-augment similar to Algorithm 13.2. There is an example which shows that this algorithm has approximation

factor $\Omega(n)$ if $M > 1$: take a single facility f with $d(f, r) = 1$, then take $K = 1$, $\delta_1 = 1$, $\gamma_1^f = 0$, and $M = 2$. For $t = 0, \ldots, n-1$ with $n \geq 2$, take a single client request at instant t on the same point as f. Note that the optimum solution buys the core edge (f, r) (cost 2) and always leases facility f (cost zero), paying a total cost of 2. In expectation, Algorithm 14.2 will sample half of the client requests, buy the core edge, and pay a cost of 1/2 to serve each client request, since if the client request is not sampled, then it has to be served by the root. Thus the expected cost is $n/2 + 2$. A similar modification of the algorithm by Eisenbrand, Grandoni, Rothvoß and Schäfer [25] also has unbounded approximation factor.

Input: $(G, d, F, r, K, \delta, \gamma, M, D_0, \ldots, D_{T-1})$
set $\gamma_K^r \leftarrow 0, \quad \mathcal{D}' \leftarrow \emptyset$;
foreach $(j, t) \in \mathcal{D}$ **do**
  add (j, t) to $\mathcal{D}'$ with probability $1/M$;
$(X, a) \leftarrow \text{FLE}(G, d, F, K, \delta, \gamma, \mathcal{D}')$;
$\mathcal{T} \leftarrow \text{ST}(G, d, \mathcal{D}' \cup \{r\})$;
$\mathcal{T} \leftarrow \mathcal{T} \cup \{(j, a(j, t)) : (j, t) \in \mathcal{D}'\}$;
foreach $(j, t) \notin \mathcal{D}'$ **do**
  $a(j, t) \leftarrow \arg\min_{(f,k,\hat{t}) \in X} d((j, t), (f, k, \hat{t}))$;
return $(X, a, \mathcal{T})$;

Algorithm 14.2: A candidate randomized sample-and-augment algorithm for CFLE.

Even if we consider more clever sample-and-augment algorithms for CFL, such as Algorithm 13.3, available analysis techniques seem insufficient. Remember from the proof of Lemma 13.6 that the cost of the returned solution is bound by the cost of a solution which utilizes only facilities that serve sampled clients in the optimum solution. In some sense, this bounds the client connection cost via the expected cost of the core tree. In CFLE, however, these costs are not so tightly related since, while core edges are permanent, facility leases cease after some time. Thus, a client which is close to the tree may have to be connected to a facility which is further apart. The primal-dual technique is another candidate for a good algorithm for CFLE, but the primal-dual algorithm for CFL by Swamy and Kumar [69] also bounds the client connection cost via the core tree cost.

Now we turn our attention to the online version of CFLE. Here, numbers T and $n := \sum_{t=0}^{T-1} |D_t|$ are unknown. Sets D_t are revealed one at a time, and we cannot remove edges from $\mathcal{T}$, change facility leases, or modify $a(j, t)$ once it is chosen. The competitive factor is $\Omega(\lg K + \lg n')$, where $n' := \min\{n, \delta_K\}$, due to the lower bounds on PP [59] and on ST [43]. We present an online algorithm for CFLE in Algorithm 14.3. It is similar to Algorithm 13.4: we maintain a virtual solution of the online algorithm for FLE [60], which we denote by OFLE, and we sample clients as they arrive with probability $1/M$. Sampled clients are connected to the tree in a greedy manner, as in the online algorithm for ST [43], the corresponding facility lease given by the virtual solution is leased, and it is connected to the tree via the client. Non-sampled clients are connected to the closest active facility lease.

The algorithm is $O(K \lg n)$-competitive when $M = 1$. This result was published in [19];

Input: $(G, d, F, r, K, \delta, \gamma, M)$
$X \leftarrow \emptyset, \quad \mathcal{D} \leftarrow \emptyset, \quad \mathcal{D}^{\mathcal{T}} \leftarrow \emptyset, \quad \mathcal{T} \leftarrow (\{r\}, \emptyset)$;
set $\gamma^r_K \leftarrow 0$;
initialize OFLE with $(G, d, F, K, \delta, \gamma)$ and let (X', a') be the virtual solution;
when D_t arrives **do**
  foreach $j \in D_t$ **do**
    send (j, t) to OFLE and update virtual solution (X', a');
    add j to $\mathcal{D}^{\mathcal{T}}$ with probability $1/M$;
    if $j \in \mathcal{D}^{\mathcal{T}}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \text{min-path}(j, \mathcal{T})$;
      $(f, k, \hat{t}) \leftarrow a'(j, t)$;
      if $(f, k, \hat{t}) \notin X$ **then**
        $X \leftarrow X \cup \{(f, k, \hat{t})\}$;
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{(f, j)\}$;
    let $(f, k, \hat{t}) \in X$ be the closest active facility to j;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(j, t)\}, \quad a(j, t) \leftarrow (f, k, \hat{t})$;
return $(X, a, \mathcal{T})$;

Algorithm 14.3: Online algorithm for CFLE.

we omit the proof since is identical to that of Theorem 14.1. The same example described above shows that its competitive factor is $\Omega(n)$ if $M > 1$. Also, if we sample clients with probability 1, the same example of Section 13.1.1 shows that the resulting algorithm has competitive factor $\Omega(M)$ if $M > 1$.

14.2 Leasing-connected facility leasing

In LECFLE, we are given a complete graph $G = (V, E)$ with a metric distance between vertices $d : V \times V \mapsto \mathbb{Q}_+$, a set $F \subseteq V$ of potential facilities, a root facility $r \in V$, and for $t = 0, \ldots, T-1$, a set $D_t \subseteq V$ of clients arriving at instant t. However, we may have different leasing types for facilities and core edges. Thus, we are given K^F facility leasing lengths $\delta^F_1, \ldots, \delta^F_{K^F} \in \mathbb{N}$, and leasing facility $f \in F$ with leasing type $k \in [K^F]$ costs $\gamma^f_k \in \mathbb{Q}_+$. We also have K^E edge leasing lengths $\delta^E_1, \ldots, \delta^E_{K^E} \in \mathbb{N}$, edge leasing factors $\gamma^E_1, \ldots, \gamma^E_{K^E} \geq 1$, and leasing core edge $e \in E$ with leasing type k_e costs $d(e) \cdot \gamma^E_{k_e}$. We wish to find a set $X \subseteq F \times [K^F] \times \mathbb{Z}_+$ of facility leases, a function $a : \mathcal{D} \mapsto X$ that assigns a facility lease $a(j, t) \in X$ which is active at instant t to each client request $(j, t) \in \mathcal{D}$, and a set of edge leases $\mathcal{T} \subseteq E \times [K^E] \times \mathbb{Z}_+$. For each client request (j, t) with $a(j, t) = (f, k, \hat{t})$, there must exist a path P from f to r in G such that each edge $e \in P$ has some edge lease in $\mathcal{T}$ which is active at instant t. The goal is to find a solution which minimizes

$$\sum_{(f,k,\hat{t}) \in X} \gamma^f_k + \sum_{(j,t) \in \mathcal{D}} d(a(j, t), j) + \sum_{(e,k_e,\hat{t}_e) \in \mathcal{T}} \gamma^E_{k_e} \cdot d(e).$$

Note that we do not lease edges connecting clients to facilities.[1]

We propose, in Algorithm 14.4, an online algorithm for LeCFLe, which is $O(K^F \lg n + \lg K^E \lg |V|)$-competitive if $\gamma_1^E = 1$, where $n := \sum_{t=0}^{T-1} |D_t|$. The algorithm utilizes as subroutines (i) OFLe, which is $O(K \lg n)$-competitive for FLe [60], (ii) algorithm FRT, which, given a metric (V, d), returns a tree metric with expected distortion $O(\lg |V|)$ (see Section 10.1), and (iii) AlgRandOPP, the randomized $O(\lg K)$-competitive online algorithm for PP by Meyerson [59].

