Group Parking Permit Problems[☆,☆☆]

Murilo Santos de Lima[a,1,*], Mário César San Felice[b,2], Orlando Lee[c,3]

[a]*Department of Computer Science, Reykjavik University, Iceland*
[b]*Department of Computing, Federal University of São Carlos (UFSCar), Brazil*
[c]*Institute of Computing, University of Campinas (UNICAMP), Brazil*

Abstract

In this paper we study some generalizations of the parking permit problem (Meyerson, FOCS'05), in which we are given a demand $r_t \in \{0,1\}$ for instant of time $t = 0,\ldots,T-1$, along with K permit types with lengths of time $\delta_1,\ldots,\delta_K$ and sub-additive costs. A permit is a pair $(k,\hat{t}) \in [K] \times \mathbb{Z}_+$, and it covers interval $[\hat{t},\hat{t}+\delta_k)$. We wish to find a minimum-cost set of permits that covers every t with $r_t = 1$. Meyerson gave deterministic $O(K)$-competitive and randomized $O(\lg K)$-competitive online algorithms for this problem, as well as matching lower bounds.

The first variant we propose is the multi parking permit problem, in which an arbitrary demand is given at each instant ($r_t \in \mathbb{Z}_+$) and we may buy multiple permits to serve it. We prove that the offline version of 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 a deterministic $O(K)$-competitive online algorithm and a randomized $O(\lg K)$-competitive online algorithm. For a leasing variant of the Steiner network problem, these results imply a $O(\lg n)$-approximation algorithm and a $O(\lg K \lg |V|)$-competitive online algorithm, where n is the number of requests and $|V|$ is the size of the input metric.

The second variant we propose is the group parking permit problem, in which we also have an arbitrary demand for each instant, and each permit of type k can be either a single permit, costing γ_k and covering one demand per instant of time, or a group permit, which costs $M \cdot \gamma_k$ for some constant $M \geq 1$ and covers an arbitrary number of demands in the interval covered by this permit. (I.e., group permits have infinite capacity.) For this version of the problem, we give an 8-approximation algorithm and a deterministic $O(K)$-competitive online algorithm. The first result yields an improvement on the previous best approximation algorithm for the leasing version of the rent-or-buy problem.

Finally, we study the 2D parking permit problem, proposed by Hu, Ludwig, Richa and Schmid (2015), in which a permit type is defined by a length of time and an integer capacity. They presented a constant approximation algorithm and a deterministic $O(K)$-competitive online algorithm for a hierarchical version of the problem, but those algorithms have pseudo-polynomial running time. We show how to turn their algorithms into polynomial time algorithms. Moreover, these results yield approximation and competitive online algorithms for a hierarchical leasing version of the buy-at-bulk network design problem. We also show that their original pseudo-polynomial offline algorithm works for a more general version of the 2D parking permit problem, which we prove to be NP-hard.

Keywords: parking permit, leasing optimization, rent-or-buy, approximation algorithms, online algorithms

[☆]DOI: 10.1016/j.dam.2019.05.013

[☆☆]A preliminary version of this paper appeared in volume 62 of ENDM, pp. 225–230. DOI: 10.1016/j.endm.2017.10.039

[*]Corresponding author

Email addresses: mslima@ic.unicamp.br (Murilo Santos de Lima), felice@ufscar.br (Mário César San Felice), lee@ic.unicamp.br (Orlando Lee)

[1]Partially supported by FAPESP/CAPES 2014/18781-1, CNPq 142161/2014-4, Icelandic Research Fund 174484-051.

[2]Partially supported by FAPESP 2017/11382-2.

[3]Partially supported by CNPq 311373/2015-1, CNPq 425340/2016-3, FAPESP 2015/11937-9.

1. Introduction

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 the global environmental issues, Johnny decided that he will always go walking when it does not rain, and he will drive otherwise. When he drives, he needs a parking permit. The parking lot has K different types of permits; a permit of type $k \in \{1, \ldots, K\}$ can be used for δ_k consecutive days, and it costs γ_k. Once a permit of type k is bought at day $\hat{t}$, it can be used at days $\hat{t}, \ldots, \hat{t} + \delta_k - 1$, so permits expire even when they are not used. The goal is to buy permits for rainy days by spending as little money as possible. We define the problem formally below. Throughout this paper, we denote by $[K] := \{1, \ldots, K\}$, $\mathbb{N}$ the set of positive natural numbers, $\mathbb{Z}_+$ the set of non-negative integers, $\mathbb{R}_+$ the set of non-negative real numbers, and $\mathbb{R}_+^*$ the set of positive real numbers.

Problem 1 (Parking Permit [32] (PP)). Given K permit types with lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$ and costs $\gamma_1, \ldots, \gamma_K \in \mathbb{R}_+^*$, along with a sequence $r_0, \ldots, r_{T-1} \in \{0, 1\}$, where $r_t = 1$ indicates that day t is rainy, find a set of permits $S \subseteq [K] \times \mathbb{Z}_+$ such that, for each rainy day t, there is some permit $(k, \hat{t}) \in S$ satisfying $t \in [\hat{t}, \hat{t} + \delta_k)$, which minimizes $\sum_{(k,\hat{t}) \in S} \gamma_k$.

This problem was proposed by Meyerson [32], and it admits a polynomial-time dynamic programming algorithm. In a more realistic online version, T is unknown and $r_0, \ldots, r_{T-1}$ are revealed one at a time. For this version, Meyerson [32] showed that the problem has 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.

PP is the seminal problem for the **leasing optimization model**, in which each resource may be leased for different periods of time, and it is more cost-effective to lease resources for longer periods. This contrasts with traditional optimization models, in which acquired resources last for unlimited duration. Leasing problems arise, for example, in the current trend among start-ups that prefer to lease servers in a cloud service rather than install their own servers [1, 7, 30]. Leasing optimization may be applied to both offline and online problems. Some literature has been devoted to leasing variants of classical optimization problems [1, 3, 7, 9, 10, 33]. Also, variants of PP and related problems were studied [12, 20, 24, 28, 31]. A traditional problem whose leasing variant was studied by Meyerson [32] is the Steiner forest problem. In the **Steiner leasing problem** (SLE), each edge can be leased for a limited amount of time; after the lease ceases, the edge is no longer available unless it is leased again. Meyerson presented a relationship between SLE and PP: if the input metric is a tree, SLE reduces to solve PP for each edge. Using the technique of approximating a metric by a tree metric [6, 11], a solution for a generic input can be obtained, losing some guarantee of quality. Meyerson obtained a $O(\lg K \lg |V|)$-competitive online algorithm for SNLE, where $|V|$ is the size of the input metric; the same idea yields a $O(\lg n)$-approximation for the offline setting, where n is the number of requests.

In this paper, we study three generalizations of PP, which can be used to solve leasing variants of three classical network design problems: the Steiner network problem, the rent-or-buy problem, and the buy-at-bulk network design problem. In the **Steiner network problem** (with edge duplication) (SN) [23, 37], we are given pairs of vertices and a demand $r(u, v)$ for each pair (u, v), and we wish to buy a minimum-cost multiset of edges that contains $r(u, v)$ edge-disjoints (u, v)-paths, for each pair (u, v). In the **rent-or-buy problem** (ROB) [25], we are given pairs of vertices, and we wish to buy some edges and, for each pair of vertices, rent some edges, so that rented and bought edges connect the corresponding pairs; the cost of buying an edge is M times the cost of renting it, for some constant $M \geq 1$. In the **buy-at-bulk network design problem** (BABND) [4], we also have pairs of vertices with demands, but we can install cables on each edge with different capacities per length, and we wish to install a minimum-cost multiset of cables that meet the total demand.

In the following paragraphs, we present the three generalizations of PP, which are the central problems studied in this paper. The first two problems are proposed by us.

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. Note that a demand greater than one can be given for each day, i.e., we receive a sequence $r_0, \ldots, r_{T-1} \in \mathbb{Z}_+$. Moreover, multiple copies of the same permit can be bought, so a solution is a multiset of permits. We give a formal definition below. Given a multiset B and an element $b \in B$, we denote by $m_B(b)$ the multiplicity of b in B.

Problem 2 (Multi Parking Permit (MPP)). Given K permit types with lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$ and costs $\gamma_1, \ldots, \gamma_K \in \mathbb{R}_+^*$, along with a sequence $r_0, \ldots, r_{T-1} \in \mathbb{Z}_+$, find a multiset of permits $S \subseteq [K] \times \mathbb{Z}_+$ such that $\sum_{\substack{(k,\hat{t}) \in S \\ t \in [\hat{t}, \hat{t}+\delta_k)}} m_S(k, \hat{t}) \geq r_t$ for each t, which minimizes $\sum_{(k,\hat{t}) \in S} m_S(k, \hat{t}) \cdot \gamma_k$.

This problem reduces to PP if $r_t \leq 1$ for every day t.

In the second problem we propose, 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. Below we give a formal definition of the problem.

Problem 3 (Group Parking Permit (GPP)). Given K permit types with lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$ and costs $\gamma_1, \ldots, \gamma_K \in \mathbb{R}_+^*$, along with a sequence $r_0, \ldots, r_{T-1} \in \mathbb{Z}_+$ and a constant $M \geq 1$, 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, 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

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

This problem reduces to PP if $r_t \leq 1$ for every day t, to MPP if $M = \infty$, and to the ski rental problem [26] if $K = 1$ and $\delta_1 = \infty$.