Input: $(G, d, F, r, K^F, \delta^F, \gamma, K^E, \delta^E, \gamma^E)$
$\mathcal{T} \leftarrow \emptyset$; **set** $\gamma^r_{K^F} \leftarrow 0$;
initialize OFLe with $(G, d, F, K^F, \delta^F, \gamma)$ and let (X, a) be the virtual solution;
$(\mathfrak{T}, c_{\mathfrak{T}}) \leftarrow \text{FRT}(V, d)$;
foreach edge e of $\mathfrak{T}$ **do**
  initialize an instance AlgRandOPP[e] with $(K^E, \delta^E, \gamma^E)$;
when D_t arrives **do**
  foreach $j \in D_t$ **do**
    $P \leftarrow \text{path}_{\mathfrak{T}}(j, r)$;
    foreach edge $e \in P$ **do**
      send 1 at instant t to AlgRandOPP[e], obtaining a permit (k_e, t_e);
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{(e, k_e, t_e)\}$;
    send (j, t) to OFLe and update the virtual solution (X, a);
    let $(f, k, \hat{t}) \leftarrow a(j, t)$; $\mathcal{T} \leftarrow \mathcal{T} \cup \{((f, j), 1, t)\}$;
return $(X, a, \mathcal{T})$;

Algorithm 14.4: Online algorithm for LeCFLe.

The algorithm builds, utilizing algorithm FRT, a tree $\mathfrak{T}$ with $V_{\mathfrak{T}} = V$ whose distances $O(\lg |V|)$-approximate the distances in G. Then, for each edge of the tree, the algorithm maintains an instance of algorithm AlgRandOPP. Besides, similarly to Algorithm 14.3, the algorithm maintains a virtual solution of algorithm OFLe. Algorithm 14.4, however, will always follow the decisions of the virtual solution. For each client (j, t) that arrives, the algorithm leases the edges in the path from j to r in $\mathfrak{T}$, using the leasing types suggested by the corresponding instances of algorithm AlgRandOPP. The algorithm leases the facility suggested by the virtual solution of algorithm OFLe, and connects the facility to the tree using an edge lease of type 1 through j.

Theorem 14.2: *Algorithm 14.4 is* $O(K^F \lg n + \lg K^E \lg |V|)$*-competitive when* $\gamma_1^E = 1$.

Proof: Given a solution returned by the algorithm, let L be the facility leasing cost, C the client connection cost, and S the core tree cost. Similarly, let L^*, C^* and S^* be those costs on an optimum solution.

Let L' be the facility leasing cost and C' be the client connection cost of the virtual solution produced by OFLe algorithm. We have that $L+C = L'+C' = O(K \lg n) \cdot \text{opt}_{\text{FLe}}$.

[1] Our definition is more general than if we leased edges between clients and facilities: since those edges would always be leased with type 1 because they are not reusable, that would be equivalent to divide all edge leasing costs by γ_1^E and have $\gamma_1^E = 1$. Similarly, in this variant we do not have a scaling parameter M; this is equivalent to multiply all edge leasing costs by M, thus our definition of the problem is more general, since edge leasing costs do not necessarily share a common divisor.

Since an optimum solution for CFLE induces a feasible solution for FLE, $\text{opt}_{\text{FLe}} \leq L^* + C^*$, so $L + C = \text{O}(K \lg n) \cdot (L^* + C^*)$.

We bound S by the cost of solving SLE on $\mathcal{D}$, plus the cost of an edge lease of type 1 between each client and its assigned facility lease, which is the connection cost for that client since $\gamma_1^E = 1$. Thus, $S \leq \text{O}(\lg K^E \lg |V|) \cdot \text{opt}_{\text{SLe}} + C$. Since the optimum core tree combined with an edge lease of type 1 between each client and its optimum facility induces a feasible solution for SLE on $\mathcal{D}$, we have that $\text{opt}_{\text{SLe}} \leq S^* + C^*$ and the theorem follows. □

We can improve this and obtain a deterministic $\text{O}((K^F + K^E) \lg n)$-competitive algorithm for $M = 1$ by using the online algorithm by Bienkowski, Kraska and Schmidt for STLE, which is $\text{O}(K \lg n)$-competitive [9]. However, this paper was published soon before the deadline for the thesis, so we do not present the improved algorithm here. It is quite similar to Algorithm 14.4, though.

If we replace algorithm ST with the algorithm by Anthony and Gupta for STLE, which is a $\text{O}(K)$-approximation [4], at Line 3 of Algorithm 14.1, then we obtain an offline algorithm for LECFLE which is a $\text{O}(K^E)$-approximation. The proof is similar to that of Theorem 14.1, so we omit it.

14.3 Multi-commodity connected facility leasing

In MCFLE, the input consists of a complete graph $G = (V, E)$, a metric distance $d : V \times V \mapsto \mathbb{Q}_+$, a set $F \subseteq V$ of potential facilities, K facility leasing lengths $\delta_1, \ldots, \delta_k \in \mathbb{N}$, a cost $\gamma_k^f \in \mathbb{Q}_+$ for leasing facility $f \in F$ with leasing type $k \in [K]$, a constant $M \geq 1$ and a sequence $P_0, \ldots, P_{T-1} \subseteq V \times V$ of sets of pairs of clients. We denote by $\mathcal{F} := F \times [K] \times \mathbb{Z}_+$ the set of possible facility leases. Let $X \subseteq \mathcal{F}$, $\mathcal{T} \subseteq E$, $u, v \in V$ and $t \in \mathbb{Z}_+$; we denote by $d_{X,\mathcal{T}}(u, v, t)$ the distance between u and v in the graph in which we add to G edges of cost zero between each pair of facility leases in X that are in the same tree in $\mathcal{T}$ and that are active at instant t. We also denote by $\mathcal{P} := \{(u, v, t) : (u, v) \in P_t \text{ for } t \in \{0, \ldots, T-1\}\}$ the set of client pair requests. The goal is to find a set $X \subseteq \mathcal{F}$ of facility leases and a forest $\mathcal{T} \subseteq E$ which minimize

$$\sum_{(f,k,t) \in X} \gamma_k^f + \sum_{(u,v,t) \in \mathcal{P}} d_{X,\mathcal{T}}(u, v, t) + M \cdot \sum_{e \in \mathcal{T}} d(e).$$

In this section we present an offline and an online algorithm for MCFLE, which are extensions of Algorithms 14.1 and 14.3. Both algorithms use as a subroutine algorithms for FLEP, in which we may choose not to serve a client request (j, t) by paying a penalty $\pi(j, t)$.

We present, in Algorithm 14.5, an offline algorithm for MCFLE; it is similar to Algorithm 13.5. The algorithm runs an instance of Algorithm 12.2, which is a 3-approximation for FLEP. For each pair $(u, v) \in P_t$, we assign a penalty of $d(u, v)/2$ to both (u, t) and (v, t). If both client requests are assigned to facility leases in the FLEP solution, then the algorithm includes a set consisting of these two clients in $\mathcal{D}'$; otherwise, the algorithm

will connect u and v directly. Then the algorithm runs the 2-approximation primal-dual algorithm for SF [2] on the pairs in $\mathcal{D}'$, and adds an edge between each client in $\mathcal{D}'$ and the facility assigned to the corresponding client request. The algorithm then returns only facility leases that serve client requests in $\mathcal{D}'$.

Input: $(G, d, F, K, \delta, \gamma, M, P_0, \ldots, P_{T-1})$
foreach $(u, v) \in P_t$ **do set** $\pi(u, t) \leftarrow \pi(v, t) \leftarrow d(u, v)/2$;
let $D_t \leftarrow \bigcup_{(u,v)\in P_t} \{u, v\}$;
$(X', a') \leftarrow \text{FLeP}(G, d, F, K, \delta, \gamma, D_0, \ldots, D_{T-1}, \pi)$;
$\mathcal{D}' \leftarrow \{\{(u, t), (v, t)\} : (u, v) \in P_t : a'(u, t) \neq \textsf{null}$ **and** $a'(v, t) \neq \textsf{null}\}$;
$X \leftarrow \{(f, k, \hat{t}) \in X' : \exists (u, t) \in \mathcal{D}' : a'(u, t) = (f, k, \hat{t})\}$;
$\mathcal{T} \leftarrow \text{SF}(G, d, \mathcal{D}') \cup \{(u, a'(u, t)) : (u, t) \in \mathcal{D}'\}$;
return $(X, \mathcal{T})$;

Algorithm 14.5: Approximation algorithm for MCFLE.

Theorem 14.3: *Algorithm 14.5 is an* 8*-approximation if* $M = 1$.

Proof: Given a solution returned by the algorithm, let L be the facility leasing cost, C the client connection cost, and S the core tree cost. Similarly, let L^*, C^* and S^* be those costs on an optimum solution.

Let L' be the facility leasing cost, C' be the client connection cost, and Π' be the penalty cost of the solution returned by the FLeP algorithm on $\mathcal{D} := \bigcup_{t=0}^{T-1} D_t$. We have that $L' + C' + \Pi' \leq 3 \cdot \text{opt}_{\text{FLeP}}$. Given an optimum solution for MCFLE on $\mathcal{P}$, we obtain a feasible solution for FLeP on $\mathcal{D}$ by paying the penalties for the clients in the pairs that are connected directly, and by assigning the other clients to the closest facility lease in the path connecting the pair. Thus, $\text{opt}_{\text{FLeP}} \leq L^* + C^*$ and $L' + C' + \Pi' \leq 3 \cdot (L^* + C^*)$.

Since Algorithm 14.5 leases a subset of the facility leases in X', clearly $L \leq L'$.

For a request (u, v, t) connected directly by Algorithm 14.5, we have that one of (u, t) and (v, t) pays for its penalty cost. Suppose w.l.o.g. that it is (u, t); then $d_{X,\mathcal{T}}(u, v, t) \leq 2 \cdot \pi(u, t)$. For a request (u, v, t) connected through the core tree, our algorithm adds to the tree a whole path between $a'(u, t)$ and $a'(v, t)$, so $d_{X,\mathcal{T}}(u, v, t) \leq d(u, a'(u, t)) + d(v, a'(v, t))$. Thus $C \leq 2 \cdot \Pi' + C'$.

Since $M = 1$, we bound S by the cost of solving SF on $\mathcal{D}'$, plus the cost of connecting each client in $\mathcal{D}'$ to its assigned facility lease. Thus, $S \leq 2 \cdot \text{opt}_{\text{SF}}(\mathcal{D}') + C' \leq 2 \cdot \text{opt}_{\text{SF}}(\mathcal{P}) + C'$. Since the optimum core tree combined with the optimum client connection edges induces a feasible solution for SF on $\mathcal{P}$, we have that $\text{opt}_{\text{SF}}(\mathcal{P}) \leq S^* + C^*$. Substituting the previous inequalities, we have that

$$\begin{aligned} L + C + S &\leq L' + 2 \cdot \Pi' + C' + 2 \cdot (S^* + C^*) + C' \\ &\leq 2 \cdot (S^* + C^*) + 2 \cdot (L' + \Pi' + C') \\ &\leq 2 \cdot (S^* + C^*) + 6 \cdot (L^* + C^*) \leq 8 \cdot (L^* + C^* + S^*) \ . \end{aligned}$$

□

In the online version of MCFLE, numbers T and $n := \sum_{t=0}^{T-1} |D_t|$ are unknown, sets P_t are revealed one at a time, and we must return sequences of sets $X_0, \ldots, X_{T-1} \subseteq \mathcal{F}$ and $\mathcal{T}_0, \ldots, \mathcal{T}_{T-1} \subseteq E$, where $X_t \supseteq X_{t-1}$ and $\mathcal{T}_t \supseteq \mathcal{T}_{t-1}$ for $t = 1, \ldots, T-1$, and for $t = 0, \ldots, T-1$, we must build $X_t, \mathcal{T}_t$ only with the information of $P_0, \ldots, P_t, X_0, \ldots, X_t$, $\mathcal{T}_0, \ldots, \mathcal{T}_t$. Also, the objective function is slightly different: minimize

$$\sum_{(f,k,\hat{t}) \in X_{T-1}} \gamma_k^f + \sum_{(u,v,t) \in \mathcal{P}} d_{X_t, \mathcal{T}_t}(u, v, t) + M \cdot \sum_{e \in \mathcal{T}_{T-1}} d(e);$$

i.e., we can only use facility leases and core edges bought up to instant t to connect clients arriving at instant t.

We give the following online algorithm for this problem. The algorithm combines ideas from Algorithms 14.5 and 14.3. We maintain a virtual solution of the $O(K \lg n)$-competitive online algorithm for FLEP by San Felice *et al.* [63], which we denote by OFLEP. We also maintain a virtual solution of the $O(\lg n)$-competitive online algorithm for SF by Berman and Coulston [8], which we denote by OSF. For each pair of clients (u, v) that arrives at instant t, we send two requests to algorithm OFLEP: (u, t) and (v, t), with $\pi(u, t) = \pi(v, t) = d(u, v)/2$. If algorithm OFLEP decides to pay the penalty for at least one of those requests, then we connect them directly. Otherwise, algorithm OFLEP leases a facility for each client request: we mimic this behavior and lease both facilities. Then we buy core edges connecting u and v via algorithm OSF. We also buy the edges between u, v and their respective facility leases.

Input: $(G, d, F, K, \delta, \gamma, M)$
1 $X_1 \leftarrow \emptyset$, $\mathcal{T}_1 \leftarrow \emptyset$;
2 initialize OFLEP with $(G, d, F, K, \delta, \gamma)$; **let** (X', a') be the virtual solution;
3 initialize OSF with (G, d) and **let** $\mathcal{T}'$ be the virtual solution;
4 **when** P_t arrives **do**
5 **if** $t > 0$ **then** $X_t \leftarrow X_{t-1}$, $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1}$;
6 **foreach** $(u, v) \in P_t$ **do**
7 **set** $\pi(u, t) \leftarrow \pi(v, t) \leftarrow d(u, v)/2$;
8 send $(u, t, \pi(u, t))$ and $(v, t, \pi(v, t))$ to OFLEP and update the virtual solution (X', a');
9 **if** $a'(u, t) \neq$ null **and** $a'(v, t) \neq$ null **then**
10 $X_t \leftarrow X_t \cup \{a'(u, t), a'(v, t)\}$;
11 send (u, v) to OSF and update the virtual solution $\mathcal{T}'$;
12 $\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \mathcal{T}' \cup \{(u, a'(u, t)), (v, a'(v, t))\}$;
13 **return** $(X_0, \ldots, X_{T-1}, \mathcal{T}_0, \ldots, \mathcal{T}_{T-1})$;

Algorithm 14.6: Online algorithm for MCFLE.