Finally, we also study the **2D parking permit problem**, which was proposed by Hu *et al.* [20]. As in MPP and GPP, at each instant we receive a non-negative 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. Below we present a formal definition.

Problem 4 (2D Parking Permit [20] (2DPP)). Given K permit types with lengths $\delta_1, \ldots, \delta_K \in \mathbb{N}$, costs $\gamma_1, \ldots, \gamma_K \in \mathbb{R}_+^*$, and capacities $\phi_1, \ldots, \phi_K \in \mathbb{N}$, along with a sequence $r_0, \ldots, r_{T-1} \in \mathbb{Z}_+$, find a multiset $S \subseteq [K] \times \mathbb{Z}_+$ of permits, such that, for $t = 0, \ldots, T-1$, we have that $\sum_{\substack{(k,\hat{t}) \in S \\ t \in [\hat{t}, \hat{t}+\delta_k)}} m_S(k, \hat{t}) \cdot \phi_k \geq r_t$, which minimizes $\sum_{(k,\hat{t}) \in S} m_S(k, \hat{t}) \cdot \gamma_k$.

We identify two important particular cases of 2DPP. The first is what we call the **orthogonal 2D parking permit problem** (O2DPP), in which 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 $\gamma_1, \ldots, \gamma_K \in \mathbb{R}_+^*$, and L capacities $\phi_1, \ldots, \phi_L \in \mathbb{N}$ with corresponding capacity scaling costs $\mu_1, \ldots, \mu_L \in \mathbb{R}_+^*$. A permit with length of time δ_k and capacity ϕ_ℓ costs $\gamma_k \cdot \mu_\ell$. Note that O2DPP reduces to GPP if $L = 2$, $\phi = (1, \infty)$ and $\mu = (1, M)$. We call the second case 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 $k > k'$. This version of the problem was proposed by Hu *et al.* [20]. 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.

Those parking permit problems can be used to solve leasing variants of SN, RoB and BaBND, via the technique of approximating a metric by a tree metric [6, 11], in a similar fashion as Meyerson used PP to solve SLe. If the input metric is a tree, then: (i) the leasing variant of SN reduces to MPP; (ii) the leasing variant of RoB reduces to GPP; (iii) the leasing variant of BaBND reduces to 2DPP. (We can also define hierarchical and orthogonal versions of leasing BaBND, as we did for 2DPP.) In Figure 1 we depict the dependency between the problems we study in this paper.

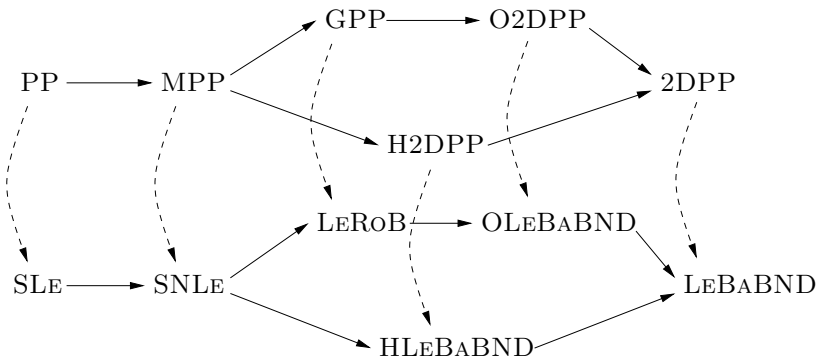

Figure 1: A graph depicting dependency between the problems we study. A full arrow $A \to B$ indicates that problem A is a particular case of problem B. A dashed arrow $A \dashrightarrow B$ indicates that parking permit problem A is a particular case of network leasing problem B, and B reduces to solving A for each edge if the input metric is a tree.

In some sense, the goal of this paper is to understand how time dynamicity, which is the central issue in PP, interacts with **capacity dynamicity**. The capacity dynamicity issue is well-solved for a fixed resource, since it consists in the ski rental problem [26]. (Even though the ski rental problem is usually defined with time as a parameter, we note that capacity is a better interpretation for it.) The ski rental problem 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 the ski rental problem, the first dealing with time dynamicity, and the second dealing with capacity dynamicity.

Going one step further, the ski rental problem can be generalized to multiple types of skis, each lasting for a different amount of time (capacity). In the offline setting, this problem has an FPTAS algorithm [21]. It also admits a 2-competitive online algorithm [4, 32]. The combination of PP with this generalization of the ski rental problem is, then, 2DPP.

In a first glance, it looks like time dynamicity and capacity dynamicity are independent issues. We conjecture that this is not true, and in particular the results we present in this paper point in the direction that more sophisticated techniques are necessary when those issues are combined in the same problem.

Related Work. Besides the work of Meyerson on PP and SLe [32], the previous main work on network leasing problems is due to Anthony and Gupta [3]. The authors reduce the problem of solving an optimization problem with K lease types in the offline setting to the corresponding K-stage stochastic problem, while losing a constant cost factor. By using previous results in K-stage stochastic optimization, the authors obtained a $O(K)$-approximation algorithm for the single-source case of SLe[4], as well as approximation algorithms for leasing variants of facility location, vertex cover and set cover. The authors also propose new K-stage stochastic algorithms for single-source RoB, single-source BaBND and multi-source BaBND. That way, they obtained $O(K)$-approximation algorithms for the single-source cases of leasing RoB and orthogonal leasing BaBND, as well as a $O(K \lg n)$-approximation for multi-source orthogonal leasing BaBND. This last result consists of a $O(K)$-approximation for a generalization of O2DPP, combined with the technique of approximating a metric by a tree metric [6, 11].

[4]Note that the single-source case of SLe is the Steiner tree leasing problem.

More recently, Bienkowski, Kraska and Schmidt [7] gave a deterministic $O(K \lg n)$-competitive online algorithm for single-source SLE. Their algorithm does not use a simple application of the metric approximation technique; instead, they build upon the technique of bounding the cost of a greedy-like algorithm via a hierarchical tree decomposition of the input metric. This technique was recently proposed by Umboh [37] for solving various online network design problems.

Other literature on leasing variants of classical optimization problems include the following. Nagarajan and Williamson [33] presented primal-dual algorithms for offline and online facility leasing. In particular, they modified and reanalyzed the primal-dual algorithm for the online facility location problem by Fotakis [14], which is then extended to solve the online facility leasing problem. Abshoff *et al.* [1] built upon the work of Nagarajan and Williamson, obtaining an improved algorithm for the online facility leasing problem; they also solve the online set multicover leasing problem. More recently, de Lima, San Felice and Lee [9] combined various algorithms for facility leasing and Steiner problems to solve a particular case of offline and online leasing variants of the connected facility location problem. In another paper, the same authors presented an approximation algorithm for the facility leasing problem with penalties [10], which is a primal-dual algorithm that combines ideas from the algorithm by Nagarajan and Williamson for the facility leasing problem [33] and the algorithm by Charikar *et al.* [8] for the facility location problem with penalties.

H2DPP was proposed by Hu *et al.* [20]. The authors presented a constant-approximation algorithm, which is based on dynamic programming, and a deterministic $O(K)$-competitive online algorithm, which is pretty similar to Meyerson's online algorithm for PP. Both those algorithms, however, have pseudo-polynomial running time. They conjectured that H2DPP is NP-hard, which we prove in this paper.

Another form of defining O2DPP is supposing we are given a sub-additive function $\mu' : \mathbb{Z}_+ \to \mathbb{R}_+$ 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 our definition [35]. There is a $O(K)$-approximation algorithm for this version of O2DPP, which was given for a more general problem by Anthony and Gupta [3] (we discuss this in Section 6.3).

MPP was studied independently by Jin, Hayashi and Tagami [24]. The authors proved that the problem admits an offline exact algorithm in the interval model, as well as deterministic $O(K)$-competitive and randomized $O(\lg K + \lg \delta_K)$-competitive online algorithms.

Other variants of PP were also studied. Li *et al.* [28] study the case when a demand does not need to be served immediately when it arrives, but instead has a deadline which must be respected. Feldkord, Markarian and auf der Heide [12] discuss what happens when leasing costs are allowed to fluctuate over time. Markarian [31] extends the leasing model for the case in which a demand can be refused by paying a penalty, and for the case in which a demand may be delayed by incurring a penalty per unit of delayed time. Mäcker [30] recently proposed a different model for cloud resource allocation, in which a resource can be leased for an arbitrary length of time and a cost per unit of time is specified.

Our Contribution. The main results of this paper concern approximation and competitive online algorithms for MPP, GPP and 2DPP.

We show that MPP has a linear program formulation whose matrix is totally unimodular, so it can be solved in polynomial time. We also show how to reduce MPP to PP while losing a constant cost factor. This reduction is based: (i) on the **Interval Model**, proposed by Meyerson [32] to add structure to permits; (ii) on a particular ordering of the permits and an assignment between permits and demands, which we call the **Hanoi tower ordering**; and (iii) on an exponential sampling, which allows us to perform the reduction in polynomial time. This approximation-preserving reduction yields deterministic $O(K)$-competitive and randomized $O(\lg K)$-competitive online algorithms for MPP. Those results are asymptotically optimal, since the lower bounds for PP [32] (deterministic $\Omega(K)$ and randomized $\Omega(\lg K)$) also apply to MPP. This problem was studied independently by Jin, Hayashi and Tagami [24], and the authors obtained similar results to ours. However, our algorithms are simpler, and they do not prove that the problem can be solved exactly in polynomial time even if the Interval Model is not assumed.