Theorem 14.4: *Algorithm 14.6 is* $O(K \lg n)$*-competitive if* $M = 1$.

Proof: Given a solution returned by the algorithm, let L be the facility leasing cost, C the client connection cost, and S the core tree cost. Similarly, let L^*, C^* and S^* be those costs on an optimum solution.

Denote by $\mathcal{D}$ the set of client requests as in Line 8 of the algorithm. Let L' be the

facility leasing cost, C' be the client connection cost, and Π' be the penalty cost of the virtual solution produced by OFLeP on $\mathcal{D}$. We have that $L'+C'+\Pi' = \mathrm{O}(K \lg n)\cdot \mathrm{opt}_{\mathrm{FLeP}}$. Given an optimum solution for MCFLe on $\mathcal{P}$, we obtain a feasible solution for FLeP on $\mathcal{D}$ by paying the penalties for the clients in the pairs that are connected directly, and by assigning the other clients to the closest facility lease in the path connecting the pair. Thus, $\mathrm{opt}_{\mathrm{FLeP}} \leq L^* + C^*$ and $L' + C' + \Pi' = \mathrm{O}(K \lg n) \cdot (L^* + C^*)$.

Since Algorithm 14.6 leases a subset of the facility leases in X', clearly $L \leq L'$.

For a request (u, v, t) connected directly by Algorithm 14.6, we have that one of (u, t) and (v, t) pays for its penalty cost. Suppose w.l.o.g. that it is (u, t); then $d_{X_t,\mathcal{T}_t}(u, v, t) \leq 2 \cdot \pi(u, t)$. For a request (u, v, t) connected through the core tree, our algorithm adds to the tree a whole path between $a'(u, t)$ and $a'(v, t)$, so $d_{X_t,\mathcal{T}_t}(u, v, t) \leq d(u, a'(u, t)) + d(v, a'(v, t))$. Thus $C \leq 2 \cdot \Pi' + C'$.

Let $\mathcal{D}'$ be the set of client requests that are connected through the core tree in Line 12 of Algorithm 14.6. Since $M = 1$, we bound S by the cost of the virtual solution produced by OSF on $\mathcal{D}'$, plus the cost of connecting each client in $\mathcal{D}'$ to its assigned facility lease. Thus, $S \leq \mathrm{O}(\lg n) \cdot \mathrm{opt}_{\mathrm{SF}}(\mathcal{D}') + C' \leq \mathrm{O}(\lg n) \cdot \mathrm{opt}_{\mathrm{SF}}(\mathcal{P}) + C'$. Since the optimum core tree combined with the optimum client connection edges induces a feasible solution for SF on $\mathcal{P}$, we have that $\mathrm{opt}_{\mathrm{SF}}(\mathcal{P}) \leq S^* + C^*$ and the theorem follows. □

We omit the presentation of the offline and online algorithms for MLeCFLe, since they follow the ideas in the algorithms for LeCFLe and MCFLe. Note, however, that the algorithm by Anthony and Gupta [4] solves STLe, not SLe. We described a $\mathrm{O}(\lg n)$-approximation algorithm for SLe on Section 10.2, which uses the same approach as the online algorithm by Meyerson [59]. Replacing this algorithm with algorithm SF on Line 6 of Algorithm 14.5, we obtain a $\mathrm{O}(\lg n)$-approximation algorithm for MLeCFLe. The online algorithm simply combines Algorithm 14.4 and Algorithm 14.6; the proof of the competitive factor is also similar.

Part V

Final Remarks

Chapter 15

Journey

In a first moment, the subject of this Ph.D. was incremental network design problems [39, 67]. Hartline [39] showed that, for any minimization covering problem that admits an α-approximation algorithm, there exists a 4α-competitive incremental algorithm. Our idea was to use study specific structural properties of network design problems and algorithms, in order to obtain better results than those yielded by this reduction. However, at that time, we concluded that this could not be achieved, and that this path was not very promising for a Ph.D. research. Nowadays, we still have the same opinion.

Then, we moved our research towards the leasing optimization model. In a first moment we proposed CFLe and LeCFLe. LeRoB, which had already been studied by Anthony and Gupta [4], inspired us to propose GPP. Our initial focus was only on online algorithms.

Our first result was Algorithm 14.3 for online CFLe, and the proof that it is $O(K \lg n)$-competitive when $M = 1$. We presented this result at ETC'16 [19]. In a first moment we believed that it was also $O(K \lg n)$-competitive for $M > 1$, but we could not prove it. We also tried to modify the online CFL algorithm by Umboh in order to obtain a good algorithm to CFLe, and eventually we obtained a very sophisticated algorithm that seemed to be competitive, but we could not analyze it.

At the same time we were working on GPP. Prof. Orlando proposed a simplification of the problem, which is MPP. The first result we obtained for those problems was Lemma 6.2. By that time I did not realize that GPP was not generalized by the 2DPP version studied by Hu *et al.* [41], so my only hope was to obtain a randomized $O(\lg K)$-competitive online algorithm for GPP. My idea was that GPP could be solved by combining contributions from single permits obtained via oracles for MPP. So we started to study how different GPP strategies worked for an hypothetical optimum oracle for MPP (since by that time we did not have an exact algorithm, even under IM). I thought that, if the strategy worked well with the optimum oracle, then it would also work well with an online oracle. In that direction, we got to rule out a simple strategy in which only permits of type k contribute to group permits of type k. We present an example which shows that this strategy has approximation factor $\Omega(K)$ in Appendix A. We also ruled out a strategy in which permits of types $1, \ldots, k$ contribute to group permits of type k, but which does not limit the contribution in each interval, as Algorithm 7.1 does. However, in this case I could only see that the strategy would lead to a $\Omega(K)$-competitive

online algorithm, even with an optimum oracle, and I could not formalize a generic family of bad instances.

In the meanwhile, Prof. Mário suggested that we tried to find an approximation algorithm for FLeP, in whose online version he had been working. We obtained then the 3-approximation algorithm of Section 12.2. Since we considered this an incremental result, we published it as a technical report [18]. Later, we published it at ETC'17 [21].

As the deadline to LAGOS'17 approached, we started to write our results, and we decided to submit two papers. We figured out that the strategy of combining Steiner and facility leasing algorithms for online CFLe when $M = 1$ also worked for the offline setting, for LeCFLe, and for multi-commodity versions of those problems. We also proved that the algorithms for CFLe had arbitrarily bad performance for $M > 1$. We submitted this first paper to LAGOS but it was rejected; we believe that the short number of pages did not help us. (Later on, we prepared a longer version to ICTCS'17 and it was accepted [20].) The other paper contained the results for MPP and SNLe [22]. As we were writing the papers for LAGOS, I also obtained the algorithm of Section 8.2 for H2DPP. Only after we submitted the paper I found out that H2DPP and O2DPP were different problems, and that GPP was not generalized by H2DPP. (Fortunately, we got to remove wrong sentences in the final version.)

By that time, I already had an intuition that Algorithm 7.1 was a 2-approximation to GPP under IM, but we could not prove it. In a certain moment, I thought that I had an example for which the algorithm had an arbitrarily bad approximation factor, until we could prove Lemma 7.3. Indeed, I believed this lemma to be true for a long time, and we already had the ideas for the rest of the proof of the approximation factor. The original proof was a bit more complicated, and involved an induction on the optimal substructure of the problem. As we began to write the proof, we figured out that one of the cases was generic enough, which yielded the proof we have now.

To obtain the online $O(K)$-competitive algorithm for GPP of Section 7.2 was a bit easier, since I was already familiar with the problem. I also believed that we had a pseudo-polynomial $O(KL)$-competitive online algorithm for O2DPP, but I figured out that it was not true when I found an error in a preliminary proof of the online algorithm for GPP.

Then we started to try to prove whether MPP, GPP and H2DPP were NP-hard or could be solved in polynomial time if we do not require IM. While trying to find a proof of NP-hardness for MPP, I got to prove that 2DPP is NP-hard by reducing CM to 2DPP. Then I remembered that, at the paper by Awerbuch and Azar on BaBND, they point that BaBND on a single edge "is an integer min-knapsack problem known to be NP-hard" [5]. If I had payed attention to that before, I could have found this result more easily, since for me it was already clear that LeBaBND on a single edge corresponds to 2DPP, and that LeBaBND is a generalization of BaBND.

The last result we found was that arbitrary instances of MPP (even not assuming IM) could be solved in polynomial time. Prof. Mário suggested that we considered primal-dual algorithms for MPP and GPP, so I decided to write a linear program formulation of those problems and to run computational experiments on random instances, in order to obtain more intuition about the problems. It turned out that the solver would always obtain an

integral solution for MPP, but for GPP there was an integrality gap even under IM. On the same day I started to search the Web for integrality gap (I did not remembered the results, which I had seen 8 years before in my Master's during Prof. Yoshiko's classes on approximation algorithms and combinatorial optimization), until I got to totally unimodular matrices (which I had seen in a Prof. Orlando's class in the first year of my Ph.D.) and figured out that MPP had such a matrix. (Indeed, it is a recurrent example of a simple totally unimodular matrix.)

Our last hope was to prove that GPP is weakly NP-hard under IM, but we could not do it. The deadline for the thesis was approaching, so we decided to stick with what we already had, and to submit the results about MPP, GPP and 2DPP to a journal [23].

Chapter 16

List of Results and Publications

Regarding parking permit problems, we obtained the following results:

1. We showed that MPP can be solved in polynomial time;
2. We showed an approximation-preserving reduction from MPP to PP, which yields a deterministic $O(K)$-competitive online algorithm and a randomized $O(\lg K)$-competitive online algorithm for MPP;
3. We obtained an 8-approximation algorithm and a $O(K)$-competitive online algorithm for GPP. We also proved that GPP has non-trivial integrality gap. We consider those to be the most interesting results in our thesis;
4. We showed how to turn the approximation and online algorithms for H2DPP by Hu *et al.* [41] into polynomial-time algorithms. We also showed that their original pseudo-polynomial approximation algorithm works for generic 2DPP;
5. We proved that 2DPP is NP-hard, even if we assume IM but do not assume HCP.

Those results yield the following results for network leasing problems:

6. For SNLe, we obtain a $O(\lg n)$-approximation algorithm and a $O(\lg K \lg |V|)$-competitive online algorithm;
7. For LeRoB, we obtain a $O(\lg n)$-approximation algorithm, which improves the previous best result, and a $O(K \lg |V|)$-competitive online algorithm;
8. For HLeBaBND, we obtain a $O(\lg n)$-approximation algorithm and a $O(K \lg |V|)$-competitive online algorithm;
9. For LeBaBND, we obtain a pseudo-polynomial $O(\lg n)$-approximation algorithm.

Those results have been submitted to

- M. S. de Lima, M. C. San Felice, and O. Lee. Group parking permit problems. *Discrete Applied Mathematics*, 2018 (submitted).

Items 2 and 6, and a mention to items 4 and 8 appeared in an earlier conference paper:

- M. S. de Lima, M. C. San Felice, and O. Lee. On generalizations of the parking permit problem and network leasing problems. In F. Bassino, F. Bonomo, L. Pournin, M. Valencia-Pabon, and J. C. V. Lizcano, editors, *LAGOS'17: IX Latin and American Algorithms, Graphs, and Optimization Symposium*, volume 62 of *Electronic Notes in Discrete Mathematics*, pages 225–230. Elsevier, 2017.

With regard to facility leasing problems, we obtained

10. a 3-approximation algorithm for FLeP,

which was published in

- M. S. de Lima, M. C. San Felice, and O. Lee. Facility leasing with penalties. In *CSBC'17: Anais do XXXVII Congresso da Sociedade Brasileira de Computação*, ETC'17: II Encontro de Teoria da Computação, pages 27–30, 2017.

We also obtained the following results for connected facility leasing problems:

11. A 7.39-approximation algorithm and a $O(K \lg n)$-competitive online algorithm for CFLe when $M = 1$;

12. A $O(K^E)$-approximation algorithm and a $O(K^F \lg n + \lg K^E \lg |V|)$-competitive online algorithm for LeCFLe when $\gamma_1^E = 1$;

13. An 8-approximation algorithm and a $O(K \lg n)$-competitive online algorithm for MCFLe when $M = 1$;

14. A $O(\lg n)$-approximation algorithm and a $O(K^F \lg n + \lg K^E \lg |V|)$-competitive online algorithm for MLeCFLe when $\gamma_1^E = 1$.

Those results were published in

- M. S. de Lima, M. C. San Felice, and O. Lee. Connected facility leasing problems. In *ICTCS'17: 18th Italian Conference on Theoretical Computer Science*, pages 162–173, 2017.

Also, the online algorithm of item 11 was published in

- M. S. de Lima, M. C. San Felice, and O. Lee. On a leasing variant of the online connected facility location problem. In *CSBC'16: Anais do XXXVI Congresso da Sociedade Brasileira de Computação*, ETC'16: I Encontro de Teoria da Computação, pages 836–839, 2016.

Chapter 17

Open Questions and Further Research Directions

Regarding parking permit problems, we leave the following questions open:

1. We do not know if, under IM, GPP is weakly NP-hard or can be solved in polynomial time. If the former is true, then an open question is whether we can obtain an FPTAS under IM.

2. If GPP is weakly NP-hard under IM, it can still be strongly NP-hard if we do not assume IM, even though the fact that MPP is polynomial is an evidence that this is not true. Another question is if we can obtain a better approximation algorithm than $4(1+\epsilon)$ if we do not assume IM.

3. We do not know if Algorithm 7.1 has approximation factor better than 2 under IM. We have a lower bound of 4/3, and experiments on random instances never attained approximation factor greater than 4/3.

4. Note that, if GPP is weakly NP-hard under IM, then O2DPP is weakly NP-hard even under 2DIM. NP-hardness and approximability questions analogue to those in items 1 and 2 apply to this problem.