The most relevant contribution in this paper are our results for GPP. We conjecture that the problem is weakly NP-hard; one evidence towards this fact is that the linear program formulation has non-trivial integrality gap. We then present an 8-approximation algorithm for GPP. This algorithm is also based

on the Interval Model and on an extension of the Hanoi tower ordering to GPP. It performs an iterative improvement of the solution, considering group permits of increasing types. In particular, the analysis of this algorithm is quite 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 algorithm by a hypothetical algorithm that knows some choices of the optimum solution. We also present a deterministic $O(K)$-competitive online algorithm, which is asymptotically optimal due to the deterministic lower bound of $\Omega(K)$ for PP [32]. This algorithm uses our approximation algorithm to update its decisions, in a similar way to Meyerson's online algorithm for PP [32], except that his algorithm uses an optimum offline solution. Our analysis, however, is different because it uses information of the offline approximate solution at instant t to update the bound on the solution obtained by the online algorithm up to instant $t-1$. This analysis builds on the concept of delimiting the "amount of regret" of the algorithm, which is an important tool for the application of online algorithms in machine learning [19]. The same result can be obtained via the algorithm by Koufogiannakis and Young for the covering problem with linear costs [27]; our algorithm, however, was obtained independently, and we add a substantial contribution in terms of analysis technique.

We then prove that 2DPP is NP-hard, which was conjectured by Hu *et al.* [20]. For H2DPP, Hu *et al.* gave a constant approximation algorithm and a deterministic $O(K)$-competitive online algorithm, which is asymptotically optimal due to the deterministic lower bound of $\Omega(K)$ for PP [32]. However, those algorithms have pseudo-polynomial running time. We show how to turn the offline algorithm, which is a dynamic programming algorithm, into a polynomial time algorithm, by using a proper data structure and a binary search on the dynamic programming. This turns the online algorithm into a polynomial time one as well, since the offline algorithm is used to update the online algorithm's decisions. We also show that their original pseudo-polynomial algorithm works for generic 2DPP, and we point that it can be implemented more efficiently than the authors claimed. Moreover, we discuss how the algorithm by Koufogiannakis and Young for the covering problem with linear costs [27] can be used to obtain some general results for 2DPP.

By combining our results for MPP, GPP and 2DPP with the technique of approximating metrics by tree metrics, we obtain the following results for network leasing problems: (i) for the leasing variant of SN, a $O(\lg n)$-approximation algorithm and a $O(\lg K \lg |V|)$-competitive online algorithm, where n is the number of request pairs and $|V|$ is the input metric size; (ii) for the leasing variant of RoB, a $O(\lg n)$-approximation algorithm and a $O(K \lg |V|)$-competitive online algorithm; (iii) for the hierarchical leasing variant of BaBND, a $O(\lg n)$-approximation algorithm and a $O(K \lg |V|)$-competitive online algorithm; (iv) for the general case of leasing BaBND, a pseudo-polynomial $O(\lg n)$-approximation algorithm. In particular, our results improve the multi-source case for leasing SN and leasing RoB[5], for which the previous best result was the $O(K \lg n)$-approximation algorithm for multi-source orthogonal leasing BaBND by Anthony and Gupta [3]. We also improve the result for hierarchical leasing BaBND, which was a $O(K \lg n)$-approximation obtained by combining the results in [27] with the metric approximation technique.

Organization of the Paper. In Section 2, we discuss the Interval Model, which is useful to solve parking permit and leasing optimization problems. Sections 3, 4 and 5 are devoted to MPP, GPP and 2DPP, respectively. In Section 6, we present the consequences of the results in the previous sections to the leasing variants of SN, RoB and BaBND. Finally, in Section 7, we present a summary of known results (see Tables 1 and 2), some open questions and future research directions.

2. The Interval Model

In order to give more structure to parking permit problems, it is useful to adopt the following assumptions [3, 32].

Hypothesis 5 (Simple Interval Model (SIM)). Permits of type k can only begin at instants $c \cdot \delta_k$, for $c \in \mathbb{Z}_+$.

[5]Even though the standard definition of RoB does not generalize SN, leasing RoB generalizes leasing SN. We discuss this in Section 6.2.

Hypothesis 6 (Hierarchical Length Property (HLP)). δ_{k-1} divides δ_k for $k = 2, \ldots, K$.

Note that SIM is a restriction on the solutions, while HLP is a hypothesis on the instance. In what follows, we state Facts 7 and 8 and Lemma 10 for 2DPP, but they extend to PP, MPP and GPP straightforwardly, since the proofs do not use any specific structure of 2DPP. Furthermore, the reduction can be performed in an online manner, so the results also apply for online algorithms. We present proofs for the sake of completion, but they are identical as those in [3, 32].

Fact 7. *An α-approximation algorithm for* 2DPP *under* SIM *is a 2α-approximation if we do not require* SIM. *Conversely, if there is an α-approximation algorithm for* 2DPP *not requiring* SIM, *then there is a 2α-approximation algorithm under* SIM.

Proof. ($\Rightarrow$) We just have to bound an optimum solution $\hat{S}$ under SIM with regard to an arbitrary optimum solution S^*. By simply replacing each permit $(k, \hat{t}) \in S^*$ with permits $(k, \lfloor \hat{t}/\delta_k \rfloor \cdot \delta_k)$ and $(k, \lceil \hat{t}/\delta_k \rceil \cdot \delta_k)$ (which may be the same permit), we obtain a solution which respects SIM, costs at most twice the cost of S^*, and costs at least the cost of $\hat{S}$ (since $\hat{S}$ is optimum under SIM).

($\Leftarrow$) Given a solution S' that does not respect SIM, we can obtain a solution that respects SIM by replacing each permit $(k, \hat{t}) \in S'$ with permits $(k, \lfloor \hat{t}/\delta_k \rfloor \cdot \delta_k)$ and $(k, \lceil \hat{t}/\delta_k \rceil \cdot \delta_k)$ (which may be the same permit). Given an optimum solution $\hat{S}$ under SIM and an arbitrary optimum solution S^*, clearly $\text{cost}(S^*) \leq \text{cost}(\hat{S})$, so the claim follows. □

Fact 8. *If there is an α-approximation algorithm for* 2DPP *under* HLP, *then there is a 2α-approximation algorithm for instances that do not satisfy* HLP.

Proof. Given an instance $I = (T, K, \delta, \gamma, \phi, r)$ that does not satisfy HLP, we obtain an instance $I' = (T, k, \delta', \gamma, \phi, r)$ that satisfies HLP in the following manner. Assume $\delta_1 \leq \cdots \leq \delta_K$, and take $\delta'_k := 2^{\lfloor \lg \delta_k \rfloor}$, i.e., round down each permit length to the closest power of 2. Any feasible solution for I' is feasible for I, since $\delta'_k \leq \delta_k$, so it remains to bound $\text{opt}(I')$ with regard to $\text{opt}(I)$. Given an optimum solution S^* for I, replace each permit $(k, \hat{t}) \in S^*$ with $(k, \hat{t})$ and $(k, \hat{t} + \delta'_k)$; since $\delta_k < 2 \cdot \delta'_k$, the resulting solution is feasible for I', thus it costs at least $\text{opt}(I')$, and it costs twice as $\text{opt}(I)$. □

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

Hypothesis 9 (Interval Model (IM)). We assume SIM and HLP.

The next lemma follows by applying Facts 7 and 8.

Lemma 10. *If there is an α-approximation algorithm for* 2DPP *under* IM, *then there is a 4α-approximation algorithm for arbitrary instances.*

The main property of the Interval Model is that optimum solutions of intervals of length δ_K are independent. This can be generalized in the following manner. We state Definitions 11 and 12 and Lemma 13 in terms of 2DPP, but those also apply to PP, MPP and GPP, since these are particular cases of the former.

Consider an instance $I = (T, K, \delta, \gamma, \phi, r)$ of 2DPP. For some forthcoming proofs and algorithms, it is convenient to define restricted instances of I in the following manner.

Definition 11 ($I[k]$). Given a permit type $k \in [K]$, let $I[k] := (T, k, \delta, \gamma, \phi, r)$ be the sub-instance of I in which we can only use permits of type $1, \ldots, k$.

Definition 12 ($I[k, \hat{t}]$). Given a permit type $k \in [K]$ and an instant of time $\hat{t} = c \cdot \delta_k$ with $c \in \mathbb{Z}_+$, let $I[k, \hat{t}] := (\delta_k, k, \delta, \gamma, \phi, (r_{\hat{t}}, \ldots, r_{\hat{t}+\delta_k-1}))$ be the sub-instance of $I[k]$ restricted to the interval of length δ_k beginning at instant $\hat{t}$.