5. Although H2DPP is weakly NP-hard if we assume IM but do not assume HCP, it is not clear whether we can obtain a polynomial-time algorithm if we do not assume IM but assume HCP.

6. It would be very nice if we could extend the ideas we developed for GPP to obtain a constant-approximation algorithm for O2DPP which runs in polynomial time. That would also improve the approximation result for OLeBABND by Anthony and Gupta [4].

7. We do not know a randomized $o(K)$-competitive online algorithm for GPP or H2DPP.

We also do not know if there is an algorithm for FLeP with approximation factor better than 3, and if there are good approximation and online algorithms for connected facility location problems when the (smallest) edge scaling factor is not a constant.

Other future research directions include:

- To study leasing variants of other facility location problems, such as capacitated facility location problems [15] and facility location problems with client latencies [13];
- To study generalizations such as GPP and 2DPP to variants of the parking permit problem, such as such as when a demand has a time window to be served after its arrival [55], or when leasing prices fluctuate along time [28]. It would be also interesting to study facility location and connected facility location problems in those leasing models.

Bibliography

[1] S. Abshoff, P. Kling, C. Markarian, F. M. auf der Heide, and P. Pietrzyk. Towards the price of leasing online. *Journal of Combinatorial Optimization*, 32(4):1197–1216, 2015.

[2] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[3] M. Andrews. Hardness of buy-at-bulk network design. In *FOCS'04: 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 115–124, 2004.

[4] B. M. Anthony and A. Gupta. Infrastructure leasing problems. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 424–438. Springer Berlin Heidelberg, 2007.

[5] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *FOCS'97: Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 542–547, 1997.

[6] B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized Steiner problem. *Theoretical Computer Science*, 324(2):313–324, 2004.

[7] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *FOCS'96: Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193, 1996.

[8] P. Berman and C. Coulston. On-line algorithms for Steiner tree problems (extended abstract). In *STOC'97: Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 344–353, 1997.

[9] M. Bienkowski, A. Kraska, and P. Schmidt. A deterministic algorithm for online Steiner tree leasing. In F. Ellen, A. Kolokolova, and J.-R. Sack, editors, *Algorithms and Data Structures*, volume 10389 of *Lecture Notes in Computer Science*, pages 169–180. Springer Berlin Heidelberg, 2017.

[10] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

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

[12] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1), 2013.

[13] D. Chakrabarty and C. Swamy. Facility location with client latencies: Linear programming based techniques for minimum latency problems. In O. Günlük and G. J. Woeginger, editors, *Integer Programming and Combinatorial Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 92–103. Springer Berlin Heidelberg, 2011.

[14] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *SODA'01: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651, 2001.

[15] X. Chen and B. Chen. Approximation algorithms for soft-capacitated facility location in capacitated network design. *Algorithmica*, 53(3):263–297, 2009.

[16] M. H. de Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, K. S. Guimarães, F. K. Miyazawa, J. C. de Pina Jr., J. A. R. Soares, and Y. Wakabayashi. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Publicações Matemáticas do IMPA, 2001.

[17] M. S. de Lima. Aproximação de métricas finitas por métricas arbóreas e aplicações. Master's thesis, Universidade de São Paulo, 2011.

[18] M. S. de Lima, M. C. San Felice, and O. Lee. Facility leasing with penalties. *arXiv preprint arXiv:1610.00575*, 2016.

[19] M. S. de Lima, M. C. San Felice, and O. Lee. On a leasing variant of the online connected facility location problem. In *CSBC'16: Anais do XXXVI Congresso da Sociedade Brasileira de Computação*, ETC'16: I Encontro de Teoria da Computação, pages 836–839, 2016.

[20] M. S. de Lima, M. C. San Felice, and O. Lee. Connected facility leasing problems. In *ICTCS'17: 18th Italian Conference on Theoretical Computer Science*, pages 162–173, 2017.

[21] M. S. de Lima, M. C. San Felice, and O. Lee. Facility leasing with penalties. In *CSBC'17: Anais do XXXVII Congresso da Sociedade Brasileira de Computação*, ETC'17: II Encontro de Teoria da Computação, pages 27–30, 2017.

[22] M. S. de Lima, M. C. San Felice, and O. Lee. On generalizations of the parking permit problem and network leasing problems. In F. Bassino, F. Bonomo, L. Pournin, M. Valencia-Pabon, and J. C. V. Lizcano, editors, *LAGOS'17: IX Latin and American Algorithms, Graphs, and Optimization Symposium*, volume 62 of *Electronic Notes in Discrete Mathematics*, pages 225–230. Elsevier, 2017.

[23] M. S. de Lima, M. C. San Felice, and O. Lee. Group parking permit problems. *Discrete Applied Mathematics*, 2018 (submitted).

[24] Z. Drezner and H. W. Hamacher, editors. *Facility Location: Applications and Theory*. Springer, 2002.

[25] F. Eisenbrand, F. Grandoni, T. Rothvoß, and G. Schäfer. Connected facility location via random facility sampling and core detouring. *Journal of Computer and System Sciences*, 76(8):709–726, 2010.

[26] A. N. Elmachtoub and R. Levi. From cost sharing mechanisms to online selection problems. *Mathematics of Operations Research*, 40(3):542–557, 2015.

[27] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[28] B. Feldkord, C. Markarian, and F. M. auf der Heide. Price fluctuation in online leasing. In X. Gao, H. Du, and M. Han, editors, *Combinatorial Optimization and Applications*, volume 10628 of *Lecture Notes in Computer Science*, pages 17–31. Springer Berlin Heidelberg, 2017.

[29] L. Fleischer, J. Könemann, S. Leonardi, and G. Schäfer. Strict cost sharing schemes for Steiner forest. *SIAM Journal on Computing*, 39(8):3616–3632, 2010.

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

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

[32] D. Fotakis and P. Koutris. Online sum-radii clustering. *Theoretical Computer Science*, 540–541:27–39, 2014.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[34] F. Grandoni and G. F. Italiano. Improved approximation for single-sink buy-at-bulk. In T. Asano, editor, *Algorithms and Computation*, volume 4288 of *Lecture Notes in Computer Science*, pages 111–120. Springer Berlin Heidelberg, 2006.

[35] F. Grandoni and T. Rothvoß. Approximation algorithms for single and multi-commodity connected facility location. In O. Günlük and G. J. Woeginger, editors, *Integer Programming and Combinatoral Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 248–260. Springer Berlin Heidelberg, 2011.

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

[37] A. Gupta, R. Ravi, K. Talwar, and S. Umboh. LAST but not least: Online spanners for buy-at-bulk. In *SODA'17: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 589–599, 2017.

[38] A. Gupta, A. Srinivasan, and É. Tardos. Cost-sharing mechanisms for network design. *Algorithmica*, 50(1):98–119, 2008.

[39] J. R. K. Hartline. *Incremental Optimization*. PhD thesis, Cornell University, 2008.

[40] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.

[41] X. Hu, A. Ludwig, A. Richa, and S. Schmid. Competitive strategies for online cloud resource allocation with discounts: The 2-dimensional parking permit problem. In *ICDCS'15: IEEE 35th International Conference on Distributed Computing Systems*, pages 93–102, 2015.

[42] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

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

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

[45] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM*, 50(6):795–824, 2003.

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

[47] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *FOCS'00: Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[48] A. R. Karlin. On the performance of competitive algorithms in practice. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 373–384. Springer Berlin Heidelberg, 1998.

[49] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1–4):79–119, 1988.