Lemma 13. *Let I be an instance of* 2DPP *and let $k \in [K]$ be a permit type. Then, under* IM*, we have that*

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

I.e., solving $I[k]$ is equivalent to solving independently each sub-instance of length δ_k and taking the union of the corresponding solutions. (The proof of this lemma follows from a standard optimal substructure argument.)

For PP, MPP and GPP, under IM, we can assume without loss of generality that

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

i.e., permit costs are sub-additive and represent economies of scale. If $\delta_1 > 1$, we can simply divide each δ_k by δ_1 and rescale the instance by having the maximum demand for each interval of length δ_1. If two permit types k and k' with $k > k'$ do not satisfy Equation (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. We discuss variations of this assumption for 2DPP in Section 5.

3. The Multi Parking Permit Problem

In this section, we discuss MPP. 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 [36] and, since $\mathbf{r}$ is integer, it follows that all extreme points of P are integral [34]. 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. We will assume IM and Equation (1).

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 2(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 2(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.

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 $R := \max_{t=0,\ldots,T-1} r_t$ and, for $j = 1, \ldots, R$, let $I^j := (T, K, \delta, \gamma, r^j)$ be an instance of PP such that, for $t = 0, \ldots, T-1$, $r^j_t = 1$ if $r_t \geq j$, and $r^j_t = 0$ otherwise. (I^j is the PP instance corresponding to level j.)

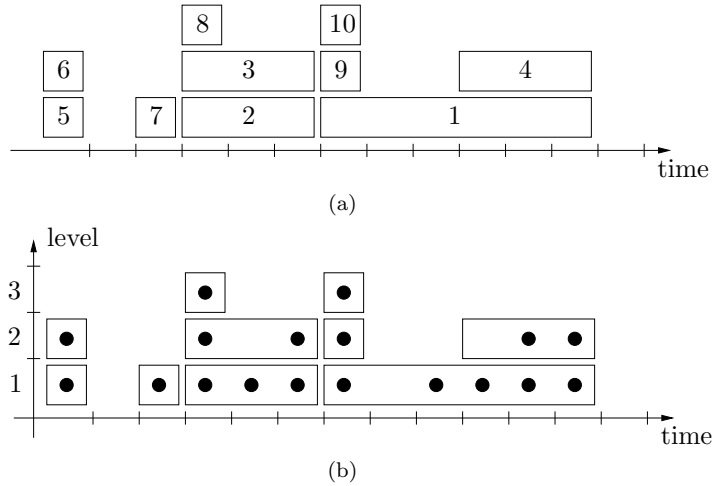

Figure 2: (a) Example of a MPP solution ordered in HTO, given an instance with $r = (2, 0, 1, 3, 1, 2, 3, 0, 1, 1, 2, 2)$ as the demand sequence. (b) Demands of this instance assigned to those permits.

Lemma 14. *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 will 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 3.) 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. □

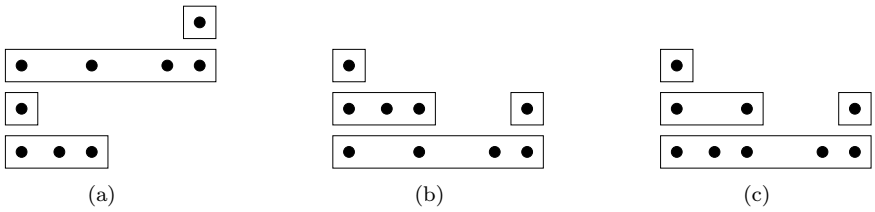

Figure 3: An illustration of the proof of Lemma 14. Consider an instance with $T = 6$, $K = 3$, $\delta = (1, 3, 6)$, $\gamma = (1, 5/2, 4)$ and $r = (3, 1, 2, 0, 1, 2)$. (a) The optimum solution is $\{(2, 0), (1, 0), (3, 0), (1, 5)\}$. (b) Optimum solution after reordering permits. (c) Optimum solution after reassigning demands; permits correspond to optimum solutions for PP instances.

So under IM, from Lemma 14, 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.[6] By Lemma 10, we have deterministic $O(K)$-competitive and randomized $O(\lg K)$-competitive online algorithms for MPP. Note that Lemma 14 is valid only if we assume IM. (See a counterexample if we do not assume IM in Figure 4.) 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 from the online algorithm for SN [37].

[6]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.

(a) (b)

Figure 4: Lemma 14 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.

Lemma 15. *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 14. By Lemma 14, 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 [32], we have the following result.

Theorem 16. *There are polynomial-time deterministic* $\mathrm{O}(K)$*-competitive and randomized* $\mathrm{O}(\lg K)$*-competitive online algorithms for* MPP.

This is asymptotically optimal due to the deterministic $\Omega(K)$ and randomized $\Omega(\lg K)$ lower bound for PP [32].

4. The Group Parking Permit Problem

In this section, we discuss GPP. In order to simplify our notation, we denote the total cost of a multiset of permits S by $\mathrm{cost}(S) := \sum_{(k,\hat{t}) \in S} m_S(k, \hat{t}) \cdot \gamma_k$. Thus, the cost of a solution (S, Q) is $\mathrm{cost}(S) + M \cdot \mathrm{cost}(Q)$. Throughout this section, we assume IM and Equation (1).

4.1. 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.[7] 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, proposed by Hu *et al.* [20]. We discuss this algorithm in Section 5.1.

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

Definition 17 ($S[k,\hat{t}]$). 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 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.

Input: $(T, K, \delta, \gamma, r, M)$
$S_0 \leftarrow \text{AlgMPP}(T, K, \delta, \gamma, r)$, $Q_0 \leftarrow \emptyset$;
for $k \leftarrow 1$ **to** K **do**
  $S_k \leftarrow S_{k-1}$, $Q_k \leftarrow Q_{k-1}$;
  for $\hat{t} \leftarrow 0$ **to** $T-1$ **step** δ_k **do**
    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**
      $S_k \leftarrow S_k \setminus S_{k-1}[k,\hat{t}]$;
      $Q_k \leftarrow (Q_k \setminus Q_{k-1}[k,\hat{t}]) \cup \{(k,\hat{t})\}$;
return (S_K, Q_K);

Algorithm 1: Approximation algorithm for GPP.

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 permits of types $1,\ldots,k$

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

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 5.

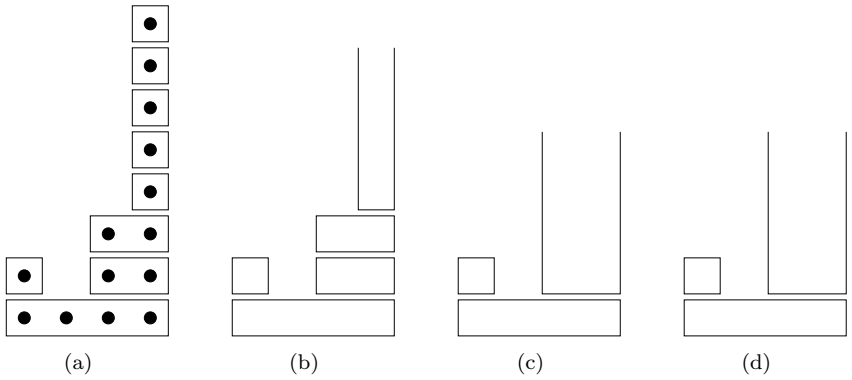

Figure 5: An execution of Algorithm 1 on an instance with $T = 4$, $K = 3$, $\delta = (1, 2, 4)$, $\gamma = (4, 6, 11)$, $r = (2, 1, 3, 8)$ and $M = 4$. (a) S_0 is the optimum solution for the corresponding MPP instance. (b) In S_1 we replace type-1 permits with type-1 group permits because this improves 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 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 1 if the considered interval is contained in the interval defined by some group permit in Q. We present a pseudocode in Algorithm 2. Note that Algorithm 1 is equivalent to running Algorithm 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 $S'_k \leftarrow S'_{k-1}$, $Q'_k \leftarrow Q'_{k-1}$;
4 **for** $\hat{t} \leftarrow 0$ **to** $T - 1$ **step** δ_k **do**
5 **if** $k \leq k'$ **and** $\hat{t} \in [t', t' + \delta_{k'})$ **for some** $(k', t') \in Q$ **then**
6 **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**
7 $S'_k \leftarrow S'_k \setminus S'_{k-1}[k, \hat{t}]$;
8 $Q'_k \leftarrow (Q'_k \setminus Q'_{k-1}[k, \hat{t}]) \cup \{(k, \hat{t})\}$;
9 **return** (S'_K, Q'_K);

Algorithm 2: A modified version of Algorithm 1, which will be useful for our analysis.

Lemma 18. *Given a* GPP *instance I, for any $Q \subseteq [K] \times \mathbb{Z}_+$, the cost of the solution returned by running Algorithm 1 on I is at most the cost of the solution returned by running Algorithm 2 on (I, Q).*

Proof. Due to IM, if Algorithm 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 2.) Thus it is easy to prove, by induction on k, that for any pair $(k, \hat{t})$ for which both Algorithm 1 and Algorithm 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 1 never increases the cost of the returned solution. Since Algorithm 1 considers each interval that Algorithm 2 does, the lemma holds. □

Our goal is to prove that Algorithm 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 1 on I. Let (S^*, Q^*) be an optimum solution of I which always utilizes the largest possible group permits, and let (S', Q') be the solution returned by Algorithm 2 on (I, Q^*). We claim that

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

The first inequality follows from Lemma 18, so it suffices to prove the second inequality.

We partition S' in two multisets, $S'_{\leq}$ and $S'_{>}$. Let $S'_{\leq} := \bigcup_{(k,\hat{t}) \in Q^*} S'[k, \hat{t}]$, i.e., $S'_{\leq}$ is the multiset of single permits that are contained in the interval defined by some group permit in Q^*. Let $S'_{>} := S' \setminus S'_{\leq}$. (See Figure 6.) Since Algorithm 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,

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

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.

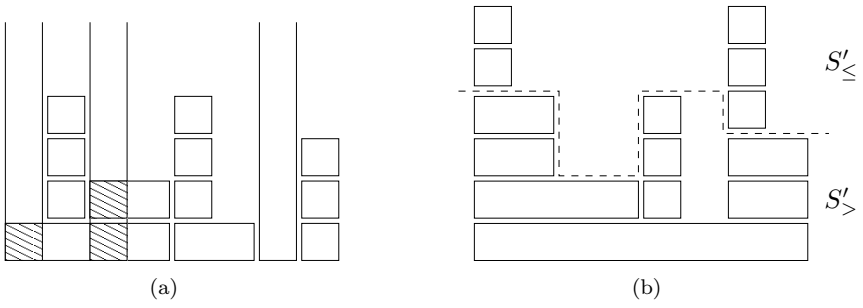

Figure 6: (a) Optimum solution; (b) Permits in $S'_{\leq}$ are contained in the interval defined by some group permit in the optimum solution.

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. Note that, due to IM, a group permit can only overlap with single permits of larger length; otherwise, we could remove the single permit.

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 8 for an example.

Lemma 19. *A group permit in Q^* overlaps with at most $\lceil M \rceil - 1$ permits in $S'_{>}$.*

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.

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 9(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 9(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 $(k, \hat{t})$).

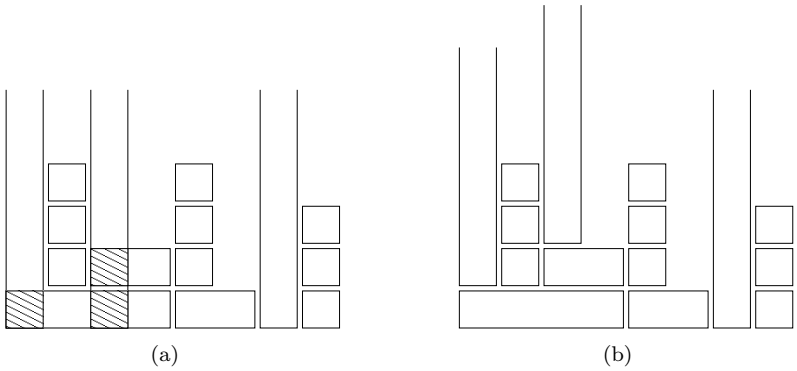

Figure 7: (a) A solution of GPP; (b) Solution (a) reorganized so that group and single permits do not overlap.

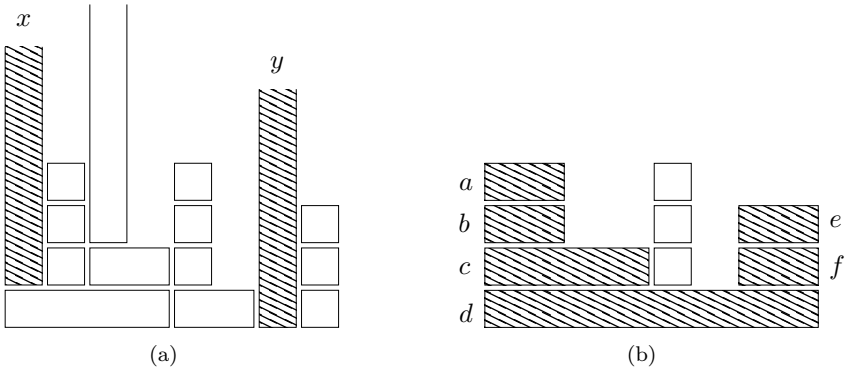

Figure 8: (a) Optimum solution; (b) $S_>$. Group permit x overlaps with permits a, b and c, and y overlaps with d, e and f.

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 9(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 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 the largest possible group permits. □

Lemma 20. $\text{cost}(S'_>) \leq \text{opt}(I)$.

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 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.

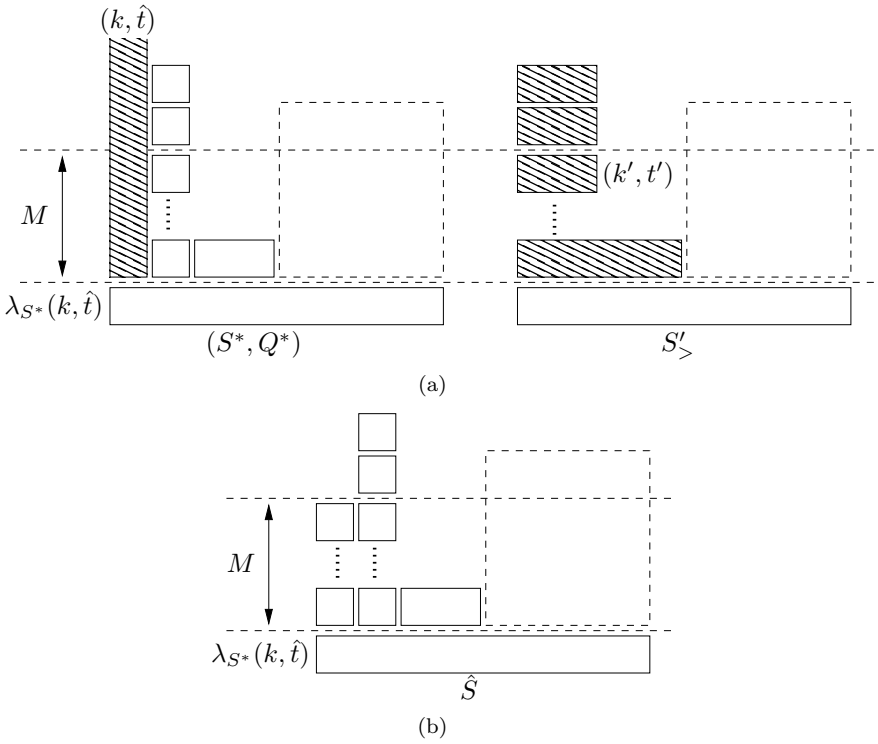

Figure 9: (a) Suppose by contradiction that $(k, \hat{t})$ overlaps at least M permits in $S'_>$; consider the M lower levels and let (k', t') be the permit that overlaps with $(k, \hat{t})$ at level $\lambda_{S^*}(k, \hat{t}) + M$. (b) Replace each group permit with M single permits of the corresponding type.

Theorem 21. *Algorithm 1 is a 2-approximation for* GPP *under* IM. *There is an 8-approximation algorithm for* GPP *with arbitrary instances.*

To conclude this section, we present an instance which shows that Algorithm 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 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 $K - 1$. We leave to the reader to check that the approximation factor for this instance approaches 4/3. An open question is whether Algorithm 1 has approximation factor smaller than 2 under IM. Experiments on random instances never attained approximation factor greater than 4/3. Another open question is whether we can obtain approximation factor better than 8 for the general case when we do not assume IM.

4.2. 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 1 and 2.

One obvious online algorithm 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. We assume IM.

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

Step 1. Run Algorithm 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).

Step 4 can be implemented in polynomial time by representing S_t by a set of tuples in the form $(\hat{t}, \ell, k, q)$, where $\hat{t}$ is a starting time, ℓ is the first demand level covered, k is a permit type and q is a multiplicity. A tuple $(\hat{t}, \ell, k, q)$ means that q copies of permit $(k, \hat{t})$ are stacked to supply the demand of levels $[\ell, \ell + q)$. In some sense, $(\hat{t}, \ell)$ is the coordinate of the bottom-left corner of the tuple, and k and q encode their width and height, respectively. (See Figure 10.) Due to HTO, permits of same k and $\hat{t}$ are used to serve contiguous levels, so we can ensure that we have at most one tuple for each pair $(k, \hat{t})$. Thus, we can find which permits cover levels $\kappa_t + 1, \ldots, r_t$ in polynomial time.

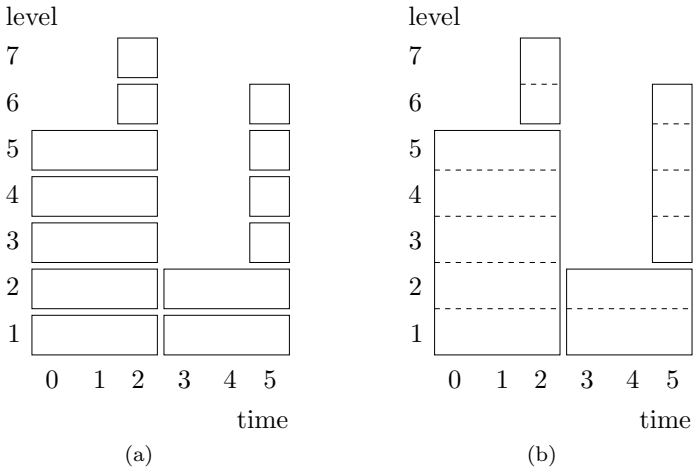

Figure 10: Consider an instance with $\delta = (1, 3, 6)$. (a) Given a multiset of permits with 5 copies of $(2, 0)$, 2 copies of $(1, 2)$, 2 copies of $(2, 3)$ and 4 copies of $(1, 5)$, (b) we represent them as tuples $(0, 1, 2, 5)$, $(2, 6, 1, 2)$, $(3, 1, 2, 2)$ and $(5, 3, 1, 4)$.