[50] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 3rd edition, 1998.

[51] G. Konjevod, R. Ravi, and F. S. Salman. On approximating planar metrics by tree metrics. *Information Processing Letters*, 80(4):213–219, 2001.

[52] C. Koufogiannakis and N. E. Young. Greedy Δ-approximation algorithm for covering with arbitrary constraints and submodular cost. *Algorithmica*, 66(1):113–152, 2013.

[53] P. Koutris. Infrastructure leasing problems. Master's thesis, National & Kapodistrian University of Athens, 2010.

[54] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

[55] S. Li, A. Mäcker, C. Markarian, F. M. auf der Heide, and S. Riechers. Towards flexible demands in online leasing problems. In D. Xu, D. Du, and D. Du, editors, *Computing and Combinatorics*, volume 9198 of *Lecture Notes in Computer Science*, pages 277–288. Springer Berlin Heidelberg, 2015.

[56] Y. Li, D. Du, N. Xiu, and D. Xu. Improved approximation algorithms for the facility location problems with linear/submodular penalties. *Algorithmica*, 73(2):460–482, 2015.

[57] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report CS-178, Department of Electrical Engineering, Princeton University, 1975.

[58] A. Meyerson. Online facility location. In *FOCS'01: Proceedings of the 42nd Symposium on Foundations of Computer Science*, pages 426–431, 2001.

[59] A. Meyerson. The parking permit problem. In *FOCS'05: 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–282, 2005.

[60] C. Nagarajan and D. P. Williamson. Offline and online facility leasing. *Discrete Optimization*, 10(4):361–370, 2013.

[61] Y. Rabinovich and R. Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete and Computational Geometry*, 19(1):79–94, 1998.

[62] M. C. San Felice, S.-S. Cheung, O. Lee, and D. P. Williamson. The online prize-collecting facility location problem. In M. Campêlo, R. Corrêa, C. Linhares-Sales, and R. Sampaio, editors, *LAGOS'15: VIII Latin-American Algorithms, Graphs and Optimization Symposium*, volume 50 of *Electronic Notes in Discrete Mathematics*, pages 151–156. Elsevier, 2015.

[63] M. C. San Felice, S.-S. Cheung, O. Lee, D. P. Williamson, and C. G. Fernandes. The online prize-collecting facility leasing problem. *(to be published)*, 2016.

[64] M. C. San Felice, C. G. Fernandes, and C. N. Lintzmayer. The online multicommodity connected facility location problem. In R. Solis-Oba and R. Fleischer, editors, *Approximation and Online Algorithms*, volume 10787 of *Lecture Notes in Computer Science*, pages 118–131. Springer Berlin Heidelberg, 2018.

[65] M. C. San Felice, D. P. Williamson, and O. Lee. A randomized O(log n)-competitive algorithm for the online connected facility location. *Algorithmica*, 76(4):1139–1157, 2016.

[66] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[67] A. M. Sharp. *Incremental algorithms: solving problems in a changing world*. PhD thesis, Cornell University, 2007.

[68] M. Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In W. J. Cook and A. S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 240–257. Springer Berlin Heidelberg, 2002.

[69] C. Swamy and A. Kumar. Primal–dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[70] K. Talwar. The single-sink buy-at-bulk LP has constant integrality gap. In W. J. Cook and A. S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 475–486. Springer Berlin Heidelberg, 2002.

[71] W. T. Tutte. Lectures on matroids. *Journal of Research National Bureau of Standards Section B*, 69:1–47, 1965.

[72] S. Umboh. Online network design algorithms via hierarchical decompositions. In *SODA'15: Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1373–1387, 2015.

[73] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[74] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

Appendix A

A Bad Example for a Simple GPP Strategy

Consider an algorithm for GPP which, given an optimum solution to MPP, buys a group permit of type k if the MPP algorithm buys $\lceil M \rceil$ permits of type k for the same interval. In principle, this seems like a good strategy, but we a have a family of examples which shows that this algorithm is a $\Omega(K)$-approximation even under IM.

The example has K types of permits, with arbitrary $K \geq 2$. The length of permit type k is $\delta_k = 2^{k-1}$, and the cost is $\gamma_k = 2^{k-1} - (2^{k-1} - 1)\epsilon$, for some $0 < \epsilon < \frac{1}{K-1}$. Note that, since $\epsilon < 1$, we have that $\gamma_k > \gamma_{k-1}$ and all permits are useful.

Also, note that $2 \cdot \gamma_{k-1} = 2 \cdot \left(2^{k-2} - (2^{k-2} - 1)\epsilon\right) = 2^{k-1} - (2^{k-1} - 2)\epsilon = \gamma_k + \epsilon > \gamma_k$, so those permits satisfy Equation (5.1).

Besides that, we show that those permits have the following property.

Fact A.1:

$$\sum_{k=1}^{K-1} \gamma_k < \gamma_K.$$

Proof:

$$\begin{aligned}
\sum_{k=1}^{K-1} \gamma_k &= \sum_{k=1}^{K-1} \left(2^{k-1} - \left(2^{k-1} - 1\right)\epsilon\right) = \sum_{k=0}^{K-2} 2^k - \epsilon \sum_{k=0}^{K-2} 2^k + (K-1)\epsilon \\
&= 2^{K-1} - 1 - \epsilon(2^{K-1} - 1) + (K-1)\epsilon \\
&< 2^{K-1} - (2^{K-1} - 1)\epsilon - 1 + (K-1) \cdot \frac{1}{K-1} = \gamma_K.
\end{aligned}$$

□

Take M integer such that

$$M > \frac{\left(\frac{K+1}{2}\right) 2^{K-1} + \epsilon}{2^{K-1} - 2^{K-1}\epsilon - \left(\frac{K-1}{2}\right)\epsilon}.$$

We define, then, for $k = 1, \ldots, K$, a sequence of requests r^k, with length 2^{k-1}. We define r^k in and inductive manner. For $k = 1$, take $r^1_0 := M - 1$. For $k > 1$,

$r_t^k := M - 1 + r_t^{k-1}$ for $0 \leq t < 2^{K-2}$, and $r_k^t := r_{(t-2^{K-2})}^{k-1}$ para $2^{K-2} \leq t < 2^{K-1}$. In Figure A.1 we show the sequence of requests for $M = 4$ and $k = 1, \ldots, 4$.[1]

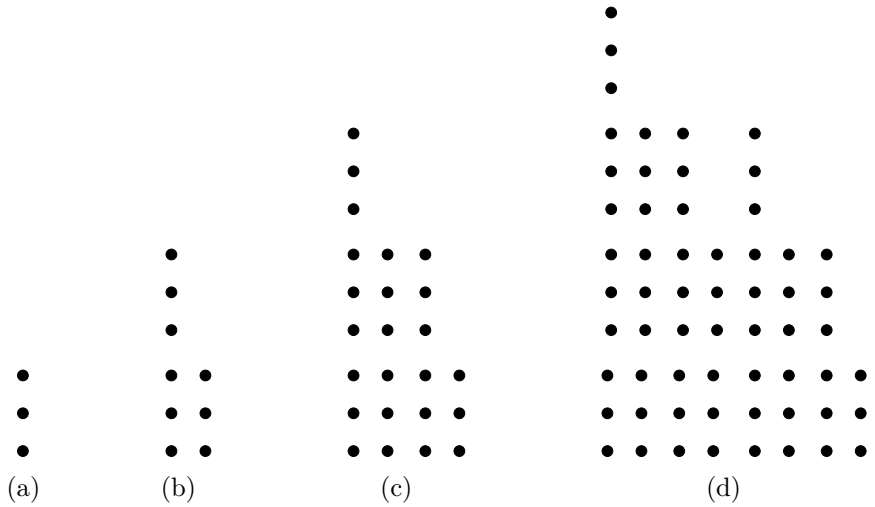


Figure A.1: Sequence r^k, with $M = 4$, for (a) $k = 1$, (b) $k = 2$, (c) $k = 3$, (d) $k = 4$.

The optimum solution for MPP for r^K buys $M - 1$ permits of type K to serve the lower $M - 1$ levels, plus the optimum solution for r^{K-1} to serve the left subinterval; for the subinterval in the right, we subdivide it in intervals of type $K - 1$, and we use the optimum solution for r^{K-2} for the left side, and proceed inductively for the right side. (See Figure A.2.) Note that, since $\sum_{k=1}^{K-1} \gamma_k < \gamma_K$, MPP does not buy permits of type K for levels above $M - 1$. Thus, we have that

$$\mathrm{opt}_{\mathrm{MPP}}(r^K) = (M-1) \cdot \gamma_K + \sum_{k=1}^{K-1} \mathrm{opt}_{\mathrm{MPP}}(r^k).$$

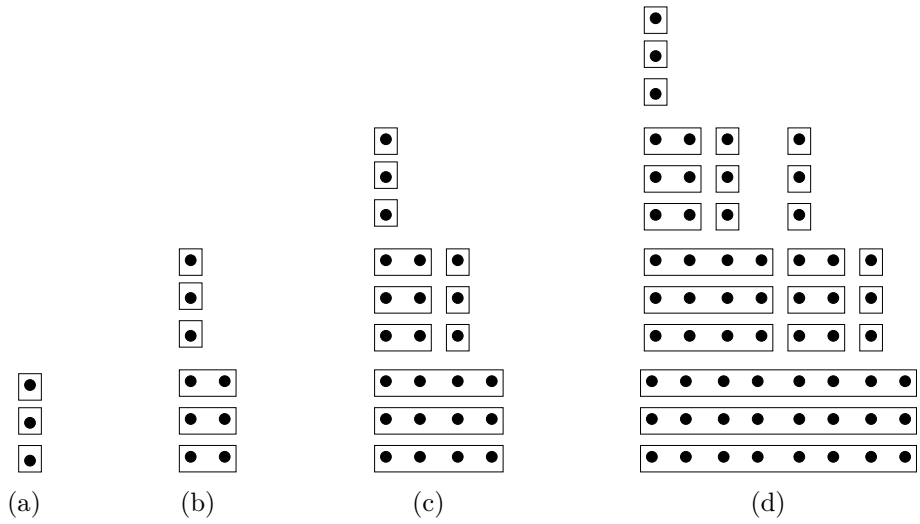


Figure A.2: Optimum MPP solution for r^k, (a) $k = 1$, (b) $k = 2$, (c) $k = 3$, (d) $k = 4$.

[1]Note that for $K = 4$ we need $M \geq 5$, but we use $M = 4$ in order to simplify the figures.

We prove by induction on K that

$$\mathrm{opt}_{\mathrm{MPP}}(r^K) = (M-1)\cdot\left(\gamma_K + \sum_{k=1}^{K-1} 2^{K-1-k}\gamma_k\right).$$

The case with $K = 1$ is trivial; for $K > 1$, by induction hypothesis, for $k = 1, \ldots, K-1$, we have that

$$\mathrm{opt}_{\mathrm{MPP}}(r^k) = (M-1)\cdot\left(\gamma_k + \sum_{\ell=1}^{k-1} 2^{k-1-\ell}\gamma_\ell\right),$$

so

$$\begin{aligned}
\mathrm{opt}_{\mathrm{MPP}}(r^K) &= (M-1)\cdot\gamma_K + \sum_{k=1}^{K-1}(M-1)\cdot\left(\gamma_k + \sum_{\ell=1}^{k-1} 2^{k-1-\ell}\gamma_\ell\right) \\
&= (M-1)\cdot\left(\gamma_K + \sum_{k=1}^{K-1}\gamma_k + \sum_{k=1}^{K-1}\sum_{\ell=1}^{k-1} 2^{k-1-\ell}\gamma_\ell\right) \\
&= (M-1)\cdot\left(\gamma_K + \sum_{k=1}^{K-1}\gamma_k + \sum_{k=1}^{K-1}\sum_{\ell=0}^{K-2-k} 2^{\ell}\gamma_k\right) \quad \text{(reordering summations)} \\
&= (M-1)\cdot\left(\gamma_K + \sum_{k=1}^{K-1}\gamma_k + \sum_{k=1}^{K-1}\gamma_k \sum_{\ell=0}^{K-2-k} 2^{\ell}\right) \\
&= (M-1)\cdot\left(\gamma_K + \sum_{k=1}^{K-1}\gamma_k + \sum_{k=1}^{K-1}\gamma_k\cdot(2^{K-1-k}-1)\right) \\
&= (M-1)\cdot\left(\gamma_K + \sum_{k=1}^{K-1} 2^{K-1-k}\gamma_k\right).
\end{aligned}$$

Note that the algorithm will never buy a group permit, since each interval of type k has at most $M-1$ permits of type k. The cost of a group permit of type K is

$$M\cdot\gamma_K = M\cdot\big(2^{K-1} - (2^{K-1}-1)\epsilon\big),$$

while the algorithm pays

$$
\begin{aligned}
& (M-1)\left(2^{K-1}-(2^{K-1}-1)\epsilon+\sum_{k=1}^{K-1}2^{K-1-k}\left(2^{k-1}-(2^{k-1}-1)\epsilon\right)\right) \\
= \;& (M-1)\left(2^{K-1}-(2^{K-1}-1)\epsilon+\sum_{k=1}^{K-1}\left(2^{K-2}-2^{K-2}\epsilon+\epsilon\cdot 2^{K-1-k}\right)\right) \\
= \;& (M-1)\left(2^{K-1}-(2^{K-1}-1)\epsilon+(K-1)2^{K-2}-(K-1)2^{K-2}\epsilon+2^{K-1}\epsilon\sum_{k=1}^{K-1}\frac{1}{2^k}\right) \\
\geq \;& (M-1)\left(2^{K-1}-(2^{K-1}-1)\epsilon+\frac{K-1}{2}2^{K-1}-\frac{K-1}{2}2^{K-1}\epsilon\right) \\
= \;& M\cdot\left(\frac{K-1}{2}\left(2^{K-1}-(2^{K-1}-1)\epsilon\right)+2^{K-1}-(2^{K-1}-1)\epsilon-\frac{K-1}{2}\epsilon\right) \\
& -\left(\frac{K+1}{2}\left(2^{K-1}-2^{K-1}\epsilon\right)+\epsilon\right) \\
\geq \;& M\frac{K-1}{2}\left(2^{K-1}-(2^{K-1}-1)\epsilon\right)+M\left(2^{K-1}-2^{K-1}\epsilon-\frac{K-1}{2}\epsilon\right) \\
& -\left(\frac{K+1}{2}2^{K-1}+\epsilon\right) \\
\geq \;& M\frac{K-1}{2}\gamma_K, \qquad \text{from the choice of } M.
\end{aligned}
$$