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

$$\text{cost}(\text{AlgOGPP}(I_t)) \leq 4K \cdot \text{opt}(I_t). \tag{2}$$

We actually prove the following stronger result:

$$\text{cost}(\text{AlgOGPP}(I_t)) \leq 2K \cdot (\text{cost}(S_t) + M \cdot \text{cost}(Q_t)), \tag{3}$$

where (S_t, Q_t) is the solution obtained by running Algorithm 1 on I_t. Thus, Equation (2) follows from Theorem 21 and Equation (3).

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 (3), we share the cost of AlgOGPP(I_t) among the permits in $\mathcal{S}_t$. More precisely, we define an assignment $\Phi_t : \mathcal{S}_t \to \mathbb{R}_+$ 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 $\Phi_t(p) \le 2K \cdot \text{cost}(p)$ (Lemma 27) for every $p \in \mathcal{S}_t$. This implies that

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

and so Equation (3) holds. We define Φ_t in an inductive manner. For $t = 0$, take $\Phi_0(p) := \text{cost}(p)$ for each $p \in \mathcal{S}_0$. Since before Step 2 is executed for $t = 0$ no demand is covered yet, clearly AlgOGPP(I_0) $= \mathcal{S}_0$, so $\Phi_0(\mathcal{S}_0) = \text{cost}(\text{AlgOGPP}(I_0))$. Now suppose Φ_{t-1} is defined, and consider the solution $\mathcal{S}'_{t-1}$ obtained by running Algorithm 2 on (I_{t-1}, Q_t). Before defining Φ_t, we define an assignment $\Phi'_{t-1} : \mathcal{S}'_{t-1} \to \mathbb{R}_+$ 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 $k' \le k$ 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 $\mathcal{S}'(p) = \{p\}$.

Fact 22. *For $t > 0$, each permit in $\mathcal{S}_{t-1}$ contains some permit in $\mathcal{S}'_{t-1}$, and each permit in $\mathcal{S}'_{t-1}$ is contained in a unique permit in $\mathcal{S}_{t-1}$.*

Proof. For the first claim, note that both $\mathcal{S}'_{t-1}$ and $\mathcal{S}_{t-1}$ are built from the same MPP solution for I_{t-1}, and Algorithm 1 executes Lines 5-7 for each pair $(k, \hat{t})$ for which Algorithm 2 executes Lines 6-8. Then the second claim follows simply from HTO. □

Now let us define Φ'_{t-1}. 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. Due to Fact 22, $\Phi'_{t-1}(\mathcal{S}'_{t-1}) = \Phi_{t-1}(\mathcal{S}_{t-1})$, 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 23. *For each $p \in \mathcal{S}_{t-1}$ and every $p' \in \mathcal{S}'_{t-1}(p)$, if $\Phi_{t-1}(p) \le \alpha \cdot \text{cost}(p)$ for some $\alpha \ge 0$, then we have that $\Phi'_{t-1}(p') \le \alpha \cdot \text{cost}(p')$.*

Proof. We have that $\text{cost}(p) \le \text{cost}(\mathcal{S}'_{t-1}(p))$, by a similar argument to the proof of Lemma 18. Thus,

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

□

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

Fact 24. *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 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 AlgMPP(I_{t-1}) is contained in some permit in AlgMPP(I_t). The uniqueness simply follows from HTO. □

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

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

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

Proof. If p is a group permit, then Algorithm 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. □

Fact 26. *If $\mathcal{S}_t$ contains a single permit $p = (k, \hat{t}, \lambda, g)$, then $\text{AlgOGPP}(I_t)$ has some permit that covers level λ for the whole interval $[\hat{t}, \hat{t} + \delta_k)$.*

Proof. Let $\tilde{t} \leq t$ be the first moment that $\text{AlgMPP}(I_{\tilde{t}})$ decides to buy p. Due to Lemma 14 and Equation (1), $r_{\tilde{t}} \geq \lambda$ and $(\tilde{t}, \lambda)$ was not covered by $\text{AlgMPP}(I_{t'})$, for $0 \leq t' < \tilde{t}$. If $(\tilde{t}, \lambda)$ is covered by $\text{AlgOGPP}(I_{\tilde{t}-1})$, it must be by a group permit that covers p (see Footnote 8 to understand why this may happen). Thus, $\text{AlgOGPP}(I_{\tilde{t}})$ covers $(\tilde{t}, \lambda)$ either by p or by a group permit that contains p.

Lemma 27. *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[8]: for each $p = (k, \hat{t}, \lambda, g) \in \mathcal{S}_t$,

(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) $\Phi_t(p) \leq (2k-1) \cdot \text{cost}(p)$, $g = \text{SINGLE}$ and $\text{AlgOGPP}(I_t)$ does not buy any group permit (k^*, t^*) with $k^* \geq k$ and $\hat{t} \in [t^*, t^* + \delta_{k^*})$.

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 $t - 1$. We claim that the induction hypothesis implies that, for each $p' = (k', t', \lambda', g') \in \mathcal{S}'_{t-1}$, we have that

(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') $\Phi'_{t-1}(p') \leq (2k' - 1) \cdot \text{cost}(p')$, $g' = \text{SINGLE}$ and $\text{AlgOGPP}(I_{t-1})$ does not buy any group permit (k^*, t^*) with $k^* \geq k'$ and $t' \in [t^*, t^* + \delta_{k^*})$.

If $g' = \text{GROUP}$, then (i') holds for p' from Facts 22 and 23. If $g' = \text{SINGLE}$, let $\bar{p} = (\bar{k}, \bar{t}, \bar{\lambda}, \bar{g}) \in \mathcal{S}_{t-1}$ such that $p' \in \mathcal{S}_{t-1}(\bar{p})$, which is unique due to Fact 22. If $\bar{g} = \text{GROUP}$, then (i') holds for p' from Facts 22 and 23. If $\bar{g} = \text{SINGLE}$, then $k' = \bar{k}$, since $\mathcal{S}_{t-1}$ and $\mathcal{S}'_{t-1}$ are built from $\text{AlgMPP}(I_{t-1})$, and thus (ii') holds for p'.

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 holds by Fact 25. Hence, (i) holds for p since group permit (k^*, t^*) contains p.

[8] 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. Intuitively, a demand of level λ that is covered by a group permit of type k^* in $\mathcal{S}_{t-1}$ may be covered by a permit of type $k < k^*$ in $\mathcal{S}_t$, because $\text{AlgMPP}(I_t)$ may decide to buy a permit of type $k' > k^*$ for some level $\lambda' < \lambda$ that was covered by the same group permit of type k^* in $\mathcal{S}_{t-1}$.

2. So assume that each permit in $\mathcal{S}'_{t-1}(p)$ satisfies (ii'), or every k^* for which it satisfies (i') is such that $k^* < k$. We have two subcases.
   (2a) If $g =$ 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 $\Phi'_{t-1}(p') \leq (2k-1) \cdot \text{cost}(p')$. Since AlgOGPP has not bought any group permit that contains p, it buys p in Step 2. Therefore,

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

   where the second inequality is due to Fact 25. 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 $k' = k$. Due to Facts 22 and 26, AlgOGPP(I_{t-1}) has bought some permit that covers p, so 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 (ii'), or every k^* for which they satisfy (i') is such that $k^* < k$. 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,

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

      where the third inequality is due to Fact 25, so p satisfies (ii).

□

Theorem 28. *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 [32] 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 [27], which we discuss in Section 5.3. We obtained Theorem 28 independently, and our analysis technique is rather different and relevant by itself.

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

5. The 2D Parking Permit Problem

In this section, we discuss 2DPP. This problem admits a $O(K)$-approximation algorithm and a $O(K)$-competitive online algorithm, both by Koufogiannakis and Young [27], which we discuss in Section 5.3. We show that this is an NP-hard problem via a reduction from the **change-making problem** (CM) [29], which is a variant of the unbounded integer knapsack problem.[9] Let us define CM: we are given a change value $R \in \mathbb{Z}_+$, and K types of coins with values $\phi_1, \ldots, \phi_K \in \mathbb{Z}_+$ and weights $\gamma_1, \ldots, \gamma_K \in \mathbb{R}^*_+$. We wish to find a minimum-weight multiset of coins, with as many coins of each type as we wish, whose value is at least R.[10] Note that 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 [21].

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

[9] 2DPP is a leasing variant of BABND on a single edge, and Awerbuch and Azar [4] pointed that BABND on a single edge corresponds to CM.

[10] Note that this is the unbounded knapsack problem with inverted signs. Also note that we do not make the usual requirement that the sum of the coins is exactly R.

each other. The proof of Fact 30 is similar to that of Fact 8: we reorder permits in non-decreasing order of capacity (which defines π), we round down each capacity to the closest power of 2, and we can cover a permit in the optimum solution of the original problem with at most two permits of the new instance. The proof of Lemma 32 is simply by composing Lemma 10 and Fact 30.

Hypothesis 29 (Hierarchical Capacity Property (HCP)). There is a permutation $\pi : [K] \to [K]$ such that $\phi_{\pi(k-1)}$ divides $\phi_{\pi(k)}$ for $k = 2, \ldots, K$.

Fact 30. *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*.*

Hypothesis 31 (2D Interval Model (2DIM)). We assume IM and HCP.

Lemma 32. *If there is an α-competitive algorithm for* 2DPP *under* 2DIM*, then there is an 8α-competitive algorithm for arbitrary instances.*

Note that Lemma 13 is true for 2DPP even if we do not assume HCP.

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.

For H2DPP under 2DIM, we can assume without loss of generality that

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

For O2DPP under 2DIM, we can assume without loss of generality that Equation (1) holds and that

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

Fact 33. CM *is no longer NP-hard 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. □

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 5.2, we present a polynomial-time algorithm for H2DPP which is exact under 2DIM.

In the following sections, we show: (i) that the pseudo-polynomial approximation algorithm for H2DPP by Hu *et al.*, which they prove to be exact under 2DIM, works for generic 2DPP under IM; (ii) how to turn that algorithm into polynomial time for H2DPP under 2DIM, turning their $O(K)$-competitive online algorithm for H2DPP into polynomial time as well; (iii) how to obtain general results for 2DPP via the work on the covering problem by Koufogiannakis and Young [27].

5.1. A Pseudo-Polynomial Algorithm for Generic 2DPP

In this section we show that the pseudo-polynomial offline algorithm that Hu *et al.* [20] proposed for H2DPP is indeed exact for generic 2DPP under IM. Moreover, our version of the algorithm is simpler, and we present it in a clearer way. 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. 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, as we illustrate in Figure 11.

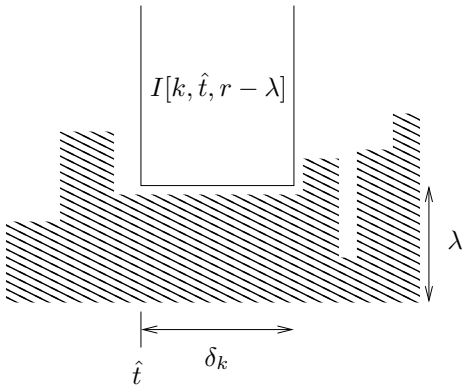

Figure 11: Given an instance I, $I[k, \hat{t}, r - \lambda]$ is the sub-instance in which we only allow permits of types $1, \ldots, k$, we only consider interval $[\hat{t}, \hat{t} + \delta_k)$, and we remove λ from the demand of each day in this interval.

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. Then,

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

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 $1, \ldots, k-1$. The proof of this recurrence follows from a standard optimal substructure argument that uses HTO.

In [20], Hu *et al.* show how to implement this recurrence in time $\mathrm{O}(T^2 \cdot R)$, where $R = \max_{t=0,\ldots,T-1} r_t$, via dynamic programming by preprocessing the input. We point that it can be implemented in time $\mathrm{O}(K \cdot T \cdot R)$, which is usually better since $K = \mathrm{o}(T)$ for typical inputs. Also, if $\delta_1 < \cdots < \delta_K$, the same implementation consumes time $\mathrm{O}(T \cdot R)$. This can be improved to $\mathrm{O}(K \cdot \bar{T} \cdot R)$, where $\bar{T}$ is the number of days with positive demand, by using linked lists. Note that this algorithm is pseudo-polynomial, since 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.

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

5.2. Hierarchical 2D Parking Permit

In this section we show how to turn the algorithm of Section 5.1 into an exact polynomial-time algorithm for H2DPP under 2DIM.

The algorithm uses the representation we discussed in Section 4.2: a multiset of permits is represented 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 $[\ell, \ell + q \cdot \phi_k)$. (See Figure 10.) 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 $t = 0, \ldots, T - 1$.

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 5.1, 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 12). Due to HTO, demand decreases as level increases, so we can perform

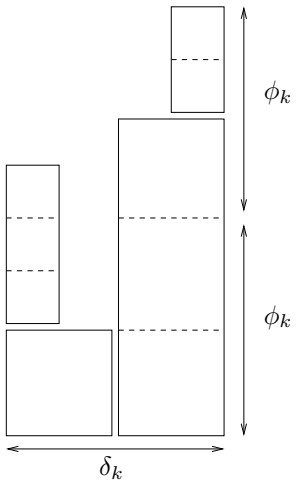

Figure 12: We can split the optimum solution for $I[k-1]$ into layers of height ϕ_k. We may split a tuple in two or more parts but, due to 2DIM, we never split a permit.

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 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 $\mathrm{O}(\lg R)$ subproblems. (See Figure 13.)

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.* [20] utilizes their offline algorithm as a black box 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 $\mathrm{O}(K)$-competitive.[11] By simply replacing their offline algorithm with ours, we obtain a polynomial-time algorithm which is also $\mathrm{O}(K)$-

[11] 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 Section 3.

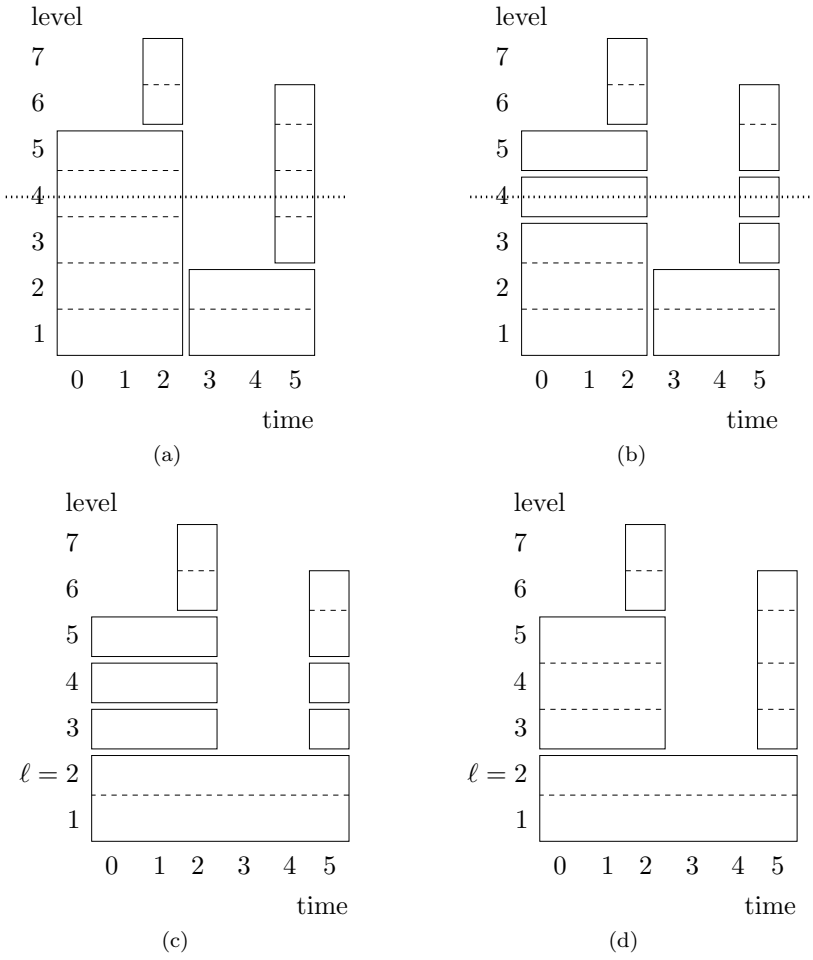

Figure 13: Consider an instance with $\delta = (1, 3, 6)$ and $\phi = (1, 1, 1)$, and let (a) be an optimum solution for $I[2]$. In order to compute the optimum for $I[3]$, we begin by splitting the solution at level 4. (b) We obtain a solution with tuples $(0, 1, 2, 3)$, $(3, 1, 2, 2)$ and $(5, 3, 1, 1)$ 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.

competitive. This result can also be obtained via the algorithm for the covering problem by Koufogiannakis and Young [27], which we discuss in the next section.

Hu *et al.* [20] 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.

5.3. 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 [27]. 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, where $R = \max_{t=0,\ldots,T-1} r_t$.

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

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 [27] 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.

6. Consequences for Network Leasing Problems

In this section we discuss the consequences of our results for MPP, GPP and 2DPP on leasing variants of SN, RoB and BaBND. Roughly speaking, each network leasing problem reduces to the corresponding parking problem if the input metric is a tree. Using the technique of approximating a metric by a tree metric [6, 11], a solution for a generic input can be obtained, losing some guarantee of quality. This idea can be formalized as follows.[12]

Theorem 34 ([6, 11]). *Given a minimization problem on a finite metric (V, d) whose objective function is a non-negative linear combination of distances in d, if there is an α-competitive algorithm for the special case of tree metrics, then there is a randomized $O(\alpha \cdot \lg |V|)$-competitive algorithm for the general case.*

Before we go into the leasing variants of the problems, let us list some literature for their traditional versions.[13] SN was proposed by Jain [23], who gave a 2-approximation algorithm. The online version of the problem with n request pairs admits a $O(\lg n)$-competitive algorithm, due to Umboh [37]. RoB was proposed by Karger and Minkoff [25]. The best current results for RoB are a 3.55-approximation for the single-source case [17] and a 5-approximation for the multi-source case [13], and there are $O(\lg n)$-competitive online algorithms [5, 37]. BaBND was proposed by Awerbuch and Azar [4], who gave a $O(\lg n)$-approximation algorithm and a $O(\lg |V|)$-competitive online algorithm, based on the technique of Theorem 34. For the single-source case of BaBND, Grandoni and Italiano [16] gave a 24.92-approximation algorithm, and Gupta *et al.* [18] gave a deterministic $O(\lg n)$-competitive online algorithm. All these problems are NP-hard and their online versions have competitive ratio $\Omega(\lg n)$, since they are generalizations of the Steiner tree problem, whose NP-hardness and online lower bound were proved in [15, 22], respectively. Also, for multi-source BaBND, there is no $O(\lg^{\frac{1}{4}-\epsilon} n)$-approximation for any constant $\epsilon > 0$, unless NP $\subseteq$ ZPTIME$(n^{\text{polylog } n})$ [2].

6.1. Steiner Network Leasing

The input for the **Steiner network leasing problem** (SNLe) consists of a complete graph $G = (V, E)$, a distance function $d : V \times V \to \mathbb{R}_+$ satisfying symmetry and triangle inequality, K leasing types with lengths of time $\delta_1, \ldots, \delta_K$ and scaling costs $\gamma_1, \ldots, \gamma_K$, 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 $S \subseteq E \times [K] \times \mathbb{Z}_+$

[12] See [38, Sections 8.5 and 8.6] for a nice presentation of the technique and its application.

[13] We give a reference for the seminal paper and for the current best result for each problem; check the latter for a broader literature review.

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. Thus, we obtain a solution to a generic graph by approximating (V, d) by a tree metric. In the offline setting, we can build a tree metric on the metric restricted to the n requested pairs, incurring in a $O(\lg n)$ distortion factor. In the online setting, requested pairs are given in an online manner, edges must be leased to connect pairs as they arrive, and no edge leases may be removed. Thus, we cannot build a tree metric only on the requested pairs. Instead, we have to approximate the entire metric (V, d), incurring in a $O(\lg |V|)$ distortion factor. Since we have a constant-approximation algorithm and an online $O(\lg K)$-competitive algorithm for MPP, we obtain the following result via Theorem 34.

Theorem 35. *There are a randomized $O(\lg n)$-approximation algorithm and a randomized $O(\lg K \lg |V|)$-competitive online algorithm for* SNLE.

6.2. Leasing Rent-or-Buy

In the **leasing rent-or-buy problem** (LEROB), we are given a complete graph $G = (V, E)$ with a distance function $d : V \times V \to \mathbb{R}_+$ satisfying symmetry and triangle inequality, K leasing types with lengths of time $\delta_1, \ldots, \delta_K$ and scaling costs $\gamma_1, \ldots, \gamma_K$, 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 [3], 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 6.3.

6.3. Leasing Buy-at-Bulk Network Design

In the **leasing buy-at-bulk network design problem** (LEBABND), we are given a distance function d between the vertices in a complete graph $G = (V, E)$, K types of cables with lengths of time $\delta_1, \ldots, \delta_K$, capacities $\phi_1, \ldots, \phi_K$ and costs per unit of distance $\gamma_1, \ldots, \gamma_K$, 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 $\delta_1, \ldots, \delta_K$ with corresponding time scaling costs $\gamma_1, \ldots, \gamma_K$, and L capacities $\phi_1, \ldots, \phi_L$ with corresponding capacity scaling costs $\mu_1, \ldots, \mu_L$. 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[14] [3], 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[15]; this yields a randomized $O(K \lg n)$-approximation for arbitrary metrics by Theorem 34. By combining the algorithm by Koufogiannakis and Young [27] with Theorem 34, we obtain a $O(KL \lg |V|)$-competitive online algorithm.

7. Discussion

In this paper we address generalizations of the parking permit problem [32] and their application to leasing variants of some classical network design problems. In particular, the problems we study in this paper combine time dynamicity, which is central to the parking permit problem, with the idea of **capacity dynamicity**.

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 and a $O(K)$-competitive online algorithm 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.

Summary of Results. In Tables 1 and 2 we summarize known approximation and competitive online results about the parking permit and network leasing problems we study.

Summary of Open Questions.

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 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.

[14] They define the problem in terms of a sub-additive capacity scaling cost function; as we mentioned in Section 1, this is equivalent to our definition up to a constant cost factor.

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

problem	offline setting			online setting	
	NP-hard?	general case	under IM / 2DIM	general case	under IM / 2DIM
PP	No	1 [32]		$4K$ [32]	K [32]
				randomized $O(\lg K)$ [32]	
MPP	No	1 (Section 3)		$4K$	K (Sec. 5.2), [27]
				randomized $O(\lg K)$ (Sec. 3)	
GPP	?	8	2 (Sec. 4.1)	$16K$	$4K$ (Sec. 4.2)
				$8K$	$2K$ [27]
O2DPP	Yes	$O(K)$ [3]		$4KL$	KL [27]
H2DPP	Yes	pseudo-8 [20]	pseudo-1 [20]	pseudo-$(8K)$ [20]	pseudo-K [20]
		8	1 (Sec. 5.2)	$4K$	K (Sec. 5.2), [27]
2DPP	Yes	pseudo-4	pseudo-1 (Sec. 5.1)	$4K$	K [27]
		$4K$	K [27]		

Table 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. All online problems have deterministic $K/3$ (under IM) and randomized $\Omega(\lg K)$ lower bounds [32].

problem	offline setting		online setting	
	single-source	multi-source	single-source	multi-source
SLe	$O(K)$ [3]	$O(\lg n)$ [32]	deterministic $O(K \lg n)$ [7]	randomized $O(\lg K \lg \lvert V\rvert)$ [32]
SNLe	$O(K)$ [3]	$O(\lg n)$ (Sec. 6.1)	$O(\lg K \lg \lvert V\rvert)$ (Sec. 6.1)	
LeRoB	$O(K)$ [3]	$O(\lg n)$ (Sec. 6.2)	$O(K \lg \lvert V\rvert)$ (Sec. 6.2), [11, 27]	
OLeBaBND	$O(K)$ [3]	$O(K \lg n)$ [3]	$O(KL \lg \lvert V\rvert)$ [11, 27]	
HLeBaBND	$O(\lg n)$ (Sec. 6.3)		$O(K \lg \lvert V\rvert)$ (Sec. 6.3), [11, 27]	
LeBaBND	pseudo-$O(\lg n)$ (Sec. 6.3)		$O(K \lg \lvert V\rvert)$ [11, 27]	
	$O(K \lg n)$ [11, 27]			

Table 2: Summary of known approximation and competitive online results for network leasing problems. A "pseudo-α" means a pseudo-polynomial α-approximation (α-competitive) algorithm. All offline problems are NP-hard [15], all offline multi-source BaBND problems have lower bound $\Omega(\lg^{1/4} n)$ [2], and all online problems have lower bound $\Omega(\lg K + \lg \min\{n, \delta_K\})$ [32, 1].

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 [3].
7. We do not know a randomized $o(K)$-competitive online algorithm for GPP or H2DPP.

Further Research Directions. Another future research direction is to study generalizations such as GPP and 2DPP to other leasing models, such as those defined in [28, 12].

[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] 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.

[3] 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.

[4] 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.

[5] B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized Steiner problem. *Theoretical Computer Science*, 324(2):313–324, 2004.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[16] 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.

[17] 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.

[18] 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.

[19] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

[20] 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.

[21] 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.

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

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

[24] Y. Jin, M. Hayashi, and A. Tagami. Online algorithms for cost-effective cloud selection with multiple demands. In *ITC 30: 2018 30th International Teletraffic Congress*, pages 37–45, 2018.

[25] 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.

[26] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1–4):79–119, 1988.

[27] C. Koufogiannakis and N. E. Young. Greedy Δ-approximation algorithm for covering with arbitrary constraints and submodular cost. *Algorithmica*, 66(1):113–152, 2013.

[28] 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.

[29] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report CS-178, Department of Electrical Engineering, Princeton University, 1975.

[30] A. Mäcker. Cost-efficient scheduling on machines from the cloud. *Journal of Combinatorial Optimization*, 36(4):1168–1194, 2018.

[31] C. Markarian. Leasing with uncertainty. In N. Kliewer, J. F. Ehmke, and R. Borndörfer, editors, *Operations Research Proceedings 2017*, pages 429–434, 2018.

[32] A. Meyerson. The parking permit problem. In *FOCS'05: 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–282, 2005.

[33] C. Nagarajan and D. P. Williamson. Offline and online facility leasing. *Discrete Optimization*, 10(4):361–370, 2013.

[34] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[35] 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.

[36] W. T. Tutte. Lectures on matroids. *Journal of Research National Bureau of Standards Section B*, 69:1–47, 1965.

[37] 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.

[38] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.